A linear eigenvalue algorithm for the nonlinear eigenvalue problem

Elias Jarlebring
Wim Michiels
Karl Meerbergen

Report TW 580, October 2010

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

A linear eigenvalue algorithm for the nonlinear eigenvalue problem

Elias Jarlebring
Wim Michiels
Karl Meerbergen

Report TW 580, October 2010

Department of Computer Science, K.U.Leuven

Abstract

The *Arnoldi method* for standard eigenvalue problems possesses several attractive properties making it robust, reliable and efficient for many problems. Our first important result is a characterization of a general *nonlinear eigenvalue problem* (NEP) as a standard but *infinite dimensional* eigenvalue problem involving an integration operator denoted $\mathcal{B}$. In this paper we present a new algorithm equivalent to the Arnoldi method for the operator $\mathcal{B}$. Although the abstract construction is infinite dimensional, it turns out that we can carry out the iteration in an *exact way* (without approximation) by using only *standard linear algebra operations* involving matrices (not operators). This is achieved by working with coefficients in a basis of scalar functions, typically polynomials. Due to the fact that the constructed method has a complete equivalence with the standard Arnoldi method, it also inherits many of its attractive properties. Another somewhat unexpected consequence of the construction is that the matrix of basis vectors should be expanded not only in the way done in standard Arnoldi. We expand the matrix of basis vectors not only with a column to the right, but also a block row below. We also show that the method can be interpreted as the standard Arnoldi method if applied to the generalized eigenvalue problem resulting from the spectral discretization of the operator. With this equivalence we reach a recommendation on how the scalar product should be chosen for an important class of nonlinear eigenvalue problems.

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. Our first important result is a characterization of a general *nonlinear eigenvalue problem* (NEP) as a standard but *infinite dimensional* eigenvalue problem involving an integration operator denoted $\mathcal{B}$. In this paper we present a new algorithm equivalent to the Arnoldi method for the operator $\mathcal{B}$. Although the abstract construction is infinite dimensional, it turns out that we can carry out the iteration in an *exact way* (without approximation) by using only *standard linear algebra operations* involving matrices (not operators). This is achieved by working with coefficients in a basis of scalar functions, typically polynomials. Due to the fact that the constructed method has a complete equivalence with the standard Arnoldi method, it also inherits many of its attractive properties. Another somewhat unexpected consequence of the construction is that the matrix of basis vectors should be expanded not only in the way done in standard Arnoldi. We expand the matrix of basis vectors not only with a column to the right, but also a block row below. We also show that the method can be interpreted as the standard Arnoldi method if applied to the generalized eigenvalue problem resulting from the spectral discretization of the operator. With this equivalence we reach a recommendation on how the scalar product should be chosen for an important class of nonlinear eigenvalue problems.

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

1 Introduction

Suppose $M(\lambda) \in \mathbb{C}^{n\times n}$ is a given parameter dependent matrix. The *nonlinear eigenvalue problem* corresponding to M is typically defined as the problem of finding scalar values $\lambda \in \mathbb{C}$, called eigenvalues, such that $M(\lambda)$ is singular. We will equivalently say that we look for pairs $(\lambda, x) \in \mathbb{C} \times \mathbb{C}^n \backslash \{0\}$ fulfilling

$$M(\lambda)x = 0, \tag{1}$$

where x is called the corresponding eigenvector. This general problem has been extensively studied in the literature. See, e.g. [29, 23, 6] and the references therein.

In this paper we present an iterative method, which in theory converges to *all* solutions of (1) if M is sufficiently smooth, but favors the solutions closest to 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.

The first result of this paper (presented in Section 2) is that solutions to the nonlinear eigenvalue problem (1) are equal to the reciprocal of the eigenvalues of an operator $\mathcal{B}$. The operator $\mathcal{B}$ acts on (essentially) arbitrary functions and the action is defined by integrating the function and adding a constant. The constant is determined by the nonlinear eigenvalue problem.

We wish to carry out an infinite dimensional algorithm involving functions using only finite arithmetic. This is achieved by representing functions by (vector) coefficients with a given basis of functions. By assuming that the basis is such that integration can be done by a linear transformation on the coefficients, we show in Section 4 that the action of $\mathcal{B}$ on a function (expressed in the given basis) can be carried out with standard linear algebra operations, without any type of approximation. The only assumption on the basis functions is the integration property. We will however pay particular attention to the Chebyshev polynomial basis.

The method we present is based on the Arnoldi method [1]. We will implement an infinite dimensional Arnoldi process (summarized in Section 3) by using the action of $\mathcal{B}$ on functions and fixing an appropriate scalar product (in Section 5). Since we focus on representing functions with Chebyshev basis functions it turns out to be easy to use the scalar product corresponding to the Euclidean scalar product on the Chebyshev coefficients. In Section 6 we show that this choice is very natural for a wide class of problems since the resulting algorithm is identical to the standard Arnoldi method applied to the matrix stemming from a spectral discretization of the corresponding differential equation. This justifies the use of Chebyshev polynomials as basis functions and scalar product and the equivalence suggests how they should be scaled when applied to the nonlinear eigenvalue problem corresponding to linear functional differential equations.

The main algorithm (also presented in Section 5) possesses several somewhat unusual properties. It is by construction equivalent to a standard but infinite dimensional Arnoldi method, but can be implemented completely in finite arithmetic. Moreover, the structure of the algorithm is the same as the structure the standard Arnoldi method, except that the matrix of basis vectors needs to be expanded not only with a column but also n rows of zeros in each iteration. The equivalence with the standard Arnoldi method implies that we expect local linear (exponential) convergence and that it is robust in the sense that eigenvalues close to the origin will converge first but also eigenvalues not very close to the origin will eventually be found. These properties are illustrated in the examples section (Section 7).

There are many numerical methods for the nonlinear eigenvalue problem available in the literature. We now list some of the available numerical methods and discuss the relation with the presented approach. Many of the algorithms can be roughly classified into two types.

- There are methods for special classes of NEPs and structures. For instance, there are methods for the quadratic eigenvalue problem (QEP) [3, 24, 27, 22] and more generally the polynomial eigenvalue problem (PEP) [30, 14] but also other structures [33, 20, 11, 17]. In a sense, these approaches achieve desirable numerical prop-

erties by exploiting the particular structure of the nonlinear eigenvalue problem. In particular, they are often robust and have desirable *global* convergence properties.
- There are several locally convergent methods for general NEPs [28,35,26,16]. See also the summary of methods in [29,23]. These methods are often generally applicable and do not involve many assumptions on the problem. They often have good *local* convergence properties and when started sufficiently close to a solution the iteration often converges quickly. The drawback is that we are often interested in several solutions and with this type of approach one typically only finds one (or a few) of the solutions close to the initial value or a shift.

