Noname manuscript No.
(will be inserted by the editor)

A linear eigenvalue algorithm for the nonlinear eigenvalue problem

Elias Jarlebring · Wim Michiels · Karl Meerbergen

Received: date / Accepted: date

Abstract The *Arnoldi method* for standard eigenvalue problems possesses several attractive properties making it robust, reliable and efficient for many problems. The first result of this paper is a characterization of the solutions to an arbitrary (analytic) *nonlinear eigenvalue problem* (NEP) as the reciprocal eigenvalues of an infinite dimensional operator denoted $\mathcal{B}$. We consider the Arnoldi method for the operator $\mathcal{B}$ and show that with a particular choice of starting function and a particular choice of scalar product, the structure of the operator can be exploited in a very effective way. The structure of the operator is such that when the Arnoldi method is started with a constant function, the iterates will be polynomials. For large class of NEPs, we show that we can carry out the infinite dimensional Arnoldi algorithm for the operator $\mathcal{B}$ in arithmetic based on standard linear algebra operations on vectors and matrices of finite size. This is achieved by representing the polynomials by vector coefficients. The resulting algorithm is by construction such that it is completely equivalent to the standard Arnoldi method and also inherits many of its attractive properties, which are illustrated with examples.

Keywords Nonlinear eigenvalue problems · The Arnoldi method · Krylov subspaces · Spectral methods · Chebyshev polynomials

1 Introduction

Consider a given function $T : \Omega \to \mathbb{C}^{n\times n}$ where $\Omega \subset \mathbb{C}$ is an open disc centered at the origin and T is assumed to be analytic in Ω. We will here consider the (analytic) nonlinear eigenvalue problem given by finding $\lambda \in \Omega$ and $x \in \mathbb{C}^n\backslash\{0\}$ such that

$$T(\lambda)x = 0. \tag{1}$$

This general problem has been extensively studied. See, e.g., the standard references [21,18] and the problem collection [4].

In the literature there exists a number of nice methods specialized for different types of structures of T, such as methods for the quadratic eigenvalue problem (QEP) [3,17] and more generally the polynomial eigenvalue problem (PEP) [22,10] and also other types of structures [23,16,7,13,11]. Apart from the specialized methods there also exist general methods which have a sense of local convergence for one or a few eigenvalues [20,25,19,12].

Our goal in this paper is to present a general algorithmic framework, applicable to a large class of NEPs allowing us, in a quite automatized and reliable way, to find all eigenvalues of (1) close to a given target. We will assume that the target is the origin. Note that there is no loss of generality to use the origin as a target in this sense, since a substitution allows us to shift the origin to an arbitrary complex point.

Fundamental for the construction is an equivalence between (1) and the eigenvalues of an operator denoted $\mathcal{B}$. The operator $\mathcal{B}$ is an integration-type operator and infinite dimensional in the sense that it is a map between infinite dimensional function spaces. In the equivalence (presented in Section 2) we show that the reciprocal eigenvalues of $\mathcal{B}$ are the solutions to (1).

We will here make a construction with the Arnoldi method, which is a common method for standard and generalized eigenvalue problems. In Section 3 we summarize the Arnoldi method, when applied to the operator $\mathcal{B}$. We subsequently show that the structure of $\mathcal{B}$ is such that when the Arnoldi method for $\mathcal{B}$ is started with a constant function, the iterates will be polynomials. By representing the polynomials in a given basis (here, either with monomials or Chebyshev polynomials) we show how we can carry out the action of $\mathcal{B}$ on a polynomial using only finite arithmetic, i.e., standard linear algebra operations applied to matrices of finite size. With this construction, we show (in Section 4) that with a particular choice of starting function and a particular scalar product, the Arnoldi method for the operator $\mathcal{B}$ can be carried out in finite arithmetic.

In Section 5 we characterize the algorithm in a different way. We show an equivalence with the presented algorithm and the Arnoldi method on a large matrix, either consisting of a companion linearization of the truncated Taylor expansion or the pencil corresponding to a spectral discretization.

Due to the fact that the algorithm only consists of standard linear algebra operations, it is suitable for large-scale problems. This is illustrated in the examples section (Section 6).

We finally wish to mention some other methods which have some components similar to our method. The method in [26] is also motivated by the Arnoldi method and it is a generalization in the sense that it reduces to the standard Arnoldi method for the linear eigenvalue problem, i.e., if $T(\lambda) = A - \lambda I$. The action of the operator $\mathcal{B}$ is integration. There are other methods for NEPs based on integration. Unlike the method presented in this paper, the methods in [2,5] are based on a contour integral formulation.

2 Operator equivalence formulation

In this paper it will be advantageous to slightly reformulate the nonlinear eigenvalue problem (1). We will first of all assume that $\lambda = 0$ is not an eigenvalue of the original problem and define,

$$B(\lambda) := T(0)^{-1}\frac{T(0) - T(\lambda)}{\lambda}, \quad \lambda \neq 0 \tag{2}$$

and $B(0) := -T(0)^{-1}T'(0)$. With this transformation, (1) yields that,

$$\lambda B(\lambda)x = x. \tag{3}$$

Note that B is analytic in Ω since T is analytic in Ω.

In order to characterize the solutions to (3), we will in this paper take an approach where the iterates can be interpreted as functions. We will commonly work with functions $\varphi : \mathbb{R} \to \mathbb{C}^n$ and denote the function variable θ.

For notational convenience we will use the concise functional analysis notation $B(\frac{d}{d\theta})$, which can consistently defined (since B is analytic), e.g., with the Taylor expansion. In particular, if we apply $B(\frac{d}{d\theta})$ to a function $\varphi : \mathbb{R} \to \mathbb{C}^n$ and evaluate at $\theta = 0$, we can express it as

$$\left(B(\frac{d}{d\theta})\varphi\right)(0) = \sum_{i=0}^{\infty} \frac{1}{i!} B^{(i)}(0)\varphi^{(i)}(0). \tag{4}$$

With the notation above we are now ready to present a characterization of (3) based on an operator denoted $\mathcal{B}$, which maps functions in $\mathcal{D}(\mathcal{B}) \subset C_\infty(\mathbb{R}, \mathbb{C}^n)$, i.e., infinitely differentiable functions $\varphi : \mathbb{R} \to \mathbb{C}^n$, to functions in $C_\infty(\mathbb{R}, \mathbb{C}^n)$. In the following we define the operator $\mathcal{B}$ and characterize some important properties. In particular, we show that the non-zero eigenvalues of $\mathcal{B}$ are the reciprocal solutions λ to (3).

Definition 1 (The operator $\mathcal{B}$) *Let $\mathcal{B}$ denote the map defined by the domain $\mathcal{D}(\mathcal{B}) := \{\varphi \in C_\infty(\mathbb{R}, \mathbb{C}^n) : \sum_{i=0}^{\infty} B^{(i)}(0)\varphi^{(i)}(0)/(i!)$ is finite$\}$ and the action*

$$(\mathcal{B}\varphi)(\theta) = \int_0^\theta \varphi(\hat{\theta})\, d\hat{\theta} + C(\varphi), \tag{5}$$

where

$$C(\varphi) := \sum_{i=0}^{\infty} \frac{B^{(i)}(0)}{i!} \varphi^{(i)}(0) = \left(B(\frac{d}{d\theta})\varphi\right)(0). \tag{6}$$

Proposition 2 (Linearity) *For any two functions $\varphi, \psi \in \mathcal{D}(\mathcal{B})$ and any two constant scalars $\alpha, \beta \in \mathbb{C}$, the map $\mathcal{B}$ satisfies the linearity property,*

$$\mathcal{B}(\alpha\varphi + \beta\psi) = \alpha\mathcal{B}\varphi + \beta\mathcal{B}\psi.$$

Proof Note that C is linear in the sense that, $C(\alpha\varphi + \beta\psi) := \sum_{i=0}^{\infty} \frac{B^{(i)}}{i!}(\alpha\varphi^{(i)}(0) + \beta\psi^{(i)}(0)) = \alpha C(\varphi) + \beta C(\psi)$. The proof is completed by noting that integration is linear and hence the sum in (5) is linear.

Theorem 3 (Operator equivalence) *Let $x \in \mathbb{C}^n \backslash \{0\}$, $\lambda \in \Omega \subset \mathbb{C}$ and denote $\varphi(\theta) := xe^{\lambda\theta}$. Then the following statements are equivalent.*

i) The pair (λ, x) is a solution to the nonlinear eigenvalue problem (3).
ii) The pair (λ, φ) is a solution to the infinite dimensional eigenvalue problem

$$\lambda\mathcal{B}\varphi = \varphi. \tag{7}$$

Moreover, all eigenfunctions of $\mathcal{B}$ depend exponentially on θ, i.e., if $\lambda\mathcal{B}\psi = \psi$ then $\psi(\theta) = xe^{\lambda\theta}$.

Proof We first show that an eigenfunction of $\mathcal{B}$ always is exponential in θ. Suppose $\varphi \in \mathcal{D}(\mathcal{B})$ satisfies (7) and consider the derivative

$$\frac{d}{d\theta}(\lambda\mathcal{B}\varphi) = \lambda\frac{d}{d\theta}(\mathcal{B}\varphi) = \varphi', \tag{8}$$

which exists since all functions of the domain of $\mathcal{B}$ are differentiable. Due to the fact that the action of $\mathcal{B}$ is integration, the left-hand side of (8) is $\lambda\varphi$. The result of (8) is hence the differential equation $\lambda\varphi = \varphi'$, for which the solution always is of the form $\varphi(\theta) = xe^{\lambda\theta}$.

