SIAM J. Matrix Anal. Appl.
Vol. 35, No. 2, pp. 411–436

COMPUTING A PARTIAL SCHUR FACTORIZATION OF NONLINEAR EIGENVALUE PROBLEMS USING THE INFINITE ARNOLDI METHOD[*]

ELIAS JARLEBRING[†], KARL MEERBERGEN[‡], AND WIM MICHIELS[‡]

Abstract. The partial Schur factorization can be used to represent several eigenpairs of a matrix in a numerically robust way. Different adaptions of the Arnoldi method are often used to compute partial Schur factorizations. We propose here a technique to compute a partial Schur factorization of a nonlinear eigenvalue problem (NEP). The technique is an extension of our algorithm from [E. Jarlebring, W. Michiels, and K. Meerbergen, *Numer. Math.*, 122 (2012), pp. 169–195], now called the *infinite Arnoldi method*. The infinite Arnoldi method is a method designed for NEPs, and can be interpreted as Arnoldi's method applied to a linear infinite-dimensional operator, whose reciprocal eigenvalues are the solutions to the NEP. As a first result we show that the invariant pairs of the operator are equivalent to invariant pairs of the NEP. We characterize the structure of the invariant pairs of the operator and show how one can carry out a modification of the infinite Arnoldi method by respecting the structure. This also allows us to naturally add the feature known as *locking*. We nest this algorithm with an outer iteration, where the infinite Arnoldi method for a particular type of structured functions is appropriately restarted. The restarting exploits the structure and is inspired by the well-known implicitly restarted Arnoldi method for standard eigenvalue problems. The final algorithm is applied to examples from a benchmark collection, showing that both processing time and memory consumption can be considerably reduced with the restarting technique.

Key words. Arnoldi's method, nonlinear eigenvalue problems, invariant pairs, restarting

AMS subject classifications. 65F15, 65H17, 65F60, 35P30, 15A18

DOI. 10.1137/110858148

1. Introduction.

The *nonlinear eigenvalue problem* (NEP) will be used in this paper to refer to the problem of finding $\lambda \in \Omega \subseteq \mathbb{C}$ and $v \in \mathbb{C}^n \backslash \{0\}$ such that

$$M(\lambda)v = 0, \tag{1.1}$$

where $M : \Omega \to \mathbb{C}^{n\times n}$ is analytic in Ω, which is an open disc centered at the origin.

This problem class has received a considerable amount of attention in the literature. See, e.g., the survey papers [22, 25] and the monographs [14, 8]. The results for (1.1) are often (but not always) presented with some restriction of the generality of M, such as the theory and algorithms for polynomial eigenvalue problems (PEPs) in [14, 15, 19, 7] and [1, Chapter 9], in particular the algorithms for quadratic eigenvalue problems (QEPs) [29, 2, 20], but also recent approaches for rational eigenvalue problems (REPs) [28, 31]. There are also general approaches based on modifications of the Arnoldi method [32] and the Jacobi–Davidson method [5]. In [11], we present an iterative algorithm equivalent to the Arnoldi method which we call the *infinite*

*Received by the editors December 7, 2011; accepted for publication (in revised form) October 31, 2013; published electronically April 8, 2014. This research was supported by the Dahlquist research fellowship, by the Programme of Interuniversity Attraction Poles of the Belgian Federal Science Policy Office (IAP P6-DYSCO), by OPTEC, the Optimization in Engineering Center of the KU Leuven, by project OT/10/038 of the KU Leuven Research Council, and by project G.0712.11N of the Research Foundation-Flanders (FWO).

http://www.siam.org/journals/simax/35-2/85814.html

†Department of Mathematics, KTH Royal Institute of Technology, Lindstedtsvägen 25, Stockholm 10044, Sweden (eliasj@kth.se).

‡Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Leuven, 3001, Belgium (karl.meerbergen@cs.kuleuven.be, wim.michiels@cs.kuleuven.be).

Arnoldi method and [30] contains comparisons with Newton-like methods, which are closely related to the Jacobi–Davison method. The results we will now present are directly related to the infinite Arnoldi method. An important aspect of the algorithm in this paper, and the infinite Arnoldi method, is *generality*. Although the algorithm and results of this paper are applicable to PEPs, QEPs, and REPs, the primary goal of the paper is not to solve problems for the most common structures, but rather to construct an algorithm which can be applied to other, less common NEPs in a somewhat automatic fashion. Some less common NEPs are given in the problem collection [4]; there exist NEPs with exponential terms [10] and implicitly stated NEPs such as [24].

In this paper we will present a procedure to compute a partial Schur factorization in the sense of the concepts of partial Schur factorizations and invariant pairs for NEPs introduced in [13]. These concepts can be summarized as follows. First note that the function M is assumed in this work to be analytic and can always be decomposed as a sum of products of constant matrices and scalar nonlinearities,

$$M(\lambda) = M_1 f_1(\lambda) + \cdots + M_m f_m(\lambda), \tag{1.2}$$

where $f_i : \Omega \to \mathbb{C}$, $i = 1, \ldots, m$, are analytic in Ω. We define

$$\mathbb{M}(Y, \Lambda) := M_1 Y f_1(\Lambda) + \cdots + M_m Y f_m(\Lambda),$$

where $Y \in \mathbb{C}^{n\times p}$, $\Lambda \in \mathbb{C}^{p\times p}$, and $f_i(\Lambda)$, $i = 1, \ldots, m$, are the matrix functions corresponding to f_i, which are well defined if $\sigma(\Lambda) \subset \Omega$. An invariant pair $(Y, \Lambda) \in \mathbb{C}^{n\times p} \times \mathbb{C}^{p\times p}$ (in the sense of [13, Definition 1]) satisfies

$$\mathbb{M}(Y, \Lambda) = 0. \tag{1.3}$$

Additional appropriate orthogonality conditions for Y and Λ yield a consistent definition of invariant pairs. In particular, if (Y, Λ) is an invariant pair and (w, λ) is an eigenpair of Λ, then λ is a solution to the NEP (1.1) with eigenvector $v = Yw$. In this setting, a partial Schur factorization corresponds to a particular invariant pair where Λ is an upper triangular matrix.

The results of this paper are based on a reformulation of the problem of finding an invariant pair of the NEP as a corresponding problem formulated with a (linear) infinite-dimensional operator denoted $\mathcal{B}$, also used in the infinite Arnoldi method [11]. In [11], we presented an algorithm which can be interpreted as Arnoldi's method applied to the operator $\mathcal{B}$. Although the operator $\mathcal{B}$ maps functions to functions, it turns out that the algorithm can be implemented with finite-dimensional linear algebra operations if the Arnoldi method (for $\mathcal{B}$) is started with a constant function. This results in a Krylov subspace consisting of polynomials. Unlike the polynomial setting in [11], in this work we will consider linear combinations of exponentials and polynomials allowing us to carry out an efficient restarting process. We will show that similar to the polynomial setting [11], the Arnoldi method for $\mathcal{B}$ applied to linear combinations of polynomials and exponentials can be carried out with finite-dimensional linear algebra operations.

The reformulation with the operator $\mathcal{B}$ allows us to adapt a procedure based on the Arnoldi method designed for the computation of a partial Schur factorization for standard eigenvalue problems. We will use a construction inspired by the *implicitly restarted Arnoldi method* (IRAM) [26, 23, 16, 17]. The construction is first outlined in section 3 and consists of two steps respectively given in sections 4 and 5. They correspond to carrying out the Arnoldi method for the operator $\mathcal{B}$ with a locked invariant pair and a procedure to restart it.

We finally wish to mention that there exist restarting schemes for algorithms for special cases of (1.1), in particular for QEPs [33, 12].

2. Reformulation as infinite-dimensional operator problem.

In order to characterize the invariant pairs of (1.1) for our setting we first need to introduce some notation. For notational convenience, we will introduce (analogous to the notation in [11]) the function $B : \Omega \to \mathbb{C}^{n\times n}$, defined by

$$B(\lambda) := M(0)^{-1}\frac{M(0) - M(\lambda)}{\lambda} \tag{2.1}$$

for $\lambda \in \Omega\backslash\{0\}$. We define it as the analytic continuation at $\lambda = 0$. Note that B is also analytic in Ω, under the condition that $\lambda = 0$ is not a solution to (1.1). We will assume that the NEP is such that $\lambda = 0$ is not an eigenvalue. If $\lambda = 0$ is an eigenvalue, the problem can be shifted with a change of variables, transforming the zero eigenvalue to a nonzero value. From the definition (2.1) we reach a transformed NEP,

$$\lambda B(\lambda)v = v. \tag{2.2}$$

We will also use a decomposition of B similar to the decomposition (1.2) of M. That is, we let

$$B(\lambda) = B_1 b_1(\lambda) + \cdots + B_m b_m(\lambda), \tag{2.3}$$

where $b_i : \Omega \to \mathbb{C}$, $i = 1, \ldots, m$, are analytic in Ω. Moreover, we will use the straightforward coupling of the decomposition of M by setting

$$B_i = M(0)^{-1}M_i, \quad b_i(\lambda) = \frac{f_i(0) - f_i(\lambda)}{\lambda}. \tag{2.4}$$

We will use the following notation in order to have a concise notation. Let the differentiation operator $B(\frac{d}{d\theta})$ be defined by the Taylor expansion in a consistent way, i.e.,

$$\left(B\left(\frac{d}{d\theta}\right)\varphi\right)(\theta) := B(0)\varphi(\theta) + \frac{1}{1!}B'(0)\varphi'(\theta) + \frac{1}{2!}B''(0)\varphi''(\theta) + \cdots,$$

where $\varphi : \mathbb{C} \to \mathbb{C}^n$ is a smooth function. We are now ready to introduce the operator which serves as the basis for the algorithm.

Definition 2.1 (the operator $\mathcal{B}$). *Let $\mathcal{B}$ denote the map defined by the domain $\mathcal{D}(\mathcal{B}) := \{\varphi \in C_\infty(\mathbb{C}, \mathbb{C}^n) : \sum_{i=0}^{\infty} B^{(i)}(0)\varphi^{(i)}(0)/(i!) < \infty\}$ and the action*

$$(\mathcal{B}\varphi)(\theta) = \int_0^\theta \varphi(\hat{\theta})\, d\hat{\theta} + C(\varphi), \tag{2.5}$$

where

$$C(\varphi) := \sum_{i=0}^{\infty} \frac{1}{i!}B^{(i)}(0)\varphi^{(i)}(0) = \left(B\left(\frac{d}{d\theta}\right)\varphi\right)(0). \tag{2.6}$$

Several properties of the operator $\mathcal{B}$ are characterized in [11]. Most importantly, the set consisting of the reciprocal eigenvalues of $\mathcal{B}$ is equal to the set of λ such that (v, λ) is a solution to (2.2) and hence also to (1.1) if $\lambda \neq 0$. In this work we will need a more general result, characterizing the invariant pairs of $\mathcal{B}$.

