PRECONDITIONING PARAMETRIZED LINEAR SYSTEMS *

ARIELLE GRIM-MCNALLY†, ERIC DE STURLER‡, AND SERKAN GUGERCIN§

Abstract. Preconditioners are generally essential for fast convergence in the iterative solution of linear systems of equations. However, the computation of a good preconditioner can be expensive. So, while solving a sequence of many linear systems, it is advantageous to recycle preconditioners, that is, update a previous preconditioner and reuse the updated version. In this paper, we introduce a simple and effective method for doing this. Although our approach can be used for matrices changing slowly in any way, we focus on the important case of sequences of the type $(s_k\mathbf{E}(\mathbf{p})+\mathbf{A}(\mathbf{p}))\mathbf{x}_k = \mathbf{b}_k$, where the right hand side may or may not change. More general changes in matrices will be discussed in a future paper.

We update preconditioners by defining a map from a new matrix to a previous matrix, for example the first matrix in the sequence, and combine the preconditioner for this previous matrix with the map to define the new preconditioner. This approach has several advantages. *The update is entirely independent from the original preconditioner, so it can be applied to any preconditioner.* The possibly high cost of an initial preconditioner can be amortized over many linear solves. The cost of updating the preconditioner is more or less constant and independent of the original preconditioner. There is flexibility in balancing the quality of the map with the computational cost.

In the numerical experiments section we demonstrate good results for several applications.

Key words. Preconditioning, Updating Preconditioners, Krylov Subspace Methods, Sparse Approximate Inverse, Parameterized Systems, Model Reduction, Transient Hydraulic Tomography, Diffuse Optical Tomography

AMS subject classifications. 65F10

1. Introduction.

We consider the efficient computation of preconditioners for sequences of systems which change slowly. Such sequences can often be represented as

$$(s_k\mathbf{E}(\mathbf{p}) + \mathbf{A}(\mathbf{p}))\mathbf{x}_k = \mathbf{b}_k, \tag{1.1}$$

where the right hand side may or may not change. Here, s_k is a shift (often related to a frequency), and the matrices $\mathbf{E}$ and $\mathbf{A}$ are functions of a parameter vector $\mathbf{p}$. For example, this is implicitly the case for the Transient Hydraulic Tomography application (THT) discussed later in this paper and for diffuse optical tomography [31]. Preconditioners are often essential for fast iterative solutions of linear systems of equations, but the computation of a good preconditioner can be expensive. Therefore, we consider *recycling preconditioners*, that is, updating a previous preconditioner and reusing the updated version for solving a new linear system. For a sequence of linear systems, this may provide a substantial reduction in cost compared with computing a new preconditioner for each system or periodically computing a new preconditioner from scratch. The latter approach includes the important case of solving all systems with a single preconditioner, which we refer to as *reusing* the initial preconditioner.

The main idea underlying our approach comes from [4]. Given a sequence of matrices, $\mathbf{A}_k$, for $k = 0, 1, 2, \ldots$, and a good preconditioner $\mathbf{P}_0$ for $\mathbf{A}_0$ such that

*This material is based upon work supported by the National Science Foundation under grant numbers NSF-DMS 1025327 and NSF-DMS 1217156, and by the Air Force Office of Scientific Research under grant number AFOSR FA9550-12-1-0442

†Mathematics Department, VA Tech, Blacksburg, VA. (arielle5@vt.edu)

‡Mathematics Department, VA Tech, Blacksburg, VA. (sturler@vt.edu).

§Department of Mathematics, VA Tech, Blacksburg, VA. (gugercin@math.vt.edu).

$\mathbf{A}_0\mathbf{P}_0$ (or $\mathbf{P}_0\mathbf{A}_0$) yields fast convergence, we could compute for each system the *ideal map* $\widehat{\mathbf{N}}_k$ such that

$$\mathbf{A}_k\widehat{\mathbf{N}}_k = \mathbf{A}_0. \tag{1.2}$$

If we define the updated preconditioner as

$$\mathbf{P}_k = \widehat{\mathbf{N}}_k\mathbf{P}_0, \tag{1.3}$$

then, as a result, $\mathbf{A}_0\mathbf{P}_0 = \mathbf{A}_1\mathbf{P}_1 = \cdots = \mathbf{A}_k\mathbf{P}_k$. Therefore, $\mathbf{A}_k\widehat{\mathbf{N}}_k\mathbf{P}_0 = \mathbf{A}_0\mathbf{P}_0$ will yield the same fast convergence as the original preconditioned system, for each k. Note that (in general) the matrix $\widehat{\mathbf{N}}_k\mathbf{P}_0$ is never computed; in an iterative method, we can just multiply vectors successively by these two matrices (which does lead to some overhead). If computing these maps can be made cheap and the initial preconditioner is very good, we obtain fast convergence for all systems at low cost.

In [4], we dealt with a very long sequence of matrices in a Markov chain Monte Carlo (MCMC) process. These matrices change by one row at a time, $\mathbf{A}_{k+1} = \mathbf{A}_k + \mathbf{e}_{i_k}\mathbf{u}_k^T$, where i_k indicates which row changes and $\mathbf{u}_k$ is the change in the row. In this case, computing the ideal map comes more or less for free, as we already need to compute $\mathbf{u}_k^T\mathbf{A}_k^{-1}\mathbf{e}_{i_k}$ for the transition probability in the MCMC process. While this update is specific to the particular application, the approach proposed in this paper generalizes the idea of recycling preconditioners to any set of closely related matrices.

Our preconditioner update is advantageous in several ways. To compute the ideal map, $\widehat{\mathbf{N}}_k$, or an approximation, knowledge of the original preconditioner, $\mathbf{P}_0$, is not required. Therefore, the map is independent of $\mathbf{P}_0$ and can be applied to any type of preconditioner. Further, the cost of updating $\mathbf{P}_0$ is more or less constant and the potentially high cost of computing a good $\mathbf{P}_0$ can be amortized over many linear solves. In practice, we do not calculate the ideal map as in (1.2), but rather an approximation, $\mathbf{N}_k$ such that

$$\mathbf{A}_k\mathbf{N}_k \approx \mathbf{A}_0.$$

Depending on how good we want our approximation to be, there is also flexibility in balancing the quality of $\mathbf{N}_k$ with the cost of computing it. While there is the additional cost of applying the map in the matrix-vector product, usually this does not outweigh the cost of computing another preconditioner from scratch. We demonstrate this for several applications in Section 4.

Our update scheme is motivated by the Sparse Approximate Inverse (SAI), and so we refer to it as a Sparse Approximate Map, or SAM update. The SAI was proposed in [18] and further developed in [19, 30, 42, 47] and references therein. To define SAIs and SAMs we need the following definitions.

DEFINITION 1.1. *A sparsity pattern for $\mathbb{C}^{n\times n}$ is any subset of $\{1,2,\ldots,n\}\times\{1,2,\ldots,n\}$.*

DEFINITION 1.2. *Let S be a sparsity pattern for $\mathbb{C}^{n\times n}$. We define the subspace $\mathcal{S}\subseteq\mathbb{C}^{n\times n}$ as $\mathcal{S} = \{X\in\mathbb{C}^{n\times n} \mid X_{ij} = 0 \text{ if } (i,j)\notin S\}$.*

DEFINITION 1.3. *For $\mathbf{P},\mathbf{A}\in\mathbb{C}^{n\times n}$ and $\mathbf{I}$ the identity matrix in $\mathbb{C}^{n\times n}$, the Sparse Approximate Inverse, $\mathbf{P}$, for a matrix, $\mathbf{A}$, is defined as the minimizer of*

$$\min_{\mathbf{P}\in\mathcal{S}} \|\mathbf{I}-\mathbf{A}\mathbf{P}\|_F. \tag{1.4}$$

The computation of a SAI is easily parallelized as n independent small least squares problems, as discussed in [42]. While our preconditioner update (or SAM) can also be computed in parallel, we do not discuss this here.

Rather than considering the identity matrix in (1.4), other work has focused on replacing it with another matrix, sometimes referred to as a target matrix [47]. The problem then becomes

$$\min_{\mathbf{P}\in\mathcal{S}} \|\mathbf{B}-\mathbf{AP}\|_F, \tag{1.5}$$

where $\mathbf{P}$ is such that $\mathbf{AP}$ targets $\mathbf{B}$. In [30, 47], (1.5) is solved in order to improve a preconditioner, $\mathbf{B}^{-1}$, such that the preconditioned system $\mathbf{APB}^{-1}$ is closer to the identity matrix than $\mathbf{AB}^{-1}$. As a preconditioner, $\mathbf{B}^{-1}$ is generally available through an approximate factorization of $\mathbf{A}$ (or of $\mathbf{A}^{-1}$). However, the columns of $\mathbf{B}$ must be computed in order to solve (1.5), and the cost of constructing these columns can be relatively high. In special cases, the structure or type of matrix can be exploited. In an example using the advection-diffusion equation and targeting the Laplacian, the authors in [47] are able to use a fast solver for the action of $\mathbf{B}^{-1}$ with good results. In order to reduce the cost of explicitly constructing $\mathbf{B}$, iterative methods with numerical dropping are used to approximate the columns of $\mathbf{B}^{-1}$ in [30].

Our update scheme involves solving

$$\mathbf{N}_k = \arg\min_{\mathbf{N}\in\mathcal{S}} \|\mathbf{A}_k\mathbf{N}-\mathbf{A}_0\|_F, \tag{1.6}$$

where $\mathcal{S}$ is the subspace defined by a chosen sparsity pattern S, as given in Definition 1.2, and $\mathbf{A}_0$ and $\mathbf{A}_k$ are matrices from a given sequence. This paper focuses on solving (1.6) for each or selected k, but we can also incrementally apply such a map where, at shift k, we solve

$$\mathbf{N}_k = \arg\min_{\mathbf{N}\in\mathcal{S}} \|\mathbf{A}_k\mathbf{N}-\mathbf{A}_{k-1}\|_F$$

and define

$$\mathbf{P}_k = \mathbf{N}_k\mathbf{P}_{k-1} = \mathbf{N}_k\mathbf{N}_{k-1}\cdots\mathbf{N}_1\mathbf{P}_0, \tag{1.7}$$

or let

$$\mathbf{N}_k = \arg\min_{\mathbf{N}\in\mathcal{S}} \|\mathbf{A}_k\mathbf{N}-\mathbf{A}_j\|_F,$$

with

$$\mathbf{P}_k = \mathbf{N}_k\mathbf{P}_j, \tag{1.8}$$

for some j such that $0 < j < k$. When applying the preconditioner from the left, we can also take advantage of row-wise changes made to $\mathbf{A}_k$, as is the case with the QMC matrices described above. We can define

$$\mathbf{N}_k = \arg\min_{\mathbf{N}\in\mathcal{S}} \|\mathbf{N}\mathbf{A}_k-\mathbf{A}_0\|_F,$$

with

$$\mathbf{P}_k = \mathbf{P}_0\mathbf{N}_k. \tag{1.9}$$

In the case of (1.9), the computation of the map can be made significantly cheaper by considering only those rows of $\mathbf{A}_k$ that differ from $\mathbf{A}_0$ when computing the least

squares minimization. Applying the maps such as in (1.7), (1.8), and (1.9) is the focus of future research.

While the minimization in (1.6) has a form very similar to (1.5), there are fundamental differences. Computing (1.5) involves improving an existing preconditioner, $\mathbf{B}$, for a fixed matrix, $\mathbf{A}$, where for most preconditioners, $\|\mathbf{B} - \mathbf{A}\|_F$ is quite large, and so an accurate solution cannot be expected. Of course, if an accurate solution would be obtained, the benefit would be faster convergence rather than maintaining the same convergence. Our approach seeks to map one matrix to another *closely related one*, so we often can expect a relatively accurate solution. The high cost of computing the columns of $\mathbf{B}$ when solving (1.5) is avoided when solving (1.6), since the columns of $\mathbf{A}_0$ are readily available, as $\mathbf{A}_0$ is a previous matrix in the sequence of linear systems.

Other update schemes for sequences of matrices have been proposed. A cheap update to the factorized approximate inverse (AINV) preconditioner is discussed in [20]. However, this update requires that $\mathbf{P}_0$ is itself of AINV type. Several incremental, or iterative, update techniques to an ILU factorization are described in [26]. But again, these updates require the initial preconditioner to be itself an ILU preconditioner. Moreover, these update techniques seem relatively expensive; they were not competitive for the problems we considered.

Although the SAM updates discussed in this paper can be used for any set of closely related matrices, here we focus on shifted matrices of the form

$$\mathbf{A}_k = s_k\mathbf{E} + \mathbf{A}, \text{ [1]} \tag{1.10}$$

which arise in model reduction [7, 12, 13, 44, 45], oscillatory and transient hydraulic tomography (OHT/THT) [28], and diffuse optical tomography (DOT) [1, 32, 49, 60]. Note that for a parameterized medium (subsurface or tissue) the latter two problems result in sequences of type (1.1). While not explicitly considered in this paper, other work has focused on shifted systems where $\mathbf{E} = \mathbf{I}$. Flexible preconditioning is used for problems of this form in [10, 43]. In [2], the authors take advantage of the form of the shifted systems in certain model reduction applications and the shift invariance of Krylov subspaces. In Section 4, we demonstrate the effectiveness of SAM updates to applications from THT and model reduction. For THT, we consider three sequences of matrices, referred to as *early*, *middle*, and *late*, where each sequence includes twenty shifts. For model reduction, we examine two applications, Rail and Flow, with multiple sequences of six shifts each. We also demonstrate the effectiveness of our approach for a discretization of the Helmholtz equation $f = -\Delta u - k^2 u$, where u is an amplitude, Δ is the Laplacian, and k is the wave number. Indefinite systems, such as the discretized Helmholtz equation, can also arise in flow control where the systems can be unstable, resulting in eigenvalues that are in both the right- and left-half planes [25].