In order to show that i) implies ii), suppose (λ, x) is a solution to (3). Let $\varphi(\theta) := xe^{\lambda\theta}$ and note that $\varphi \in \mathcal{D}(\mathcal{B})$ since

$$\sum_{i=0}^{\infty}\frac{1}{i!}B^{(i)}(0)\varphi^{(i)}(0) = \sum_{i=0}^{\infty}\frac{\lambda^i}{i!}B^{(i)}(0)\varphi(0) = B(\lambda)x,$$

is finite. We here used that $\lambda \in \Omega$ implies that the series is convergent. Now note that the derivative of the left-hand side of (7) is $\lambda\frac{d}{d\theta}(\mathcal{B}\varphi) = \lambda\varphi$, since $\mathcal{B}\varphi$ is a primitive function of φ. From the fact that an eigenfunction takes the form $\varphi(\theta) = xe^{\lambda\theta}$ it follows that the derivative of the right-hand side of (7) is $\lambda\varphi$. Hence, the derivative of the function equality (7) is satisfied. It remains to show that (7) holds in one point. Consider (7) evaluated at $\theta = 0$. The left-hand side is $\lambda(\mathcal{B}\varphi)(0) = \lambda C(\varphi)$ and the right-hand side $\varphi(0) = x$. It follows that the difference is,

$$\lambda C(\varphi) - x = \lambda(B(\frac{d}{d\theta})\varphi)(0) - x = \lambda B(\lambda)\varphi(0) - x = 0,$$

where in the last step we used that (λ, x) is an eigenpair of (3). We have shown i) implies ii).

In order to show the converse, suppose $(\lambda, \varphi) \in (\mathbb{C}, \mathcal{D}(\mathcal{B}))$ is a solution to (7). We already know that a solution to (7) is an exponential times a vector (which we call x), i.e., $\varphi(\theta) = xe^{\lambda\theta}$. Now evaluate the difference between the left and right-hand side of (7) at $\theta = 0$,

$$0 = \lambda(\mathcal{B}\varphi)(0) - \varphi(0) = \lambda C(\varphi) - x = \lambda(B(\frac{d}{d\theta})\varphi)(0) - x = \lambda B(\lambda)x - x. \tag{9}$$

In the last step we used $\varphi(\theta) = xe^{\lambda\theta}$, which implies that $\varphi^{(i)} = \lambda^i\varphi$ for any i. Since (9) is the nonlinear eigenvalue problem (3), we have completed the proof.

Remark 1 (Eigenvalue properties) The set of solutions λ of (1) is exactly the root set of the characteristic equation $\det(T(\lambda)) = 0$. Recall that T and hence $\det(T(\lambda))$ is assumed to be an analytic function in Ω. A direct consequence of Cauchy's residue theorem (and the principle of argument) is that an analytic function which is not identically zero will only have a finite number of zeros in any compact subset of the complex plane. Hence, (1) will not have any clustering points in Ω (unless $\lambda = 0$ is an eigenvalue). Note that this carries over to the operator with the equivalence in Theorem 3, implying that the reciprocal eigenvalues of $\mathcal{B}$ will not have any clustering points in Ω.

Remark 2 (Connection with differential equation in work by Gohberg, Lancaster and Rodman) In several works of Gohberg, Lancaster and Rodman, e.g., [8], the authors use a differential equation associated with the polynomial eigenvalue problem. It is straightforward to show that the equation in the domain of $\mathcal{B}^{-1}$ is precisely this associated differential equation. We can hence interpret $\mathcal{B}$ and Theorem 3 as follows. The operator $\mathcal{B}$ and Theorem 3 corresponds to an (integration) operator formulation of the differential equation associated with the polynomial (or nonlinear) eigenvalue problem.

3 The Arnoldi method in a function setting

Now consider the Arnoldi method (first presented in [1]) where instead of applying it to a matrix, which is the common construction, we apply it to the operator $\mathcal{B}$.

Fundamental in the Arnoldi method is the *Krylov subspace*, for the operator $\mathcal{B}$ defined as,

$$\mathcal{K}_k(\mathcal{B},\varphi) := \operatorname{span}(\varphi, \mathcal{B}\varphi, \dots, \mathcal{B}^{k-1}\varphi) \subset C_\infty(\mathbb{C},\mathbb{C}^n),$$

where $\varphi \in \mathcal{D}(\mathcal{B}) \subset C_\infty(\mathbb{R},\mathbb{C}^n)$. The Arnoldi method can be interpreted as a construction of an orthogonal basis of $\mathcal{K}_k(\mathcal{B},\varphi)$ and simultaneously forming an orthogonal projection of $\mathcal{B}$ onto this subspace. The orthogonal projection is achieved by a Gram-Schmidt orthogonalization process associated with a (for the moment) arbitrary scalar product. For a given scalar product we can directly formulate the Arnoldi method in an abstract setting. This is summarized in Algorithm 1.

In this paper we use the notation for the elements of the Hessenberg matrix common when working with the Arnoldi method. The upper block of the rectangular Hessenberg matrix $\underline{H}_k \in \mathbb{C}^{(k+1)\times k}$ is denoted $H_k \in \mathbb{C}^{k\times k}$ and the (i,j) element of $\underline{H}_k$ is denoted $h_{i,j}$.

Algorithm 1 The Arnoldi method for $\mathcal{B}$

```
Require: $\varphi_1 \in \mathcal{D}(\mathcal{B})$ such that $<\varphi_1,\varphi_1>=1$
 1: for $k = 1,2,\dots$ until converged do
 2:     $\psi = \mathcal{B}\varphi_k$
 3:     for $i = 1,\dots,k$ do
 4:         $h_{i,k} = <\psi,\varphi_i>$
 5:         $\psi = \psi - h_{i,k}\varphi_i$
 6:     end for
 7:     $h_{k+1,k} = \sqrt{<\psi,\psi>}$
 8:     $\varphi_{k+1} = \psi/h_{k+1,k}$
 9: end for
10: Compute the eigenvalues $\{\mu_i\}_{i=1}^k$ of the Hessenberg matrix $H_k$
11: Return eigenvalue approximations $\{1/\mu_i\}_{i=1}^k$
```

Recall that our ultimate goal is to construct a method which favors solutions of (3) close to the origin. From this perspective, it is quite natural to consider the Arnoldi method corresponding to $\mathcal{B}$, since the Arnoldi method favors extreme isolated eigenvalues of $\mathcal{B}$. We know from Theorem 3 that the reciprocal eigenvalues of $\mathcal{B}$ are solutions to (3) and the reciprocal of extreme isolated eigenvalues are usually indeed the solutions λ close to origin, since the extreme eigenvalues are often of large magnitude. This is consistent with the common construction for matrices of shift-invert Arnoldi method, e.g., used in the software package ARPACK [14].

4 Finite arithmetic implementation of Algorithm 1

It is now for our purposes important to note that Algorithm 1 has some free components. In Algorithm 1, we are free to choose

i) a starting function φ_1; and
ii) a scalar product $< \cdot, \cdot >$.

Algorithm 1 is an iteration where the iterates are elements of the infinite dimensional vector space $C_\infty(\mathbb{R}, \mathbb{C}^n)$. We will now see that with particular choice of starting function and a particular choice of scalar product we can reformulate Algorithm 1 to an iteration involving only standard linear algebra operations on matrices and vectors of *finite* dimension.

4.1 Constant starting function and action on polynomials

First note that $\mathcal{B}$ satisfies a closure property over polynomials of growing degree, in the sense that if φ is a vector of polynomials of degree k, then $\mathcal{B}\varphi$ is a vector of polynomials of degree $k+1$. This follows from the fact that the action of $\mathcal{B}$ corresponds to forming a primitive function. Also note that the linear combination of two polynomials is also a polynomial.

Algorithm 1 consists of applying $\mathcal{B}$ to function and forming corresponding linear combinations. Hence, we can start Algorithm 1 with a constant function φ_1 and we get iterates which are polynomials. After N steps we have the Krylov subspace $\mathcal{K}_N(\mathcal{B}, \varphi) = \text{span}(\varphi_1, \ldots, \varphi_N)$, which is a space of vectors of polynomials of degree $N-1$.

In the following result we formalize the closure property of $\mathcal{B}$ for polynomials, when considering a given polynomial basis.