To this end we first define the application of the operator $\mathcal{B}$ to block functions and say that if $\Psi : \mathbb{C} \to \mathbb{C}^{n\times p}$ with columns given by

$$\Psi(\theta) = (\psi_1(\theta), \ldots, \psi_p(\theta)),$$

then $\mathcal{B}\Psi$ is interpreted in a column fashion, i.e.,

$$(\mathcal{B}\Psi)(\theta) := (\mathcal{B}\psi_1(\theta), \ldots, \mathcal{B}\psi_p(\theta)).$$

With this notation, we can now consistently define an invariant pair as a pair (Ψ, R) of the operator $\mathcal{B}$, where $\Psi : \mathbb{C} \to \mathbb{C}^{n\times p}$ and $R \in \mathbb{C}^{p\times p}$ such that

$$(\mathcal{B}\Psi)(\theta) = \Psi(\theta)R \quad \text{for all } \theta \in \mathbb{C}. \tag{2.7}$$

The following theorem explicitly shows the structure of the function Ψ and relates invariant pairs of the operator with invariant pairs (1.3), i.e., invariant pairs in the setting in [13].

Theorem 2.2 (invariant pairs of $\mathcal{B}$). *Suppose $\Lambda \in \mathbb{C}^{p\times p}$ is invertible and suppose (Ψ, Λ^{-1}) is an invariant pair of $\mathcal{B}$. Then, Ψ can be expressed as*

$$\Psi(\theta) = Y \exp(\theta\Lambda) \tag{2.8}$$

for some matrix $Y \in \mathbb{C}^{n\times p}$. Moreover, given $\Lambda \in \mathbb{C}^{p\times p}$ and $Y \in \mathbb{C}^{n\times p}$, where Λ is invertible, the following statements are equivalent:

(i) *The pair (Ψ, Λ^{-1}), where $\Psi(\theta) := Y \exp(\theta\Lambda)$, is an invariant pair of the operator $\mathcal{B}$, i.e.,*

$$(\mathcal{B}\Psi)(\theta) = \Psi(\theta)\Lambda^{-1}.$$

(ii) *The pair (Y, Λ) is an invariant pair of the NEP (1.1) in the sense of* [13, *Definition* 1]*, i.e.,*

$$\mathbb{M}(Y, \Lambda) = 0. \tag{2.9}$$

Proof. Suppose (Ψ, Λ^{-1}) is an invariant pair of $\mathcal{B}$. By differentiating the function equality (2.7) with respect to θ, and using that the action of $\mathcal{B}$ is integration, we find that Ψ satisfies the matrix differential equation,

$$\Psi(\theta) = \Psi'(\theta)R,$$

with $\Lambda^{-1} = R$. All solutions to this differential equation are of the form $Y \exp(\theta\Lambda)$, i.e., Ψ can be expressed as (2.8). The equivalence between statements (i) and (ii) follows directly from the fact that $M(0)$ is invertible (since Λ is invertible and $\lambda = 0$ is not an eigenvalue) and the application of Lemma A.1. ∎

3. Outline of the algorithm.

We now know (from Theorem 2.2) that an invariant pair of the NEP (1.1) is equivalent to an invariant pair of the linear operator $\mathcal{B}$. The general idea of the procedure we will present in later sections is inspired by the procedures used to compute partial Schur factorizations for standard eigenvalue problems with the Arnoldi method [17, 26, 27]. We will carry out a variant of the corresponding algorithm for the operator $\mathcal{B}$. More precisely, we will repeat the following two steps.

In the first step (described in section 4) we compute, in a particular way, an orthogonal projection of the operator $\mathcal{B}$ onto a Krylov subspace. The projection is constructed such that it possesses the feature known as *locking*. This means that given a partial Schur factorization (or an approximation of the partial Schur factorization) the projection respects the invariant subspace and returns an approximation containing the invariant pair (without modification) and also approximations of further eigenvalues. This prevents repeated convergence to the eigenvalues in the locked partial Schur factorization in a robust way.

In the literature (for standard eigenvalue problems) this projection is often computed with a variation of the Arnoldi method. More precisely, we will start the Arnoldi algorithm with a state containing the (locked) partial Schur factorization. The result of the infinite Arnoldi method can be expressed as what is commonly called an *Arnoldi factorization*,

$$(\mathcal{B}F_k)(\theta) = F_{k+1}(\theta)\underline{H}_k, \tag{3.1}$$

where $\underline{H}_k \in \mathbb{C}^{(k+1)\times k}$ is a Hessenberg matrix, $F_{k+1} : \mathbb{C} \to \mathbb{C}^{n\times k}$ is an orthogonal basis of the Krylov subspace, and F_k is the first k columns of F_{k+1}. In this paper we use a common notation for Hessenberg matrices; the first k rows of the matrix $\underline{H}_k$ will be denoted $H_k \in \mathbb{C}^{k\times k}$. A property of the locking feature is that the Hessenberg matrix $\underline{H}_k$ has the structure

$$\underline{H}_k = \begin{pmatrix} (\underline{H}_k)_{1,1} & (\underline{H}_k)_{1,2} \\ & (\underline{H}_k)_{2,2} \end{pmatrix},$$

where $R = (\underline{H}_k)_{1,1} \in \mathbb{C}^{p_\ell \times p_\ell}$ is an upper triangular matrix. The upper left block of the $\underline{H}_k$ is called the *locked* part, since the first p_ℓ columns of (3.1) are the equation for an invariant pair (2.7).

In the first step we show how the Arnoldi method with locking can be carried out if we represent the functions in the algorithm (and in the factorization (3.1)) in a structured way. Unlike the infinite Arnoldi method in [11] we will need to work with functions which are linear combinations of exponentials and polynomials. It turns out that, similar to [11], the action of the operator as well as the entire Arnoldi algorithm can be carried out with finite-dimensional arithmetic, while the use of exponentials is benificial also for the second step.

In the second step (described in section 5), i.e., after computing the Arnoldi factorization (3.1), we process the factorization such that two types of information can be extracted:

- We extract converged eigenvalues from the Arnoldi factorization (3.1) and store those in a partial Schur factorization. Due to the locking feature, the updated partial Schur factorization will be of the same size as the locked part of (3.1) or larger.
- We extract a function with favorable approximation properties for those eigenvalues of interest, which have not yet converged.

This information is extracted in a fashion similar to the implicitly restarted Arnoldi method (IRAM) [23, 16, 17, 27]. However, several modifications are necessary in order to restart with the structured functions.

The two steps are subsequently iterated by starting the (locked version) of Arnoldi's method with the extracted function and with (the possibly larger) partial Schur factorization. Thus, we nest the infinite Arnoldi method with a restarting scheme which is expected to eventually converge to a partial Schur factorization.

4. The infinite Arnoldi method with locked invariant pair.

In the first step of the conceptual algorithm described in section 3, we need to carry out an Arnoldi algorithm for $\mathcal{B}$ with the preservation feature that the given partial Schur factorization is not modified. In an infinite-dimensional setting, the adaption to achieve this feature with the Arnoldi method is straightforward by initiating the state of the Arnoldi method with the invariant pair. The procedure is given in Algorithm 1, where the basis of the invariant subspace associated with the partial Schur factorization is assumed to be orthogonal with respect to a given inner product $\langle\cdot,\cdot\rangle$ defined on the space of n-vector valued analytical functions.

4.1. Representation of structured functions.

In later sections we will provide a specialization of all the steps in the abstract algorithm (Algorithm 1) such that we can implement it in finite-dimensional arithmetic. The first step in the conversion of Algorithm 1 into a finite-dimensional algorithm is to select an appropriate starting function and an appropriate finite-dimensional representation of the functions.

In this work, we will consider functions which are sums of exponentials and polynomials with the structure

$$\varphi(\theta) = Ye^{S\theta}c + q(\theta), \tag{4.1}$$

where $Y \in \mathbb{C}^{n\times p}$, $S \in \mathbb{C}^{p\times p}$, $c \in \mathbb{C}^p$, and $q : \mathbb{C} \to \mathbb{C}^n$ is a vector of polynomials. We will see that these functions allow us to represent approximations of invariant pairs with an exploitable structure suitable to be used in a restarting. Moreover, we let S be a block triangular matrix

$$S = \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} \tag{4.2}$$

and set $S_{11} = R^{-1} \in \mathbb{C}^{p_\ell \times p_\ell}$, where $p_\ell \le p$, where R will later be chosen such that it is an approximation of the matrix in the Schur factorization. This structure has a number of favorable properties important for our situation:

Algorithm 1. Arnoldi's method for $\mathcal{B}$ with locking.

Input: A partial Schur factorization of $\mathcal{B}$ represented by (Ψ, R) and an analytical vector-valued function $f : \mathbb{C} \to \mathbb{C}^n$ such that $\langle f, f\rangle = 1$ and such that f is orthogonal to the columns of Ψ with respect to $\langle\cdot,\cdot\rangle$.

Output: An Arnoldi factorization of $\mathcal{B}$ represented by $(\varphi_1, \dots, \varphi_{k_{\max}})$ and $H_{k_{\max}+1,k_{\max}}$

```
1:  Set $H_{p_\ell,p_\ell} = R$
2:  Set $(\varphi_1, \dots, \varphi_{p_\ell}) = \Psi$
3:  Set $\varphi_{p_\ell+1} = f$
4:  for $k = p_\ell + 1, \dots, k_{\max}$ do
5:      $\psi = \mathcal{B}\varphi_k$ and $\psi_\perp = \psi$
6:      for $i = 1, \dots, k$ do
7:          $h_{i,k} = \langle \psi, \varphi_i \rangle$
8:          $\psi_\perp = \psi_\perp - h_{i,k}\varphi_i$
9:      end for
10:     Repeat Gram–Schmidt process (Steps 6–9) if necessary
11:     $h_{k+1,k} = \sqrt{\langle \psi_\perp, \psi_\perp \rangle}$
12:     $\varphi_{k+1} = \psi/h_{k+1,k}$
13: end for
```

- The action of $\mathcal{B}$ applied to functions of the type (4.1) can be carried out in an efficient way using only finite-dimensional operations. This stems from the property that the action of $\mathcal{B}$ corresponds to integration and the set of polynomials and exponentials under consideration is closed under integration. Algorithmic details will be given in section 4.2.
- This particular structure allows the storing and orthogonalization against an invariant subspace, which, according to Theorem 2.2, has exponential structure.
- The structure provides freedom to choose the blocks S_{12} and S_{22}. This allows us to appropriately restart the algorithm. Due to the exponential structure illustrated in Theorem 2.2, it will turn out to be natural to impose an exponential structure on the Ritz functions in order to construct a function f to be used in the restart. The precise choice of S_{12}, S_{22}, and Y will be further explained in section 5.