The shifts in these sequences can be real, as with the Rail and Flow matrices, or complex, as with the THT matrices. Often, the magnitude of the shifts is not large, and the shifted matrices are closely related. However, in model reduction applications the magnitude of the shifts can be quite large. More detail for each application will be provided in Section 4.

In Section 2, we analyze SAM preconditioner updates and their effect on the convergence of GMRES. In Section 3, we discuss how to implement SAMs. We present

[1] Analogous results are obtained for $\mathbf{A}_k = s_k\mathbf{E} - \mathbf{A} = s_k\mathbf{E} + \widetilde{\mathbf{A}}$ with $\widetilde{\mathbf{A}} = -\mathbf{A}$.

the results of the SAM preconditioner updates for the applications described above in Section 4. Finally, we provide some conclusions in Section 5.

2. Analysis of Sparse Approximate Maps and their Effect on Convergence. We assume that our matrices take the form (1.10). We define the ideal map, $\widehat{\mathbf{N}}_k$, such that (1.2) holds. As we consider applications in which a sequence of linear systems must be solved, we assume that the matrices $\mathbf{A}_k$ are invertible for all k. Hence, we can also write $\widehat{\mathbf{N}}_k$ as

$$\widehat{\mathbf{N}}_k = \mathbf{A}_k^{-1}\mathbf{A}_0, \tag{2.1}$$

and, clearly, $\widehat{\mathbf{N}}_k$ is invertible (for all k). For a given subspace, $\mathcal{S}$, as defined in Definition 1.2, the *least squares (LS) map*, $\mathbf{N}_k$, is the solution to (1.6). We define $\mathbf{R}_k$, the residual of the LS map at shift k, as

$$\mathbf{R}_k = \mathbf{A}_k\mathbf{N}_k - \mathbf{A}_0. \tag{2.2}$$

We would like to analyze the convergence of GMRES when using these maps. The following theorem bounds the convergence of GMRES for matrices that can be expressed as a small (in norm) perturbation of the identity. This result is well known and given here for ease of reference.

THEOREM 2.1. *Let* $\mathbf{I}, \mathbf{C} \in \mathbb{C}^{n\times n}$ *and* $\mathbf{A} = \mathbf{I}+\mathbf{C}$*, where* $\mathbf{I}$ *denotes the identity and* $\|C\|_2 \leq \rho < 1$*. Let* $\mathbf{r}_m = \mathbf{b} - \mathbf{A}\mathbf{x}_m$ *be the GMRES residual, with* $\mathbf{x}_m \in \mathcal{K}^m(\mathbf{A}; \mathbf{r}_0) = \text{Span}(\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\})$*. Then* $\|\mathbf{r}_m\|_2 \leq \rho^m\|\mathbf{r}_0\|_2$*, and* $\rho_m \to 0$ *for* $m \to \infty$*.*

Proof. We refer to [27, 40, 50] for a proof of this theorem and related discussion. ∎

Note that in exact arithmetic $\mathbf{r}_n = 0$, but in practice n is very large, and we are interested in good convergences for $m \ll n$. Clearly, in Theorem 2.1, the smaller ρ is ($0 \leq \rho < 1$), the faster GMRES will converge. However, ρ need not be very small in order for GMRES to converge rapidly (consider, for example, $\rho = 1/3$). When the residual $\mathbf{R}_k$ is sufficiently small and the initial preconditioner is good, we can use Theorem 2.1 to guarantee rapid convergence of GMRES for the preconditioned matrix $\mathbf{A}_k\mathbf{P}_k = \mathbf{A}_k\mathbf{N}_k\mathbf{P}_0$. Since $\mathbf{A}_k\mathbf{N}_k = \mathbf{A}_0 + \mathbf{R}_k$, we have

$$\mathbf{A}_k\mathbf{P}_k = \mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 = \mathbf{A}_0\mathbf{P}_0 + \mathbf{R}_k\mathbf{P}_0. \tag{2.3}$$

We assume that the initial preconditioner $\mathbf{P}_0$ is a good approximation to $\mathbf{A}_0^{-1}$ such that $\mathbf{A}_0\mathbf{P}_0 = \mathbf{I} + \mathbf{K}$, with $\|\mathbf{K}\|_2 \leq \delta < 1$. From (2.3), we get

$$\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 = \mathbf{I} + \mathbf{K} + \mathbf{R}_k\mathbf{P}_0. \tag{2.4}$$

COROLLARY 2.2. *Let the preconditioned system* $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0$ *be as in (2.4) with with* $\mathbf{K}$ *as above. Then GMRES will converge if* $\|\mathbf{R}_k\mathbf{P}_0\|_2 < (1-\delta)$ *or* $\|\mathbf{R}_k\|_2 < (1-\delta)\|\mathbf{P}_0\|_2^{-1}$*.*

From (2.3) we see that $\mathbf{R}_k\mathbf{P}_0 = \mathbf{A}_k\mathbf{P}_k - \mathbf{A}_0\mathbf{P}_0$ represents the 'deterioration' of the preconditioned system from the original preconditioned system.

In Section 3 we show how to compute SAMs efficiently, and in Section 4 we show that these maps are (relatively) cheap to compute. Therefore, computing a very good preconditioner such that we satisfy the assumptions of Corollary 2.2 is reasonable.

While our convergence results have been defined in terms of $\|\cdot\|_2$, we note that $\|\mathbf{R}_k\|_2 \leq \|\mathbf{R}_k\|_F$, which is available more or less for free while computing (1.6). $\|\mathbf{P}_0\|_F$

(or $\|\mathbf{P}_0\|_2$) can also be estimated. If $\mathbf{P}_0$ is a sparse approximate inverse itself, then computing $\|\mathbf{P}_0\|_F$ is trivial. Often, $\mathbf{P}_0$ is available in a factorized form, such as the incomplete LU factorization used in this paper. In this case, norm estimators can be used [46, Chapters 8, 14]. If necessary or cost effective, we can make $\|\mathbf{R}_k\|_F$ smaller by extending the sparsity pattern of $\mathbf{N}_k$ guided by Corollary 2.2. In practice, it may be sufficient for fast convergence to satisfy Corollary 2.2 only approximately, as a clustered spectrum with a few outliers generally leads to fast convergence as well [41, Chapter 3] and [62, Chapter 6]. Several strategies for choosing and extending the nonzero patterns of SAIs can be adapted to achieve a good map, $\mathbf{N}_k$. Adaptive strategies for computing the SAI are discussed in [42]. However, fixing the sparsity pattern greatly reduces the cost of computing the SAI. In our numerical experiments, we choose the nonzero pattern of $\mathbf{N}_k$ to be that of $\mathbf{A}_0$ or $\mathbf{A}_0^2$, though higher powers of $\mathbf{A}_0$ can also be chosen. Other a priori and adaptive choices are discussed in [29, 48] and the references therein.

Regardless of how the sparsity pattern is chosen, for a sequence of nested patterns with increasing numbers of nonzero entries, the following theorem guarantees a monotonic decrease in the size of $\|\mathbf{R}_k\|_F$.

THEOREM 2.3. *For a sequence of nested patterns,*

$$S_1 \subseteq S_2 \subseteq \cdots \subseteq S_t,$$

and the corresponding subspaces $\mathcal{S}_j$, let $\mathbf{R}_k^{(j)} = \mathbf{A}_k\mathbf{N}^{(j)} - \mathbf{A}_0$ with $\mathbf{N}^{(j)}$ the minimizer of (1.6) for $\mathcal{S} = \mathcal{S}_j$. Then

$$\|\mathbf{R}_k^{(t)}\|_F \leq \|\mathbf{R}_k^{(t-1)}\|_F \leq \cdots \leq \|\mathbf{R}_k^{(1)}\|_F. \tag{2.5}$$

Proof. Since the minimization problem (1.6) has a solution for any sparsity pattern, the proof follows directly from the fact that the sequence of nested patterns leads to a sequence of nested (sub)spaces. ∎

In the THT application, we observe that both the relative and absolute residuals tend to be small (in norm), with the exception of the larger shifts for the matrices from the 'early' sequence. We demonstrate this in Figure 2.1, which shows the relative and absolute residuals of the early, middle, and late THT matrices. When these residuals are small, GMRES converges rapidly and GMRES convergence deteriorates for the larger residuals; see Section 4.1.

To understand under which conditions we can expect small residuals for practical sparsity patterns (patterns that are not much denser than the system), we analyze the ideal map and its distance to the LS map, the solution to (1.6).

We first show that, in an appropriate norm, $\mathbf{N}_k$ is the best approximation to $\widehat{\mathbf{N}}_k$.

DEFINITION 2.4. *For any nonsingular matrix $\mathbf{B} \in \mathbb{C}^{n\times n}$, we define the Frobenius $\mathbf{B}$-norm of $\mathbf{X} \in \mathbb{C}^{n\times n}$ as*

$$\|\mathbf{X}\|_{F,\mathbf{B}} = \|\mathbf{B}\mathbf{X}\|_F.$$

It is easy to verify that the Frobenius $\mathbf{B}$-norm satisfies the properties of a norm.[2]

THEOREM 2.5. *Let S be a sparsity pattern, then $\mathbf{N}_k$ is the best approximation to $\widehat{\mathbf{N}}_k$ in the Frobenius $\mathbf{A}_k$-norm over the space $\mathcal{S}$.*

[2]In general, $\|\mathbf{X}\mathbf{Y}\|_{F,\mathbf{A}_k} \leq \|\mathbf{X}\|_{F,\mathbf{A}_k}\|\mathbf{Y}\|_{F,\mathbf{A}_k}$ does not hold. While this property is occasionally considered part of the definition of a norm for matrices, we consider only the four standard properties of a norm, omitting the submultiplicative property as a requirement.

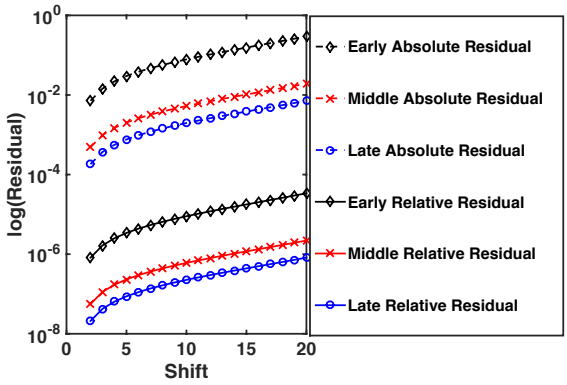


Fig. 2.1: The absolute residual $\|\mathbf{A}_k\mathbf{N}_k - \mathbf{A}_0\|_F$ (dotted lines) and relative residual $\frac{\|\mathbf{A}_k\mathbf{N}_k-\mathbf{A}_0\|_F}{\|\mathbf{A}_0\|_F}$ (solid lines) of the early ('◊'), middle ('×'), and late ('○') THT matrices.