Theorem 4 (General coefficient map) *Let $\{q_i\}_{i=0}^\infty$ be a sequence of polynomials such that q_i is of degree i and has a non-zero leading coefficient and let $q_0(\theta) = 1$. Moreover, let $L_N \in \mathbb{R}^{N \times N}$ denote the integration map of the polynomials $q_0, \ldots, q_{N-1}$, in the sense that for any $N \in \mathbb{N}$,*

$$\begin{bmatrix} q_0(\theta) \\ q_1(\theta) \\ \vdots \\ q_{N-1}(\theta) \end{bmatrix} = L_N \begin{bmatrix} q_1'(\theta) \\ q_2'(\theta) \\ \vdots \\ q_N'(\theta) \end{bmatrix}. \tag{10}$$

Let the columns of $(x_0, \ldots, x_{N-1}) =: X \in \mathbb{C}^{n \times N}$ denote the vector coefficients in the basis $\{q_i\}_{i=0}^\infty$ and denote the corresponding vector of polynomials φ,

$$\varphi(\theta) := \sum_{i=0}^{N-1} q_i(\theta) x_i. \tag{11}$$

Correspondingly, let $y_0, \ldots, y_N$ denote the coefficients of $\psi := \mathcal{B}\varphi$, i.e.,

$$\psi(\theta) = (\mathcal{B}\varphi)(\theta) = \sum_{i=0}^{N} q_i(\theta) y_i.$$

Then, the coefficients of $\mathcal{B}\varphi$ are given by

$$(y_1,\ldots,y_N) = XL_N, \tag{12}$$

and

$$y_0 = \left(\sum_{i=0}^{N-1} B(\frac{d}{d\theta})q_i(\theta)x_i\right)(0) - \sum_{i=1}^{N} q_i(0)y_i. \tag{13}$$

Proof From the expansion of φ, i.e., (11), and the integration map L_N we find that

$$\int_0^\theta \varphi(\hat\theta)\,d\hat\theta = \int_0^\theta X \begin{bmatrix} q_0(\hat\theta) \\ \vdots \\ q_{N-1}(\hat\theta)\end{bmatrix} d\hat\theta = XL_N \begin{bmatrix} q_1(\theta) \\ \vdots \\ q_N(\theta)\end{bmatrix} - XL_N \begin{bmatrix} q_1(0) \\ \vdots \\ q_N(0)\end{bmatrix}. \tag{14}$$

We can now insert (14) into (5) and find that

$$(y_0,\ldots,y_N)\begin{bmatrix} q_0(\theta) \\ \vdots \\ q_N(\theta)\end{bmatrix} = \left(C(\varphi) - XL_N \begin{bmatrix} q_1(0) \\ \vdots \\ q_N(0)\end{bmatrix}, XL_N \right) \begin{bmatrix} q_0(\theta) \\ \vdots \\ q_N(\theta)\end{bmatrix}, \tag{15}$$

where we used that $q_0(\theta) := 1$. Note that the polynomials $q_0,\ldots,q_N$ are linearly independent. Hence, the matrix in front of the coefficients on the left-hand side in (15) equals the matrix in front of the coefficients on the right-hand side. The relation (12) follows from the corresponding last N columns and (13) follows from the first column and (12).

4.2 Action in monomial basis and Chebyshev basis

In Theorem 4 we characterized the map $\mathcal{B}$ for vectors of polynomials, where the vectors of polynomials were given as vector coefficients in a given polynomial basis. The best choice of basis depends on the problem at hand and in order to have a general algorithmic framework we now provide the specialization of Theorem 4 for two choices of basis functions.

The Taylor coefficients correspond to a simple way of representing polynomials. That is, we represent the polynomial with the coefficients in the monomial basis,

$$q_i(\theta) = \theta^i. \tag{16}$$

Since $q_i'(\theta) = \frac{1}{i}q_{i-1}$, the integration map L_N in (10) is the diagonal matrix

$$L_N = L_{T,N} := \begin{bmatrix} 1 & & & \\ & \frac{1}{2} & & \\ & & \ddots & \\ & & & \frac{1}{N}\end{bmatrix}. \tag{17}$$

In Theorem 4 we also need find an expression for y_0. Note that from the definition of $B(\frac{d}{d\theta})$ we have property (4) which for formula (13) simplifies to,

$$\left(B(\frac{d}{d\theta})q_i(\theta)\right)(0) = \left(B(\frac{d}{d\theta})\theta^i\right)(0) = B^{(i)}(0).$$

Hence, from definition (13), we can express y_0 in terms of derivatives of B,

$$y_0 = \left(\sum_{i=0}^{N-1} B(\frac{d}{d\theta}) q_i(\theta) x_i \right)(0) - \sum_{i=1}^{N} q_i(0) y_i = \sum_{i=0}^{N-1} B^{(i)}(0) x_i. \tag{18}$$

By using that $\sum_{i=1}^{N} q_i(\theta) y_i$ is a primitive function of $\sum_{i=0}^{N-1} q_i(\theta) x_i$ we can also express the formula for y_0 in terms of the original nonlinear eigenvalue problem (1). If we insert the Taylor expansion of T in the definition of B, i.e., (2), and simplify (18) we find that

$$y_0 = -T(0)^{-1} \sum_{i=1}^{N} T^{(i)}(0) y_i. \tag{19}$$

We have shown that if a vector of polynomials is given in terms of vector coefficients in a monomial basis, the action of $\mathcal{B}$ can be computed by first computing $y_1, \ldots, y_N$ with the matrix-matrix multiplication (12) with $L_{T,N}$ given by (17) and then computing y_0 using formula (18) or (19). These operations only involve linear algebra operations applied to matrices and vectors of length n.

Apart from the monomials, we will in this work also consider functions represented in the Chebyshev basis. Here, we define the shifted and scaled Chebyshev polynomials for an interval $[a, b] \subset \mathbb{R}$ as

$$\hat{T}_i(\theta) := T_i\left(k\theta + c\right), \; c = \frac{a+b}{a-b} \text{ and } k = \frac{2}{b-a}, \tag{20}$$

where $T_i(\theta) := \cos(i \arccos(\theta))$.

By using the properties of Chebyshev polynomials, in particular $\hat{T}_i'(\theta) = ikU_{i-1}(k\theta + c)$, where U_i is the Chebyshev polynomial of the second kind, it is straightforward to verify that the integration map L_N for Chebyshev polynomials $\hat{T}_i$ is given by

$$L_N = L_{C,N} = \frac{b-a}{4} \begin{bmatrix} 2 & & & & & \\ 0 & \frac{1}{2} & & & & \\ -1 & 0 & \frac{1}{3} & & & \\ & -\frac{1}{2} & 0 & \frac{1}{4} & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\frac{1}{N-2} & 0 & \frac{1}{N} \end{bmatrix} \tag{21}$$

Since we want to implement our algorithm with arithmetic operations on a computer, we need to be able to evaluate (13) for a given problem B, we also need to derive a computable expression for

$$y_0 = \left(\sum_{i=0}^{N-1} B(\frac{d}{d\theta}) \hat{T}_i(\theta) x_i \right)(0) - \sum_{i=1}^{N} T_i(c) y_i. \tag{22}$$

The last term in (22) is already easy to evaluate and it remains to study the first term. In order to find an explicit efficient expression we will decompose B into a sum of scalar nonlinearities,

$$B(\lambda) = B_0 b_0(\lambda) + \cdots + B_m b_m(\lambda), i = 0, \ldots, m. \tag{23}$$

where $b_i : \mathbb{C} \to \mathbb{C}$ are analytic functions in Ω. Note that (23) is not a restriction of generality *in theory* since $B_0, \ldots, B_m \in \mathbb{C}^{n \times n}$ can be chosen as the n^2 unit matrices with $m = n^2 - 1$. In many examples, such as those in the example collection [4], m can be chosen small. The first term in (22) can now be written as

$$\sum_{i=0}^{N-1}((B(\frac{d}{d\theta})\hat{T}_i)x_i)(0) = \sum_{j=0}^{m} B_j \sum_{i=0}^{N-1}\left(b_j(\frac{d}{d\theta})\hat{T}_i x_i\right)(0). \tag{24}$$

The problem is hence reduced to a sum of scalar problems, given by

$$\left(\sum_{i=0}^{N-1} b(\frac{d}{d\theta})\hat{T}_i x_i\right)(0), \tag{25}$$

for $b = b_j$, $j = 0, \ldots, m$. The handling of the expression (25) is a matter of analysis with properties of Chebyshev polynomials. In Appendix A we give the explicit expression for several elementary functions b and also present a constructive procedure for computing formulas for (25) given the Taylor expansion of b.

Finally, note that in many situations the coefficients B_i involve an inverse, due to the definition (2). This inverse should not be computed explicitly. It is often possible to rearrange terms in the formula such that we need to solve only one linear system for each evaluation of y_0.

4.3 Scalar product consistent with polynomial basis

We showed above that if we represent the vector of polynomials φ with vector coefficients using either the monomial basis or Chebyshev basis, we can express the action of $\mathcal{B}\varphi$ using only matrices and vectors of size n.

We will now propose a scalar product (between vectors of polynomials) which can also be implemented using only vectors when applied to vectors of polynomials. The scalar product can be defined in a way which is consistent with the basis. Consider (for the moment) any of the two polynomials bases $\{q_i\}_{i=0}^{\infty}$ in the previous section and consider two functions

$$\varphi(\theta) = \sum_{i=0}^{N} q_i(\theta)x_i, \quad \psi(\theta) = \sum_{i=0}^{N} q_i(\theta)y_i. \tag{26}$$

We will propose to use the following construction, which consists of summing the Euclidean scalar product of the coefficients,

$$<\varphi, \psi> := \sum_{i=0}^{N} y_i^H x_i. \tag{27}$$

It is important to note that this does indeed define a scalar product, in the following sense. The proof consists of direct verification of the properties in the definition of a scalar product.

Lemma 5 (Scalar product) *The map defined by* (27) *satisfies the properties of a scalar product when* φ *and* ψ *are given by* (26)*, where* $q_i(\theta) = \theta^i$ *or* $q_i(\theta) = \hat{T}_i(\theta)$*.*

We will respectively refer to the scalar product corresponding to $q_i(\theta) = \theta^i$ and $q_i(\theta) = \hat{T}_i(\theta)$ as the Taylor scalar product (and denote it $< \cdot, \cdot >_T$) and the Chebyshev scalar product (and denote it $< \cdot, \cdot >_C$).

Further characterization of the scalar product is given in Section 5.

4.4 Algorithm

We are now ready to combine Section 3, i.e., the function setting Arnoldi method with the results in Section 4.1-4.3 into a practical algorithm. Algorithm 2 is started with a constant function, we represent the functions in a basis $\{q_k\}_{k=0}^{\infty}$, where q_k is the monomial basis or the Chebyshev basis, and use the consistent scalar product defined by (27).