With our method we aim to achieve the positive properties of both of these classes of approaches. We wish to have good *global* convergence properties and still maintain generality such that it is applicable to essentially *arbitrary* nonlinear eigenvalue problems. This is achieved by maintaining an equivalence with the standard Arnoldi method, which in a sense already has these properties.

There are already other methods for NEPs motivated by the Arnoldi method. An important method is the nonlinear Arnoldi method in [36] which reduces to the standard Arnoldi method for the linear eigenvalue problem, i.e., if $M(\lambda) = A - \lambda I$. The method [36] does apparently not have an interpretation as a standard Arnoldi method for the general case. The algorithm has however turned out to be very successful in practice and robustness and a sense of global convergence can be achieved if a minmax characterization is found for the nonlinear eigenvalue problem. The method we present here is equivalent to an Arnoldi method for any nonlinearity and directly inherits the global convergence properties of the standard Arnoldi method.

Some methods for the general nonlinear eigenvalue problems are based on a formulation of a representation involving a contour integral [2,7]. Although one formulation of the method we present involves integration, there appears to be no simple connection with these results. Our method is based on the Arnoldi method in an operator setting by (exact) integration of functions and the methods in [2,7] are based on (numerical) integration of contour integrals.

The algorithm we present here can be seen as an iteration in function spaces and we focus on representing functions using Chebyshev polynomials. There is a software package called chebfun [5,10] which also uses a representation of functions with Chebyshev polynomials. In our approach, the full algorithm is an iteration equivalent to the Arnoldi method, which only involves operations on vectors and uses the Euclidean scalar product. In this way, the method is based on purely algebraic operations, although at each point there is an interpretation in terms of functions. With this approach there is no overhead in the representation of functions and the scalar product is the simplest possible. This allows us to solve large scale problems. In Section 7 we successfully solve a nonlinear eigenvalue problem of dimension $n = 9956$.

2 Operator formulation

There are a number of eigenvalue algorithms for the standard eigenvalue problem which in some way focus on first finding eigenvalues close to the origin. In this methods, it is common to reformulate the eigenvalue problem by considering the inverse matrix, resulting in methods like *inverse iteration* (see e.g. [28]). We will use a similar con-

struction. For our purposes it will be easier to work with the form,

$$\lambda B(\lambda)x = x, \tag{2}$$

where the elements of $B(\lambda)$ are assumed to be analytic at the origin. Note that this is an algebraic reformulation of the standard formulation (1) if we assume that $\lambda = 0$ is not an eigenvalue. We will typically use the transformation

$$B(\lambda) = M(0)^{-1}\frac{M(0) - M(\lambda)}{\lambda}, \tag{3}$$

where, without restriction, we can define $B(0) := -M(0)^{-1}M'(0)$ by analytic continuation. The elements of B are entire functions in many applications. As usual for standard eigenvalue problems, we will only use the matrix inverse as a theoretical tool. The inverse is never explicitly computed in the algorithm. Instead we use the common approach to solve corresponding linear systems possibly using a pre-computed factorization.

The nonlinear eigenvalue problem (2) involves a matrix of dimension n depending on λ in a nonlinear way. We will now see how this (arbitrary) nonlinearity can be transformed into a linear first-order representation with a linear infinite dimensional operator, denoted $\mathcal{B}$, which reciprocal eigenvalues are the solutions to (2).

The characterization exploits the smoothness of B and the neighborhood at the origin in which it is analytic, will be important. Let Ω denote any (large) closed disc centered at the origin in which B is analytic. Since the radius of the disc corresponds to the convergence radius of the Taylor expansion, we have that

$$B(\lambda) = \sum_{i=0}^{\infty} B_i \lambda^i, \tag{4}$$

if $\lambda \in \Omega$, where $B_0, B_1, \ldots$ are the coefficients of B given by the Taylor expansion. Now consider the following operator defined using the power series expansion (4). The definition is followed by the first result of the paper stating that in the region Ω the reciprocal eigenvalues of the operator are the solutions to (2).

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

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

where

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

Theorem 1 (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 (2).
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 $\varphi \in \mathcal{D}(\mathcal{B})$ fulfills (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 solution to the differential equation $\lambda\varphi = \varphi'$ are of the form $\varphi(\theta) = xe^{\lambda\theta}$.

In order to show that i) implies ii), suppose (λ, x) is a solution to (2). Let $\varphi(\theta) := xe^{\lambda\theta}$ and note that $\varphi \in \mathcal{D}(\mathcal{B})$ since

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

exists. We here used that $\lambda \in \Omega$ implies that the series is convergent. From the fact that an eigenfunction takes the form $\varphi(\theta) = xe^{\lambda\theta}$ it follows that the derivative of (7) holds for any θ. 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 (2). 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 (2), we have completed the proof.

Remark 1 (Connection with differential equation in work by Lancaster and Gohberg) In several works of Lancaster and Gohberg, e.g., [12], 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 1 as follows. The operator $\mathcal{B}$ and Theorem 1 corresponds to an (integration) operator formulation of the differential equation associated with the polynomial (or nonlinear) eigenvalue problem.

Remark 2 (The delay eigenvalue problem) Some results in this paper are true generalizations of the results for time-delay systems in [15]. The operator $\mathcal{B}$ can be interpreted in the setting of time-delay systems (and in particular [15]) as follows. The characteristic equation of a time-delay system with a single delay can be written as a nonlinear eigenvalue problem with an exponential term,

$$M(\lambda) = -\lambda I_n + A_0 + A_1e^{-\tau\lambda}.$$

We bring the nonlinear eigenvalue problem to the form (2) using (3) and have that

$$B(\lambda) = (A_0 + A_1)^{-1}(I_n + A_1 q(\lambda)) \text{ with } q(\lambda) := \frac{1 - e^{-\tau\lambda}}{\lambda}. \tag{10}$$

Throughout this paper we will tacitly define $q(0)$ as the analytic extension of q. In order to explicitly form $\mathcal{B}$ we need the relation

$$\left(q(\frac{d}{d\theta})\varphi\right)(0) = \int_{-\tau}^{0} \varphi(\theta)\, d\theta, \tag{11}$$

which is easy to show by inserting the Taylor expansion of q and φ into (11) and comparing coefficients. The action of the operator is now given by,

$$(\mathcal{B}\varphi)(\theta) = \int_0^\theta \varphi(\theta)\, d\theta + (A_0 + A_1)^{-1} \left(\varphi(0) + A_1 \int_{-\tau}^{0} \varphi(\theta)\, d\theta \right). \tag{12}$$

By straightforward calculations we find that the inverse of $\mathcal{B}$ is

$$\begin{aligned} \mathcal{D}(\mathcal{B}^{-1}) &= \{\varphi : \varphi(\theta) - \varphi(0) + (A_0 + A_1)^{-1} \left[\varphi'(0) + A_1(\varphi(0) - \varphi(-\tau))\right] = \varphi(\theta)\} \\ \mathcal{B}^{-1}\varphi &= \varphi'. \end{aligned}$$

We rearrange and cancel some terms in the domain condition of $\mathcal{B}^{-1}$ and find that $\varphi \in \mathcal{D}(\mathcal{B}^{-1})$ is equivalent to

$$\varphi'(0) = A_0\varphi(0) + A_1\varphi(-\tau). \tag{13}$$

An operator defined by differentiation action and the domain given by (13) is in the field of time-delay systems known as *the infinitesimal generator* [13] and [25, Chapter 1]. With this we have shown the following. If $A_0 + A_1$ is non-singular, then the operator $\mathcal{B}$ associated with the nonlinear eigenvalue problem (10) corresponding to a time-delay system, equals the inverse of the infinitesimal generator of the time-delay system. Hence, the operator $\mathcal{B}^{-1}$ is a generalization of the infinitesimal generator which is a starting point in [15].