Proof. We can represent the residual in terms of the ideal and LS maps,

$$\mathbf{R}_k = \mathbf{A}_k\mathbf{N}_k - \mathbf{A}_0 = \mathbf{A}_k\mathbf{N}_k - \mathbf{A}_k\widehat{\mathbf{N}}_k = \mathbf{A}_k(\mathbf{N}_k - \widehat{\mathbf{N}}_k). \tag{2.6}$$

Hence, solving (1.6) is equivalent to solving

$$\mathbf{N}_k = \arg\min_{\mathbf{N}\in\mathcal{S}} \|\mathbf{A}_k(\mathbf{N} - \widehat{\mathbf{N}}_k)\|_F = \arg\min_{\mathbf{N}\in\mathcal{S}} \|\mathbf{N} - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k}. \tag{2.7}$$

□

From (2.6)-(2.7) we have $\|\mathbf{R}_k\|_F = \|\mathbf{N}_k - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k}$, which immediately leads to the following Corollary (to Theorem 2.3)

COROLLARY 2.6. *For a sequence of nested patterns,*

$$S_1 \subseteq S_2 \subseteq \cdots \subseteq S_t,$$

and the corresponding subspaces $\mathcal{S}_j$, let $\mathbf{N}^{(j)}$ be the minimizer of (1.6) for $\mathcal{S} = \mathcal{S}_j$. Then

$$\|\mathbf{N}^{(t)} - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k} \le \|\mathbf{N}^{(t-1)} - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k} \le \cdots \le \|\mathbf{N}^{(1)} - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k} \tag{2.8}$$

Next, we examine the ideal map, $\widehat{\mathbf{N}}_k$. Following the analysis in [8], we consider the generalized eigenvalues and eigenvectors of $\mathbf{A}$ and $\mathbf{E}$ in (1.10).

$$\mathbf{AVM} = \mathbf{EV} \Leftrightarrow \mathbf{AVMV}^{-1} = \mathbf{E}, \tag{2.9}$$

where $\mathbf{M} = \operatorname{diag}(\mu_i)$ is the matrix of generalized eigenvalues (we assume diagonalizability). Substituting (2.9) into (1.2), and recalling the form of our shifted matrices (1.10), we have

$$(s_k\mathbf{AVMV}^{-1} + \mathbf{A})\widehat{\mathbf{N}}_k = (s_0\mathbf{AVMV}^{-1} + \mathbf{A}) \iff$$

$$\widehat{\mathbf{N}}_k = \mathbf{V}(s_k\mathbf{M} + \mathbf{I})^{-1}(s_0\mathbf{M} + \mathbf{I})\mathbf{V}^{-1} = \mathbf{VDV}^{-1} \tag{2.10}$$

and

$$\mathbf{D} = \text{diag}\left(\frac{s_0\mu_i + 1}{s_k\mu_i + 1}\right). \tag{2.11}$$

The assumption that all matrices $s_j\mathbf{E} + \mathbf{A}$ in a given sequence are invertible implies that the diagonal matrices $s_j\mathbf{M} + \mathbf{I}$ are invertible, and hence that $s_k\mu_i + 1 \neq 0$ and $s_0\mu_i + 1 \neq 0$ for $i = 1, \ldots, n$ in (2.11). With (2.10) a similarity transformation, the eigenvalues of $\widehat{\mathbf{N}}_k$ are the (diagonal) entries of $\mathbf{D}$, d_i. Equation (2.11) suggests clustering of the eigenvalues if $|\mu_i| \gg |s_0|, |s_k|$ for most of the eigenvalues, or if s_k is relatively close to s_0 with respect to most $|\mu_i|$. We will see that, if the condition number of $\mathbf{V}$, $\kappa_{F,\mathbf{A}_k}(\mathbf{V})$, is modest, clustering leads to a good approximation of the ideal map by the LS map.

In two of the applications discussed in this paper, we can expect the diagonal entries of (2.11) - and therefore the eigenvalues of $\widehat{\mathbf{N}}_k$ - to be clustered (possibly with some outliers). For stable dynamical systems, all eigenvalues have negative real part. Therefore, in model reduction for such systems, the shifts are generally computed (for example, by IRKA [45]), such that $\text{Re}(s_i)$ is close to zero and the shifts are often relatively close to one another. This reflects the fact that the reduced model needs to represent most accurately the modes of the system that decay slowest (corresponding to the eigenvalues with the smallest absolute real part). In addition, a stable dynamical system may have many eigenvalues with large absolute real part (corresponding to modes that decay very rapidly). Therefore, $s_0\mu_j \approx s_k\mu_j$ and $d_j \approx 1$, or $|\text{Re}(\mu_j)|$ is very large compared to $\text{Re}(s_i)$, and $\frac{s_0\mu_j+1}{s_k\mu_j+1} \approx \frac{\mu_j}{\mu_j} = 1$.

We also expect clustering of the eigenvalues of the ideal map for the THT matrices. In this application, the shifts come from a modified Talbot contour and tend to be quite small, particularly for larger values of time, as shown in Figure 4.1. For more information on how these contours, and the parameters which define them, are determined, we refer to [63]. When these shifts are small and relatively close to one another, as is the case with the middle and late THT matrices, $s_0\mu_j$ and $s_k\mu_j$ are both small enough that $d_j \approx 1$. More information on the THT matrices is provided later in this section as well as in Section 4.1, and more detail on the model reduction matrices is given in Section 4.2.

Clustering implies a small ϵ_D such that, for all i,

$$|\bar{d} - d_i| < \frac{\epsilon_D}{\sqrt{n}}.$$

where $\bar{d}$ is the average of all d_i or another appropriate center for the cluster, such as, $\bar{d} = \arg\min_{\mathbb{C}} \max_i |\bar{d} - d_i|$. Note that the average minimizes $\|\mathbf{D} - \bar{d}\mathbf{I}\|_F$, whereas the minimax solution minimizes $\|\mathbf{D} - \bar{d}\mathbf{I}\|_2$. Writing $\mathbf{D} = \bar{d}\mathbf{I} + \widehat{\mathbf{F}}$, with $\widehat{\mathbf{F}} = (\mathbf{D} - \bar{d}\mathbf{I})$ and therefore $\|\widehat{\mathbf{F}}\|_F < \epsilon_D$, we have

$$\widehat{\mathbf{N}}_k = \mathbf{V}\mathbf{D}\mathbf{V}^{-1} = \bar{d}\mathbf{I} + \mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}. \tag{2.12}$$

Now assume that the chosen sparsity pattern, S, contains the diagonal. This is often the case and can easily be ensured. Then

$$S \supseteq S_0 = \{(1,1), (2,2), \ldots, (n,n)\}.$$

Since $\bar{d}\mathbf{I} \in \mathcal{S}_0$, the subspace corresponding to S_0, by Theorem 2.3,

$$\|\mathbf{N}_k - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k} \leq \|\bar{d}\mathbf{I} - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k} = \|\bar{d}\mathbf{I} - \bar{d}\mathbf{I} - \mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}\|_{F,\mathbf{A}_k} = \|\mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}\|_{F,\mathbf{A}_k}.$$

Therefore,

$$\|\mathbf{R}_k\|_F = \|\mathbf{N}_k - \widehat{\mathbf{N}}_k\|_{F,\mathbf{A}_k} \leq \|\mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}\|_{F,\mathbf{A}_k}. \tag{2.13}$$

Hence, for modest $\kappa_{F,\mathbf{A}_k}(\mathbf{V})$, $\|\mathbf{R}_k\|_F$ will also be small, and we can expect good convergence. Note that, in general, this bound is rather pessimistic, as $\mathbf{N}_k$ can provide a much better approximation than $\bar{d}\mathbf{I}$, and a further analysis of these approximation problems is a topic of further research.

Since the eigenvalues of $\widehat{\mathbf{N}}_k$ are not always perfectly clustered, we also consider $\|\mathbf{R}_k\|_F$ when $\widehat{\mathbf{N}}_k$ has clustered eigenvalues with a few outliers. We expect that, in that case,

$$\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 = \mathbf{I} + \widetilde{\mathbf{K}} + \mathbf{H}, \tag{2.14}$$

where $\|\widetilde{\mathbf{K}}\|_F$ is small and $\text{rank}\,(\mathbf{H}) = p \ll n$, but $\|\mathbf{H}\|_F$ is not small. We can still consider (2.12) for some appropriate $\bar{d}$; however, some of the (diagonal) coefficients of $\widehat{\mathbf{F}}$, $d_i - \bar{d}$, will not be small, and $\mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}$ will be the sum of a matrix with small norm and a low rank matrix with (typically) larger norm. Writing

$$\mathbf{N}_k = \bar{d}\mathbf{I} + \widetilde{\mathbf{N}},$$

where $\widetilde{\mathbf{N}}$ has the same sparsity pattern as $\mathbf{N}_k$, gives

$$\mathbf{R}_k = \mathbf{A}_k(\mathbf{N}_k - \widehat{\mathbf{N}}_k) = \mathbf{A}_k(\widetilde{\mathbf{N}} - \mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}). \tag{2.15}$$

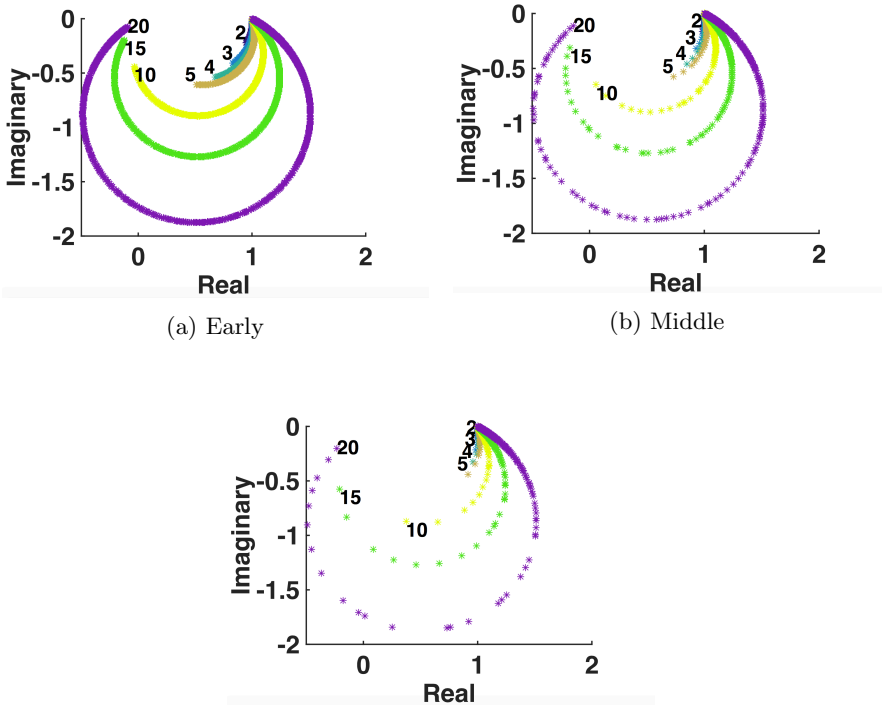
From (2.15) and Theorem 2.5 follows that $\widetilde{\mathbf{N}}$ is the best approximation of $\mathbf{V}\widehat{\mathbf{F}}\mathbf{V}^{-1}$ in the Frobenius $\mathbf{A}_k$-norm. Although a formal proof appears complicated, we expect $\mathbf{R}_k$, and hence $\mathbf{R}_k\mathbf{P}_0$, to also be the sum of a matrix with small norm and a low rank matrix with (typically) larger norm. We numerically verify this for the THT matrices below. Future work will focus on the conditions under which we can prove this to be true.

Figure 2.2 shows the eigenvalues of the ideal maps for the THT matrices.[3] Note that they are clustered for the first several shifts. This corresponds to when the residual of the SAMs is small, as shown in Figure 2.1. In Tables 4.3 and 4.9, it can be seen that the number of GMRES iterations for these shifts is low. Also in the case of the LS map, we observe clustering of the eigenvalues, as shown in Figure 2.3.

Figure 2.2 also shows that for the middle and late THT matrices, the eigenvalues of $\widehat{\mathbf{N}}_k$ are mostly clustered with relatively few outliers. We show the eigenvalues of the ideal map for shifts 11 through 15 of the late THT matrices in Figure 2.4(a). Examining the 60 largest singular values of $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 - \mathbf{I}$ for these same matrices and shifts in Figure 2.4(b), we see that there are a few singular values larger than 1 and only about ten larger than 0.5.[4] This shows that, for the THT matrices, it is reasonable to represent $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 - \mathbf{I}$ as a small perturbation of a low rank matrix (2.14) when the eigenvalues of $\widehat{\mathbf{N}}_k$ are clustered with few outliers. Figures 2.4(c) and 2.4(d) show that when the eigenvalues of $\widehat{\mathbf{N}}_k$ are not clustered, there are many more singular values of $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 - \mathbf{I}$ larger than 1.

[3] Figures 2.2 and 2.3 show the eigenvalues of the maps beginning at shift two, since an ILUTP factorization of $\mathbf{A}_0$ is computed for the first shift and the SAMs are applied at subsequent shifts.

[4] Our use of the word "small" is relative to Theorem 2.1.



(a) Early (b) Middle

(c) Late

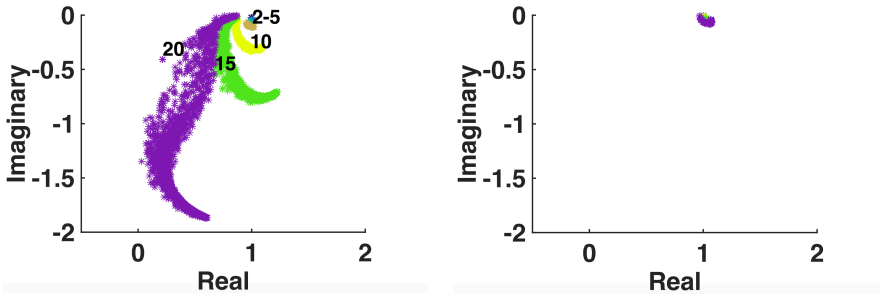
Fig. 2.2: Eigenvalues of the ideal map $\widehat{\mathbf{N}}_k$ for selected shifts (2-5, 10, 15, 20) of the THT matrices. Note the clustering for the first few shifts for each sequence (early, middle, and late). Note the clustering with relatively few outliers for the middle and late sequences ($n = 10\,201$).

3. Implementation. To efficiently compute the SAM updates given a sparsity pattern S, the solution of (1.6) must be implemented in sparse-sparse fashion. Furthermore, the nonzero pattern of the matrices often does not change, as is the case in the applications in this paper. Then the structure of the small least squares (LS) problems will be the same for every update. This makes it efficient to setup the data structures for the small LS problems just once, in advance.