It turns out to be advantageous to store the coefficients of the vectors of polynomials in vectorized form, i.e., stacking of the coefficients on top of each other. The scalar product (27) can in this way be carried out as the Euclidean inner product of the vectorized coefficients. This is efficient since it is only based on a few matrix operations and it naturally suggests the use of reorthogonalization to remedy possible loss of orthogonality. With this stacking of coefficients we arrive at an algorithm where the basis matrix is growing with one column and one block row in each iteration, which is visualized in Figure 1 and explicitly given in Algorithm 2.

In the previous sections we have presented two versions of the function representation and scalar products. We will call the construction with monomials (corresponding to (17)-(19)) the *Taylor version* and the construction with Chebyshev polynomials (corresponding to (21) and (22)) the *Chebyshev version*. The Taylor version and Chebyshev version are denoted *a*) and *b*) respectively in Algorithm 2.

Note that we have constructed the algorithm such that it is a (finite arithmetic) implementation of Algorithm 1.

- The Taylor version of Algorithm 2 is equivalent to Algorithm 1 started with the constant function $\varphi_1(\theta) = x_0$ and the scalar product $< \cdot, \cdot >_T$.
- The Chebyshev version of Algorithm 2 is equivalent to Algorithm 1 started with the constant function $\varphi_1(\theta) = x_0$ and the scalar product $< \cdot, \cdot >_C$.

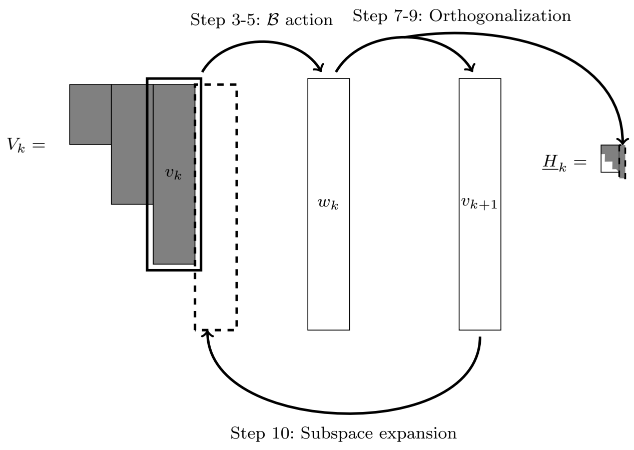

Step 3-5: $\mathcal{B}$ action
Step 7-9: Orthogonalization
$V_k =$
v_k
w_k
v_{k+1}
$\underline{H}_k =$
Step 10: Subspace expansion

Fig. 1 Visualization of Algorithm 2, illustrating growth of the basis matrix V_k.

Remark 3 (Implementation issues) Several implementation issues need to be taken into account when implementing Algorithm 2. We use the same techniques for eigenvector

Algorithm 2 A finite arithmetic implementation of Algorithm 1

```
Require: $x_0 \in \mathbb{C}^n$
 1: Let $V_1 = x_0/\|x_0\|_2$, $k = 1$, $\underline{H}_0$ =empty matrix
 2: for $k = 1, 2, \ldots$ until converged do
 3:    Let $\operatorname{vec}(X) = v_k$
 4:    Compute $y_1, \ldots, y_{k+1}$ according to (12) with sparse $L_k$
         a)   using $L_k = L_{T,k}$ given by (17); or
         b)   using $L_k = L_{C,k}$ given by (21).
 5:    Compute $y_0$ by either
         a)   using $y_0$ given by (18) or (19); or
         b)   using $y_0$ given by (22).
 6:    Expand $V_k$ with one block row (zeros)
 7:    Let $w_k := \operatorname{vec}(y_0, \ldots, y_{k+1})$, compute $h_k = V_k^H w_k$ and then $\hat{w}_k = w_k - V_k h_k$
 8:    Compute $\beta_k = \|\hat{w}_k\|_2$ and let $v_{k+1} = \hat{w}_k/\beta_k$
 9:    Let $\underline{H}_k = \begin{bmatrix} \underline{H}_{k-1} & h_k \\ 0 & \beta_k \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}$
10:    Expand $V_k$ into $V_{k+1} = [V_k, v_{k+1}]$
11: end for
12: Compute the eigenvalues $\{\mu_i\}_{i=1}^k$ of the Hessenberg matrix $H_k$
13: Return approximations $\{1/\mu_i\}_{i=1}^k$
```

extraction, reorthogonalization, stopping criteria and related issues as described in [11, Section 3.2].

5 Interpretations as Arnoldi's method on a matrix

In order to provide further insight into what version of Algorithm 2 is suitable given for a given problem type, we will now give characterizations for the two versions in a different way. We will provide reasoning based on the result that k steps of the Taylor version as well as k steps of the Chebyshev version both have an equivalence with the (standard) Arnoldi's method applied to a (finite) matrix of size $Nn \times Nn$ for any $N > k$.

5.1 Interpretation of Taylor version and companion linearization

Consider the truncated Taylor expansion of B,

$$B(\lambda) = \sum_{k=0}^{\infty} \frac{B^{(k)}(0)}{k!}\lambda^k \approx \sum_{k=0}^{N} \frac{B^{(k)}(0)}{k!}\lambda^k. \tag{28}$$

If we insert this approximation in (3), we arrive at the polynomial eigenvalue problem

$$\left(-I + \tilde{\lambda}B(0) + \tilde{\lambda}^2\frac{B^{(1)}(0)}{1!} + \tilde{\lambda}^3\frac{B^{(2)}(0)}{2!} + \cdots + \tilde{\lambda}^{N+1}\frac{B^{(N)}(0)}{N!}\right)\tilde{x} = 0.$$

The standard approach to solve and study polynomial eigenvalue problems is by means of the transformation called companion linearization. We will here consider the companion linearization

$$\tilde{\lambda} C_N \begin{bmatrix} \tilde{x} \\ \tilde{\lambda}\tilde{x} \\ \frac{\tilde{\lambda}^2}{2!}\tilde{x} \\ \vdots \\ \frac{\tilde{\lambda}^N}{N!}\tilde{x} \end{bmatrix} = \begin{bmatrix} \tilde{x} \\ \tilde{\lambda}\tilde{x} \\ \frac{\tilde{\lambda}^2}{2!}\tilde{x} \\ \vdots \\ \frac{\tilde{\lambda}^N}{N!}\tilde{x} \end{bmatrix},$$

where

$$C_N = \begin{bmatrix} B(0) & B^{(1)}(0) & \cdots & B^{(N-1)}(0) & B^{(N)}(0) \\ I & & & & \\ & \frac{1}{2}I & & & \\ & & \ddots & & \\ & & & \frac{1}{N}I & 0 \end{bmatrix}. \tag{29}$$

Now note that due to the approximation (28) we expect that the reciprocal eigenvalues of C_N approximate the eigenvalues of (3). We will now consider the standard Arnoldi method applied to C_N. By comparison of the formulas for the Taylor version of Algorithm 2 given by (17), i.e., the formula for L_k, and (18), i.e., the formula for y_0, with the action of C_N we reach the following equivalence.

Theorem 6 (Equivalence with Taylor version of Algorithm 2) *Let k, N be such that $N > k$. The result of k steps of the standard Arnoldi method for matrix C_N started with $(x_0^T, 0\ldots, 0)^T$ is equivalent to k steps of the Taylor version of Algorithm 2 started with x_0. The equivalence holds in the sense that the Hessenberg as well as the matrix of basis vectors are equal.*

Remark 4 (Generality of Taylor version) With the above reasoning we reach the indication that the Taylor version works well when the truncated Taylor expansion is an accurate approximation of the function B. Due to the fact that B is analytic in Ω, it has a convergent power series expansion (and hence also Taylor expansion) and we expect the Taylor version of Algorithm 2 to work in general. We expect it to work particularly well when the power series expansion of B converges quickly.

5.2 Interpretation of Chebyshev version for functional differential equations

For the moment, consider a slightly different form of the nonlinear eigenvalue problem

$$\lambda x = A(\lambda)x. \tag{30}$$

This formulation is common for nonlinear eigenvalue problems stemming from linear functional differential equations (FDEs) acting on an interval

$$\tilde{I} = [\tilde{a}, \tilde{b}]. \tag{31}$$

Here, by FDE we mean (as usual in e.g. [9])

$$\dot{z}(t) = f(z_t) = (A(\frac{d}{d\theta})z_t)(0), \tag{32}$$

where f is a (linear) functional and $z_t : [\tilde{a}, \tilde{b}] \to \mathbb{C}^n$ denotes the function segment of z given by $z_t(\theta) = z(t + \theta)$, $\theta \in [\tilde{a}, \tilde{b}]$. The function $A : \mathbb{C} \to \mathbb{C}^{n\times n}$, which also characterizes the eigenvalues of (32) via (30), is often a simple function, e.g., for a retarded delay-differential equation with a single delay, $A(\lambda) := C_0 + C_1 e^{-\tau\lambda}$.