In practice we also need to store the structured functions in some fashion, preferably with matrices and vectors. It is tempting to store the exponential part and polynomial part of (4.1) separately, i.e., to store the exponential part with the variables Y, S, and c and the polynomial part by coefficients in some polynomial basis, e.g., the coefficients $y_0,\dots,y_{N-1}$ in the monomial basis $q(\theta)=y_0+y_1\theta+\cdots+y_{N-1}\theta^{N-1}$. Although such an approach is natural from a theoretical perspective, it is not adequate from a numerical perspective. This can be seen as follows. Note that the Taylor expansion of the structured function (4.1) is

(4.3)
$$\varphi(\theta)=(Yc+y_0)+\left(\frac{1}{1!}YSc+y_1\right)\theta+\cdots+\left(\frac{1}{(N-1)!}YS^{N-1}c+y_{N-1}\right)\theta^{N-1} + \left(\frac{1}{N!}YS^{N}c\right)\theta^{N}+\left(\frac{1}{(N+1)!}YS^{N+1}c\right)\theta^{N+1}+\cdots.$$

A potential source of cancellation is apparent for the first N terms in (4.3) if the polynomial $q(\theta)$ approximates $-Y\exp(\theta S)c$. This turns out to be a situation appearing in practice in this algorithm, making the storing of the structured functions in this separated form inadequate.

We will instead use a function representation where the coefficients in the Taylor expansion are not formed by sums. This can be achieved by replacing the first N terms in (4.3) by new coefficients $x_0,\dots,x_{N-1}$, i.e.,

(4.4)
$$\varphi(\theta)=x_0+x_1\theta+\cdots+x_{N-1}\theta^{N-1}+\frac{1}{N!}(YS^{N}c)\theta^{N}+\frac{1}{(N+1)!}(YS^{N+1}c)\theta^{N+1}+\cdots.$$

The structured functions (4.1) will be represented with the four variables $Y\in\mathbb{C}^{n\times p}$, $S\in\mathbb{C}^{p\times p}$, $c\in\mathbb{C}^{p}$, $x\in\mathbb{C}^{Nn}$, where $x^T=(x_0^T,\dots,x_{N-1}^T)$. Note that this representation does not suffer from the potential cancellation effects present in the naive representation (4.3).

Throughout this work we will need to carry out many manipulations of functions represented in the form (4.4) and we need a concise notation. Let $\exp_N$ denote the remainder of the truncated Taylor expansion of the exponential, i.e.,

$$\exp_N(\theta S) := \exp(\theta S) - I - \frac{1}{1!}\theta S - \cdots - \frac{1}{N!}\theta^N S^N. \tag{4.5}$$

This can equivalently be expressed as

$$\exp_N(\theta S) = \frac{1}{(N+1)!}\theta^{N+1}S^{N+1} + \frac{1}{(N+2)!}\theta^{N+2}S^{N+2} + \cdots \tag{4.6}$$

with

$$\exp_{-1}(\theta S) := \exp(\theta S).$$

With this notation, we can now concisely express (4.4) with $\exp_N$ and Kronecker products,

$$\varphi(\theta) = Y\exp_{N-1}(\theta S)c + \big((1,\theta,\theta^2,\ldots,\theta^{N-1})\otimes I_n\big)\,x. \tag{4.7}$$

Remark 4.1 (*other representations of functions*). Note that we here propose to use a representation of functions consisting of sums of polynomials and exponentials with monomials and the remainder term in the Taylor expansion of the exponential as in (4.7). We have decided to use this representation as it simplifies the Gram–Schmidt orthogonalization (presented in section 4.4) and it has better numerical stability properties in comparison to (4.3) in general. Depending on application and the inner product used in the Arnoldi method, different choices of the function representations might be suitable, e.g., by using coefficients of Chebyshev polynomials (as we have presented in [11]) instead of monomial coefficients or using function values at a Chebyshev grid, as is the representation in the software package `chebfun` [3].

4.2. Action for structured functions. We have now (in section 4.1) introduced the function structure and shown how we can represent these functions with matrices and vectors. An important component in Algorithm 1 is the action of $\mathcal{B}$. We will now show how we can compute the action of $\mathcal{B}$ applied to a function given with the representation (4.7).

It will be convenient to introduce notation for the remainder of the NEP $\mathbb{M}$ after a Taylor expansion to order N, analogous to the definition of $\exp_N$. We define

$$\begin{aligned}\mathbb{M}_N(Y,S) := {}& \mathbb{M}(Y,S) - M(0)Y - \frac{1}{1!}M'(0)YS \\ & - \frac{1}{2!}M''(0)YS^2 - \cdots - \frac{1}{N!}M^{(N)}(0)YS^N\end{aligned} \tag{4.8}$$

or, equivalently,

$$\mathbb{M}_N(Y,S) = \frac{1}{(N+1)!}M^{(N+1)}(0)YS^{N+1} + \frac{1}{(N+2)!}M^{(N+2)}(0)YS^{N+2} + \cdots. \tag{4.9}$$

Note that with this definition

$$\mathbb{M}_{-1}(Y,S) = \mathbb{M}(Y,S).$$

We are now ready to express the action of $\mathcal{B}$ applied to functions with the structure (4.7). Note that the construction of the new function $\varphi_+ = \mathcal{B}\varphi$ in the following result involves only standard linear algebra operations of matrices and vectors.

THEOREM 4.2 (action for structured functions). *Let $S \in \mathbb{C}^{p\times p}$ and $c \in \mathbb{C}^p$ be given constants, where S is invertible. Suppose*

$$\varphi(\theta) = Y \exp_{N-1}(\theta S)c + \left((1,\theta,\theta^2,\ldots,\theta^{N-1}) \otimes I_n\right) x. \tag{4.10}$$

Then,

$$\varphi_+(\theta) := (\mathcal{B}\varphi)(\theta) = Y \exp_N(\theta S)c_+ + \left((1,\theta,\theta^2,\ldots,\theta^{N}) \otimes I_n\right) x_+, \tag{4.11}$$

where

$$c_+ = S^{-1}c, \tag{4.12}$$

$$(x_{+,1},\ldots,x_{+,N}) = (x_0,\ldots,x_{N-1}) \begin{pmatrix} 1 & & & \\ & \frac{1}{2} & & \\ & & \ddots & \\ & & & \frac{1}{N} \end{pmatrix}, \tag{4.13}$$

and

$$x_{+,0} = -M(0)^{-1}\left(\mathbb{M}_N(Y,S)c_+ + \sum_{i=1}^{N} M^{(i)}(0)x_{+,i}\right). \tag{4.14}$$

Proof. We show that φ_+ constructed by (4.12), (4.13), and (4.14) satisfies

$$\mathcal{B}\varphi = \varphi_+ \tag{4.15}$$

by first showing that the derivative of the left-hand side and the derivative of the right-hand side of (4.15) are equal and then showing that they are also equal in one point $\theta = 0$. From the property (4.6), we have

$$\frac{d}{d\theta}\exp_N(\theta S) = \frac{1}{N!}\theta^N S^{N+1} + \frac{1}{(N+1)!}\theta^{N+1}S^{N+2} + \cdots = \exp_{N-1}(\theta S)S. \tag{4.16}$$

Moreover, the relation (4.13) implies that

$$\frac{d}{d\theta}\left((1,\theta,\theta^2,\ldots,\theta^{N}) \otimes I_n\right) x_+ = \left((1,\theta,\theta^2,\ldots,\theta^{N-1}) \otimes I_n\right) x. \tag{4.17}$$

Note that $\mathcal{B}$ corresponds to integration and the left-hand side of (4.15) is φ. The right-hand side can be differentiated using (4.16) and (4.17). We reach that the right-hand side of (4.15) is φ by using (4.12).

We have shown that the derivative of the left-hand side and the derivative of the right-hand side of (4.15) are equal.

We now evaluate (4.15) at $\theta = 0$. From the definition of $\mathcal{B}$ we have that $(\mathcal{B}\varphi)(0) = \left(B(\frac{d}{d\theta})\varphi\right)(0)$, i.e., we wish to show that

$$(\mathcal{B}\varphi)(0) = \left(B\left(\frac{d}{d\theta}\right)\varphi\right)(0) = \varphi_+(0) = x_0 \tag{4.18}$$

when $N > 0$. (The relation obviously holds for $N = 0$.) Note that by construction φ_+ is a primitive function of φ. From the relations between f_i, b_i, M_i, B_i, in (2.4) it follows that (4.18) is equivalent to

$$0 = \left(\left(M_1 f_1\left(\frac{d}{d\theta}\right) + \cdots + M_m f_m\left(\frac{d}{d\theta}\right)\right)\varphi_+\right)(0) = \left(M\left(\frac{d}{d\theta}\right)\varphi_+\right)(0). \tag{4.19}$$

We now consider the terms of φ_+ in (4.11) separately. Note that for any analytic function $g : \Omega \to \mathbb{C}$, we have

$$\left(g\left(\frac{d}{d\theta}\right)\exp_N(\theta S)\right)(0) = g_N(S),$$

where g_N is the remainder term in the truncated Taylor expansion, analogous to $\exp_N$. It follows that

$$\left(\left(M_1 f_1\left(\frac{d}{d\theta}\right) + \cdots + M_m f_m\left(\frac{d}{d\theta}\right)\right) Y \exp_N(\theta S)c_+\right)(0) = \mathbb{M}_N(Y,S)c_+. \tag{4.20}$$

For the polynomial part of φ_+ we have

$$\left(M\left(\frac{d}{d\theta}\right)\left((1,\theta,\theta^2,\ldots,\theta^N)\otimes I_n\right)x_+\right)(0) = \sum_{i=0}^{N} M^{(i)}(0)x_{+,i}. \tag{4.21}$$

Note that $(M(\frac{d}{d\theta})\varphi_+)(0)$ is the sum of (4.20). Hence, we have shown (4.19) (and hence also (4.18)) by using (4.20), (4.21), and the definition of $x_{+,0}$ in (4.14). $\square$

4.3. Inner product and finite-dimensional specialization of Algorithm 1. Since the goal is to completely specify all operations in Algorithm 1 in a finite-dimensional setting, we also need to provide an inner product. In [11] we worked with polynomials and we defined the inner product via the Euclidean inner product on monomial or Chebyshev coefficients. The structured functions described in section 4.1 are not polynomials. We can, however, still define the inner product consistent with [11]. In this work we restrict the presentation to the consistent extension of the definition of the inner products via the monomial coefficients. Given two functions

$$\varphi(\theta) = \sum_{j=0}^{\infty}\theta^j x_j,\quad \psi(\theta) = \sum_{j=0}^{\infty}\theta^j z_j,$$

we define

$$\langle\varphi,\psi\rangle := \sum_{i=0}^{\infty} z_i^H x_i. \tag{4.22}$$

It is straightforward to show that (4.22) satisfies the properties of an inner product and that the sum in (4.22) is always finite for functions of the considered structure. The computational details for the inner product and the orthogonalization process are postponed until the next section (section 4.4).

The combination of the above results, i.e., the choice of the representation of the function structure (section 4.1), the operator action (section 4.2), and the inner product (4.22), forms a complete specialization of all the operations in Algorithm 1. For reasons of numerical efficiency, we will slightly modify the direct implementation of the operations.