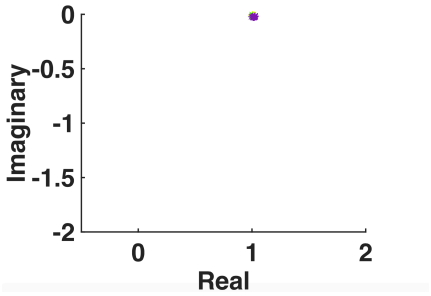
For ease of notation, we drop the indices and consider the problem

$$\mathbf{N} = \arg\min_{\widetilde{\mathbf{N}} \in \mathcal{S}} \|\mathbf{A}\widetilde{\mathbf{N}} - \widehat{\mathbf{A}}\|_F. \tag{3.1}$$

Given the pattern S, let s_k be the set of indices of the (potential) nonzeros in column k of $\mathbf{N}$: $s_k = \{i \,|\, (i,k) \in S\}$ and let $\mathcal{S}_k = \{\mathbf{x} \in \mathbb{C}^n \mid \mathbf{x}_i = 0 \text{ if } i \notin s_k\}$. Thus, for computing $\mathbf{n}_k$ (the kth column of $\mathbf{N}$) only the columns $\mathbf{a}_j$ with $j \in s_k$ of $\mathbf{A}$ matter. These columns themselves are sparse, and for the small LS problem defining $\mathbf{n}_k$ we need only consider rows i such that $a_{i,j} \neq 0$ for some $j \in s_k$. Note that if $\hat{a}_{i,k} \neq 0$ but $a_{i,j} = 0$ for all $j \in s_k$, row i is irrelevant for computing $\mathbf{n}_k$ since $\mathbf{e}_k \perp \text{Span}(\{\mathbf{a}_j \,|\, j \in s_k\})$. However, if we wish to compute the residual ($\mathbf{R} = \mathbf{AN} - \widehat{\mathbf{A}}$)



(a) Early (b) Middle



(c) Late

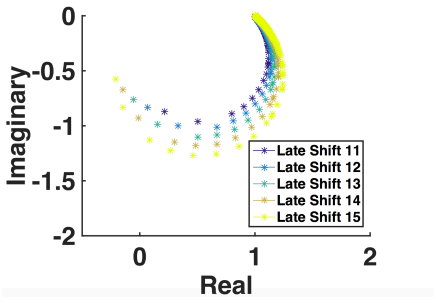
Fig. 2.3: Eigenvalues of the LS map, $\mathbf{N}_k$, for selected shifts (2-5, 10, 15, 20) of the THT Matrices.

or its norm, in addition to $\mathbf{N}$, we need to include such rows as well. If the matrices $\mathbf{A}$ and $\widehat{\mathbf{A}}$ have the same sparsity pattern and the pattern of $\mathbf{N}$ includes at least the diagonal, this is not an issue. Let r_k be the set of indices of rows in $\mathbf{A}$ that are relevant for the kth small LS problem. Then the least squares problem for $\mathbf{n}_k$ is defined as

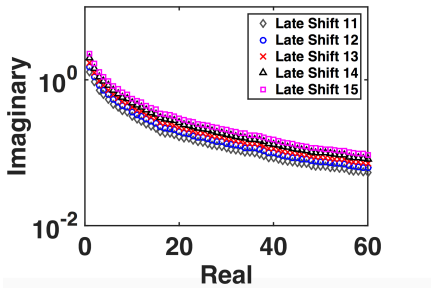
$$\mathbf{n}_k = \arg\min_{\widetilde{\mathbf{n}}_k \in \mathcal{S}_k} \|\mathbf{A}(r_k, s_k)\widetilde{\mathbf{n}}_k(r_k) - \widehat{\mathbf{A}}(r_k, k)\|_2,$$

where $\mathbf{A}(r_k, s_k)$ is the block submatrix of $\mathbf{A}$ indexed by $r_k \times s_k$, $\widetilde{\mathbf{n}}_k(r_k)$ is the subvector of $\widetilde{\mathbf{n}}_k$ indexed by r_k, and $\widehat{\mathbf{A}}(r_k, k)$ is the corresponding subvector of the kth columns of $\widehat{\mathbf{A}}$.

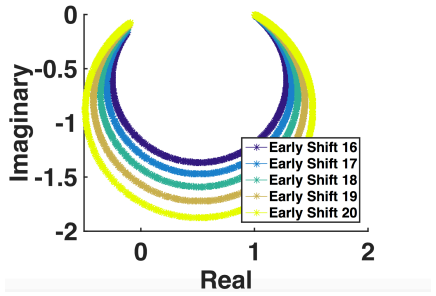
We preprocess the matrices or their sparsity patterns to be able to efficiently select the relevant rows, r_k, and columns, s_k, of the small LS problem for each column k. It may also be efficient to (only once) allocate memory space for solving these least squares problems. This can be a single memory allocation sufficiently large for each of the problems or multiple allocations to allow for parallelism. For matrices that derive from some discretization, the size of these least squares problems typically depends only on the sparsity pattern, not on the size of the matrix. So, while n may be large, each of the least squares problems solved is very small (and most are about the same



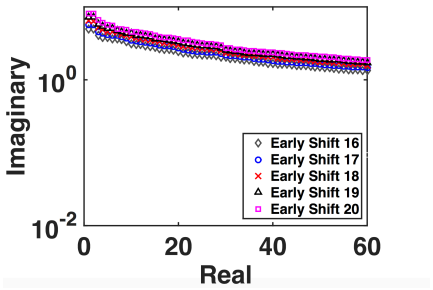
(a) Eigenvalues of the ideal map, $\widehat{\mathbf{N}}_k$, for the late THT matrices for shifts 11 through 15.



(b) The 60 largest singular values of $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 - \mathbf{I}$ for the late THT matrices for shifts 11 through 15.



(c) Eigenvalues of the ideal map, $\widehat{\mathbf{N}}_k$, for the early THT matrices for shifts 16 through 20.



(d) The 60 largest singular values of $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 - \mathbf{I}$ for the early THT matrices for shifts 16 through 20.

Fig. 2.4: Comparison of the singular values of $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0 - \mathbf{I}$ with the eigenvalues of the late and early THT matrices when the eigenvalues are clustered with few outliers and when the eigenvalues are not clustered.

size). For example, the average size of these least squares problems for the THT matrices is 18×7. In general, the matrices or the underlying problems have structure that should be exploited. For example, if a matrix derives from discretization on some mesh or grid, finding the nonzero patterns of powers of the matrix can be done very efficiently using the information defining the mesh or grid.

Finally, it is essential to store the matrices in an appropriate (sparse) format and to generate $\mathbf{N}$ in an appropriate format. For example, in MATLAB®it is inefficient to generate a sparse matrix one column at a time, even if the total space is allocated in advance (presumably because of the required manipulation of sparse matrix data structures for each column). Therefore, we generate $\mathbf{N}$ first in coordinate format (COO) [58, 59], and after the whole matrix has been computed we convert this temporary data structure into a MATLAB®sparse matrix using the command `sparse`. In Algorithm 1, the statement $t = \texttt{find}(\mathbf{a}_j)$ (with reference to the MATLAB®command

`find`), it is important for efficiency that $\mathbf{A}$ is stored as a sparse matrix, and that its columns are easily accessible.

The algorithm for preprocessing is given in Algorithm 1; the algorithm for computing the SAM itself is given in Algorithm 2.

Algorithm 1 Preprocessing for Computing Sparse Approximate Maps

Given sparsity pattern S, and matrix $\mathbf{A}$
$maxSk = 0$; $maxRk = 0$; { initialize max num of columns, max num of rows }
for $k = 1 : n$ **do** { for each column do }
  $s_k = \{i \,|\, (i,k) \in S\}$ { get indices; typically defined in advance }
  $r_k = \emptyset$ { Initialize set of rows for kth LS problem }
  for all $j \in s_k$ **do**
    $t = \texttt{find}(\mathbf{a}_j)$ { find indices of nonzeros in column $\mathbf{a}_j$ }
    $r_k = r_k \cup t$
  end for
  $nnz_k = \#(s_k)$ { #() gives number of elements in a set }
  if $nnz_k > maxSk$ **then**
    $maxSk = nnz_k$
  end if
  if $\#(r_k) > maxRk$ **then**
    $maxRk = \#(r_k)$
  end if
end for
Allocate $maxRk \times maxSk$ array for storing the LS matrices, $maxRk$ vector for storing the right hand side, and $maxSk$ vector for storing the solution.

Algorithm 2 Computing $\mathbf{N} = \arg\min_{\widetilde{\mathbf{N}} \in \mathcal{S}} \|\mathbf{A}\widetilde{\mathbf{N}} - \widehat{\mathbf{A}}\|_F$

$cnt = 0$ { counts number of nonzeros in preconditioner }
for $k = 1 : n$ **do**
  $\mathbf{A}_{\text{tmp}} = \mathbf{A}(r_k, s_k)$ { get submatrix indexed by r_k and s_k for LS problem }
  $\mathbf{f} = \widehat{\mathbf{A}}(r_k)$ { get rhs for LS problem }
  Solve LS $\mathbf{A}_{\text{tmp}}\mathbf{z} = \mathbf{f}$
  (possibly save residual, norm of residual, etc.)
  $rowN[cnt + 1 : cnt + nnz_k] = s_k$ { assign indices in the order of values in $\mathbf{z}$ }
  $colN[cnt + 1 : cnt + nnz_k] = j$
  $valN[cnt + 1 : cnt + nnz_k] = \mathbf{z}$
end for
$\mathbf{N} = \text{sparse}(rowN, colN, valN)$ { convert into sparse matrix }

4. Numerical Experiments. We apply the strategies of reusing and recycling preconditioners to several applications, focusing both on total computation time as well as total GMRES iterations. We define total computation time to be the time to compute the preconditioner or SAM update plus the time for GMRES to converge for all shifts. We also report the times for the computation of individual preconditioners and SAMs as well as the number of iterations and runtime per system solve. These numbers provide good insight into the merits of reusing a preconditioner, possibly

including a previous SAM update, computing a new preconditioner, or computing a new SAM update. Of course, the actual costs of these computations, the number of iterations saved, and the cost per iteration are all problem dependent. We compare the results of computing a new ILUTP preconditioner for each shift, reusing the initial $\mathbf{P}_0$ for all shifts, updating $\mathbf{P}_0$ with a new SAM update for all shifts, and updating $\mathbf{P}_0$ with a SAM update only at selected shifts. The first approach, a new ILUTP preconditioner for every shift, is always the most expensive option in runtime, but it provides a useful benchmark in terms of the number of iterations. The last approach, to compute a SAM update only at selected shifts, is usually the winner in runtime. The exceptions are the linear systems for the late THT matrices and the matrices from the model reduction test problem Rail. For these problems reusing the initial ILUTP for all systems leads to the lowest runtime. For brevity, we do not provide data for recomputing the ILUTP at selected shifts. This, of course, can be better than computing the ILUTP for all shifts, but it was never the fastest - this can easily be derived directly from the high cost of computing the preconditioner.

Finally, for longer sequences of systems and other problems, many other variations of computing preconditioners and updates may be effective. Although we experimented with several *indicator functions* to decide when to do a SAM update and the results were encouraging, we did not find a single good indicator. Hence we leave a further analysis and discussion of these for future work. A simple and effective strategy is to compute a new SAM or preconditioner based on (1) the time for this computation and (2) the (relative) increase in the number of iterations or the solution time for a single system.

For the THT matrices, we also compare with the AINV preconditioner and update. The algorithms for calculating both the AINV preconditioner as well as the AINV updates can be found in [15, 20, 21, 22, 23, 24, 56]. The implementation of the ILUTP preconditioner is computed using an implementation based on that in [58].

4.1. Transient Hydraulic Tomography[5]. Transient Hydraulic Tomography (THT) is a method for imaging the earth's subsurface (see [28] for a detailed description of THT). Water is pumped at a constant rate in pumping wells and the measured drawdown curves of pressure response at the observation wells is recorded. A subset of this data is used in a nonlinear inversion to recover the parameters of interest, namely hydraulic conductivity and specific storage. In the example described later in this section, we choose three key time points (corresponding to *early*, *middle*, and *late* times). The governing equations of groundwater flow through an aquifer with domain Ω are given by,

$$\begin{aligned} S_s(x)\frac{\partial \phi(x,t)}{\partial t} - \nabla \cdot (\kappa(x)\nabla\phi(x,t)) = &\quad q(t)\delta(x-x_s), && x \in \Omega \\ \phi(x,t) = &\quad 0, && x \in \partial\Omega_D \\ \nabla\phi(x,t)\cdot n = &\quad 0, && x \in \partial\Omega_N \end{aligned} \tag{4.1}$$

where x_s denotes the location of the pumping well, $q(t)$ is the pumping rate, $\kappa(x)$ is the hydraulic conductivity, S_s is the specific storage, $q(t)\delta(x-x_s)$ is the pumping source, and $\phi(x,t)$ is the hydraulic head (pressure). Ω_D and Ω_N denote the parts of the boundary where the Dirichlet and Neumann boundary conditions are defined,

[5]We would like to thank to Tania Bakhos, Arvind Saibaba, and Peter Kitanidis for providing the description of THT as well as the matrices used.

respectively. The differential equation (4.1) and its corresponding boundary conditions are discretized by standard linear finite elements using FEnICS [51, 52, 53]. We obtain the semi-discrete system of equations,

$$\mathbf{M}\partial_t\phi_h + \mathbf{K}\phi_h = q(t)\mathbf{b} \tag{4.2}$$

where $\mathbf{K}$ and $\mathbf{M}$ denote the stiffness and mass matrices respectively. Instead of using a traditional time-stepping scheme to solve these equations, the Laplace transform-based exponential time integrator is used, as described in [9]. The main idea is that a contour integral representation of the inverse Laplace transform chosen on the modified Talbot contour can be used to efficiently solve the groundwater equations [63]. The solution at a given time t is given by,

$$\phi_h(t) \approx \sum_{k=1}^{N_z} w_k(\mathbf{K} + z_k\mathbf{M})^{-1}\left(\mathbf{M}\phi_0 + \hat{q}(z_k)\mathbf{b}\right) \tag{4.3}$$

with w_k and z_k being the weights and nodes of the quadrature scheme, respectively. Then (4.3) amounts to solving a shifted system of equations for each time point,

$$(\mathbf{K} + z_k\mathbf{M})\, X_k = [\mathbf{b}, \quad \mathbf{M}\phi_0], \qquad k = 1, \ldots, N_z/2 \tag{4.4}$$

In the experiments presented later in this section, we solve for $\mathbf{b}$.