3 The Arnoldi method in a function setting

One reason for the success of the Arnoldi method (first presented in [1]) is that the only way the matrix appears in the algorithm is in combination with multiplication of the matrix and a vector. The method is therefore particularly suited for problems where the matrix vector product is simple or computationally cheap. Correspondingly, we now have an operator $\mathcal{B}$ with an action which is quite simple, and it will turn out that we can construct the Arnoldi algorithm for $\mathcal{B}$. In this section we introduce the Arnoldi method for the operator $\mathcal{B}$ in an abstract setting. In later sections we will show how this infinite dimensional algorithm can be implemented (without approximation) in finite arithmetic.

Recall that our ultimate goal is to construct a method which favors solutions of (2) 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 1 that the reciprocal eigenvalues of $\mathcal{B}$ are solutions

to (2) and the reciprocal of extreme isolated eigenvalues are usually indeed the solutions λ close to origin. This is consistent with the common construction for matrices of shift-invert Arnoldi method, e.g., used in the software package ARPACK [18].

The span of the elements of a power sequence, known as a *Krylov subspace*, is fundamental in the Arnoldi method. In this operator setting it is defined as,

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

where $\varphi \in \mathcal{D}(\mathcal{B})$. The Arnoldi method can be seen 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 scalar product. In the standard (finite dimensional) Arnoldi method, the natural scalar product is the Euclidean scalar product. In this infinite dimensional construction, there are several natural choices for the scalar product and we will select a discuss a scalar product in Section 5. In an abstract setting with a fixed 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 on a function

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

The Arnoldi method in an infinite dimensional operator setting is by no means new. In some literature, this type of construction is treated as an *orthogonal Galerkin projection* and has been studied for many decades. See the bibliographic references in [9, pg 178] and the discussion of projection methods for operators in [9, Chapter 4]. The construction with the operator $\mathcal{B}$ corresponding to a nonlinear eigenvalue problem and the result that Algorithm 1 can, without approximation, be implemented in finite arithmetic (which we shall show in the following sections) is to our knowledge new.

4 A coefficient map representation of $\mathcal{B}$

In Step 2 of Algorithm 1 we need to compute the action of $\mathcal{B}$ applied to a function. In order to construct a finite arithmetic version of Algorithm 1 we will now show how

the action of $\mathcal{B}$ can be implemented in finite arithmetic, if we work with coefficients in a function basis. We give the action for a general set of basis functions in Section 4.1 and show how this can be specialized when working with Chebyshev polynomials in Section 4.2 and Section 4.3.

4.1 A general coefficient map

Consider an infinite sequence of functions $q_i : \mathbb{C} \to \mathbb{C}$, $i \in \mathbb{N}$, possessing the property that the functions can be integrated by a linear transformation. It turns out that we can express the action of $\mathcal{B}$ by representing the function φ in such a basis. More precisely, the transformation of the coefficients can be expressed as follows.

Lemma 1 (General coefficient map) *Let $\{q_i\}_{i=0}^{\infty}$ be a sequence of analytic functions such that $\{q_i\}_{i=0}^{N}$ is a linearly independent set for any $N \in \mathbb{N}$. Moreover, suppose that $q_0(\theta) \equiv 1$ and that the sequence of functions has an integration map $L_{N,N} \in \mathbb{R}^{N\times N}$, defined by*

$$\begin{pmatrix} q_0(\theta) \\ q_1(\theta) \\ \vdots \\ q_{N-1}(\theta) \end{pmatrix} = L_{N,N} \begin{pmatrix} q_1'(\theta) \\ q_2'(\theta) \\ \vdots \\ q_N'(\theta) \end{pmatrix}. \tag{14}$$

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 functions φ,

$$\varphi(\theta) =: \sum_{i=0}^{N-1} q_i(\theta) x_i. \tag{15}$$

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

$$(y_1, \ldots, y_N) = X L_{N,N}, \tag{16}$$

and

$$y_0 = \left(\sum_{i=0}^{N-1} B(\frac{d}{d\theta}) q_i(\theta) x_i \right)(0) - \sum_{i=1}^{N} q_i(0) y_i. \tag{17}$$

Proof From the expansion of φ, i.e., (15), and the integration map $L_{N,N}$ we find that

$$\int_0^\theta \varphi(\theta)\, d\theta = \int_0^\theta X \begin{pmatrix} q_0(\theta) \\ \vdots \\ q_{N-1}(\theta) \end{pmatrix} d\theta = X L_{N,N} \begin{pmatrix} q_1(\theta) \\ \vdots \\ q_N(\theta) \end{pmatrix} - X L_{N,N} \begin{pmatrix} q_1(0) \\ \vdots \\ q_N(0) \end{pmatrix}. \tag{18}$$

We can now insert (18) into (5) and find that

$$(y_0, \ldots, y_N) \begin{pmatrix} q_0(\theta) \\ \vdots \\ q_N(\theta) \end{pmatrix} = \left(C(\varphi) - XL_{N,N} \begin{pmatrix} q_1(0) \\ \vdots \\ q_N(0) \end{pmatrix}, XL_{N,N} \right) \begin{pmatrix} q_0(\theta) \\ \vdots \\ q_N(\theta) \end{pmatrix}, \tag{19}$$

where we used that $q_0(\theta) := 1$. Note that the functions $q_0, \ldots, q_N$ are (by assumption) linearly independent. Hence, the matrix in front of the coefficients on the left-hand side in (19) equals the matrix in front of the coefficients on the right-hand side. The relation (16) follows from the corresponding last N columns and (17) follows from the first column and (16).