Instead of representing the individual functions $\varphi_1,\ldots,\varphi_k$ of the basis $(\varphi_1,\ldots,\varphi_k)$ we will use a block representation and denote

$$F_k(\theta) = (\varphi_1,\ldots,\varphi_k).$$

Now note that variables Y and S in the function structure (4.7) are not modified in Theorem 4.2 and obviously not modified when forming linear combinations. Hence, the variables Y and S can be kept constant throughout the algorithm. This allows us to also use the structured representation (4.7) directly for the block function F_k instead of individually for $\varphi_1, \ldots, \varphi_k$. In every point in the algorithm, there exist matrices C_k and V_k such that

$$F_k(\theta) = Y \exp_{N-1}(\theta S)C_k + ((1, \theta, \ldots, \theta^{N-1}) \otimes I)V_k \tag{4.23}$$

with an appropriate choice of N.

The variable N defining the length of the polynomial part of the structure needs to be adapted during the iteration. This stems from the fact that functions φ_+ and φ in Theorem 4.2 are represented with polynomial parts of different length ($N-1$ and N). Hence, we need to increase N by one after each application of $\mathcal{B}$. Fortunately, the corresponding increase of N can be easily achieved by treating the leading element of the exponential part as an element of the polynomial part. Here, this means using the fact that

$$\begin{aligned} F_k(\theta) &= Y \exp_{N-1}(\theta S)C_k + ((1, \theta, \ldots, \theta^{N-1}) \otimes I)V_k \\ &= Y \exp_N(\theta S)C_k + ((1, \theta, \ldots, \theta^N) \otimes I) \begin{pmatrix} V_k \\ \frac{YS^N C_k}{N!} \end{pmatrix}. \end{aligned} \tag{4.24}$$

In this work, the starting function f will be an exponential function, and after the first application of $\mathcal{B}$, we need to expand the polynomial part with one block consisting of $\frac{YS^N C_k}{N!}$ with $N = 0$. Since $k = p_\ell + 1$ at the first application of $\mathcal{B}$, for an iteration corresponding to a given k, we need to expand the polynomial part of F_k with one block row consisting of $\frac{YS^N C_k}{N!}$ with $N = k - p_\ell - 1$.

With the block structure representation (4.23) we can now specialize Algorithm 1 for the structured functions. The finite-dimensional implementation of Algorithm 1 is given in Algorithm 2 and visually illustrated in Figure 1.

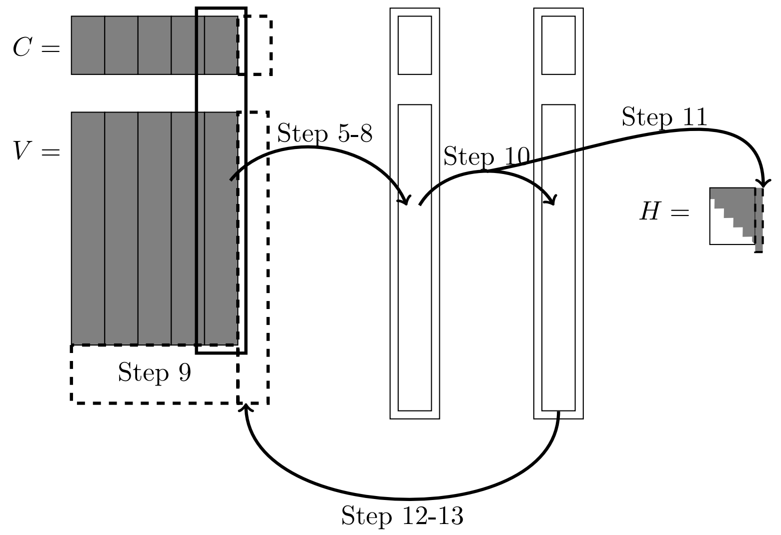

FIG. 1. *Visualization of the infinite Arnoldi method with structured functions (Algorithm 2).*

Algorithm 2. Infinite Arnoldi method with structured functions and locked pair visualized in Figure 1 and equivalent to Algorithm 1.
$[V_k, C, H_k] =$ `infarn_exp`$(c, S, Y, p_\ell, k_{\max})$.

Input: Number of iterations $k_{\max}$, coefficients $Y \in \mathbb{C}^{n\times p}$, $S \in \mathbb{C}^{p\times p}$, $c \in \mathbb{C}^p$, representing the normalized function f given by (4.25) and the locked part of the factorization corresponding to the invariant pair (Ψ, R) with Ψ given by (4.26) and $R \in \mathbb{C}^{p_\ell\times p_\ell}$ given from the structure of S in (4.27). The functions corresponding to the columns of Ψ as well as the function f must be orthogonal.

Output: $V_{k_{\max}+1} \in \mathbb{C}^{(k_{\max}+1)n\times(k_{\max}+1)}$, $C_{k_{\max}+1} \in \mathbb{C}^{p\times(k_{\max}+1)}$, $\underline{H}_{k_{\max}} \in \mathbb{C}^{(k_{\max}+1)\times k_{\max}}$ representing the factorization (4.28)

```
1: Set $H_{p_\ell,p_\ell} = R$
2: Set $C_{p_\ell+1} = \begin{pmatrix} e_1 & \dots & e_{p_\ell} & c\end{pmatrix}$
3: Set $V_{p_\ell+1} =$ empty matrix of size $0 \times (p_\ell + 1)$
4: for $k = p_\ell + 1, \dots, k_{\max}$ do
5:     Compute $c_+$ according to (4.12) where $c = c_k$, i.e., $k$th column of $C_k$
6:     Let $x \in \mathbb{C}^{(k-p_\ell-1)n}$ be the $k$th column of $V_k$
7:     Compute $x_{+,1}, \dots, x_{+,k-p_\ell-1} \in \mathbb{C}^n$ according to (4.13)
8:     Compute $x_{+,0}$ according to (4.14) with $N = k - p_\ell - 1$
9:     Expand $V_k$ with one block row:
           $$\underline{V}_k = \begin{pmatrix} V_k \\ \frac{YS^{k-p_\ell-1}C_k}{(k-p_\ell-1)!}\end{pmatrix}$$
10:    $[c_\perp, x_\perp, h_k, \beta] =$ gram_schmidt$(c_+, x_+, C_k, \underline{V}_k)$
11:    Let $\underline{H}_k = \begin{bmatrix} \underline{H}_{k-1} & h_k \\ 0 & \beta \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}$
12:    Expand $C_k$ by setting $C_{k+1} = (C_k, c_\perp)$
13:    Expand $V_k$ by setting $V_{k+1} = (\underline{V}_k, x_\perp)$
14: end for
```

The input and output of the algorithm should be interpreted as follows. The variables Y, S, c specify the starting function f as well as the locked part of the factorization. The starting function is given by

$$f(\theta) = Y \exp(\theta S)c \tag{4.25}$$

and the locked part of the factorization (in Algorithm 1 denoted (Ψ, R)) corresponds to

$$\Psi(\theta) = Y \exp(\theta S) \begin{pmatrix} I_{p_\ell} \\ 0 \end{pmatrix}, \tag{4.26}$$

where $R \in \mathbb{C}^{p_\ell\times p_\ell}$ is defined as the inverse of the leading block of S. Recall that S is assumed to have the block triangular structure (4.2), i.e.,

$$S = \begin{pmatrix} R^{-1} & S_{12} \\ 0 & S_{22} \end{pmatrix}. \tag{4.27}$$

The output is a finite-dimensional representation of the factorization

$$(\mathcal{B}F_{k_{\max}})(\theta) = F_{k_{\max}+1}(\theta)\underline{H}_{k_{\max}}, \tag{4.28}$$

where the block function $F_{k_{\max}+1}$ is given by

$$F_{k_{\max}+1}(\theta) = Y \exp_{k_{\max}}(\theta S) C_{k_{\max}+1} + ((1, \theta, \dots, \theta^{k_{\max}}) \otimes I_n) V_{k_{\max}+1}, \tag{4.29}$$

and $F_{k_{\max}}$ is the first $k_{\max}$ columns of $F_{k_{\max}+1}$.

4.4. Gram–Schmidt orthogonalization.

4.4.1. Computing the inner product for structured functions. For structured functions, i.e., functions of the form (4.4), the consistent extension of the definition (4.22) is the following. Let

$$\varphi(\theta) = Y \exp_N(\theta S) c + ((1, \theta, \dots, \theta^N) \otimes I_n) x \tag{4.30}$$

and

$$\psi(\theta) = Y \exp_N(\theta S) d + ((1, \theta, \dots, \theta^N) \otimes I_n) z. \tag{4.31}$$

Then, (4.22) reduces to

$$\langle \varphi, \psi \rangle = \sum_{i=0}^{N} z_i^H x_i + \sum_{i=N+1}^{\infty} \frac{d^H (S^i)^H Y^H Y S^i c}{(i!)^2}. \tag{4.32}$$

In practice, we can compute the inner product by truncating the infinite sum and exploiting the structure of the sum.

LEMMA 4.3 (computation of inner product). *Suppose the two functions $\varphi : \mathbb{C} \to \mathbb{C}^n$ and $\psi : \mathbb{C} \to \mathbb{C}^n$ are given by* (4.30) *and* (4.31)*. Then*

$$\langle \varphi, \psi \rangle = \sum_{i=0}^{N} z_i^H x_i + d^H W_{N+1, N_{\max}} c + \varepsilon_{N_{\max}} \tag{4.33}$$

with

$$W_{N,M} = \sum_{i=N}^{M} \frac{(S^i)^H Y^H Y S^i}{(i!)^2} \tag{4.34}$$

provides an approximation to accuracy

$$|\varepsilon_{N_{\max}}| \le \|d\|_2 \|Y^H Y\|_2 \|c\|_2 \frac{e^{2\|S\|_2} \|S\|_2^{2(N_{\max}+1)}}{((N_{\max}+1)!)^2}. \tag{4.35}$$

Proof. By comparing the infinite sum (4.32) with (4.33), we can solve for $\varepsilon_{N_{\max}}$ and bound the modulus,

$$|\varepsilon_{N_{\max}}| \le \|d\|_2 \|Y^H Y\|_2 \|c\|_2 \sum_{i=N_{\max}+1}^{\infty} \frac{\|S\|_2^{2i}}{(i!)^2} \le \|d\|_2 \|Y^H Y\|_2 \|c\|_2 \left(\sum_{i=N_{\max}+1}^{\infty} \frac{\|S\|_2^{i}}{i!} \right)^2.$$

The sum in the right-hand side can be interpreted as the remainder term in the Taylor approximation of $\exp(\|S\|_2)$. The bound (4.35) follows by applying Taylor's theorem. ∎

The lemma above has some properties important from a computational perspective:

- The sum in (4.34) involves only matrices of size $p \times p$, i.e., it does not involve very large matrices, under the condition that $Y^H Y$ is precomputed.
- The matrix $W_{N,M}$ defined by (4.34) is constant if S and Y are constant. Hence, in combination with Algorithm 2 it only needs to be computed once in order to construct the Arnoldi factorization.
- An appropriate value of $N_{\max}$ such that $\varepsilon_{N_{\max}}$ is smaller than or comparable to machine precision can be computed from $\|S\|$ by increasing $N_{\max}$ until the right-hand side of (4.35) is sufficiently small.