We consider a 2D depth-averaged aquifer with zero Dirichlet boundary conditions on all boundaries. The domain size is square of size 100m $\times$ 100m. For the log conductivity field, we use a randomly generated field from the exponential covariance kernel,

$$\kappa(x, y) = 4\exp(-2\|x - y\|_2/100) \tag{4.5}$$

The mean conductivity was chosen to be $\mu_K = 10^{-3.5}$ [m^2/s] and the variance was chosen to be $\sigma_K^2 = 1.6$. The specific storage in this example was chosen to be constant with $S_s = 10^{-5}$. There is one pumping source located at $(50, 50)$, pumping at a constant rate of 0.85 L/s. We are interested in the solution at three time instances which we will refer to as *early* (1 min), *middle* (15 min) and *late* (40 min) - the shifts z_k are shown in Figure 4.1. Note that for the middle and late matrices, the shifts are clustered together more tightly. The matrices have system size 10201 $\times$ 10201.

We provide results for the early and middle THT matrices.[6] Tables 4.1 and 4.7 show that computing a new ILUTP preconditioner for every system results in the lowest number of GMRES iterations, but the longest overall time. Tables 4.2, 4.3, 4.8, and 4.9 show that when we apply the SAM update at each shift, we achieve fewer total GMRES iterations as compared with just reusing $\mathbf{P}_0$ for all shifts.

While the computation of a SAM update is a factor ten cheaper than that of an ILUTP, computing an update at every shift is still too expensive. Therefore, it makes sense to do an update at selected shifts. A simple choice is to do a single SAM update at shift 10 (halfway), and reuse that update for all subsequent systems. For comparison, we also try single SAM updates for shifts 5 or 15. We present the results for the early and middle THT matrices in Tables 4.4-4.6 and 4.10-4.12. We experimented with indicators for updating, but although results were good we did not

[6] For the late THT matrices, the magnitude of each shift is small enough that reusing $\mathbf{P}_0$ leads to the lowest total runtime.

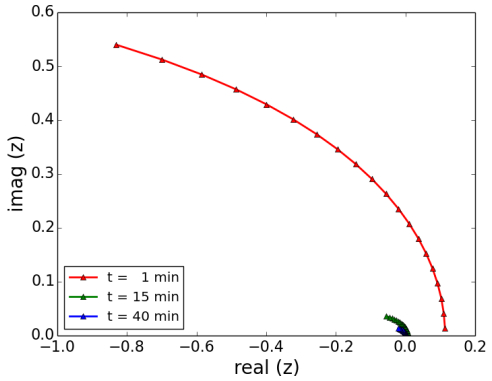


Fig. 4.1: Plot of contours corresponding to $N_z = 40$. Because of symmetry, only half the contour plot is shown. Here N_z is the number of quadrature points in Equation (4.3). $t = 1$ min corresponds to the shifts for the early matrices, $t = 15$ min to those for the middle matrices, and $t = 40$ min to those for the late matrices.

find a single indicator that was best in all cases. However, our results demonstrate the potential of a SAM update at selected shifts.

Note that when applying the SAM update at each shift, the number of GMRES iterations increases as the norm of the residual of the LS map increases. GMRES iterations are also fewer when the eigenvalues of the ideal map are clustered, potentially with some outliers (see Figures 2.2(a) and 2.2(b)), and when the eigenvalues of the preconditioned matrix are clustered near one. For the early THT matrices, Figures 4.2 and 4.3 show that the eigenvalues of $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0$ are more clustered than those of $\mathbf{A}_k\mathbf{P}_0$, especially for the later shifts.

Column pivoting in computing the ILUTP results in a matrix $\mathbf{Q}$ such that $\mathbf{A}_0\mathbf{Q} \simeq \mathbf{LU}$ and $\mathbf{A}_0^{-1} \simeq \mathbf{QU}^{-1}\mathbf{L}^{-1} = \mathbf{P}_0$.[7] When applying the SAM update to the ILUTP preconditioner, we get

$$\mathbf{N}_k\mathbf{P}_0 = (\mathbf{N}_k\mathbf{Q})\mathbf{U}^{-1}\mathbf{L}^{-1} = \widetilde{\mathbf{N}}_k\mathbf{U}^{-1}\mathbf{L}^{-1},$$

with $\widetilde{\mathbf{N}}_k = \mathbf{N}_k\mathbf{Q}$. So, an additional advantage (in this case) of the SAM is that it absorbs this permutation, which saves a bit of time in the preconditioned matrix-vector product. When reusing the initial ILUTP, however, the $\mathbf{U}$ factor must be permuted. In MATLAB®this leads to a slightly higher runtime for the back-substitution with $\mathbf{U}$. This explains the slightly increased time for GMRES iterations when reusing the ILUTP preconditioner compared with using a new ILUTP preconditioner for a similar number of iterations.

We also apply the AINV updates to the THT matrices. This update scheme specifies that we compute an AINV preconditioner for $\mathbf{P}_0 = \mathbf{ZD}^{-1}\mathbf{W}^T \approx \mathbf{A}^{-1}$, with $\mathbf{Z}$, $\mathbf{W}$ unit upper triangular and $\mathbf{D}$ diagonal. For this application, it turns out that this type of preconditioner is expensive to compute and substantially less effective than ILUTP. While computation of the updates is inexpensive (and the updated preconditioner preserves GMRES iterations for the first few shifts), this preconditioner

[7] In practice, we never invert these matrices but rather use forward/backward solves for $\mathbf{L}$ and $\mathbf{U}$. Applying $\mathbf{Q}$ (or $\mathbf{Q}^{-1}$) amounts to reordering the components of a vector.

type and update are not effective for this class of problems. Results are provided in Tables 4.13-4.15.

Shift	Prec Time	GMRES Time	Iter
1	8.89	0.076	22
2	8.88	0.081	22
3	8.89	0.078	22
4	8.88	0.082	23
5	8.90	0.086	23
10	8.98	0.095	24
15	9.10	0.098	27
20	9.29	0.11	28

Table 4.1: Timings for selected shifts for THT early matrices with ILUTP computed at each shift (total time 182.23 s, total iterations 498).

Shift	Prec Time	GMRES Time	Iter
1	9.01	0.14	22
2	0	0.19	23
3	0	0.18	25
4	0	0.19	27
5	0	0.23	30
10	0	0.47	54
15	0	1.11	129
20	0	2.93	325

Table 4.2: Timings for selected shifts for THT early matrices with initial ILUTP reused (total time 25.69 s, total iterations 1975).

Shift	Prec Time	GMRES Time	Iter
1	8.84	0.068	22
2	0.95	0.087	23
3	0.88	0.085	24
4	0.89	0.089	25
5	0.77	0.10	27
10	0.78	0.17	41
15	0.87	0.45	87
20	0.85	0.90	190

Table 4.3: Timings for selected shifts for THT early matrices with ILUTP computed at first shift and SAM updates computed for remaining shifts (total time 30.38 s, total iterations 1312).

Shift	Prec Time	GMRES Time	Iter
Shift 1	8.88	0.072	22
2	0	0.14	23
3	0	0.15	25
4	0	0.17	27
5	0.93	0.099	27
10	0	0.22	47
15	0	0.53	115
20	0	1.43	301

Table 4.4: Timings for selected shifts for THT early matrices with ILUTP computed at first shift and SAM update only computed at shift 5 and reused for subsequent shifts (total time 18.085 s, total iterations 1763).

Shift	Prec Time	GMRES Time	Iter
Shift 1	8.94	0.073	22
2	0	0.14	23
3	0	0.15	25
4	0	0.17	27
5	0	0.19	30
10	0.94	0.18	41
15	0	0.48	100
20	0	1.54	282

Table 4.5: Timings for selected shifts for THT early matrices with ILUTP computed at first shift and SAM update only computed at shift 10 and reused for subsequent shifts (total time 18.68 s, total iterations 1645).

Shift	Prec Time	GMRES Time	Iter
Shift 1	9.015	0.075	22
2	0	0.14	23
3	0	0.15	25
4	0	0.17	27
5	0	0.19	30
10	0	0.40	54
15	0.98	0.40	87
20	0	1.055	226

Table 4.6: Timings for selected shifts for THT early matrices with ILUTP computed at first shift and SAM update only computed at shift 15 and reused for subsequent shifts (total time 18.80 s, total iterations 1564).

4.2. Interpolatory Model Reduction. Another set of linear systems arise in interpolatory model reduction, in particular, in the Iterative Rational Krylov Algorithm (IRKA) [45]. Consider the single-input/single-output linear dynamical system

$$\mathbf{E}\dot{\mathbf{x}}(t) + \mathbf{A}\mathbf{x}(t) = \mathbf{b}u(t), \quad y(t) = \mathbf{c}^T\mathbf{x}(t), \tag{4.6}$$

where $\mathbf{E}, \mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$, and $u(t), y(t) \in \mathbb{R}$. In (4.6), $\mathbf{x}(t) \in \mathbb{R}^n$, $u(t)$, and $y(t)$ are called, respectively, the state, the input, and the output of the underlying dynamical system. By taking the Laplace transform of (4.6), one obtains the transfer function

$$H(s) = \mathbf{c}^T(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}. \tag{4.7}$$

Shift	Prec Time	GMRES Time	Iter
1	9.057	0.085	24
2	9.032	0.082	24
3	8.99	0.08	24
4	8.99	0.080	24
5	8.94	0.080	24
10	9.049	0.091	26
15	9.008	0.096	27
20	9.098	0.10	28

Table 4.7: Timings for selected shifts for THT middle matrices with ILUTP computed at each shift (total time 182.023 s, total iterations 514).

Shift	Prec Time	GMRES Time	Iter
1	8.95	0.083	24
2	0	0.20	24
3	0	0.21	25
4	0	0.20	25
5	0	0.22	26
10	0	0.31	35
15	0	0.44	47
20	0	0.99	111

Table 4.8: Timings for selected shifts for THT middle matrices with ILUTP reused (total time 16.80 s, total iterations 880).

Shift	Prec Time	GMRES Time	Iter
1	9.00	0.081	24
2	1.026	0.086	24
3	0.86	0.091	24
4	0.83	0.091	25
5	0.83	0.095	26
10	0.86	0.13	32
15	0.84	0.19	42
20	0.81	0.39	79

Table 4.9: Timings for selected shifts for THT middle matrices with ILUTP computed at first shift and SAM update computed for remaining shifts (total time 28.50 s, total iterations 736).

Shift	Prec Time	GMRES Time	Iter
1	8.97	0.081	24
2	0	0.21	24
3	0	0.22	25
4	0	0.22	25
5	1.069	0.11	26
10	0	0.19	34
15	0	0.28	45
20	0	0.59	102

Table 4.10: Timings for selected shifts for THT middle matrices with ILUTP computed at first shift and SAM update only applied at shift 5 (total time 15.084 s, total iterations 846).

Shift	Prec Time	GMRES Time	Iter
1	8.95	0.078	24
2	0	0.21	24
3	0	0.22	25
4	0	0.22	25
5	0	0.23	26
10	1.06	0.13	32
15	0	0.24	44
20	0	0.62	98

Table 4.11: Timings for selected shifts for THT middle matrices with ILUTP computed at first shift and SAM update only applied at shift 10 (total time 15.24 s, total iterations 814).

Shift	Prec Time	GMRES Time	Iter
1	9.00	0.082	24
2	0	0.21	24
3	0	0.22	25
4	0	0.21	25
5	0	0.23	26
10	0	0.25	35
15	0.98	0.20	42
20	0	0.46	91

Table 4.12: Timings for selected shifts for THT middle matrices with ILUTP computed at first shift and SAM update only applied at shift 15 (total time 15.14 s, total iterations 797).