4.2 Chebyshev coefficient map

The basis used to represent functions and polynomials should be chosen carefully. An inappropriate choice can have severe impact in practice since the sensitivity of quantities with respect to perturbations in the coefficients may be very large. Although our method and most of our results are applicable to an arbitrary basis, we will here present more details for the method when using the (scaled and shifted) Chebyshev polynomials $\hat{T}_i$ for an interval $I = [a, b] \subset \mathbb{R}$, defined by

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

We pay special attention to Chebyshev polynomials since we will later (in Section 6.1) make a connection with a spectral discretization approach where the use of Chebyshev polynomials is well established. This connection also allows us to derive suggestions for how the interval I should be chosen. Our general experience with Chebyshev polynomials in numerical examples is also positive for examples where the theory in Section 6.1 is not applicable.

In the following result, which follows from Lemma 1, we have a formula for the Chebyshev coefficients of the function $\mathcal{B}\varphi$ in terms of the Chebyshev coefficients of the function φ.

Theorem 2 (Chebyshev coefficient mapping) *Let φ denote an arbitrary vector of polynomials of degree N and $(x_0, \ldots, x_{N-1}) =: X$ the corresponding coefficients in a Chebyshev basis, for Chebyshev polynomials scaled to the interval $I = [a, b]$, i.e.,*

$$\varphi(\theta) =: \sum_{i=0}^{N-1} \hat{T}_i(\theta) x_i.$$

Moreover, let $y_0, \ldots, y_N$ denote the coefficients of $\mathcal{B}\varphi$, i.e.,

$$(\mathcal{B}\varphi)(\theta) =: \sum_{i=0}^{N} \hat{T}_i(\theta) y_i.$$

Then,

$$(y_1, \ldots, y_N) = XL_{N,N}, \tag{21}$$

where

$$L^T_{N,N} = \frac{b-a}{4}\begin{pmatrix} 2 & 0 & -1 & & & \\ \frac{1}{2} & 0 & -\frac{1}{2} & & & \\ & \frac{1}{3} & 0 & \ddots & & \\ & & \frac{1}{4} & \ddots & -\frac{1}{N-2} & \\ & & & \ddots & 0 & \\ & & & & \frac{1}{N} & \end{pmatrix} \tag{22}$$

and

$$y_0 = \left(\sum_{i=0}^{N-1} B(\frac{d}{d\theta})\hat{T}_i(\theta)x_i\right)(0) - \sum_{i=1}^{N} T_i(c)y_i. \tag{23}$$

Proof The proof consists of deriving an integration map for the Chebyshev polynomials and invoking Lemma 1. We use properties of the Chebyshev polynomials of the section kind, denoted U_i, in order to derive the integration map. In particular, the following properties will be used

$$T_i'(t) = iU_{i-1}(t), \quad i \geq 1 \tag{24}$$
$$T_0(t) = U_0(t) \tag{25}$$
$$T_1(t) = \frac{1}{2}U_1(t) \tag{26}$$
$$T_i(t) = \frac{1}{2}(U_i(t) - U_{i-2}(t)), \quad i \geq 2. \tag{27}$$

We can now form the relation between the derivatives as follows,

$$\begin{pmatrix} \hat{T}_0(\theta) \\ \vdots \\ \hat{T}_{N-1}(\theta) \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2 & & & \\ -1 & 1 & & \\ \ddots & \ddots & \ddots & \\ & -1 & 1 \end{pmatrix}\begin{pmatrix} U_0(k\theta + c) \\ \vdots \\ U_{N-1}(k\theta + c) \end{pmatrix} =$$
$$\frac{1}{2k}\begin{pmatrix} 2 & & & \\ -1 & 1 & & \\ \ddots & \ddots & \ddots & \\ & -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & & \\ & \frac{1}{2} & \\ & \ddots & \\ & & \frac{1}{N} \end{pmatrix}\begin{pmatrix} \hat{T}_1'(\theta) \\ \vdots \\ \hat{T}_N'(\theta) \end{pmatrix}. \tag{28}$$

In the first equality we used (25),(26),(27) and for the last equality (24). The equation (28) is an integration map corresponding to L_N in Lemma 1. The proof is completed by invoking Lemma 1 and inserting the Chebyshev polynomials for y_0 into (17).

4.3 Computing y_0 for Chebyshev polynomials

Since we want to implement our algorithm with arithmetic operations on a computer, we need to able to evaluate (23), i.e., compute y_0 in Theorem 2, for a given problem

B. The last term in (23) is already easy to evaluate and it remains to study the first term,

$$\sum_{i=0}^{N-1}((B(\frac{d}{d\theta})\hat{T}_i)x_i)(0) = \sum_{j=0}^{m} B_j \sum_{i=0}^{N-1}\left(b_j(\frac{d}{d\theta})\hat{T}_i x_i\right)(0), \tag{29}$$

where we have denoted

$$B(\lambda) = B_0 b_0(\lambda) + \cdots + B_m b_m(\lambda), \;\; b_i : \mathbb{C} \to \mathbb{C}, \;\; i = 0, \ldots, m. \tag{30}$$

Note that (30) is not a restriction of generality since $B_0, \ldots, B_m \in \mathbb{C}^{n\times n}$ can be chosen as the n^2 unit matrices with $m = n^2 - 1$. Consider one of the terms in (29), $b = b_j$ and define the vector,

$$\hat{b}^T := \left((b(\frac{d}{d\theta})\hat{T}_0)(0), \ldots, (b(\frac{d}{d\theta})\hat{T}_{N-1})(0)\right). \tag{31}$$

If we can compute this vector, we can evaluate (29) since one term in the outer sum of (29) can be expressed as,

$$\sum_{i=0}^{N-1}(b(\frac{d}{d\theta})\hat{T}_i x_i)(0) = (x_0, \ldots, x_{N-1})\hat{b}. \tag{32}$$

Hence, the problem of computing y_0 is solved if we can compute $\hat{b}$ for every scalar nonlinearity $b = b_j$. In the following result we see how the vector $\hat{b}$ can be computed from the Taylor expansion of the b.

Theorem 3 (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

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

with columns defined by

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

Then,

$$\begin{pmatrix} (b(\frac{d}{d\theta})\hat{T}_0)(0) \\ \vdots \\ (b(\frac{d}{d\theta})\hat{T}_{N-1})(0) \end{pmatrix} = Z_N \begin{pmatrix} b_0 \\ \vdots \\ b_{N-1} \end{pmatrix}. \tag{33}$$

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 $\hat{b}$,

$$\hat{b} = b_0 \begin{pmatrix} \hat{T}_0(0) \\ \vdots \\ \hat{T}_{N-1}(0) \end{pmatrix} + b_1 \begin{pmatrix} \hat{T}_0'(0) \\ \vdots \\ \hat{T}_{N-1}'(0) \end{pmatrix} + \cdots + b_{N-1} \begin{pmatrix} \hat{T}_0^{(N-1)}(0) \\ \vdots \\ \hat{T}_{N-1}^{(N-1)}(0) \end{pmatrix}. \tag{34}$$