4.4.2. Computing the Gram–Schmidt orthogonalization for structured functions. One step of the Gram–Schmidt orthogonalization process can be seen as a way of computing the *orthogonal complement*, followed by normalizing the result. When working with matrices, the process is compactly expressed as follows. Consider an orthogonal matrix $X \in \mathbb{C}^{n\times k}$. The *orthogonal complement* of a vector $u \in \mathbb{C}$ with respect to the space spanned by the columns of X and the Euclidean inner product is given by

$$u_\perp = u - Xh, \tag{4.36}$$

where

$$h = X^H u. \tag{4.37}$$

In the setting of Arnoldi's method, the orthogonalization coefficients h and the norm of the orthogonal complement β needs to be returned to the Arnoldi algorithm.

Due to the fact that the considered inner product (4.32) is the Euclidean inner product on the Taylor coefficients, we can, similar to (4.36) and (4.37), compute the orthogonal complement using matrices. The corresponding operations for our setting are presented in the following theorem.

THEOREM 4.4 (orthogonal complement). *Let $Y \in \mathbb{C}^{n\times p}$, $S \in \mathbb{C}^{p\times p}$, $C \in \mathbb{C}^{p\times k}$, $V \in \mathbb{C}^{n(N+1)\times k}$ be the matrices representing the block function $F : \mathbb{C} \to \mathbb{C}^{n\times k}$,*

$$F(\theta) = Y \exp_N(\theta S)C + ((1, \theta, \ldots, \theta^N) \otimes I_n)V,$$

where the columns are orthonormal with respect to $\langle\cdot,\cdot\rangle$ defined by (4.32). Consider the function φ, represented by $c_+ \in \mathbb{C}^p$ and $x_+ \in \mathbb{C}^{n(N+1)}$ and defined by

$$\varphi(\theta) = Y \exp_N(\theta S)c_+ + ((1, \theta, \ldots, \theta^N) \otimes I_n)x_+,$$

and let $h \in \mathbb{C}^k$,

$$h := V^H x_+ + C^H \left(\sum_{i=N+1}^{\infty} \frac{(S^i)^H Y^H Y S}{(i!)^2} \right) c_+.$$

Then, the function $\varphi_\perp$, represented by the vectors

$$c_\perp = c_+ - Ch \in \mathbb{C}^k, \quad x_\perp = x_+ - Vh \in \mathbb{C}^{n(N+1)},$$

and defined by

$$\varphi_\perp(\theta) := Y \exp_N(\theta S)c_\perp + ((1, \theta, \ldots, \theta^N) \otimes I_n)x_\perp,$$

is the orthogonal complement of φ with respect to the space spanned by the columns of F and the inner product $\langle\cdot,\cdot\rangle$ defined by (4.32).

ALGORITHM 3. Gram–Schmidt orthogonalization for the inner product (4.32).
$[c_\perp, x_\perp, h, \beta] = \texttt{gram\_schmidt}(c, x, C, V)$.

Input: Vectors $c \in \mathbb{C}^n$, $x \in \mathbb{C}^{(N+1)n}$ representing the function

$$\varphi(\theta) := Y \exp_N(\theta S)c + ((1, \theta, \cdots, \theta^N) \otimes I_n)x$$

and $C \in \mathbb{C}^{n\times k}$, $V \in \mathbb{C}^{(N+1)n\times k}$, representing the block function, $F : \mathbb{C} \to \mathbb{C}^{n\times k}$,

$$F(\theta) = Y \exp_N(\theta S)C + ((1, \theta, \cdots, \theta^N) \otimes I_n)V,$$

whose columns are orthogonal with respect to $\langle\cdot,\cdot\rangle$ defined by (4.32).

Output: Orthogonalization coefficients $h \in \mathbb{C}^k$, $\beta \in \mathbb{C}$ and vectors $c_\perp \in \mathbb{C}^{pn}$ and $x_\perp \in \mathbb{C}^{(k+1)n}$ representing the normalized orthogonal complement of φ,

$$\varphi_\perp(\theta) := Y \exp_N(\theta S)c_\perp + ((1, \theta, \cdots, \theta^N) \otimes I_n)x_\perp.$$

```
1:  $h = V^H x + C^H(W_{N+1,N_{\max}} c)$, where $W_{N+1,N_{\max}}$ is given by (4.34)
2:  $c_\perp = c - Ch$
3:  $x_\perp = x - Vh$
4:  $g = V^H x_\perp + C^H(W_{N+1,N_{\max}} c\perp)$
5:  if $\|g\|$ >REORTH_TOL then
6:      $c_\perp = c_\perp - Cg$
7:      $x_\perp = x_\perp - Vg$
8:      $h = h + g$
9:  end if
10: $\beta = x_\perp^H x_\perp + c_\perp^H(W_{N+1,N_{\max}} c_\perp)$
11: $c_\perp = c_\perp/\beta$
12: $x_\perp = x_\perp/\beta$
```

Proof. The construction is such that $\varphi_\perp$ is a linear combination of φ and the columns of F (due to linearity in coefficients c and x). It remains to check that $\varphi_\perp$ is orthogonal to columns of F. ∎

The Gram–Schmidt process with reorthogonalization hence can be efficiently implemented with operations on matrices and vectors. This is presented in Algorithm 3, where we used iterative reorthogonalization [6] with (as usual) at most two steps. In the numerical simulations we used REORTH_TOL= $\sqrt{\varepsilon_{\text{mach}}}$. In the description of the algorithm we have for simplicity ignored the breakdown that can occur if $\beta \approx 0$. Similar to Arnoldi's method for standard eigenvalue problems, such a situation implies that the Krylov subspace is an invariant subspace.

5. Extracting and restarting.

Recall the general outline described in section 3 and recall that we have (in the section 4) described the first step in detail. In what follows we discuss the second step. We propose a procedure to carry out some operations of the result of the first step, i.e., Algorithm 2, and restart it such that we expect that the outer iteration eventually converges to a partial Schur factorization.

5.1. Manipulations of the Arnoldi factorization.

First recall that Algorithm 2 is an Arnoldi method in a function setting and the output corresponds to an Arnoldi factorization,

$$(\mathcal{B}F_k)(\theta) = F_{k+1}(\theta)\underline{H}_k, \tag{5.1}$$

where the block function F_{k+1} is given by the output of Algorithm 2 with the definition

$$F_{k+1}(\theta) = Y \exp_k(\theta S)C_{k+1} + ((1, \theta, \ldots, \theta^k) \otimes I_n)V_{k+1}, \tag{5.2}$$

and F_k is the first k columns of F_{k+1}. To ease the notation, we have denoted $k = k_{\max}$.

Although the Arnoldi factorization (5.1) is a function relation, we will now see that several parts of the steps for implicit restarting (cf. [23, 16, 17, 27]) for Arnoldi's method (for linear matrix eigenvalue problems) can be carried out in a similar way by working with functions.

We will start by computing an ordered Schur factorization of H_k,

$$Q^* H_k Q = (Q_1, Q_2, Q_3)^* H_k (Q_1, Q_2, Q_3) = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ & R_{22} & R_{23} \\ & & R_{33} \end{pmatrix}, \tag{5.3}$$

where $R_{11} \in \mathbb{C}^{p_\ell \times p_\ell}$, $R_{22} \in \mathbb{C}^{(p-p_\ell)\times(p-p_\ell)}$, and $R_{33} \in \mathbb{C}^{(k-p)\times(k-p)}$ are upper triangular matrices. The ordering is such that the eigenvalues of R_{11} are very accurate (and from now on called the locked Ritz values), the eigenvalues of R_{22} are wanted eigenvalues (selected according to some criteria) which have not converged, and the eigenvalues of R_{33} are unwanted.

Hence,

$$\begin{pmatrix} Q^* & \\ & 1 \end{pmatrix} \underline{H}_k (Q_1, Q_2, Q_3) = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ & R_{22} & R_{23} \\ & & R_{33} \\ a_1^T & a_2^T & a_3^T \end{pmatrix}. \tag{5.4}$$

Note that a_1 is a measure of the (unstructured) backward error of the corresponding eigenvalues of R_{11} and $\|a_1\|$ is often used as stopping criteria. Hence, $\|a_1\|$ will be zero if the eigenvalues of R_{11} are exact and will in general be small (or very small) relative to $\underline{H}_k$ since the eigenvalues of R_{11} are very accurate solutions. By successive application of Householder reflections (see, e.g., [21]) we can now construct an orthogonal matrix P_2 such that

$$\begin{pmatrix} I_{p_\ell} & & \\ & P_2^* & \\ & & 1 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \\ a_1^T & a_2^T \end{pmatrix} \begin{pmatrix} I_{p_\ell} & \\ & P_2 \end{pmatrix} = \begin{pmatrix} R_{11} & M \\ 0 & \hat{H} \\ a_1^T & e_{p-p_\ell}^T \beta \end{pmatrix}, \tag{5.5}$$

where

$$\underline{\hat{H}} := \begin{pmatrix} \hat{H} \\ e_{p-p_\ell}^T \beta \end{pmatrix}$$

is a Hessenberg matrix.

By considering the leading two blocks and columns of (5.4) and the result of the Householder reflection transformation (5.5) we find that

(5.6)
$$\begin{pmatrix}(Q_1,Q_2P_2)^* & \\ & 1\end{pmatrix}\underline{H}_k(Q_1,Q_2P_2) \quad = \begin{pmatrix}R_{11} & Z\\ 0 & \hat{H}\\ a_1^T & e_{p-p_\ell}^T\beta\end{pmatrix} = \begin{pmatrix}R_{11} & Z\\ 0 & \underline{\hat{H}}\end{pmatrix} + O(\|a_1\|).$$

These operations yield a transformation of the Arnoldi factorization where the first block is triangular (to order $O(\|a_1\|)$). We reach the following result, which is an Arnoldi factorization similar to (5.1) but only of length p. Moreover, the Hessenberg matrix does not contain the unwanted eigenvalues and has a leading block which is almost triangular.