We can directly approach this problem with the main algorithm of this paper (Algorithm 2). If we set $T(\lambda) = A(\lambda) - \lambda I_n$ and use the transformation (2) we have that,

$$B(\lambda) = A(0)^{-1}\frac{A(0) - A(\lambda) + \lambda I_n}{\lambda}. \tag{33}$$

One common approach to compute the eigenvalues of FDEs similar to (32) consists of doing a spectral discretization of the corresponding operator. This approach is taken in, e.g., [6]. We will use a discretization very similar to [11, Section 2] and only point out the elements of the derivation which need to be modified. The FDE (32) is first discretized for an (at this moment) arbitrary interval $I = [a, b]$ using a spectral method. We will use a grid which generalizes the grid in [11, Section 2.3]. The grid is given by,

$$\theta_i = \frac{\alpha_i - c}{k},\ \ \alpha_i = \cos\frac{\pi i}{N+1}, i = 1, \dots, N \text{ and } \theta_{N+1} = 0, \tag{34}$$

where c and k are given in (20). By defining the matrices,

$$R_i := (A(\frac{d}{d\theta})\hat{T}_i)(0),$$

the steps in the derivation of the discretization [11, Section 2.3] can be followed and result in the discretized eigenvalue problem

$$(\lambda \mathit{\Pi}_N - \mathit{\Sigma}_N)z = 0,\ \ z \neq 0, \tag{35}$$

where

$$\mathit{\Pi}_N = \frac{b-a}{4}\begin{bmatrix} \frac{4}{b-a}\hat{T}_0(0) & \frac{4}{b-a}\hat{T}_1(0) & \frac{4}{b-a}\hat{T}_2(0) & \cdots & \frac{4}{b-a}\hat{T}_{N-1}(0) & \frac{4}{b-a}\hat{T}_N(0) \\ 2 & 0 & -1 & & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & & \\ & & \frac{1}{3} & 0 & \ddots & \\ & & & \ddots & \ddots & -\frac{1}{N-1} \\ & & & & \frac{1}{N} & 0 \end{bmatrix} \otimes I_n, \tag{36}$$

and

$$\mathit{\Sigma}_N = \left(\begin{array}{c|c} R_0 & R_1 \ \cdots \ R_N \\ \hline 0 & I_{Nn} \end{array}\right). \tag{37}$$

This grid and this type of formulation of the discretization has the property that $\mathit{\Pi}_{N_1}$ and $\mathit{\Sigma}_{N_1}$ are the leading submatrices of $\mathit{\Pi}_{N_2}$ and $\mathit{\Sigma}_{N_2}$ if $N_2 > N_1$. This structure will allow us to now form a connection with the Chebyshev version of Algorithm 2.

We first show that the action of $\mathit{\Sigma}_N^{-1}\mathit{\Pi}_N$ is equivalent to the action of $\mathcal{B}$ in the sense of the following lemma. The proof is available in Appendix B.

Lemma 7 (Matrix-vector product equivalence) *Let $N > k$ and let the columns of $(x_0, \dots, x_k)$ and $(y_0, \dots, y_{k+1})$ be coefficients of two polynomials given by*

$$\varphi(\theta) := \sum_{i=0}^{k}\hat{T}_i(\theta)x_i \text{ and } \psi(\theta) := \sum_{i=0}^{k+1}\hat{T}_i(\theta)y_i,$$

such that the coefficients fulfill

$$\Sigma_N^{-1}\Pi_N \operatorname{vec}(x_0,\ldots,x_k,0,\ldots,0) = \operatorname{vec}(y_0,\ldots,y_{k+1},0,\ldots,0), \tag{38}$$

where Σ_N and Π_N are given by (36)-(37) *and correspond to the discretization of* (32)*. Then, the operator $\mathcal{B}$ corresponding to the nonlinear eigenvalue problem* (33) *is equivalent to $\Sigma_N^{-1}\Pi_N$ in the sense that,*

$$\psi = \mathcal{B}\varphi. \tag{39}$$

A discretization approach to compute eigenvalues of (32) typically consists of first discretizing the functional differential equation (32), yielding a large generalized eigenvalue problem, similar to (35). The second step normally consists of computing the eigenvalues of the generalized eigenvalue problem with a general purpose method for eigenvalue problems. Suppose we now use the standard Arnoldi algorithm to solve (35).

We saw that the action of $\Sigma_N^{-1}\Pi_N$ was (in the sense of Lemma 7) equivalent to the action of $\mathcal{B}$. Using this result we reach the conclusion that the two-step approach of a discretization and the Arnoldi method is equivalent to Algorithm 1 and hence also equivalent to Algorithm 2. The equivalence holds in the following sense.

Theorem 8 (Equivalence with Chebyshev version of Algorithm 2) *Let k, N be such that $N > k$. The result of k steps of the standard Arnoldi method for $\Sigma_N^{-1}\Pi_N$ started with $(x_0^T, 0\ldots,0)^T$ is equivalent to k steps of the Chebyshev version of Algorithm 2 with interval $[a,b]$ started with x_0. The equivalence holds in the sense that the Hessenberg as well as the matrix of basis vectors are equal.*

5.3 Choice of the interval for functional differential equations

In the equivalence in Section 5.2, we saw that if we discretize an FDE (32) acting on an interval $\tilde{I} = [\tilde{a},\tilde{b}]$ using the grid (34), with $\theta_1,\ldots,\theta_N \in I$, and apply the standard Arnoldi algorithm to the resulting GEP, the approximations are equal to the approximations of the Chebyshev version of Algorithm 2 where the Chebyshev polynomials are scaled to the interval $[a,b]$. We will now set the discretization interval I, equal to the interval of the FDE $\tilde{I}$, i.e.,

$$[a,b] = [\tilde{a},\tilde{b}]. \tag{40}$$

The assumption (40) is very common in literature on discretization of FDEs similar to (32), e.g. [6] and references therein. An intuitive reasoning is that it is natural to distribute the points such that the function values of interest are well approximated. On the contrary, if we would choose a discretization interval I which is larger than the FDE interval $\tilde{I}$, we would also approximate function values not relevant for the FDE. For functional differential equations the interval normally involves the origin, i.e., $\theta_{N+1} = 0 \in \tilde{I}$. Hence, if we set the intervals equal (as in (40)), all grid points (34) are in the discretization interval $I = \tilde{I}$.

In spectral discretization approaches it is common to distribute the points in a non-uniform manner with more grid points at the boundary. Grids which asymptotically have a Chebyshev distribution are in some sense optimal [24, Chapter 5]. The grid points (34) are asymptotically distributed in this way.

Hence, under the condition (40), i.e., that we set the intervals equal, the spectral discretization in Section 5.2 is *good* in the sense that,

- it corresponds to approximating the correct interval; and
- the grid distribution (34) is a Chebyshev like distribution.

The above arguments lead to a natural choice (40) of the interval $I = [a, b]$. Furthermore, the fact that with the choice (40) the Chebyshev version of Algorithm 2 corresponds to a good spectral discretization of the problem suggests the use of the Chebyshev version of Algorithm 2 with interval (40) for functional differential equations.

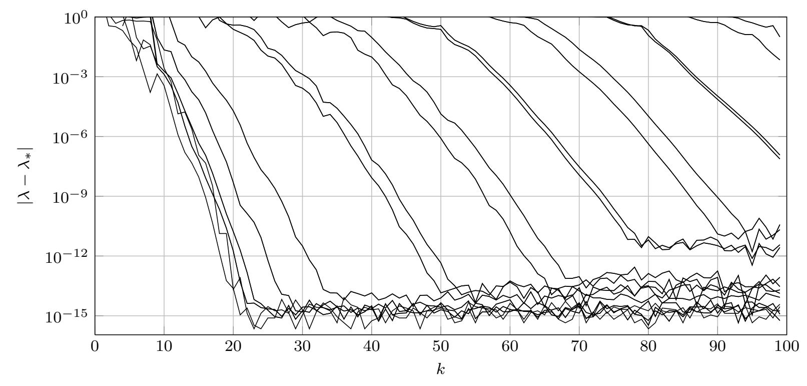


Fig. 2 Convergence history for the Taylor version of Algorithm 2 applied to the example in Section 6.1. After $k = 80$ iterations 22 eigenvalues have been found (with absolute error less than 10^{-10})

6 Examples

6.1 Delay eigenvalue problem with a quadratic term

Although the method is primarily designed for large scale problems, we will for illustrative purposes first consider a small nonlinear eigenvalue problem. This allows us to study the impact of the different versions of the algorithm and the scalar product.

Consider a nonlinear eigenvalue problem of the form,

$$T(\lambda) = -\lambda^2 I_n + A_0 + A_1 e^{-\tau\lambda},$$

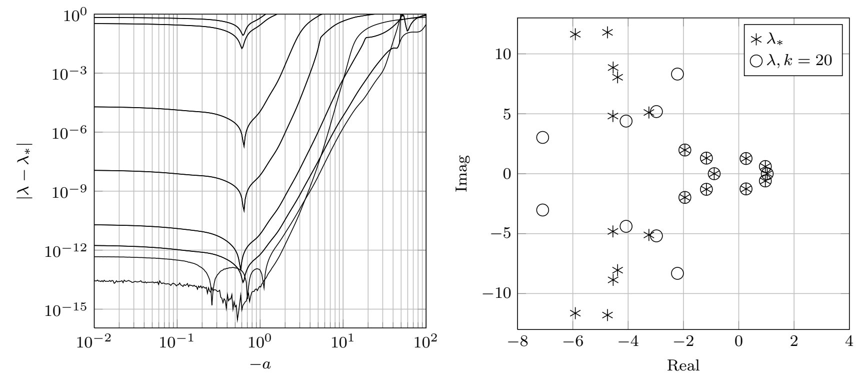
which can be seen as the characteristic equation of a second order time-delay system, i.e., a combination of a QEP and a DEP. We choose A_0 and A_1 in a random way,

$$A_0 = \frac{1}{10}\begin{bmatrix} 3 & -6 & 0 & 4 \\ -3 & 4 & -8 & 19 \\ 1 & -16 & -13 & 0 \\ -14 & -9 & 2 & 9 \end{bmatrix}, \quad A_1 = \frac{1}{10}\begin{bmatrix} 8 & 2 & -13 & -3 \\ -11 & 9 & 12 & 5 \\ 5 & 2 & -16 & -13 \\ 7 & 4 & -4 & 0 \end{bmatrix}.$$

For the Taylor version of Algorithm 2 we use the formula for y_0 in (19) and the Taylor expansion of $e^{-\tau\lambda}$ and find that

$$y_0 = (A_0 + A_1)^{-1}\left(y_1 - A_0\sum_{i=1}^{N}(-\tau)^i y_i\right).$$

The convergence diagram for the Taylor version of Algorithm 2 is given in Figure 2.