We will now use the inverse of the integration map $L_{N,N}$ given in (22) in order to compute the derivatives. Consider only one term in (34) and apply $L_{i,i}^{-1}$ several times,

$$\begin{pmatrix} (\hat{T}_0^{(i)})(0) \\ \vdots \\ (\hat{T}_{N-1}^{(i)})(0) \end{pmatrix} = \begin{pmatrix} 0 \\ L_{N-1,N-1}^{-1} \end{pmatrix} \begin{pmatrix} (\hat{T}_0^{(i-1)})(0) \\ \vdots \\ (\hat{T}_{N-2}^{(i-1)})(0) \end{pmatrix} = \cdots = \\ \begin{pmatrix} 0 \\ L_{N-1,N-1}^{-1} \end{pmatrix} \cdots \begin{pmatrix} 0 \\ L_{N-i,N-i}^{-1} \end{pmatrix} \begin{pmatrix} \hat{T}_0(0) \\ \vdots \\ \hat{T}_{N-i-1}(0) \end{pmatrix}. \tag{35}$$

The proof is completed by defining the matrix Z_N as the columns given by (35), $i = 0, \ldots, N-1$ and using (34).

The theorem above directly yields formulas for $\hat{b}$ for several cases. We summarize some useful formulas in Table 1. The first three rows in Table 1 follow directly from Theorem 3. Further analysis is needed for the delay eigenvalue problem. We again consider B corresponding to a time-delay system (10). We saw in Remark 2 that B could be expressed using a function q in (12).

We will here use that $\psi = \mathcal{B}\varphi$ in Theorem 2, is given by the coefficients of y_i and is a primitive function of φ. Hence, with $b = q$ in (23) and using that y_i correspond to the coefficients of a primitive function of φ, we have that

$$\sum_{i=0}^{N-1} (q(\frac{d}{d\theta})\hat{T}_i x_i)(0) = \int_{-\tau}^{0} \varphi(\theta)\, d\theta = \psi(0) - \psi(-\tau) = \sum_{i=1}^{N} \left(\hat{T}_i(0) - \hat{T}_i(-\tau)\right) y_i. \tag{36}$$

We can generalize the method to neutral time-delay systems by deriving the formula corresponding to $b(\lambda) = e^{-\tau\lambda}$. We derive it by forming the Taylor expansion of b from which it follows that,

$$(b(\frac{d}{d\theta})\varphi)(0) = \varphi(-\tau) = \sum_{i=0}^{N-1} \hat{T}_i(-\tau) x_i. \tag{37}$$

The last two rows of Table 1 follow from (36) and (37).

Remark 3 (Computing y_0) We propose two ways to compute and find formulas for y_0. For each nonlinearity $b = b_j$, $j = 0, \ldots, m$ one can either take an algebraic approach or use symbolic software.

i) We saw in Table 1 that for some common choices of b, there is an explicit simple analytic expression. The table is not exhaustive and in a given situation it is recommended to first attempt algebraic derivation starting from the definition of $\hat{b}$ and use (32) to compute y_0.

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	$(x_0,\dots,x_{N-1})\begin{pmatrix}0\\ L^{-1}_{N-1,N-1}\end{pmatrix}\cdots\begin{pmatrix}0\\ L^{-1}_{N-p,N-p}\end{pmatrix}\begin{pmatrix}T_0(c)\\ \vdots\\ T_{N-p-1}(c)\end{pmatrix}$
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 (23). 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_i are defined in Theorem 2, and k and c are the constants in the scaling of the Chebyshev polynomials defined in (20).

ii) The Taylor expansion of a function is often quite easy to compute by hand. Theorem 3 provides a direct way to get $\hat{b}$ by multiplying the Taylor coefficients with the triangular matrix Z_N. Note that numerical stability issues have to be taken into account with this approach. There is a high risk of cancellation effects. The elements of Z_N grow exponentially, the Taylor coefficients decay exponentially and the elements of $\hat{b}$ typically increase exponentially. This process can however still be completely automated by using high precision arithmetic combined with software for symbolic manipulations. In the example in Section 7.2 we use this approach to compute the 50 first Taylor coefficients and $\hat{b}$ for a nonlinear eigenvalue problem involving a square-root function. The corresponding matrix Z_N is also computed with software for high precision arithmetic and the high precision values of $\hat{b}$ are rounded to standard (double) precision before executing the algorithm.

Note also that in many situations the coefficients B_i involve an inverse, which should not be computed explicitly. It is often possible to rearrange the operations such that we only need to solve one linear system for each evaluation of y_0.

5 Finite arithmetic implemenation

The main algorithm of this paper is a finite arithmetic implementation of Algorithm 1. We already saw above that the action of $\mathcal{B}$ can be carried out in finite arithmetic if we work with coefficients in a function basis, where we pay special attention to the Chebyshev polynomials. The remaining infinite dimensional operation in Algorithm 1 is the scalar product. Since the functions will be represented as coefficients in a basis, a simple scalar product (in terms of algebraic operations) is the Euclidean scalar product

with the coefficients. In other words, we define the scalar product of the functions

$$\varphi(\theta) = \sum_{i=0}^{\infty} \hat{T}_i(\theta) x_i, \quad \psi(\theta) = \sum_{i=0}^{\infty} \hat{T}_i(\theta) y_i,$$

by

$$< \varphi, \psi > = < \varphi, \psi >_C := \sum_{i=0}^{\infty} x_i^H y_i. \tag{38}$$

Further interpretations of the scalar product (38) will given in Section 6.

Suppose we start Algorithm 1 with a constant function $\varphi_1(\theta) = x_0$. By a simple induction argument, we find that the sequence of functions $\varphi_1, \ldots, \varphi_k$ are polynomials of increasing order. This is due to the fact that ψ (constructed in Step 2) is the integration of φ_k and a polynomial of one order higher than φ_k. The orthogonalization steps (Step 4-5) do not change the order of the polynomials since we always form linear combination with previous iterates, i.e., linear combinations with polynomials of lower order.

We will now stack the coefficients of the functions $\varphi_1, \ldots, \varphi_k$ into the columns of the matrix $V_k \in \mathbb{C}^{kn \times k}$, where we truncated the matrix such that it is an upper block triangular matrix. Since we only wish to store the non-zero coefficients, we need, in order to carry out one more iteration of Algorithm 2, to increase the size of the matrix by one column to the right as well as one block vector below.

Due to the fact that (38) is the Euclidean scalar product with respect to the coefficients, we can simplify the orthogonalization process. In fact, the Gram-Schmidt process can be done using the matrix of basis vectors V_k as in standard Arnoldi. With the construction above and the evaluation of the action of $\mathcal{B}$ in Section 4, we have reached the main algorithm of the paper, summarized in Algorithm 2. The meaning of the equivalence with Algorithm 1 is made precise in the theorem that follows.