THEOREM 5.1. *Consider an Arnoldi factorization given by* (5.1) *and let* $F_{k+1}(\theta) = (F_k(\theta), f(\theta))$. *Let* Q_1 *and* Q_2 *represent the leading blocks in the ordered Schur decomposition* (5.3) *and let* P_2, R_{11}, *and* $\underline{\hat{H}}$ *be the result of the Householder reflections in* (5.6). *Moreover, let*

$$G_p(\theta) := F_k(\theta)(Q_1, Q_2P_2),\ \ G_{p+1}(\theta) := (G_p(\theta), f(\theta)). \tag{5.7}$$

Then, G_{p+1} *approximately satisfies the length* $p < k$ *Arnoldi factorization*

$$(\mathcal{B}G_p)(\theta) = G_{p+1}(\theta)\begin{pmatrix}R_{11} & Z\\ 0 & \underline{\hat{H}}\end{pmatrix} + O(\|a_1\|). \tag{5.8}$$

5.2. Extraction and imposing structure. Restarting in standard IRAM for matrices essentially consists of assigning the algorithmic state of the Arnoldi method to that corresponding to the factorization in Theorem 5.1. The direct adaption of this procedure is not suitable in our setting due to a growth of the polynomial part of the structured functions. This can be seen as follows. Suppose we start Algorithm 1 with a constant function (as done in [11]) and carry out the construction of G_p as in Theorem 5.1. Then, G_{p+1} will be a matrix with polynomials of degree k. We hence need to start with a state consisting of polynomials of degree k. The degree of the polynomial will grow with each restart and after M restarts, the polynomials will be of degree Mk. The representation of this polynomial will hence quickly limit the efficiency of the restarting scheme.

Instead of restarting with polynomials we will perform an explicit restart using Algorithm 2 with a particular choice of the input which we here denote $\hat{Y}$, $\hat{S}$, $\hat{c}$. This choice is *inspired* by the factorization in Theorem 5.1.

We will first impose exponential structure on G_p in the sense that we consider a function $\hat{G}_p$, with the property

$$G_p(0) = \hat{G}_p(0)$$

and defined by

$$\hat{G}_p(\theta) := G_p(0)\exp(\hat{S}\theta), \tag{5.9}$$

where

$$\hat{S} = \begin{pmatrix}R_{11} & Z\\ 0 & \hat{H}\end{pmatrix}^{-1}. \tag{5.10}$$

Note that we can express $G_p(0)$ explicitly from (5.7) as

$$G_p(0) = (V_{k+1,1}Q_1,\ V_{k+1,1}Q_2P_2) =: \hat{Y}. \tag{5.11}$$

where $V_{k+1,1}$ is the upper $n \times (k+1)$-block of V_k.

Assume for the moment that $\|a_1\| = 0$. Then, the first p_ℓ columns of (5.8) correspond to the definition of an invariant pair (Ψ, R), where $\Psi(\theta) = G_{p_\ell}(\theta)$ and $R = R_{11}$. From Theorem 2.2 we know that Ψ is of exponential structure, and imposing the structure as in (5.9) does not modify the function, i.e., if $\|a_1\| = 0$, then $\hat{G}_{p_\ell}(\theta) = G_{p_\ell}(\theta)$. Hence, the first p_ℓ columns of (5.8) are preserved also if we replace $G_p(\theta)$ with $\hat{G}_p(\theta)$. Due to the fact that $\|a_1\|$ is small (or very small) we expect that imposing the structure as in (5.9) gives an approximation of the p_ℓ columns of (5.8), i.e.,

$$(\mathcal{B}\hat{G}_{p_\ell})(\theta) \approx \hat{G}_{p_\ell+1}(\theta)\begin{pmatrix} R_{11} \\ 0 \end{pmatrix}, \tag{5.12}$$

if $\|a_1\|$ is small and equality is achieved if $\|a_1\| = 0$.

Algorithm 4. Structured explicit restarting with locking.
$[S, Y] =$ `infarn_restart` $(x_0, \lambda_0, k_{\max}, p)$.

Input: $x_0 \in \mathbb{C}^n$, λ_0 representing the function

$$f(\theta) = \exp(\lambda_0\theta)x_0,$$

maximum size of subspace $k_{\max}$, number of wanted eigenvalues p

Output: S, Y such that (Y, S) represents an invariant pair

```
1:  Normalize $f$ by setting $x_0 = \frac{1}{\|x_0\|\sqrt{W_{0,N_{\max}}}}x_0$, where $W_{0,N_{\max}}$ is given by (4.34) with $S = \lambda_0$
2:  Set $Y_0 = (x_0, 0, \dots, 0) \in \mathbb{C}^{n\times p}$
3:  Set $S = \mathrm{diag}(\lambda_0, 1, \dots, 1) \in \mathbb{C}^{p\times p}$
4:  Set $c = e_1 \in \mathbb{C}^p$, $p_\ell = 0$
5:  while $p_\ell < p$ do
6:     $[V, C, \underline{H}_{k_{\max}}] =$infarn_exp$(c, S_j, Y_j, p_\ell, k_{\max})$
7:     For every eigenvalue of $H_{k_{\max}}$ classify it as, lock, wanted or unwanted, and let $p_\ell$ denote the number of locked eigenvalues
8:     Compute ordered Schur factorization of $H_{k_{\max}}$ partitioned according to (5.3)
9:     Compute the $a_2$ vector in (5.4)
10:    Compute the orthogonal matrix $P_2$ according to (5.5)
11:    Compute $Z$ and $\hat{H}$ from (5.6)
12:    Set $Y_{j+1} = \hat{Y}$ and $S_{j+1} = \hat{S}$ according to (5.11) and (5.10)
13:    Reorthogonalize the function $F(\theta) = Y_{j+1}\exp(\theta S_{j+1})(e_1, \dots, e_{p_\ell})$
14:    $[c, \cdot, \cdot, \cdot] =$gram_schmidt$(e_{p_\ell+1}, \cdot, C_{j+1}, \cdot)$
15:    Set $j = j + 1$
16: end while
```

With the above reasoning we have a justification to use the first p_ℓ columns of (5.9), i.e.,

$$\hat{Y}\exp(\hat{S}\theta)\begin{pmatrix} I_{p_\ell} \\ 0 \end{pmatrix},$$

in the initial state for the restart. In the approximation of the $p-p_\ell$ last columns of G_p by the $p-p_\ell$ last columns of (5.9), the Arnoldi relation in the function setting is in general lost, because Ritz functions only have exponential structure upon convergence. Therefore, we will only use the $(p_\ell+1)$st column in the restart, from which the Krylov space will be extended again in the next inner iteration. This leads us to a restart with the function

$$\hat{Y}\exp(\hat{S}\theta)\begin{pmatrix} I_{p_\ell+1} \\ 0 \end{pmatrix},$$

which corresponds to setting $\hat{c} = e_{p_\ell+1}$ and the initial function

$$f(\theta) = \hat{Y}\exp(\hat{S}\theta)e_{p_\ell+1}.$$

By these modifications of the factorization (5.8) we have now reached a choice of $\hat{Y}$ given by (5.11), $\hat{S}$ given by (5.10), and $\hat{c} = e_{p_\ell+1}$. This choice of variables satisfies all the properties necessary for the input of Algorithm 2, except the orthogonality condition. The first columns of $\hat{G}_{p_\ell}$ are automatically orthogonal (at least if $\|a_1\| = 0$). The $(p_\ell+1)$st column, however, in general will not be orthogonal to $\hat{G}_{p_\ell}$, which is an assumption needed for Algorithm 2. It is fortunately easily remedied by orthogonalizing the function corresponding to $\hat{c} = e_{p_\ell+1}$ using the function `gram_schmidt`, i.e., Algorithm 3.

The details of this selection as well as the manipulations in section 5.1 are summarized in the outer iteration Algorithm 4.

Remark 5.2 (explicit restart without locking). Note that a restart which is theoretically very similar to what we have proposed here can be achieved by starting the infinite Arnoldi method with the function of the first column of (5.9), without taking the "locked part" of the factorization directly into account. Such an explicit restarting technique (without locking) unfortunately does have unfavorable numerical properties and will not be persued here. From reasoning similar to [23] we know that the first column of (5.9) is an approximation of an element of an invariant subspace and the first p_ℓ steps of Arnoldi's method started with this vector are expected to recompute the p_ℓ converged Ritz vectors after p_ℓ iterations. In the $(p_\ell+1)$st iteration, the Arnoldi vector is corrupted due to cancellation.

6. Examples.

6.1. A small example of Hadeler.

The NEP presented in [9], which is available as `hadeler` in the problem collection [4], is given by

$$M(\lambda) = -A_0 + (\lambda+\mu)^2 A_1 + (e^{\lambda+\mu}-1)A_2,$$

where $A_i \in \mathbb{R}^{n\times n}$, $i = 0,\ldots,2$, with $n = 8$ and μ is a shift which we will use to select a point close to which we will find the eigenvalues.

In order to apply Algorithm 2 we need to derive a formula for $x_{+,0}$ in (4.14). The derivatives for M are straightforward to compute and we compute $\mathbb{M}_N$, using (4.8) and (4.9). More precisely, we use the following computational expressions:

TABLE 1
The indicator value and number of locked Ritz values for the two runs of the example of Hadeler in section 6.1. Run 1 corresponds to Figure 2 and Run 2 corresponds to Figure 3. The outer iteration count represents the number of loops carried out in Algorithm 4.

	Run 1		Run 2	
Outer iteration	p_ℓ	γ	p_ℓ	γ
1	0		0	0
2	1	1.0×10^{-15}	0	0
3	2	5.7×10^{-14}	3	6.4×10^{-14}
4	2	5.7×10^{-14}	3	6.4×10^{-14}
5	3	5.8×10^{-14}	3	1.4×10^{-14}
6	3	5.8×10^{-14}	4	1.4×10^{-14}
7	4	7.3×10^{-13}	5	1.4×10^{-14}
8	10	2.3×10^{-13}		

$$\begin{aligned}
\mathbb{M}_{-1}(Y,S)c_+ &= -A_0Yc_+ + A_1Y(S+\mu I)^2c_+ + A_2Y(\exp(S+\mu I)-I)c_+,\\
\mathbb{M}_0(Y,S)c_+ &= A_1Y(S^2+2\mu S)c_+ + e^\mu A_2Y(\exp(S)-I)c_+,\\
\mathbb{M}_1(Y,S)c_+ &= A_1Y(S^2c_+) + e^\mu A_2Y(\exp(S)-I-S)c_+,\\
\mathbb{M}_N(Y,S)c_+ &\approx e^\mu A_2Y\left(\sum_{i=N+1}^{i_{\max}}\frac{S^ic_+}{i!}\right), \quad N>1.
\end{aligned}$$

In the last formula, $i_{\max}$ is chosen such that the expression has converged to machine precision. Since this is not computationally expensive, we can roughly overestimate $i_{\max}$. In this example it was sufficient to take $i_{\max} = 40$.

In the outer algorithm (Algorithm 4) we classified a Ritz value as converged (locked) when the absolute residual was smaller than $1000 \times \varepsilon_{\text{mach}}$. We selected the largest eigenvalues of H_k as the wanted eigenvalues.