Dynamical systems with large state-space dimension n appear in many applications, ranging from nonlinear parameter inversion to optimal control to circuit design. Simulations in these large-scale settings could be overwhelming. The goal of model reduction is, then, to replace the original large-scale dynamical system with one having a much smaller state-space dimension without losing much accuracy in the system response. In other words, the goal is to construct the dynamical system

$$\mathbf{E}_r\dot{\mathbf{x}}_r(t) + \mathbf{A}_r\mathbf{x}_r(t) = \mathbf{b}_r u(t), \quad y_r(t) = \mathbf{c}_r^T\mathbf{x}_r(t), \tag{4.8}$$

where $\mathbf{E}_r, \mathbf{A}_r \in \mathbb{R}^{r\times r}$, $\mathbf{b}_r, \mathbf{c}_r \in \mathbb{R}^r$, and $r \ll n$, such that $y_r(t) \approx y(t)$ in an appropriate norm for a wide range of input selections $u(t)$. The reduced model quantities in (4.8) are obtained by constructing two matrices $\mathbf{V}_r, \mathbf{W}_r \in \mathbb{R}^{n\times r}$ (the model reduction bases) and performing a Petrov-Galerkin projection

$$\mathbf{A}_r = \mathbf{W}_r^T\mathbf{A}\mathbf{V}_r, \quad \mathbf{E}_r = \mathbf{W}_r^T\mathbf{E}\mathbf{V}_r, \quad \mathbf{b}_r = \mathbf{W}_r^T\mathbf{b}, \quad \mathbf{c}_r = \mathbf{V}_r^T\mathbf{c}. \tag{4.9}$$

Shift	Prec Time	GMRES Time	Iter
1	156.18	1.19	241
2	0.22	1.21	246
3	0.23	1.29	262
4	0.23	1.45	285
5	0.23	1.73	346
10	0.23	6.13	1239
15	0.23	20.84	4150
20	0.23	50.24	10202

Table 4.13: Timings for selected shifts for THT early matrices with (full) robust AINV computed once with AINV updates applied to remaining shifts (total time 449.88 s, total iterations 57951).

Shift	Prec Time	GMRES Time	Iter
1	161.25	1.31	265
2	0.22	1.34	277
3	0.22	1.38	282
4	0.22	1.47	298
5	0.22	1.49	298
10	0.22	2.54	513
15	0.22	5.67	1147
20	0.22	25.38	5101

Table 4.14: Timings for selected shifts for THT middle matrices with (full) robust AINV computed once with AINV updates applied to remaining shifts (total time 278.31 s, total iterations 22879).

Shift	Prec Time	GMRES Time	Iter
1	149.55	1.24	247
2	0.22	1.26	250
3	0.22	1.26	251
4	0.22	1.29	261
5	0.22	1.36	281
10	0.22	1.75	356
15	0.22	2.73	560
20	0.23	7.24	1467

Table 4.15: Timings for selected shifts for THT late matrices with (full) robust AINV computed once with AINV updates applied to remaining shifts (total time 204.085 s, total iterations 10251).

The difference between model reduction techniques results from the choices for $\mathbf{V}_r$ and $\mathbf{W}_r$. We refer the reader to [6, 11, 17] for an overview of various model reduction methods. In this paper, we focus on interpolatory model reduction, for which solving linear systems plays a fundamental role.

Similar to (4.7), the transfer function of the reduced model (4.8) is given by

$$H_r(s) = \mathbf{c}_r^T(s\mathbf{E}_r - \mathbf{A}_r)^{-1}\mathbf{b}_r. \tag{4.10}$$

While $H(s)$ is a degree-n rational function, $H_r(s)$ is a degree-r rational function. Interpolatory model reduction, then, aims to construct the reduced model matrices and thus $H_r(s)$, via projection as in (4.9), so that $H_r(s)$ interpolates $H(s)$ at selected points. Assume that a set of interpolation points $\{s_1, s_2, \ldots, s_r\}$ is given. We want to construct $H_r(s)$ so that it is a Hermite interpolant to $H(s)$ at these points, i.e.,

$$H_r(s_j) = H(s_j) \quad \text{and} \quad H_r'(s_j) = H'(s_j), \text{ for } j = 1, 2, \ldots, r.$$

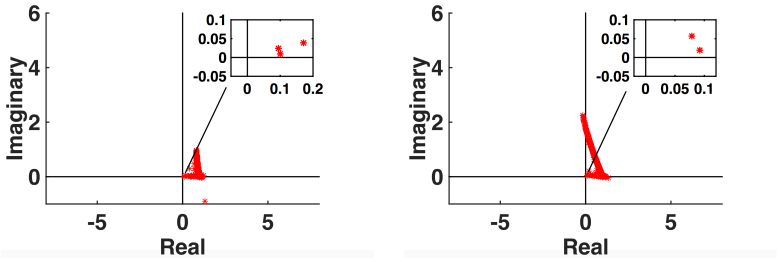
To achieve this, we build the model reduction matrices

$$\mathbf{V}_r = [(s_1\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}, \ldots, (s_r\mathbf{E} - \mathbf{A})^{-1}\mathbf{b}], \tag{4.11}$$

$$\mathbf{W}_r = [(s_1\mathbf{E}^T - \mathbf{A}^T)^{-1}\mathbf{c}, \ldots, (s_r\mathbf{E}^T - \mathbf{A}^T)^{-1}\mathbf{c}], \tag{4.12}$$

and obtain the reduced model using (4.9). For more details on interpolatory model reduction, see [7, 13].

The construction of $\mathbf{V}_r$ and $\mathbf{W}_r$ reveals that computing $H_r(s)$, a rational Hermite interpolant to $H(s)$, requires solving $2r$ linear systems. However, the number of shifted



(a) Shift 5 (b) Shift 10

(c) Shift 15 (d) Shift 20 (eigenvalue at $(5.72, -2.478)$ omitted).

Fig. 4.2: Eigenvalues of the preconditioned early THT matrices, $\mathbf{A}_k\mathbf{P}_0$, for selected shifts with the ILUTP preconditioner for $\mathbf{A}_0$ reused for all shifts.

linear systems to be solved increases rapidly if the goal is to construct an *optimal* interpolant. The construction above is valid for an arbitrary set of interpolation points, as long as they do not coincide with the eigenvalues of the matrix pencil $\lambda\mathbf{E} - \mathbf{A}$. The reduced model, however, does not inherit any optimality and could be a poor approximation. Interpolatory optimal $\mathcal{H}_2$ model reduction resolves this issue.

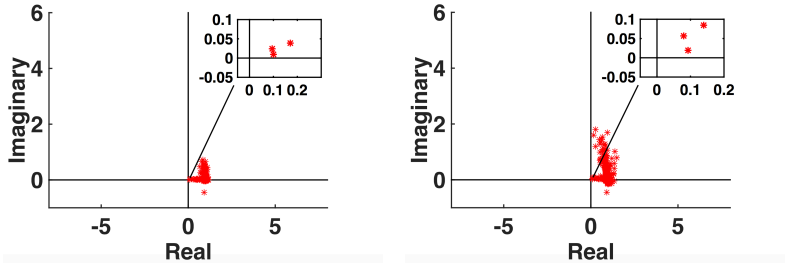
In optimal $\mathcal{H}_2$ approximation, one seeks a reduced model with transfer function $H_r(s)$ that minimizes the $\mathcal{H}_2$ error measure $\|H(s) - H_r(s)\|_{\mathcal{H}_2}$, where

$$\|H(s) - H_r(s)\|_{\mathcal{H}_2} = \left(\frac{1}{2\pi}\int_{-\infty}^{\infty} \mid H_r(\imath\omega) - H(\imath\omega) \mid^2 d\omega\right)^{1/2}.$$

The $\mathcal{H}_2$ error norm has a direct implication for the time domain error in the system response $y(t) - y_r(t)$; namely,

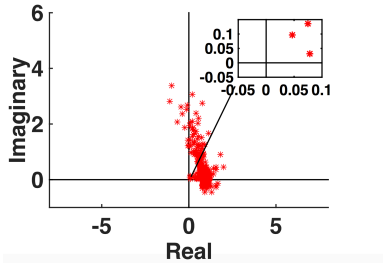
$$\|y(t) - y_r(t)\|_{L_\infty} \leq \|H(s) - H_r(s)\|_{\mathcal{H}_2}\|u(t)\|_{L_2}.$$

In other words, in order to minimize the L_∞ distance between $y(t)$ and $y_r(t)$ for every bounded input in the L_2 norm, one has to minimize the $\mathcal{H}_2$ distance between $H(s)$ and $H_r(s)$. The connection to Hermite interpolation is immediately revealed in the first-order optimality conditions: If $H_r(s)$ is the best degree-r rational approximation to $H(s)$, then $H_r(s)$ is a Hermite interpolant to $H(s)$ at the mirror images of the poles

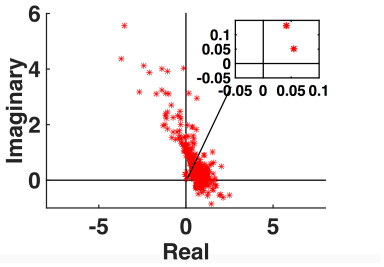


(a) Shift 5

(b) Shift 10



(c) Shift 15



(d) Shift 20

Fig. 4.3: Eigenvalues of the preconditioned early THT matrices, $\mathbf{A}_k\mathbf{N}_k\mathbf{P}_0$, for selected shifts with SAM updates applied to shifts 2 through 20.

of $H_r(s)$, i.e., at the mirror images of the eigenvalues of the reduced matrix-pencil $\lambda\mathbf{E}_r - \mathbf{A}_r$; see, e.g., [54, 45]. In other words, the optimal interpolation points depend on the reduced model to be constructed, and therefore an iterative algorithm is required. The Iterative Rational Krylov Algorithm (IRKA) [45] precisely achieves this task. IRKA in its original formulation is a fixed point iteration. The interpolation points are initialized by the user, but within the algorithm the points $\{s_j\}$ are updated to be the mirror images of the generalized eigenvalues of the intermediate reduced model quantities $\mathbf{A}_r$ and $\mathbf{E}_r$ [45]. Until convergence of the fixed points, s_j, is achieved, the algorithm iteratively updates the reduced order model by building the new matrices $\mathbf{V}_r$ and $\mathbf{W}_r$ given in (4.11) and (4.12) for the current interpolation points. The matrices $\mathbf{V}_r$ and $\mathbf{W}_r$ give the new $\mathbf{A}_r$, $\mathbf{E}_r$, $\mathbf{b}_r$, and $\mathbf{c}_r$ as in (4.9). Since it may take many iterations until IRKA converges to the final set of $\{s_j\}$, many shifted systems must be solved in computing (4.11) and (4.12). We refer to the set of shifts for an IRKA iteration as a *batch*. For more detail on interpolatory model reduction and IRKA see [7, 12, 13, 44, 45].

Other approaches have been used to reduce the cost of the linear solves in (4.11) and (4.12). In [14], inexact solves within a Petrov-Galerkin framework are used to reduce this cost. In [5], the recycling BiCG algorithm is proposed and applied effectively to a parametric model order reduction example, while recycling BiCGSTAB is used (also for parametric model reduction) in [3]. Further discussion of recycling

Krylov subspace methods specifically applied to model reduction applications can be found in [38, 39].

We give results for two sets of matrices, Rail and Flow, which can be found in [16] along with additional information. The Rail matrices come from a semi-discretized heat transfer equation for the cooling of steel beams. The matrices $\mathbf{A}$ and $\mathbf{E}$ are very sparse with $\mathbf{A}$ symmetric negative definite, $\mathbf{E}$ positive definite, and $n \approx 80\ 000$. We give results for three batches of six shifts, which are real and range from $O(10^{-5})$ to $O(10)$. The shifts are provided in Table 4.16.

For the Rail matrices, the number of nonzeros in $\mathbf{A}$ is 553921, that in $\mathbf{E}$ is 554913, making the matrices very sparse and therefore the matrix-vector product very cheap. In addition, the ILUTP preconditioner for these matrices is remarkably good. As a result, the best choice is to reuse the initial preconditioner for all shifted matrices, which achieves the fastest overall computation time. However, we do see that applying the SAM updates at each shift substantially reduces the total number of GMRES iterations as shown in Tables 4.18 and 4.19. We omit the results for computing the preconditioner at each shift. While the total GMRES iterations were the fewest when computing the ILUTP for each shift, the overall computation time was very high (over two hours).

The Flow matrices arise in a simulation of the heat exchange between a solid body and a fluid flow. Rather than using computational fluid dynamics, which is quite expensive, an alternative approach is to include a flow region with a given velocity profile [55]. However, this requires that the number of elements is drastically increased. To deal with this, model reduction is used to effectively describe the system [55]. For further description of the problem as well as how model reduction is applied, we refer to [55, 57]. The model reduction involves the sparse matrices $\mathbf{A}$ and $\mathbf{E}$, where $n \approx 10\ 000$. We use three batches of six shifts, which are real and range from $O(1)$ to $O(10^4)$. The shifts for the Flow matrices are provided in Table 4.17.