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 $\mathrm{vec}(X) = v_k$
4: Compute $y_1, \ldots, y_{k+1}$ according to (21) with sparse L_k
5: Compute y_0 according to (23)
6: Expand V_k with one block row (zeros)
7: Let $w_k := \mathrm{vec}(y_0, \ldots, y_{k+1})$, compute $h_k = V_k^* w_k$ 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 $\{1/\mu_i\}_{i=1}^k$

Theorem 4 (Equivalence between Algorithm 1 and Algorithm 2) *The result of k steps of Algorithm 2 started with x_0 is equivalent to k steps of Algorithm 1 with the scalar product*

$$< \varphi, \psi > = < \varphi, \psi >_C,$$

and started with the constant function $\varphi_1(\theta) = x_0$. The equivalence holds in the sense that the Hessenberg matrices are equal and the orthogonal basis functions $\varphi_1, \ldots, \varphi_k$ are equal to the functions stemming from the interpretation of the blocks in the basis matrix as coefficients in a Chebyshev basis.

Remark 4 (Implementation issues) Several implementation issues need to be taken into account when implementing Algorithm 2. We use the same techniques for eigenvector extraction, reorthogonalization, stopping criteria and related issues as described in [15, Section 3.2].

Remark 5 Unlike the standard Arnoldi method, Algorithm 2 never breaks down. The matrix V_k grows with one vector in the bottom right corner which is always non-zero and the new vector is hence never in the subspace spanned by the previous iterates.

This has the somewhat remarkable consequence that when we apply the method to a problem with a finite number of solutions, the method will eventually give approximations which do not correspond to solutions of the problem. For instance, PEPs have a finite number of solutions. It is easy to show that for the scalar case with a PEP of order m, the Hessenberg matrix has $k - m$ zero eigenvalues after $k > m$ iterations. For the non-scalar case, we observe similarly that some eigenvalues of the Hessenberg matrix are very small in magnitude. Note that a PEP is in a sense a finite dimensional eigenvalue problem since it can be transformed to a standard eigenvalue problem with a companion linearization. In the operator formulation (Theorem 1) we embed the problem into an infinite dimensional standard eigenvalue problem. The existence of approximations corresponding to zero eigenvalues of the Hessenberg matrix, i.e., "infinite" eigenvalues of the nonlinear eigenvalue problem (2), is a natural consequence of the infinite dimensional embedding of a finite dimensional problem.

Note that these spurious solutions do not have a substantial impact on the reliability of the method in general. In a post-processing step we discard solutions for which the residual norm is too large, in the same way done in [15, Remark 3.2]. The spurious roots will not have a small residual.

6 Discretization interpretation and the scalar product

The choice of scalar product for the standard Arnoldi method is in general a difficult problem and still an active topic of research (see e.g. [32, 21, 4]). There currently appears to exist no final answer of the choice of the scalar product and the motivations for different problems are based on quite different strategies. Here, we motivate the choice of the scalar product $< \cdot, \cdot >_C$ and establish an appropriate interval for the Chebyshev polynomials when applying Algorithm 2 to nonlinear eigenvalue problems stemming from a functional differential equation. This is achieved by making a connection with a spectral discretization approach. In Section 6.1 we present a discretization and in Section 6.2 we show how the discretized problem is related to Algorithm 2. We make conclusions about the scalar product and the choice of the interval for the Chebyshev polynomials in Section 6.3.

6.1 A spectral discretization

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

$$\lambda x = A(\lambda)x. \tag{39}$$

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

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

Here, by FDE we mean (as usual in e.g. [13])

$$\dot{z}(t) = f(z_t) = (A(\frac{d}{d\theta})z_t)(0), \tag{41}$$

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 (41) via (39), 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 $M(\lambda) = A(\lambda) - \lambda I_n$ and use the tranformation (3) we have that,

$$B(\lambda) = A(0)^{-1}\frac{A(0) - A(\lambda) + \lambda I_n}{\lambda}. \tag{42}$$

With this explicit form of B we can carry out Algorithm 2 by appropriately deriving formulas for y_0 as in Section 4.3. We will now see that the resulting algorithm can be interpreted in a different way.

One common approach to compute the eigenvalues of FDEs similar to (41) consists of doing a spectral discretization of the corresponding operator. This approach is taken in, e.g., [8]. We will use a discretization very similar to [15, Section 2] and only point out the elements of the derivation which need to modified. The FDE (41) 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 [15, Section 2.3]. The grid is given by,

$$\theta_i = \frac{\alpha_i - c}{k}, \quad \alpha_i = \cos\frac{\pi i}{N+1}, i = 1, \ldots, N \text{ and } \theta_{N+1} = 0, \tag{43}$$

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 derivation of the discretization [15, Section 2.3] can be followed and result in the discretized eigenvalue problem

$$(\lambda\Pi_N - \Sigma_N)z = 0, \quad z \neq 0, \tag{44}$$

where

$$\Pi_N = \frac{b-a}{4} \begin{pmatrix} \frac{4}{b-a}\hat{T}_0(0) & \frac{4}{b-a}\hat{T}_1(0) & \frac{4}{b-a}\hat{T}_2(0) & \cdots & \frac{4}{b-a}\hat{T}_{N-1}(0) & \frac{4}{b-a}\hat{T}_N(0) \\ 2 & 0 & -1 & & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & & \\ & & \frac{1}{3} & 0 & \ddots & \\ & & & \ddots & \ddots & -\frac{1}{N-1} \\ & & & & \frac{1}{N} & 0 \end{pmatrix} \otimes I_n, \tag{45}$$

and

$$\Sigma_N = \left(\begin{array}{c|ccc} R_0 & R_1 & \cdots & R_N \\ \hline 0 & & I_{Nn} & \end{array} \right). \tag{46}$$

This grid and this type of formulation of the discretization has the property that Π_{N_1} and Σ_{N_1} are the leading submatrices of Π_{N_2} and Σ_{N_2} if $N_2 > N_1$. This structure will allow us to form a connection with Algorithm 2 in the next subsection.

With this we have shown that the eigenvalues corresponding to the spectral discretization of (41) using the grid (43) are the solutions to the generalized eigenvalue problem (44) where the matrices are given by (45) and (46).

6.2 Spectral discretization equivalence with Algorithm 2

We will now see that if we apply the standard Arnoldi method to the discretized eigenvalue problem (44) in an appropriate way, the resulting iteration is equivalent to Algorithm 2.

The first step in making a connection between Algorithm 2 applied to (42) and a spectral discretization approach, is the following result stating that a particular matrix vector product corresponding to the discretized problem, i.e., the generalized eigenvalue problem (44), can be interpreted as the action of $\mathcal{B}$ on polynomials. The equivalence holds in the sense that block elements of the vectors should be interpreted as coefficients in a Chebyshev expansion of corresponding functions.