The convergence is illustrated for two runs in Figures 2 and 3. In order to illustrate the similarity with implicit restarting in [26], we also carried out the infinite Arnoldi method with true implicit restarting by restarting only with polynomials, instead of using Algorithm 4. We clearly see that at least in the beginning of the iteration, the convergence of Algorithm 4 is similar to the convergence of IRAM. Note that IRAM in this setting exhibits a growth of the basis matrix and it is hence considerably slower. We show the number of locked Ritz values in Table 1. Moreover, we quantify the impact of the procedure to impose the structure in the restart by inspecting the approximation in (5.12). We define γ as the norm of the difference of the left-hand side and right-hand side of (5.12). Lemma A.1 shows that this difference is independent of θ and provides a computable expression. More precisely,

$$\begin{aligned}
\gamma &:= \left\|(\mathcal{B}\hat{G}_{p_\ell})(\theta) - \hat{G}_{p_\ell+1}(\theta)\begin{pmatrix}R_{11}\\0\end{pmatrix}\right\|_2 \\
&= \left\|(\mathcal{B}\hat{G}_{p_\ell})(\theta) - \hat{G}_{p_\ell}(\theta)R_{11}\right\|_2 = \|M(0)^{-1}\mathbb{M}(Y,S)S^{-1}\|_2,
\end{aligned} \tag{6.1}$$

where we used that $\hat{G}_{p_\ell}$ has the structure $\hat{G}_{p_\ell}(\theta) = Y\exp(\theta R_{11}^{-1})$. The values of γ are also given in Table 1. They are, as expected, of the same order of magnitude as the locking tolerance. The solution and the computed approximations are given in Figure 4.

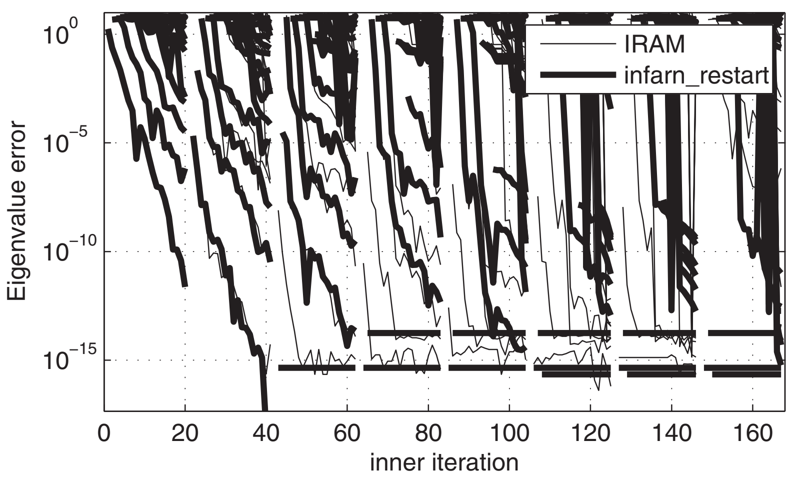

FIG. 2. *Convergence of Algorithm* 4 *(thick) and IRAM* [26] *(thin) for the Hadeler example in section* 6.1 *(*$k_{\max} = 20$*,* $p = 10$*,* $\mu = -1$*).*

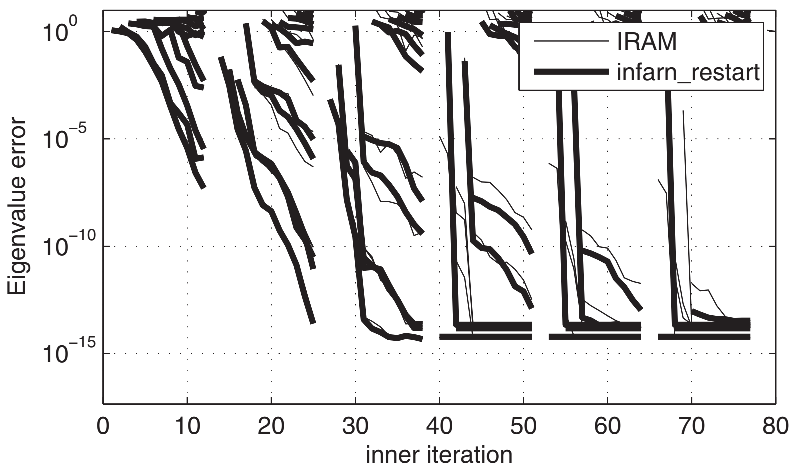

FIG. 3. *Convergence of Algorithm* 4 *(thick) and IRAM* [26] *(thin) for the Hadeler example in section* 6.1 *(*$k_{\max} = 12$*,* $p = 5$*,* $\mu = 3 + 5\iota$*).*

6.2. A large-scale square root example. We considered the same example as in [11, section 7.2], which is the problem called gun in the problem collection [4] and stems from [18]. It is currently the largest example, among those examples in the collection [4], which are neither PEPs nor REPs. In [18] the problem was solved using Newton-like methods with carefully selected starting values close to the solutions. In our approach we will only select a shift, which specifies a general region of interest and is not necessarily an accurate approximation of an eigenvalue. We introduce (as in [11]) a shift μ and a scaling γ, for which the NEP is

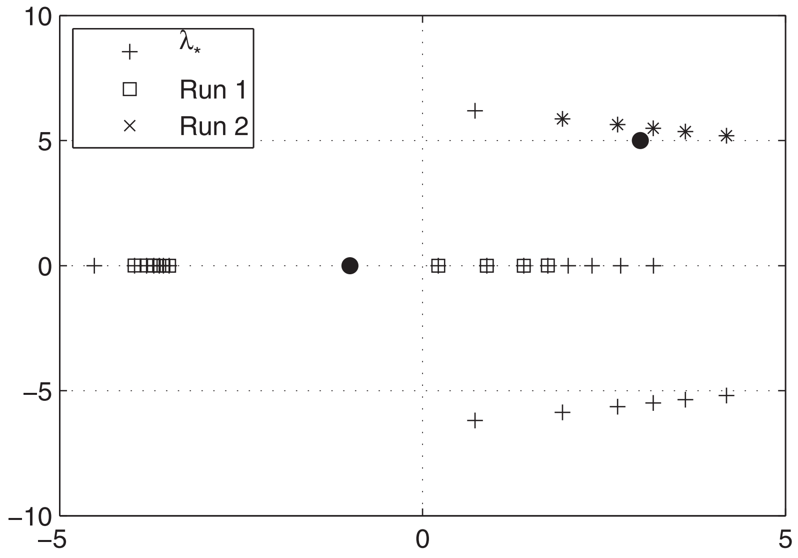

FIG. 4. *Computed eigenvalues and shifts for the Hadeler example. The shifts are marked with bullets* ●.

$$M(\lambda) = A_0 - (\gamma\lambda + \mu)A_1 + \iota\sqrt{\gamma\lambda + \mu - \sigma_1^2}A_2 + \iota\sqrt{\gamma\lambda + \mu - \sigma_2^2}A_3,$$

where $\sigma_1 = 0$ and $\sigma_2 = 108.8774$ and $\iota^2 = -1$. We selected $\gamma = 300^2 - 200^2$ and $\mu = 250^2$ since this transforms the region of interest to being essentially within the unit circle.

In order to compute a formula for $x_{+,0}$ in (4.14), we need in particular

$$\begin{aligned}\mathbb{M}(Y, S) = {} & A_0Y - A_1Y(\gamma S + \mu I_p)\\ & + \iota A_2Y\sqrt{\gamma S + (\mu - \sigma_1^2)I_p} + \iota A_3Y\sqrt{\gamma S + (\mu - \sigma_2^2)I_p},\end{aligned}$$

where $\sqrt{Z}$ denotes the matrix square root (principal branch).

We will partially base the formulas on the Taylor coefficients of the square root in order to compute $\mathbb{M}_N$ (needed in the computation of $x_{0,+}$ in (4.14)). We will use

$$\sqrt{\gamma\lambda + \mu - \sigma_j^2} = \alpha_{0,j} + \alpha_{1,j}\lambda + \alpha_{2,j}\lambda^2 + \cdots,$$

where

$$\alpha_{0,j} = \sqrt{\mu - \sigma_j^2}, \tag{6.2a}$$

$$\alpha_{k,j} = \left(\frac{\gamma}{2}\right)\left(-\frac{\gamma}{2}\right)\left(-\frac{3\gamma}{2}\right)\cdots\left(-\frac{(2k-3)\gamma}{2}\right)(\mu - \sigma_j^2)^{1/2-k}, \quad k > 0. \tag{6.2b}$$

TABLE 2

Consumption of memory resources and profiling times for some choices of the restart parameter $k_{\max}$ *and* $p = 10$. *Memory in megabytes (MB) and CPU time in seconds.*

	$k_{\max} = 50$	$k_{\max} = 30$	$k_{\max} = 25$
No. of restarts	0	1	3
Total CPU	35.7s	21.0s	23.7s
LU decomp.	2.1s	2.1s	2.1s
`gram_schmidt`	23.7s	12.9s	15.1s
Computing x_+	6.8s	6.9s	3.4s
Memory usage	$\sim$ 200 MB	$\sim$ 78 MB	$\sim$ 58 MB

This can be used to compute $\mathbb{M}_N$, as follows:

$$\mathbb{M}_0(Y,S)c_+ = \mathbb{M}(Y,S)c_+ - M(0)Yc_+, \tag{6.3a}$$

$$\mathbb{M}_1(Y,S)c_+ = \mathbb{M}_0(Y,S)c_+ - M'(0)Y(Sc_+), \tag{6.3b}$$

$$\begin{aligned}\mathbb{M}_N(Y,S)c_+ &= \sum_{i=N+1}^{\infty} \frac{1}{i!} M^{(i)}(0) Y S^i c_+ \\ &\approx \iota A_2 \left(Y \sum_{i=N+1}^{i_{\max}} \alpha_{i,1} S^i c_+ \right) \\ &\quad + \iota A_3 \left(Y \sum_{i=N+1}^{i_{\max}} \alpha_{i,2} S^i c_+ \right), \quad N > 1.\end{aligned} \tag{6.3c}$$

Note that the sums in (6.3c) are operations with vectors of relatively small dimension and can be computed efficiently. We selected the number of terms $i_{\max}$ adaptively such that $\|S^{i_{\max}} c_+\| |\alpha_{i_{\max},k}| \ll \varepsilon_{\text{mach}}$.

This results in the next formulas, which we used for the computation of $x_{+,0}$,

$$x_{+,0} = -M(0)^{-1}(\mathbb{M}_0(Y,S)c_+) \text{ for } N = 0,$$
$$x_{+,0} = -M(0)^{-1}(\mathbb{M}_1(Y,S)c_+ + M'(0)x_{+,1}) \text{ for } N = 1,$$

and for $N > 1$,

$$x_{+,0} = -M(0)^{-1} \left(\mathbb{M}_N(Y,S)c_+ + \iota A_2 \sum_{j=1}^{N} x_{+,j}(\alpha_{j,1}(j!)) + \iota A_3 \sum_{j=1}^{N} x_{+,j}(\alpha_{j,2}(j!)) \right).$$

The matrix $M(0)$ was factorized (with an LU-factorization) before starting the iteration, such that $M(0)^{-1}b$ could be computed efficiently.