(a) Accuracy of the eigenvalues after $k = 20$ iterations for different intervals.

(b) Eigenvalue approximations after $k = 20$ iterations (Chebyshev version)

Fig. 3 Illustration of the approximations of the Chebyshev version of Algorithm 2 for the problem in Section 6.1.

In order to implement the Chebyshev version we first need to transform the problem to the form (3), i.e., find an expression for B. The result of the reformulation (2) is

$$B(\lambda) = (A_0 + A_1)^{-1}(\lambda I_n + A_1 q(\lambda)),$$

where q is given by (47). In order to study the convergence as a function of the interval we will now derive the method for the interval $I = [a, 0]$, where $a < 0$ is treated as a free parameter. From the formulas in Table 1 and the same manipulations as those leading up to (49), we find that y_0 in (22) can be simplified to

$$y_0 = (A_0 + A_1)^{-1}\left(\sum_{i=1}^{N-1}\left(\frac{2i}{a}U_{i-1}(1)x_i\right) - A_0\sum_{i=1}^{N} y_i - A_1\sum_{i=1}^{N} T_i(1 + 2\tau/a)y_i\right).$$

By carrying out several runs, we study the accuracy of the solution after $k = 20$ for different choices of a. This is visualized in Fig. 3a, where we see that choosing $-a = \tau = 1$ produces high accuracy for many eigenvalue approximations. From the figure it is also clear that choosing $-a$ not equal to the delay, can slow down convergence considerably, in particular if the interval is chosen much larger than the delay. This is consistent with the theory in Section 5.3 which suggests that we should choose $a = -\tau$.

By comparing the convergence diagrams Fig. 2 and Fig. 4 we see that with the correct choice of the interval, the Chebyshev version has faster convergence than the Taylor version.

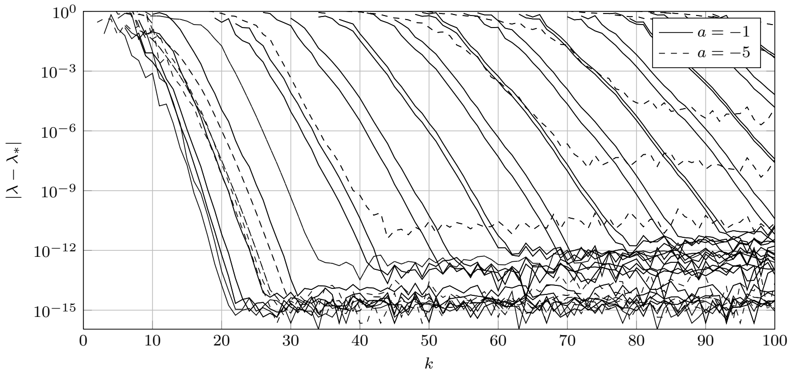


Fig. 4 Convergence history for the Chebyshev version of Algorithm 2 applied to the example in Section 6.1. The illustration shows two different choices of a. The first eigenvalue reaches accuracy 10^{-10} at $k = 17$ and $k = 23$ for $a = -1$ and $a = -5$ respectively. After $k = 80$ iterations, the method finds 30 and 10 eigenvalues (with error less that 10^{-10}) for $a = -1$ and $a = -5$ respectively.

6.2 A large nonlinear eigenproblem involving a square root

The standard Arnoldi method has turned out to be very useful for large eigenvalue problems. We will now illustrate that this appears to be the case also for Algorithm 2. We apply it to a non-standard nonlinearity, in this case a function which is not an entire function. With this we also wish to illustrate the generality of our approach. Let

$$T(\lambda) = A_0 - \lambda A_1 + i\sqrt{\lambda - \sigma_1^2}A_2 + i\sqrt{\lambda - \sigma_2^2}A_3,$$

where $\sigma_1 = 0$ and $\sigma_2 = 108.8774$. The notation complex square root, $\sqrt{\cdot}$ denotes the principal branch and the domain of interest are such that Re $(\lambda) > \sigma_1^2$, i.e., bounded away from the branch points $\lambda = 0$ and $\lambda = \sigma_2^2$. This problem appears in the simulation related to accelerator modeling in [15] and the sparse matrices $A_0, A_1, A_2, A_3 \in \mathbb{R}^{n\times n}$, where $n = 9956$, are available in the problem collection [4].

We shift and scale the problem by $\hat{\lambda} = \kappa\lambda + \mu$. The scaling is selected (to $\kappa = 300^2 - 200^2$) such that it corresponds to a transformation of the region of interest for a similar problem [15, Fig. 1] to be roughly within unit magnitude. In the standard (shift-and-invert) Arnoldi method, the general rule-of-thumb is to pick the shift close to the eigenvalues of interest. Note that T has branch points at $\lambda = \sigma_2^2$ and $\lambda = 0$, i.e., points where T is not analytic. We only have guaranteed convergence for eigenvalues within Ω which is small if σ is close to any of the branch points. Hence, when using Algorithm 2 on a problem where B is not an entire function, we additionally need to take into account that the region of guaranteed convergence is smaller when the shift is close to a branch point.

We carry out the algorithm for two different shifts in order to illustrate the importance of the shift and the region of guaranteed convergence. We use $\mu = \mu_0 = 146.71^2$, in the first run, since one eigenvalue of interest is close to this point [4]. Inspired by the region of interest for a similar problem [15, Fig. 1] we also carry out the algorithm with the shift $\mu = \mu_1 = 250^2$, which corresponds to a larger guaranteed region of convergence.

For completeness, we carry out the algorithm with the Taylor version as well as the Chebyshev version. The problem does not correspond to a differential equation on a finite interval and the reasoning in Section 5.3 does not provide a recommendation about how the interval should be chosen. For simplicity we use the unscaled Chebyshev polynomials ($I = [-1, 1]$) and note that the behavior of the algorithm for this problem is very similar for many other choices of the interval. The formula for y_0 was derived using the automatic symbolic procedure for $\hat{b}$, i.e., symbolic representation of Taylor coefficients and symbolic differentiation and the application of Theorem 9.

The convergence diagram for the Taylor version and Chebyshev version are given in Fig. 5 and Fig. 6. We use, as in [15], the quantity

$$E(\lambda, v) := \frac{\|T(\lambda)v\|_2}{\|A_0\|_1 + \|A_1\|_1|\lambda| + \sqrt{|\lambda|}\|A_2\|_1 + \sqrt{|\lambda - \sigma_2^2|}\|A_3\|_1}, \tag{41}$$

to measure the convergence. Similar convergence and properties are observed for both versions, with the exception that the Taylor version sometimes produced an overflow error in the evaluation of y_0.

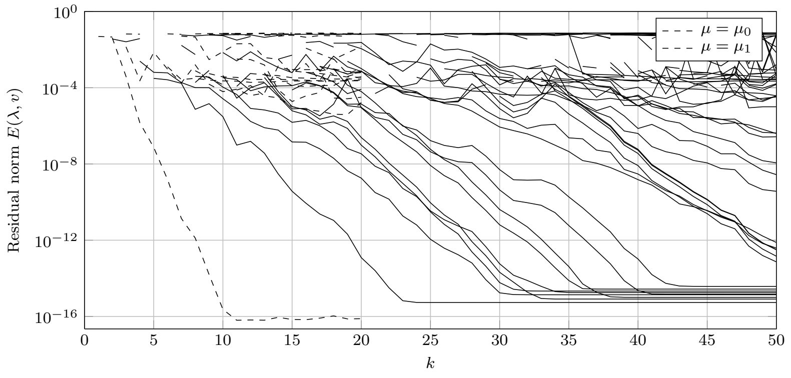


Fig. 5 Convergence history for the Taylor version of Algorithm 2 applied to the example in Section 6.2. The figure visualizes the simulation for two different shifts μ. The error indicator is the relative residual norm (41). Note that when $\mu = \mu_0$, overflow occurs (for $N = 21$) in the evaluation of y_0.

Note that when we select the shift $\mu = \mu_0$, only one eigenvalue has converged after 50 iterations and the convergence to the other eigenvalues appears stagnated. For the shift $\mu = \mu_1$, we find 23 eigenvalues accurately after $k = 50$ iterations. The dramatic difference in the shift, is actually quite natural when taking into account the eigenvalues and the region of guaranteed convergence, both visualized in Fig. 7. We clearly see that there is only one eigenvalue within Ω_0 the region of convergence for

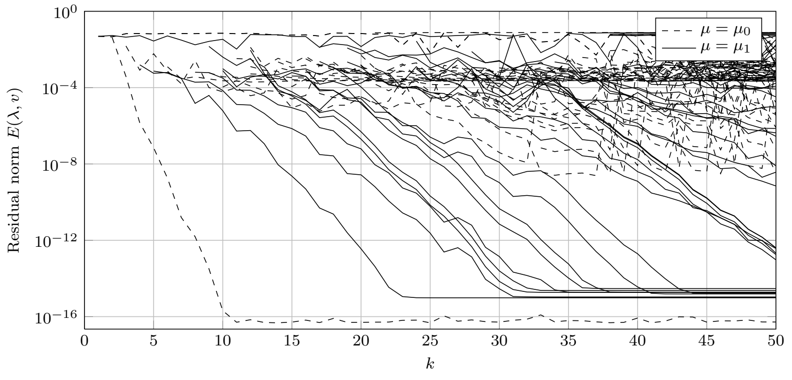


Fig. 6 Convergence history for the Chebyshev version of Algorithm 2 applied to the example in Section 6.2. The figure visualizes the simulation for two different shifts μ. The error indicator is the relative residual norm (41).