Although $\mathbf{A}$ is not symmetric, and therefore the generalized eigenvalues of $\mathbf{A}$ and $\mathbf{E}$ may be complex, it turns out that for the three steps of IRKA that we used for our experiments, the shifts remain real. Table 4.20 shows that computing a new ILUTP preconditioner at each shift results in the the lowest total number of iterations but the highest overall computation time. When we apply the SAM updates at each shift, the number of iterations increases, but this still yields a lower total computation time compared with reusing the initial ILUTP for all shifts, as shown in Tables 4.21 and 4.22. In fact, at the first shift of batch three, the initial ILUTP is not a good preconditioner and GMRES fails to converge. However, computing a SAM update at this shift produces a preconditioner for which GMRES does converge.[8] Here we see that reusing the ILUTP cannot be done for all shifts. Again note that when reusing the ILUTP, the $\mathbf{U}$ factor must absorb any permutation that results from the initial factorization of $\mathbf{A}_0$. However, when using the SAM updates, the map absorbs the permutation and $\mathbf{U}$ remains upper triangular. This is why GMRES takes longer for a similar number of iterations.

As with the THT matrices, we also apply the SAM update at selected shifts of the Flow matrices. We compute an ILUTP preconditioner for the first shift of the first batch. We apply a SAM update at the sixth and largest shift of each batch and again at the start of each new batch. It is at these shifts that we see some of the largest numbers of GMRES iterations when reusing the ILUTP for all shifts. As seen

[8] The maximum number of iterations is set to 5000; (5001) in Table 4.22 indicates GMRES does not converge.

	Batch 1	Batch 2	Batch 3
Shift 1	1e-05	1.8347e-05	1.8447e-05
2	0.00013804	0.00032778	0.0003295
3	0.0019055	0.0046185	0.0046991
4	0.026303	0.056949	0.067322
5	0.36308	0.52928	0.72306
6	5.0119	8.4359	10.9255

Table 4.16: Shifts for the Rail Matrices.

	Batch 1	Batch 2	Batch 3
Shift 1	1.4091	1.4106	1.411
2	28.1234	29.3905	29.7024
3	150.6975	158.8259	160.8
4	669.2639	683.6133	687.1313
5	3536.6535	3555.1646	3559.7271
6	17329.4291	17344.2626	17347.9311

Table 4.17: Shifts for the Flow Matrices.

B/S	Prec Time	GMRES Time	Iter
1/1	437.30	1.14	52
1/2	0	0.49	30
1/3	0	0.28	15
1/4	0	0.19	8
1/5	0	0.36	20
1/6	0	1.59	77
2/1	0	0.89	48
2/2	0	0.42	25
2/3	0	0.24	12
2/4	0	0.20	9
2/5	0	0.41	24
2/6	0	2.39	97
3/1	0	0.87	48
3/2	0	0.42	25
3/3	0	0.24	12
3/4	0	0.20	9
3/5	0	0.48	27
3/6	0	2.67	109

Table 4.18: Timings for Rail matrices with ILUTP for the first system reused for all shifts (total time 450.77 s, total iterations 647), B/S = Batch Number/Shift Number.

B/S	Prec Time	GMRES Time	Iter
1/1	438.38	1.075	52
1/2	29.96	0.59	30
1/3	10.36	0.32	15
1/4	10.13	0.22	8
1/5	10.40	0.35	17
1/6	10.13	0.51	26
2/1	11.74	0.99	48
2/2	10.36	0.49	25
2/3	10.43	0.27	12
2/4	10.47	0.23	9
2/5	10.19	0.39	20
2/6	10.48	0.74	38
3/1	11.51	0.96	48
3/2	10.24	0.48	25
3/3	10.29	0.27	12
3/4	10.25	0.23	9
3/5	10.29	0.43	22
3/6	10.12	2.32	93

Table 4.19: Timings for Rail matrices with ILUTP computed once and SAM updates applied for all other shifts (total time 646.60 s, total iterations 509). B/S = Batch Number/Shift Number.

in Table 4.23, we achieve the lowest overall computation time when we apply the SAM updates at these selected shifts.

4.3. Indefinite Matrices. In our previous applications, computing a new ILUTP for each system is better in terms of keeping GMRES iterations low, but it is, in general, too expensive in terms of time. Here, we consider linear systems of equations where the computation of the ILUTP preconditioner may fail or may be unstable (resulting in poor preconditioners). This is the case, for example, for indefinite systems [59, Chapter 10]. Indefinite matrices may arise when discretizing the 2D Helmholtz equation for a wave problem in an inhomogeneous medium [36, 37]. In such cases, we can select a matrix from the set, or choose an additional matrix, for which the ILUTP algorithm computes an effective preconditioner. Then we use SAM updates to (approximately) map matrices for which ILUTP may fail to this matrix.

This same idea has also been applied to the Helmholtz equation using other preconditioning approaches. Previous work has successfully used operator-based preconditioners in order to achieve fast convergence of Krylov methods. The shifted Laplace preconditioner (SLP) is used along with multilevel Krylov methods in [36, 37, 61], while a sweeping preconditioner is constructed layer-by-layer in [35]. Preconditioning by replacing a subset of the Sommerfield-like boundary conditions of the discretized Helmholtz equation with either Dirichlet or Neumann boundary conditions is examined in [33, 34].

	Prec Time	GMRES Time	Iter
1/1	4.47	4.90	3591
1/2	4.23	0.0254	13
1/3	4.087	0.029	11
1/4	4.18	0.023	10
1/5	3.95	0.020	7
1/6	3.61	0.021	7
2/1	4.20	0.80	579
2/2	4.23	0.026	13
2/3	4.33	0.026	11
2/4	4.22	0.025	10
2/5	4.059	0.020	7
2/6	3.74	0.020	7
3/1	4.34	0.81	584
3/2	4.39	0.026	13
3/3	4.35	0.026	11
3/4	4.25	0.022	10
3/5	4.011	0.021	7
3/6	3.74	0.021	7

Table 4.20: Timings for Flow matrices with ILUTP recomputed for each shift (total time 81.25 s, total iterations 4898). B/S = Batch Number/Shift Number.

	Prec Time	GMRES Time	Iter
1/1	4.17	4.88	3591
1/2	0.44	0.028	18
1/3	0.35	0.034	24
1/4	0.34	0.14	78
1/5	0.35	0.19	98
1/6	0.34	0.61	207
2/1	0.35	0.94	596
2/2	0.36	0.028	18
2/3	0.38	0.037	24
2/4	0.36	0.32	139
2/5	0.36	0.19	98
2/6	0.37	0.62	205
3/1	0.34	1.89	1204
3/2	0.36	0.027	18
3/3	0.35	0.034	24
3/4	0.37	0.26	123
3/5	0.35	0.090	59
3/6	0.36	0.63	207

Table 4.21: Timings for Flow matrices with ILUTP computed once and SAM updates applied for all other shifts (total time 21.23 s, total iterations 6731). B/S = Batch Number/Shift Number.

	Prec Time	GMRES Time	Iter
1/1	4.39	4.98	3591
1/2	0	0.061	18
1/3	0	0.13	41
1/4	0	0.97	205
1/5	0	1.00	211
1/6	0	1.066	229
2/1	0	7.47	1623
2/2	0	0.056	18
2/3	0	0.13	41
2/4	0	0.97	202
2/5	0	0.99	211
2/6	0	1.14	239
3/1	0	22.63	(5001)
3/2	0	0.056	18
3/3	0	0.13	41
3/4	0	0.96	207
3/5	0	0.98	211
3/6	0	1.029	232

Table 4.22: Timings for Flow matrices with ILUTP for the first system reused for all shifts (total time 49.12 s, total iterations 12339). B/S = Batch Number/Shift Number.

	Prec Time	GMRES Time	Iter
1/1	4.10	4.85	3591
1/2	0	0.059	18
1/3	0	0.13	41
1/4	0	0.96	205
1/5	0	0.97	211
1/6	0.42	0.62	207
2/1	0.35	0.92	596
2/2	0	0.028	19
2/3	0	0.30	134
2/4	0	0.62	207
2/5	0	0.62	211
2/6	0.34	0.63	205
3/1	0.36	1.82	1204
3/2	0	0.027	19
3/3	0	0.058	42
3/4	0	0.62	206
3/5	0	0.64	212
3/6	0.34	0.62	207

Table 4.23: Timings for Flow matrices with ILUTP computed once and SAM updates applied at selected shifts and reused until next SAM update computed (total time 20.40 s, total iterations 7535). B/S = Batch Number/Shift Number.

Our purpose for this section is to demonstrate another possible use of SAM updates; we do not mean to suggest it is a more effective approach than the ones just mentioned.

Let $\mathbf{K}_0$ and right hand side $\mathbf{b}$ be generated using a vertex-based finite volume discretization for the 2D Laplacian on the unit square, with the Dirichlet boundary conditions

$$u(x,0) = 1, \quad u(0,y) = 1, \quad u(x,1) = 0, \quad u(1,y) = 0.$$

$\mathbf{K}_0$ is symmetric, positive definite and of size 100×100. We compute an ILUTP

preconditioner, $\mathbf{P}_0$, for $\mathbf{K}_0$, and use GMRES to solve the preconditioned system. We set the maximum number of GMRES iterations to 100, the relative convergence tolerance to 10^{-10}, and we take the zero vector as the initial guess. Next, we consider the systems

$$\mathbf{K}_i = \mathbf{K}_0 - s_i\mathbf{I}, \tag{4.13}$$

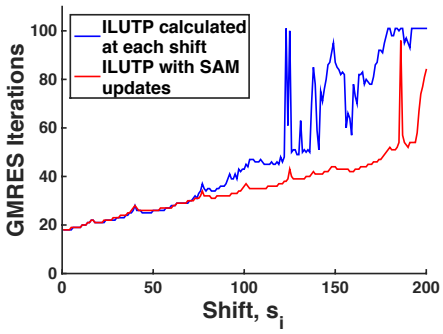
with $\mathbf{I}$ the identity matrix. We take $s_i = i\Delta s$ with $\Delta s = 0.01$, for $i = 1, 2, \ldots, 200$. We solve these systems with preconditioned GMRES as above, computing either a new ILUTP preconditioner at every shift or updating $\mathbf{P}_0$ with a SAM for each system.

For these small problems, we are not concerned with runtime; we only demonstrate the superior convergence behavior obtained with the updated preconditioners compared with ILUTP. After some number of shifts, the ILUTP preconditioners deteriorate due to instability and the number of iterations to convergence rapidly increases. To show that the problem is not due to choices in our implementation of the ILUTP preconditioner, we also give results for MATLAB's `ilu`. Figure 4.4(c) shows the eigenvalues for selected $\mathbf{K}_i$; $\mathbf{K}_i$ becomes indefinite by the twentieth shift. Figures 4.4(a) and 4.4(b) show that while $\mathbf{K}_i$ becomes indefinite around the twentieth shift, ILUTP produces good preconditioners until approximately the 125^{th} shift. At this shift and subsequent shifts, both our implementation of ILUTP and MATLAB's `ilu` fail to produce a good preconditioner, and the number of GMRES iterations increases substantially (or GMRES fails to converge). However, the SAM updates combined with $\mathbf{P}_0$ are successful in keeping the GMRES iterations low for almost all shifts.

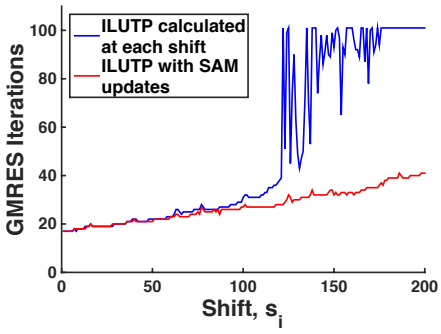
5. Conclusions and Future Work. In applications that involve many linear systems, recycling a preconditioner can be advantageous. We develop a flexible update to arbitrary preconditioners that we call the Sparse Approximate Map, or SAM update. This map is motivated by the Sparse Approximate Inverse; however, rather than approximately inverting a matrix, the SAM update approximately maps one matrix to another nearby matrix for which a good preconditioner is already available. In this paper, we discuss several applications with systems defined by a shift, but the SAM update can be applied to any set of closely related systems. The cost of computing a very good preconditioner for a chosen matrix can be amortized over many matrices in a sequence of systems, since our map is cheap to compute. Further, the map is independent of preconditioner type and quality.

When the residual of the LS map is small (in norm) and the initial preconditioner is sufficiently good, we can guarantee rapid convergence of GMRES for the preconditioned system. We show that the LS map is the best approximation to the ideal map in the Frobenius $\mathbf{A}_k$-norm. This results in a monotonic decrease in the Frobenius-norm of the residual for a sequence of nested sparsity patterns. We show for several applications that the SAM update leads to a good (updated) preconditioner.

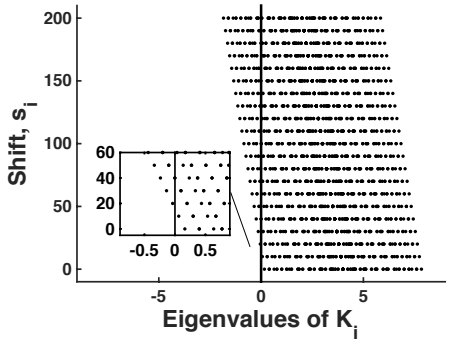
Future work will consider applying the SAM update incrementally as in (1.7) and (1.8) as well as applying the update from the left as in (1.9). Another important topic is to exploit the possibility of only updating a few columns or rows of the map to match localized changes in the matrix $\mathbf{A}_k$. We also plan to further analyze how well the LS map approximates the ideal map under more general conditions as well as determine the conditions under which our preconditioned matrices - using the SAM updates - take the form (2.14). We also plan to analyze the SAM updates for more general sequences of matrices. Finally, we will further analyze and develop good indicators for when a new map should be computed.