We first wish to illustrate that the restarting and structure exploitation can considerably reduce both memory and CPU usage. In Table 2 we compare runs for the standard version of the infinite Arnoldi method [11] (first column) with the restarting algorithm (Algorithm 4) for two choices of the parameter $k_{\max}$. The iteration was terminated when $p = 10$ eigenvalues were found. We clearly see that for the choices of $k_{\max}$ there is a considerable reduction in memory and some reduction in computation time.

In Figures 5 and 6 we illustrate that the algorithm scales reasonably well with p, i.e., the number of wanted eigenvalues. When we increase p, we need more outer iterations, but eventually the algorithm usually converges for reasonably large p.

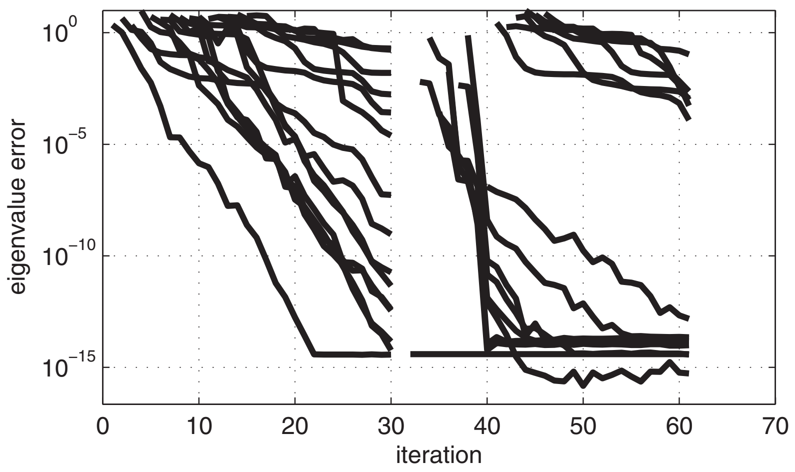

FIG. 5. *Convergence history for Algorithm* 4 *with the example involving a square root in section* 6.2 *($p = 9$).*

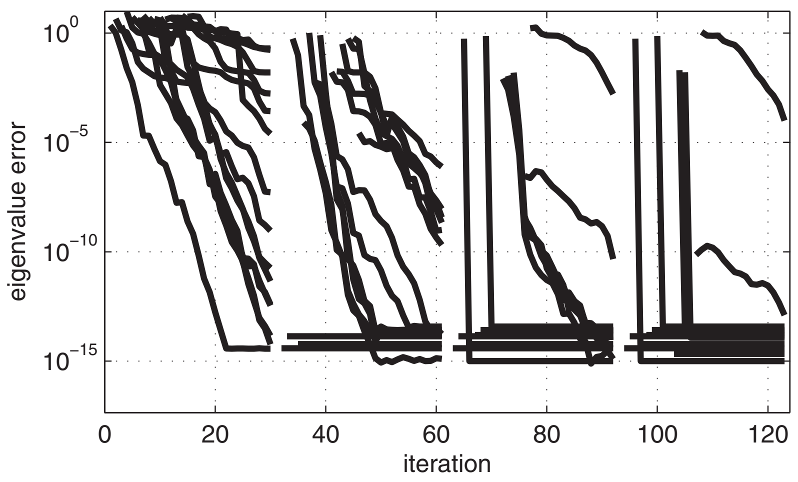

FIG. 6. *Convergence history for Algorithm* 4 *with the example involving a square root in section* 6.2 *($p = 14$).*

7. Concluding remarks. We have in this work shown how the partial Schur factorization of an operator $\mathcal{B}$ can be computed using a variation of the procedures to compute partial Schur factorization for matrices. Several variations of the results for matrices appear to be possible to adapt. Concepts like thick restarting, purging, and other selection strategies appear to carry over but deserve further attention. In this paper we used an inner product based on the Euclidean inner product for the monomial coefficients, which could essentially be replaced by any suitable inner product.

Appendix A. A technical lemma.

LEMMA A.1. *Consider $Y \in \mathbb{C}^{n\times p}$ and $S \in \mathbb{C}^{p\times p}$, where S is invertible. Let $F(\theta) := Y\exp(\theta S)$. Then,*

$$(\mathcal{B}F)(\theta) - F(\theta)S^{-1} = -M(0)^{-1}\mathbb{M}(Y,S)S^{-1}. \tag{A.1}$$

Proof. We prove the theorem by showing that the derivative of the function relation (A.1) holds for any θ and that the relation holds in one point $\theta = 0$. Note that the right-hand side of (A.1) is constant (with respect to θ) and the derivative of the left-hand side reduces to

$$F(\theta) - F'(\theta)S^{-1} = F(\theta) - Y\exp(\theta S)SS^{-1} = 0$$

by definition of $\mathcal{B}$ and differentiation of $\exp(\theta S)$.

From the definition of $\mathcal{B}$ and evaluation of the left-hand side of (A.1) at $\theta = 0$ we have

$$(\mathcal{B}F)(0) - F(0)S^{-1} = \left(B\left(\frac{d}{d\theta}\right)Y\exp(\theta S)\right)(0) - YS^{-1}. \tag{A.2}$$

Note that for an analytic scalar function $b : \mathbb{C} \to \mathbb{C}$,

$$\left(b\left(\frac{d}{d\theta}\right)\exp(\theta S)\right)(0) = b(S).$$

Hence,

$$\begin{aligned}&\left(B\left(\frac{d}{d\theta}\right)Y\exp(\theta S)\right)(0)\\ &\quad= B_1Y\left(b_1\left(\frac{d}{d\theta}\right)\exp(\theta S)\right)(0) + \cdots + B_mY\left(b_m\left(\frac{d}{d\theta}\right)\exp(\theta S)\right)(0)\\ &\quad= B_1Yb_1(S) + \cdots + B_mYb_m(S).\end{aligned}$$

Moreover, by using the relation between b_i and f_i and M_i and B_i given by (2.4) we have

$$\begin{aligned}&B_1Yb_1(S) + \cdots + B_mYb_m(S)\\ &\quad= M(0)^{-1}\left[M_1Y(f_1(0)I - f_1(S))S^{-1} + \cdots + M_mY(f_m(0)I - f_m(S))S^{-1}\right]\\ &\quad= M(0)^{-1}\left[\mathbb{M}(Y,0)S^{-1} - \mathbb{M}(Y,S)S^{-1}\right] = YS^{-1} - M(0)^{-1}\mathbb{M}(Y,S)S^{-1}.\end{aligned} \tag{A.3}$$

The proof is completed by canceling the term YS^{-1} when inserting (A.3) into (A.2). ▯

REFERENCES

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. A. van der Vorst, eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, Software Environ. Tools 11, SIAM, Philadelphia, 2000.

[2] Z. Bai and Y. Su, *SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 640–659.

[3] Z. Battles and L. N. Trefethen, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., 25 (2004), pp. 1743–1770.

[4] T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur, *NLEVP: A collection of nonlinear eigenvalue problems*, ACM Trans. Math. Software, 39 (2013) pp. 1–28.

[5] T. Betcke and H. Voss, *A Jacobi-Davidson type projection method for nonlinear eigenvalue problems*, Future Generation Comput. Syst., 20 (2004), pp. 363–372.

[6] Å. Björck, *Numerics of Gram-Schmidt orthogonalization*, Linear Algebra Appl., 179 (1994), pp. 297–316.

[7] H. Fassbender, D. Mackey, N. Mackey, and C. Schröder, *Structured polynomial eigenproblems related to time-delay systems*, Electron. Trans. Numer. Anal., 31 (2008), pp. 306–330.

[8] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.

[9] K. Hadeler, *Mehrparametrige und nichtlineare Eigenwertaufgaben*, Arch. Ration. Mech. Anal., 27 (1967) pp. 306–328.

[10] E. Jarlebring, K. Meerbergen, and W. Michiels, *A Krylov method for the delay eigenvalue problem*, SIAM J. Sci. Comput., 32 (2010), pp. 3278–3300.

[11] E. Jarlebring, W. Michiels, and K. Meerbergen, *A linear eigenvalue algorithm for the nonlinear eigenvalue problem*, Numer. Math., 122 (2012), pp 169–195.

[12] Z. Jia and Y. Sun, *A Refined Second-Order Arnoldi (RSOAR) Method for the Quadratic Eigenvalue Problem and Implicit Restarting*, Technical report, arXiv:1005.3947, 2010.

[13] D. Kressner, *A block Newton method for nonlinear eigenvalue problems*, Numer. Math., 114 (2009), pp. 355–372.

[14] P. Lancaster, *Lambda-Matrices and Vibrating Systems*, Dover Publications, Mineola, NY, 2002.

[15] P. Lancaster and P. Psarrakos, *On the pseudospectra of matrix polynomials*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 115–129.

[16] R. B. Lehoucq, *Implicitly restarted Arnoldi methods and subspace iteration*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 551–562.

[17] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.

[18] B.-S. Liao, Z. Bai, L.-Q. Lee, and K. Ko, *Solving Large Scale Nonlinear Eigenvalue Problems in Next-Generation Accelerator Design*, Technical report SLAC-PUB-12137, Stanford University, Stanford, CA, 2006.

[19] D. S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann, *Structured polynomial eigenvalue problems: Good vibrations from good linearizations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 1029–1051.

[20] K. Meerbergen, *Locking and restarting quadratic eigenvalue solvers*, SIAM J. Sci. Comput., 22 (2001), pp. 1814–1839.

[21] K. Meerbergen, *The quadratic Arnoldi method for the solution of the quadratic eigenvalue problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1463–1482.

[22] V. Mehrmann and H. Voss, *Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods*, GAMM-Mitt., 27 (2004), pp. 121–152.

[23] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.

[24] O. Rott and E. Jarlebring, *An iterative method for the multipliers of periodic delay-differential equations and the analysis of a PDE milling model*, in Proceedings of the 9th IFAC Workshop on Time-Delay Systems, Prague, 2010, pp. 1–6.

[25] A. Ruhe, *Algorithms for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 10 (1973), pp. 674–689.

[26] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[27] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.

[28] Y. Su and Z. Bai, *Solving rational eigenvalue problems via linearization*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 201–216.

[29] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.

[30] R. Van Beeumen, K. Meerbergen, and W. Michiels, *A rational Krylov method based on Hermite interpolation for nonlinear eigenvalue problems*, SIAM J. Sci. Comput., 35 (2013), pp. A327–A350.

[31] H. Voss, *A maxmin principle for nonlinear eigenvalue problems with application to a rational spectral problem in fluid-solid vibration*, Appl. Math. (Praha), 48 (2003), pp. 607–622.

[32] H. Voss, *An Arnoldi method for nonlinear eigenvalue problems*, BIT, 44 (2004), pp. 387–401.

[33] L. Zhou, L. Bao, Y. Lin, Y. Wei, and Q. Wu, *Restarted generalized second-order Krylov subspace methods for solving quadratic eigenvalue problems*, Internat. J. Comput. Math. Sci., 4 (2010), pp. 148–155.