$\mu = \mu_0$, and the theory only supports the convergence to this eigenvalue. With the shift $\mu = \mu_1$ we successfully find the eigenvalues given in [15]. Note that we also find eigenvalues outside the region of guaranteed convergence Ω_1.

The computational effort is more or less the same for both shifts and both versions of the algorithm. The LU decomposition carried out before the iteration starts was done in 2.5s. The Arnoldi iteration (Algorithm 2 excluding LU decomposition) finished in 37.0s, of which the matrix vector product, i.e., computing y_0, in total took 5.7s and the orthogonalization 28.0s.

The robustness and attractive global convergence properties of Algorithm 2 for this example can be observed in two ways. The convergence shown in Fig. 5 and Fig. 6, behaves in a very regular way. In order to find more eigenvalues, we just have to carry out more iterations. In Fig. 7 we see that we find more eigenvalues in the region of interest than the local correction schemes used in [15]. This illustrates the property that Algorithm 2 is reliable in the sense that it is not likely to miss solutions. As usual, the local correction schemes, e.g., those in [15], are however likely to be faster.

7 Concluding remarks

The two most important properties of the algorithm we have presented here is that it is equivalent to the Arnoldi method and it is applicable to a wide class of NEPs. This has the nice consequence that many properties of the Arnoldi method are inherited. We also wish to point out that the Arnoldi method is well understood. The equivalence hence opens up possibilities to improve the method presented here in the same way the Arnoldi method has been improved.

The resources required for the orthogonalization is substantial and even dominating in the example in Section 6.2. Hence, it can be worthwhile to work with other scalar products. One may consider only using the first n components of the matrix of basis vectors for the orthogonalization. This would be cheaper, but it is in general not a scalar product but only a semidefinite bilinear form. In related methods, e.g., [3], this

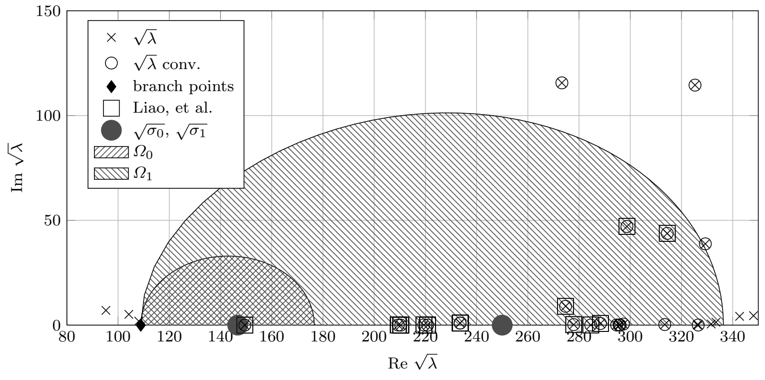


Fig. 7 The figure is a visualization of the simulations in Section 6.2 in a square-root-scale as in [15]. It shows the approximate eigenvalues, the shifts and the region of guaranteed convergence. There is apparently only one eigenvalue within the region of guaranteed convergence for $\mu = \mu_0 = 146.71^2$ (Ω_0). For $\mu = \mu_1 = 250^2$ all solutions (from [15]) within the region of guaranteed convergence Ω_1 are found.

type of orthogonalization is combined with the solving of a projected small nonlinear eigenvalue problem instead of computing the eigenvalues of a Hessenberg matrix.

Finally, we wish to point out that the main algorithm of this paper is a framework in the sense that it can be adapted to the problem at hand. In this paper we work out formulas for the scalar products associated with the monomials and Chebyshev polynomials and show that they both can be interpreted as Arnoldi's method on a companion matrix and the matrix stemming from a spectral discretization. As for the standard Arnoldi method, depending on the problem at hand, a modified scalar product may be more efficient. Our framework allows the use of different scalar products.

Acknowledgements This work has been supported 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 K.U. Leuven, and by the project STRT1-09/33 of the K.U. Leuven Research Council. We thank Dr. Ben-Shan Liao for the assistance in reproducing the data of [15] in Fig. 7.

References

1. W. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. appl. Math.*, 9:17–29, 1951.
2. J. Asakura, T. Sakurai, H. Tadano, T. Ikegami, and K. Kimura. A numerical method for nonlinear eigenvalue problems using contour integrals. *JSIAM Letters*, 1:52–55, 2009.
3. 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(3):640–659, 2005.
4. T. Betcke, N. J. Higham, V. Mehrmann, C. Schröder, and F. Tisseur. NLEVP: A collection of nonlinear eigenvalue problems. Technical report, Manchester Institute for Mathematical Sciences, 2008.
5. W. J. Beyn. An integral method for solving nonlinear eigenvalue problems. Technical report, Bielefeld University, 2010.
6. D. Breda, S. Maset, and R. Vermiglio. Pseudospectral approximation of eigenvalues of derivative operators with non-local boundary conditions. *Applied Numerical Mathematics*, 56:318–331, 2006.

7. H. Fassbender, D. Mackey, N. Mackey, and C. Schröder. Structured polynomial eigenproblems related to time-delay systems. *Electronic Transactions on Numerical Analysis*, 31:306–330, 2008.
8. I. Gohberg, P. Lancaster, and L. Rodman. *Matrix polynomials.* Academic press, 1982.
9. J. Hale and S. M. Verduyn Lunel. *Introduction to functional differential equations.* Springer-Verlag, 1993.
10. M. E. Hochstenbach and G. L. Sleijpen. Harmonic and refined Rayleigh-Ritz for the polynomial eigenvalue problem. *Numer. Linear Algebra Appl.*, 15(1):35–54, 2008.
11. E. Jarlebring, K. Meerbergen, and W. Michiels. A Krylov method for the delay eigenvalue problem. Technical Report TW558, K.U. Leuven, 2010. to appear in SIAM J. Sci. Comp.
12. D. Kressner. A block Newton method for nonlinear eigenvalue problems. *Numer. Math.*, 114(2):355–372, 2009.
13. D. Kressner, C. Schröder, and D. S. Watkins. Implicit QR algorithms for palindromic and even eigenvalue problems. *Numer. Algorithms*, 51(2):209–238, 2009.
14. R. Lehoucq, D. Sorensen, and C. Yang. *ARPACK user's guide. Solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods.* SIAM publications, 1998.
15. B.-S. Liao, Z. Bai, L.-Q. Lee, and K. Ko. Nonlinear Rayleigh-Ritz iterative method for solving large scale nonlinear eigenvalue problems. *Taiwanese Journal of Mathematics*, 14(3):869–883, 2010.
16. D. Mackey, N. Mackey, C. Mehl, and V. Mehrmann. Numerical methods for palindromic eigenvalue problems: Computing the anti-triangular Schur form. *Numer. linear Algebr.*, 16:63–86, 2009.
17. K. Meerbergen. The quadratic Arnoldi method for the solution of the quadratic eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 30(4):1463–1482, 2008.
18. V. Mehrmann and H. Voss. Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods. *GAMM Mitteilungen*, 27:121–152, 2004.
19. A. Neumaier. Residual inverse iteration for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 22:914–923, 1985.
20. G. Peters and J. Wilkinson. Inverse iterations, ill-conditioned equations and Newton's method. *SIAM Rev.*, 21:339–360, 1979.
21. A. Ruhe. Algorithms for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 10:674–689, 1973.
22. G. L. Sleijpen, A. G. Booten, D. R. Fokkema, and H. A. van der Vorst. Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems. *BIT*, 36(3):595–633, 1996.
23. Y. Su and Z. Bai. Solving rational eigenvalue problems via linearization. Technical report, Department of Computer Science and Mathematics, University of California, Davis, 2008.
24. L. N. Trefethen. *Spectral Methods in MATLAB.* SIAM Publications, Philadelphia, 2000.
25. H. Unger. Nichtlineare Behandlung von Eigenwertaufgaben. *Z. Angew. Math. Mech.*, 30:281–282, 1950. English translation: http://www.math.tu-dresden.de/~schwetli/Unger.html.
26. H. Voss. An Arnoldi method for nonlinear eigenvalue problems. *BIT*, 44:387 – 401, 2004.

A Computing y_0 for the Chebyshev basis

Consider one of the terms in (24), $b = b_j$ and define the vector,

$$\hat{b}^T := \left((b(\frac{d}{d\theta})\hat{T}_0)(0), \ldots, (b(\frac{d}{d\theta})\hat{T}_{N-1})(0) \right). \tag{42}$$

If we can compute this vector, we can evaluate (24) since one term in the outer sum of (24) can be expressed as,

$$\sum_{i=0}^{N-1} (b(\frac{d}{d\theta})\hat{T}_i x_i)(0) = (x_0, \ldots, x_{N-1})\hat{b}. \tag{43}$$

We present two procedures to compute the vector $\hat{b}$.

A.1 Computing $\hat{b}$ from the Taylor expansion

The function b is a scalar function and we first present results for the case where the Taylor expansion of b is available. The Taylor expansion can be computed, e.g., by hand, with methods for symbolic manipulations or using some high accuracy numerical approach. This computation can be done as a precomputation to Algorithm 2, and will for large-scale systems not dominate the computation-time of the algorithm.