Lemma 2 (Matrix-vector product equivalence) *Let $N > k$ and let the columns of $(x_0, \ldots, x_k)$ and $(y_0, \ldots, y_{k+1})$ 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

$$\Sigma_N^{-1} \Pi_N \mathrm{vec}(x_0, \ldots, x_k, 0, \ldots, 0) = \mathrm{vec}(y_0, \ldots, y_{k+1}, 0, \ldots, 0), \tag{47}$$

where Σ_N and Π_N are given by (45)-(46) and correspond to the discretization of (41). Then, the operator $\mathcal{B}$ corresponding to the nonlinear eigenvalue problem (42) is equivalent to $\Sigma_N^{-1}\Pi_N$ in the sense that,

$$\psi = \mathcal{B}\varphi. \tag{48}$$

Proof 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

$$\Pi_N \mathrm{vec}(x_0, \ldots, x_k, 0, \ldots, 0) = \mathrm{vec}(X_k t_k,\ X_k L_{k+1,k+1}). \tag{49}$$

and

$$\Sigma_N \mathrm{vec}(y_0, \ldots, y_{k+1}, 0, \ldots, 0) = \mathrm{vec}\left(R_0 y_0 + R_1 y_1 + \cdots + R_{k+1} y_{k+1},\ \ y_1, \ldots, y_{k+1}\right). \tag{50}$$

The equality of (49) and (50) can be interpreted as conditions on the functions φ and ψ. From the last $k+1$ block rows of (49) and (50) it follows that

$$\psi'(\theta) = \varphi(\theta) \tag{51}$$

and the first column correspondingly gives the condition that

$$\varphi(0) = (A(\frac{d}{d\theta})\psi)(0). \tag{52}$$

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 (52) for $\psi(0)$ and use (51),

$$\begin{aligned}\psi(0) = A_0^{-1}(\varphi(0) - A_1\psi'(0) - A_2\psi''(0) - \cdots) \\ = A_0^{-1}(\varphi(0) - A_1\varphi(0) - A_2\varphi'(0) - \cdots).\end{aligned} \tag{53}$$

When we insert the expansion of A into B into the definition (42) and compare with (53) we see that,

$$(B(\frac{d}{d\theta})\varphi)(0) = A(0)^{-1}(\varphi(0) - A_1\varphi(0) - A_2\varphi'(0) - \cdots) = \psi(0). \tag{54}$$

From (51) and (54) it follows that ψ is the action of $\mathcal{B}$ onto φ, i.e., (48) holds. This completes the proof.

A discretization approach to compute eigenvalues of (41) typically consists of first discretizing the functional differential equation (41), yielding (as above) a large generalized eigenvalue problem. 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 (44).

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

Theorem 5 (Equivalence with 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 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.*

Proof We will show that the standard Arnoldi algorithm applied to $\Sigma_N^{-1}\Pi_N$ is equivalent to Algorithm 1 in the same sense that Algorithm 1 and Algorithm 2 are equivalent in Theorem 4.

Note that the Chebyshev scalar product $< \cdot, \cdot >_C$, defined by (38), is equivalent to the Euclidean scalar product if the (block) vectors are interpreted as coefficients in a Chebyshev expansion. In the standard Arnoldi method, we use the Euclidean scalar product. From Lemma 2 we know that the action $\mathcal{B}$ is also equivalent to $\Sigma_N^{-1}\Pi_N$ if the vector is interpreted in the same way. The same equivalence holds for the starting function φ and $(x_0^T, 0, \ldots, 0)^T$. Hence, all operations are equivalent if (block) vectors are interpreted as coefficients in a Chebyshev basis. The result follows by induction and application of Theorem 4.

6.3 Choice of the interval for functional differential equations

In the equivalence in Section 6.2, we saw that if we discretize an FDE (41) acting on an interval $\tilde{I}$ using the grid (43), 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 Algorithm 2. We will now set the discretization interval I, equal to the interval of the FDE $\tilde{I}$, i.e.,

$$[a, b] = [\tilde{a}, \tilde{b}]. \tag{55}$$

The assumption (55) is very common in literature on discretization of FDEs similar to (41), e.g. [8] 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 (55)), all grid points (43) 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 [34, Chapter 5]. The grid points (43) are asymptotically distributed in this way.

Hence, under the condition (55), i.e., that we set the intervals equal, the spectral discretization in Section 6.1 is *good* in the sense that,

- it corresponds to approximating the correct interval; and
- the grid distribution (43) is a Chebyshev like distribution.

The above arguments lead to a natural choice (55) of the interval $I = [a, b]$. Furthermore, the fact that with the choice (55) Algorithm 2 corresponds to a good spectral discretization of the problem and that Algorithm 2 is derived from Algorithm 1 by taking the scalar product $< \cdot, \cdot >=< \cdot, \cdot >_C$, justify this choice of the scalar product. Since choosing a different interval for the Chebyshev polynomials (in Algorithm 2) would correspond to discretizing a different interval, it is advisable to use (55) for problems stemming from FDEs.

7 Examples

7.1 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 scalar product. We will in particular show that when using the Chebyshev scalar product the choice of the interval can have a dramatic impact in practice. This is consistent with the theory in Section 6.3,

Consider a nonlinear eigenvalue problem of the form,

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

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. Let

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

We first need to transform the problem to the form (2), i.e., find an expression for B. The result of the reformulation (3) is

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

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 is treated as a free parameter. From the formulas in Table 1 and the same manipulations as leading up to (36), we find that y_0 in (23) in the coefficient map (Theorem 2) can be simplified to

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

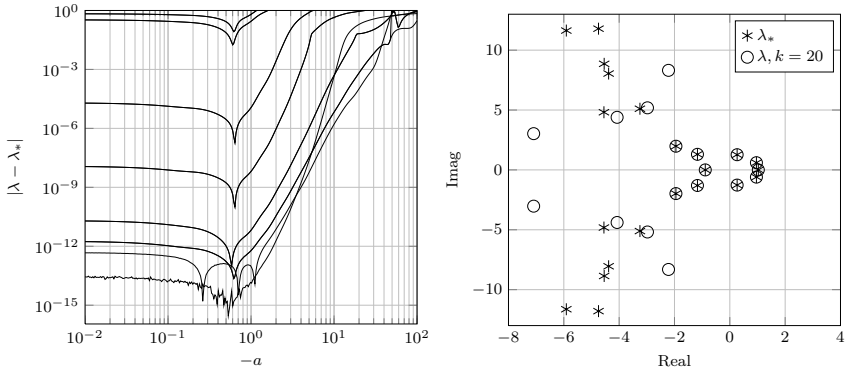
By carrying out several runs, we study the accuracy of the solution after $k = 20$ for different choices of a. This is visualized in Fig. 1a, where we see that choosing $-a = \tau = 1$ produces high accuracy for many eigenvalue approximations. From the figure it is also clear that choosing $-a$ different from the delay, can slow down convergence considerably, in particular if the interval is chosen much larger than the delay.