(a) The number of GMRES iterations to converge for the discretized Helmholtz problem computing a new ILUTP preconditioner for each $\mathbf{K}_i$ (blue line) vs. computing an ILUTP preconditioner for $\mathbf{K}_0$ and computing SAM updates for all subsequent $\mathbf{K}_i$ (red line). These results are based on our implementation of the ILUTP preconditioner.



(b) The number of GMRES iterations to converge for the discretized Helmholtz problem computing a new ILUTP preconditioner for each $\mathbf{K}_i$ (blue line) vs. computing an ILUTP preconditioner for $\mathbf{K}_0$ and computing SAM updates for all subsequent $\mathbf{K}_i$ (red line). These results are based on the MATLAB® ILUTP preconditioner, `ilu`.



(c) Eigenvalues of every tenth matrix, $\mathbf{K}_i$. At the twentieth shift, the matrices become indefinite.

Fig. 4.4: GMRES convergence and selected eigenvalues for a discretized Helmholtz problem.

6. Acknowledgements. We would like to thank Tania Bakhos, Arvind Saibaba, and Peter Kitanidis for providing the description of the Transient Hydraulic Tomography method as well as the THT matrices.

REFERENCES

[1] Alireza Aghasi, Misha E. Kilmer, and Eric L. Miller, *Parametric level set methods for inverse problems*, SIAM Journal on Imaging Sciences, 4 (2011), pp. 618–650.

[2] Mian Ilyas Ahmad, Daniel B. Szyld, and Martin B. van Gijzen, *Preconditioned multishift BiCG for H_2-optimal model reduction*, Tech. Report 12-06-15, Department of Mathematics, Temple University, June 2012. Revised March 2013 and June 2015.

[3] Kapil Ahuja, Peter Benner, Eric de Sturler, and Lihong Feng, *Recycling BiCGSTAB with an application to parametric model order reduction*, SIAM Journal on Scientific Computing, 37 (2015), pp. S429–S446.

[4] Kapil Ahuja, Bryan K. Clark, Eric de Sturler, David M. Ceperley, and Jeongnim Kim, *Improved scaling for quantum Monte Carlo on insulators*, SIAM Journal on Scientific Computing, 33 (2011), pp. 1837–1859.

[5] Kapil Ahuja, Eric de Sturler, Eun R. Chang, and Serkan Gugercin, *Recycling BiCG with an application to model reduction*, SIAM J. Sci. Comput., 34 (2012), pp. A1925–A1949.

[6] Anthanasios C. Antoulas, *Approximation of large-scale dynamical systems (Advances in Design and Control)*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[7] Athanasios C. Antoulas, Christopher A. Beattie, and Serkan Gugercin, *Interpolatory model reduction of large-scale dynamical systems*, in Efficient Modeling and Control of Large-Scale Systems, Javad Mohammadpour and Karolos M. Grigoriadis, eds., Springer, 2010, pp. 3–58.

[8] Zhaojun Bai and Karl Meerbergen, *The Lanczos method for parametrized symmetric linear systems with multiple right-hand sides*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 1642–1662.

[9] Tania Bakhos, Arvind K. Saibaba, and Peter K. Kitanidis, *A fast algorithm for parabolic PDE-based inverse problems based on Laplace transforms and flexible Krylov solvers*, Journal of Computational Physics, 299 (2015), pp. 940–954.

[10] Manuel Baumann and Martin B. van Gijzen, *Nested Krylov methods for shifted linear systems*, SIAM Journal on Scientific Computing, 37 (2015), pp. S90–S112.

[11] Ulrike Baur, Peter Benner, and Lihong Feng, *Model order reduction for linear and nonlinear systems: a system-theoretic perspective*, Archives of Computational Methods in Engineering, 21 (2014), pp. 331–358.

[12] Christopher A. Beattie and Serkan Gugercin, *Interpolatory projection methods for structure-preserving model reduction*, Systems and Control Letters, 58 (2009), pp. 225–232.

[13] ———, *Model reduction by rational interpolation*, in Model Reduction and Approximation: Theory and Algorithms, Peter Benner, Albert Cohen, Mario Ohlberger, and Karen Willcox, eds., SIAM, Philadelphia, PA, 2016.

[14] Christopher A. Beattie, Serkan Gugercin, and Sarah A. Wyatt, *Inexact solves in interpolatory model reduction*, Linear Algebra and its Applications, 436 (2012), pp. 2916–2943.

[15] Stefania Bellavia, Daniele Bertaccini, and Benedetta Morini, *Nonsymmetric preconditioner updates in Newton-Krylov methods for nonlinear systems*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2595–2619.

[16] Peter Benner, Gene Golub, Volker Mehrmann, and Daniel C. Sorensen, *Oberwolfach model reduction benchmark collection*. https://portal.uni-freiburg.de/imteksimulation/downloads/benchmark, 2015. IMTEK Simulation, University of Freiburg.

[17] Peter Benner, Serkan Gugercin, and Karen Willcox, *A survey of projection-based model reduction methods for parametric dynamical systems*, SIAM Review, 57 (2015), pp. 483–531.

[18] Maurice W. Benson, *Iterative solution of large scale linear systems*, master's thesis, Lakehead University, Thunder Bay, Canada, 1973.

[19] Maurice W. Benson and Paul O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math, 22 (1982), pp. 154–155.

[20] Michele Benzi and Daniele Bertaccini, *Approximate inverse preconditioning for shifted linear systems*, BIT Numerical Mathematics, 43 (2003), pp. 231–244.

[21] Michele Benzi, Jane K. Cullum, and Miroslav Tuma, *Robust approximate inverse preconditioning for the Conjugate Gradient method*, SIAM Journal on Scientific Computing, 22 (2000), pp. 1318–1332.

[22] Michele Benzi, John C. Haws, and Miroslav Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM Journal on Scientific Computing, 22 (2000), pp. 1333–1353.
[23] Michele Benzi, Reijo Kouhia, and Miroslav Tuma, *Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics*, Computer Methods in Applied Mechanics and Engineering, 190 (2001), pp. 6533–6554.
[24] Michele Benzi and Miroslav Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM Journal on Scientific Computing, 19 (1998), pp. 968–994.
[25] Jeffrey T. Borggaard and Serkan Gugercin, *Model reduction for DAEs with an application to flow control*, in Active Flow and Combustion Control, Rudibert King, ed., Springer International Publishing, 2015, pp. 381–396.
[26] Caterina Calgaro, Jean-Paul Chehab, and Yousef Saad, *Incremental incomplete LU factorizations with applications*, Numerical Linear Algebra with Applications, 17 (2010), pp. 811–837.
[27] Stephen L. Campbell, Ilse C.F. Ipsen, C. Tim Kelley, and Carl D. Meyer, *GMRES and the minimal polynomial*, BIT Numerical Mathematics, 36 (1996), pp. 664–675.
[28] Michael Cardiff and Warren Barrash, *3-D Transient hydraulic tomography in unconfined aquifers with fast drainage response*, Water Resources Research, 47 (2011).
[29] Edmund Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1804–1822.
[30] Edmund Chow and Yousef Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM Journal on Scientific Computing, 19 (1998), pp. 995–1023.
[31] Eric de Sturler, Serkan Gugercin, Misha E. Kilmer, Saifon Chaturantabut, Christopher A. Beattie, and Meghan O'Connell, *Nonlinear parametric inversion using interpolatory model reduction*, SIAM Journal on Scientific Computing, 37 (2015), pp. B495–B517.
[32] Eric de Sturler and Misha E. Kilmer, *A regularized Gauss-Newton trust region approach to imaging in diffuse optical tomography*, SIAM Journal on Scientific Computing, 33 (2011), pp. 3057–3086.
[33] Howard C. Elman and Dianne P. O'Leary, *Efficient iterative solution of the three-dimensional Helmholtz equation*, Journal of Computational Physics, 142 (1998), pp. 163–181.
[34] ———, *Eigenanalysis of some preconditioned Helmholtz problems*, Numerische Mathematik, 83 (1999), pp. 231–257.
[35] Björn Engquist and Lexing Ying, *Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation*, Communications on Pure and Applied Mathematics, 64 (2011), pp. 697–735.
[36] Yogi A. Erlangga and Reinhard Nabben, *On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian*, Electronic Transactions on Numerical Analysis, 31 (2008), pp. 403–424.
[37] Yogi A. Erlangga, Cornelis Vuik, and Cornelis W. Oosterlee, *Comparison of multigrid and incomplete LU shifted-Laplace preconditioners for the inhomogeneous Helmholtz equation*, Applied Numerical Mathematics, 56 (2006), pp. 648–666.
[38] Lihong Feng, Peter Benner, and Jan G. Korvink, *Parametric model order reduction accelerated by subspace recycling*, in 48th IEEE Conference on Decision & Control and 28th Chinese Control Conference, 2009, pp. 4328–4333.
[39] ———, *Subspace recycling accelerates the parametric macro-modeling of MEMS*, International Journal for Numerical Methods in Engineering, 94 (2013), pp. 84–110.
[40] Nabil Gmati and Bernard Philippe, *Comments on the GMRES convergence for preconditioned systems*, in Large-scale scientific computing, Springer, 2008, pp. 40–51.
[41] Anne Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, PA, 1997.
[42] Marcus J. Grote and Thomas Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM Journal on Scientific Computing, 18 (1997), pp. 838–853.
[43] Guiding Gu, Xianli Zhou, and Lei Lin, *A flexible preconditioned Arnoldi method for shifted linear systems*, Journal of Computational Mathematics, 25 (2007), pp. 522–530.
[44] Serkan Gugercin and Anthanasios C. Antoulas, *A survey of model reduction by balanced truncation and some new results*, International Journal of Control, 77 (2004), pp. 748–766.
[45] Serkan Gugercin, Anthanasios C. Antoulas, and Christopher A. Beattie, *H_2 model reduction for large-scale linear dynamical systems*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 609–638.
[46] Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, 2002.
[47] Ruth M. Holland, Andy J. Wathen, and Gareth J. Shaw, *Sparse approximate inverses*

and target matrices, SIAM Journal on Scientific Computing, 26 (2005), pp. 1000–1011.
[48] Thomas Huckle, *Approximate sparsity patterns for the inverse of a matrix and preconditioning*, Applied Numerical Mathematics, 30 (1999), pp. 291–303.
[49] Misha E. Kilmer and Eric de Sturler, *Recycling subspace information for diffuse optical tomography*, SIAM Journal on Scientific Computing, 27 (2006), pp. 2140–2166.
[50] Ming Li, Vishwas Rao, and Eric de Sturler, *Effective truncation of low-rank preconditioner updates*. Submitted for publication, 2015.
[51] Anders Logg, Kent-Andre Mardal, and Garth N. Wells, eds., *Automated Solution of Differential Equations by the Finite Element Method*, vol. 84, Springer, 2012.
[52] Anders Logg and Garth N. Wells, *DOLFIN: Automated finite element computing*, ACM Transactions on Mathematical Software, 37 (2010), p. 20.
[53] Anders Logg, Garth N. Wells, and Johan Hake, *DOLFIN: a C++/Python finite element library*, in Automated Solution of Differential Equations by the Finite Element Method, Anders Logg, Kent-Andre Mardal, and Garth N. Wells, eds., Springer, 2012, ch. 10.
[54] L. Meier and D.G. Luenberger, *Approximation of linear constant systems*, IEE. Trans. Automat. Contr., 12 (1967), pp. 585–588.
[55] Christian Moosmann, Evgenii B. Rudnyi, Andreas Greiner, and Jan G. Korvink, *Model order reduction for linear convective thermal flow*, in Proceedings of 10th International Workshops on THERMal INvestigations of ICs and Systems, THERMINIC2004, vol. 29, 2004, pp. 317–322.
[56] Amin Rafiei, *Left-looking version of AINV preconditioner with complete pivoting strategy*, Linear Algebra and its Applications, 445 (2014), pp. 103–126.
[57] Evgenii B. Rudnyi and Jan G. Korvink, *Review: Automatic model reduction for transient simulation of MEMS-based devices*, Sensors Update, 11 (2002), pp. 3–33.
[58] Yousef Saad, *SPARSKIT: A basic took-kit for sparse matrix computations*. http://www-users.cs.umn.edu/ saad/software/SPARSKIT/, 1994.
[59] ———, *Iterative Methods for Sparse Linear Systems, 2nd Ed.*, SIAM, 2003.
[60] Arvind K. Saibaba, Tania Bakhos, and Peter K. Kitanidis, *A flexible Krylov solver for shifted systems with application to oscillatory hydraulic tomography*, SIAM Journal on Scientific Computing, 35 (2013), pp. A3001–A3023.
[61] Abdul H. Sheikh, Domenico Lahaye, and Cornelis Vuik, *On the convergence of shifted Laplace preconditioner combined with multilevel deflation*, Numerical Linear Algebra with Applications, 20 (2013), pp. 645–662.
[62] Henk A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, 2003.
[63] Jacob Andre C. Weideman, *Optimizing Talbot's contours for the inversion of the Laplace transform*, SIAM Journal on Numerical Analysis, 44 (2006), pp. 2342–2362.