In the following result we see how the vector $\hat{b}$ can be computed from the Taylor expansion of the b. When using Theorem 9 in practice it is advisable to represent the Taylor coefficient with high-precision arithmetic and also solve the linear systems in Theorem 9 with high-precision arithmetic.

Theorem 9 (Computing $\hat{b}$ from the Taylor expansion of b) *Let $\{b_j\}_0^\infty$ be the coefficients in the power series expansion of an arbitrary function $b : \mathbb{C} \to \mathbb{C}$, i.e.,*

$$b(\lambda) = \sum_{j=0}^{\infty} b_j \lambda^j.$$

Consider the matrix

$$Z_N = (z_0, \dots, z_{N-1}) \in \mathbb{R}^{N\times N},$$

with columns defined by

$$z_i = \begin{bmatrix} 0 \\ L_{C,N-1}^{-1} \end{bmatrix} \cdots \begin{bmatrix} 0 \\ L_{C,N-i}^{-1} \end{bmatrix} \begin{bmatrix} \hat{T}_0(0) \\ \vdots \\ \hat{T}_{N-i-1}(0) \end{bmatrix}.$$

Then,

$$\begin{bmatrix} (b(\frac{d}{d\theta})\hat{T}_0)(0) \\ \vdots \\ (b(\frac{d}{d\theta})\hat{T}_{N-1})(0) \end{bmatrix} = Z_N \begin{bmatrix} b_0 \\ \vdots \\ b_{N-1} \end{bmatrix}. \tag{44}$$

Proof First note that since $\hat{T}_i$ is a polynomial of order i, we only need a finite number of Taylor coefficients in the definition of $\hat{b}$,

$$\hat{b} = b_0 \begin{bmatrix} \hat{T}_0(0) \\ \vdots \\ \hat{T}_{N-1}(0) \end{bmatrix} + b_1 \begin{bmatrix} \hat{T}_0'(0) \\ \vdots \\ \hat{T}_{N-1}'(0) \end{bmatrix} + \cdots + b_{N-1} \begin{bmatrix} \hat{T}_0^{(N-1)}(0) \\ \vdots \\ \hat{T}_{N-1}^{(N-1)}(0) \end{bmatrix}. \tag{45}$$

We will now use the inverse of the integration map $L_{C,N}$ given in (21) in order to compute the derivatives. Consider only one term in (45) and apply $L_{C,i}^{-1}$ several times,

$$\begin{bmatrix} (\hat{T}_0^{(i)})(0) \\ \vdots \\ (\hat{T}_{N-1}^{(i)})(0) \end{bmatrix} = \begin{bmatrix} 0 \\ L_{C,N-1}^{-1} \end{bmatrix} \begin{bmatrix} (\hat{T}_0^{(i-1)})(0) \\ \vdots \\ (\hat{T}_{N-2}^{(i-1)})(0) \end{bmatrix} = \cdots = \begin{bmatrix} 0 \\ L_{C,N-1}^{-1} \end{bmatrix} \cdots \begin{bmatrix} 0 \\ L_{C,N-i}^{-1} \end{bmatrix} \begin{bmatrix} \hat{T}_0(0) \\ \vdots \\ \hat{T}_{N-i-1}(0) \end{bmatrix}. \tag{46}$$

The proof is completed by defining the matrix Z_N as the columns given by (46), $i = 0, \dots, N-1$ and using (45).

Used in NEP	$b(\lambda)$	$(b(\frac{d}{d\theta})\varphi)(0) = \sum_{i=0}^{N-1}(b(\frac{d}{d\theta})\hat{T}_i x_i)(0)$
GEP	1	$\sum_{i=0}^{N-1} T_i(c)x_i$
QEP	λ	$\sum_{i=1}^{N-1} kiU_{i-1}(c)x_i$
PEP	λ^p	$(x_0,\dots,x_{N-1})\begin{bmatrix}0\\L_{C,N-1}^{-1}\end{bmatrix}\cdots\begin{bmatrix}0\\L_{C,N-p}^{-1}\end{bmatrix}\begin{bmatrix}T_0(c)\\ \vdots\\ T_{N-p-1}(c)\end{bmatrix}$
DEP	$q(\lambda)$	$\sum_{i=1}^{N}(\hat{T}_i(0)-\hat{T}_i(-\tau))y_i$
Neutral DEP	$e^{-\tau\lambda}$	$\sum_{i=0}^{N-1}\hat{T}_i(-\tau)x_i$

Table 1 Formulas for scalar nonlinearities appearing in some common nonlinear eigenvalue problems: generalized eigenvalue problems (GEPs), quadratic eigenvalue problems (QEPs), polynomial eigenvalue problems (PEPs), delay eigenvalue problems (DEPs) and neutral DEPs. These are to be used in the derivation of expressions for y_0 in (22). The Chebyshev polynomials of the second kind are denoted U_i. The variables $x_0,\dots,x_{N-1}$,$y_1,\dots,y_N$ and $L_{C,i}$ are defined in Theorem 4 and (21), and k and c are the constants in the scaling of the Chebyshev polynomials defined in (20). The function $q(\lambda)$ is defined in (47).

A.2 Formulas for $\hat{b}$ for some common elementary functions

Direct manipulations of the definition of $\hat{b}$ in (43) and using properties of Chebyshev polynomials can in several situations result in quite simple formulas. The formulas for some common nonlinearities are summarized in Table 1. We briefly summarize the derivations.

The first three rows in Table 1 follow directly from Theorem 9. For the delay eigenvalue problem

$$T(\lambda) = -\lambda I + A_0 + A_1 e^{-\tau\lambda},$$

we find from (2) that

$$B(\lambda) = (A_0 + A_1)^{-1}(I_n + A_1 q(\lambda)) \text{ with } q(\lambda) := \frac{1-e^{-\tau\lambda}}{\lambda}. \tag{47}$$

We have tacitly defined $q(0)$ as the analytic extension of q. Now note that q can be interpreted as integration, in the sense that

$$\left(q(\frac{d}{d\theta})\varphi\right)(0) = \int_{-\tau}^{0}\varphi(\hat{\theta})\,d\hat{\theta}. \tag{48}$$

This can be established by comparing the terms in the Taylor expansion of the left and right-hand side in (48).

We will here use that $\psi = \mathcal{B}\varphi$ in Theorem 4 is given by the coefficients of y_i and is a primitive function of φ. Hence, we have that

$$\sum_{i=0}^{N-1}(q(\frac{d}{d\theta})\hat{T}_i x_i)(0) = \int_{-\tau}^{0}\varphi(\hat{\theta})\,d\hat{\theta} = \psi(0)-\psi(-\tau) = \sum_{i=1}^{N}\left(\hat{T}_i(0)-\hat{T}_i(-\tau)\right)y_i. \tag{49}$$

The nonlinear eigenvalue problem corresponding to time-delay systems known as *neutral* time-delay system have terms involving $\lambda e^{-\lambda\tau}$. By the transformation (3) we arrive at a decomposition (23) with a scalar nonlinearity $b(\lambda) = e^{-\tau\lambda}$. We derive it by forming the Taylor expansion of b from which it follows that,

$$(b(\frac{d}{d\theta})\varphi)(0) = \varphi(-\tau) = \sum_{i=0}^{N-1}\hat{T}_i(-\tau)x_i. \tag{50}$$

The last two rows of Table 1 follow from (49) and (50).

B Proof of Lemma 7

Let $t_N^T := (\hat{T}_0(0), \ldots, \hat{T}_N(0))$ and $X_k := (x_0, \ldots, x_k)$. From the definition of Σ_N and Π_N it follows that

$$\Pi_N \operatorname{vec}(x_0, \ldots, x_k, 0, \ldots, 0) = \operatorname{vec}(X_k t_k,\ X_k L_{k+1}). \tag{51}$$

and

$$\Sigma_N \operatorname{vec}(y_0, \ldots, y_{k+1}, 0, \ldots, 0) = \operatorname{vec}\left(R_0 y_0 + R_1 y_1 + \cdots + R_{k+1} y_{k+1},\ \ y_1, \ldots, y_{k+1}\right). \tag{52}$$

The equality of (51) and (52) can be interpreted as conditions on the functions φ and ψ. From the last $k+1$ block rows of (51) and (52) it follows that

$$\psi'(\theta) = \varphi(\theta) \tag{53}$$

and the first column correspondingly gives the condition that

$$\varphi(0) = (A(\frac{d}{d\theta})\psi)(0). \tag{54}$$

Consider the Taylor expansion of A, and denote the coefficients, $A(\lambda) = A_0 + \lambda A_1 + \lambda^2 A_2 + \cdots$. We now solve (54) for $\psi(0)$ and use (53),

$$\begin{aligned}\psi(0) = A_0^{-1}(\varphi(0) - A_1\psi'(0) - A_2\psi''(0) - \cdots)&\\ = A_0^{-1}(\varphi(0) - A_1\varphi(0) - A_2\varphi'(0) - \cdots).&\end{aligned} \tag{55}$$

When we insert the expansion of A into B in the definition (33) and compare with (55) we see that,

$$(B(\frac{d}{d\theta})\varphi)(0) = A(0)^{-1}(\varphi(0) - A_1\varphi(0) - A_2\varphi'(0) - \cdots) = \psi(0). \tag{56}$$

From (53) and (56) it follows that ψ is the action of $\mathcal{B}$ onto φ, i.e., (39) holds. This completes the proof.