Similar conclusions can be made from the convergence diagram in Fig. 2. We have considerably slower convergence for $a = -5$ than for $a = -1$.

7.2 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

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



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

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

Fig. 1 Illustration of the approximations from Algorithm 2 for the problem in Section 7.1.

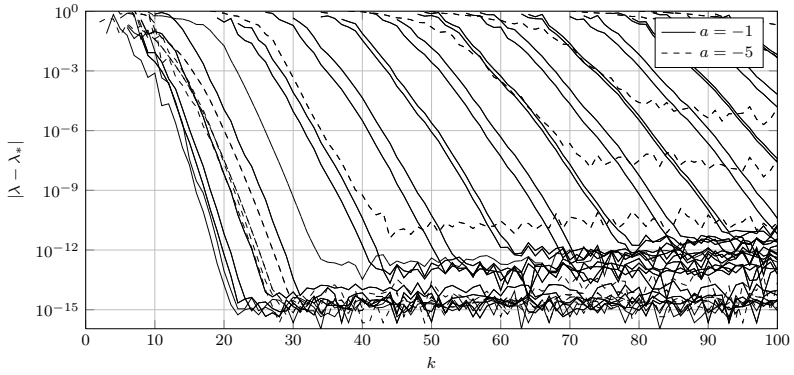


Fig. 2 Convergence history for two different choices of a for the example in Section 7.1 using Algorithm 2. The first eigenvalue reaches accuracy 10^{-10} at $k = 17$ and $k = 23$ for $a = -1$ and $a = -5$ correspondingly. After $k = 80$ iterations, the method finds 30 and 10 eigenvalues (with error less that 10^{-10}) for $a = -1$ and $a = -5$ correspondingly.

where $\sigma_2 = 108.8774$ and $n = 9956$. This problem appears in the simulation in [19] and the sparse matrices A_0, A_1, A_2 and A_3 are available in the problem collection [6].

We shift and scale the problem by $\lambda = \kappa\mu + \sigma$. 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 [19, Fig. 1] to be roughly within unit magnitude. In the standard Arnoldi method, the general rule-of-thumb is to pick the shift close to the eigenvalues of interest. Note that M is non-analytic in $\lambda = \sigma_2^2$ and $\lambda = 0$. We only have guaranteed convergence for eigenvalues within Ω which is small if σ is close to any of the non-analytic 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 non-analytic 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 $\sigma = \sigma_0 = 146.71^2$, in the first run, since one eigenvalue of interest is close to this point [6]. Inspired by the region of interest for a similar problem [19, Fig. 1] we also do simulations with the shift $\sigma = \sigma_1 = 250^2$, which corresponds to a larger guaranteed region of convergence.

The problem does not correspond to a differential equation on a finite interval and the reasoning in Section 6.3 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, described in point ii) in Remark 3.

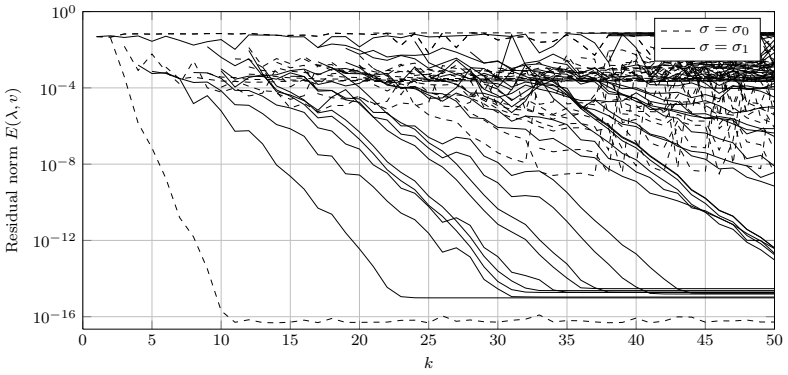


Fig. 3 The convergence of Algorithm 2 for the example in Section 7.2. The error indicator is the relative residual norm also used in [19].

The convergence of the algorithm (in the sense of the relative residual used in [19]) is visualized in Fig. 3. Note that when we select the shift $\sigma = \sigma_0$, only one eigenvalue has converged after 50 iterations and the convergence to the other eigenvalues appears stagnated. For the shift $\sigma = \sigma_1$, 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. 4. We clearly see that there is only one eigenvalue within Ω_0 the region of convergence for $\sigma = \sigma_0$, and the theory only supports the convergence to the one eigenvalue within Ω_0. With the shift $\sigma = \sigma_1$ we successfully find the eigenvalues given in [19]. 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 runs. 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 $28.0s$.

We use, as in [19], the quantity

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

to measure the convergence. The convergence of the iteration is visualized in Fig. 3.

The robustness and attractive global convergence properties of Algorithm 2 for this example can be observed in two ways. The convergence shown in Fig. 3, behaves in a very regular way. In order to find more eigenvalues, we just have to carry out more iterations. In Fig. 4 we see that we find more eigenvalues in the region of interest than the local correction schemes used in [19]. This illustrates the property that Algorithm 2 is reliable in the sense that is not likely to miss solutions. As usual, the local correction schemes, e.g., those in [19], are however likely to be faster.

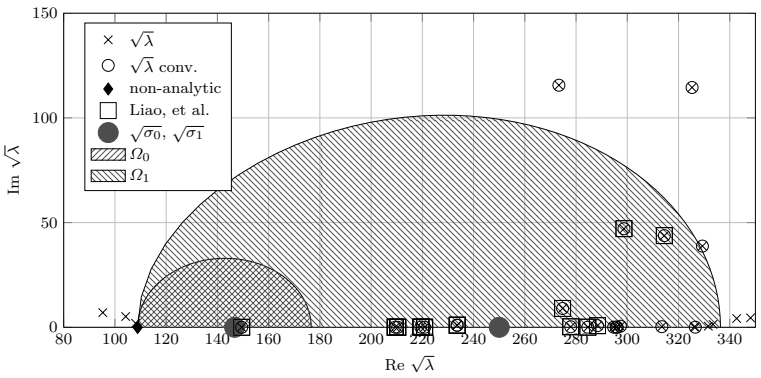


Fig. 4 The figure is a visualization of the simulations in Section 7.2 in a square-root-scale as in [19]. It shows the approximate eigenvalues, the shifts and the region of guaranteed convergence. There is apparently only one eigenvalue within the region of guaranteed convergence for $\sigma = \sigma_0 = 146.71^2$ (Ω_0). For $\sigma = \sigma_1 = 250^2$ all solutions (from [19]) within the region of guaranteed convergence Ω_1 are found.

8 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 arbitrary nonlinear eigenvalue problems. 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.

For instance, there are techniques for restarting implemented in ARPACK [18]. Implicit restarting as in [31] carries over directly with one major difference. In a function setting, the initial function φ_1, after restart would not be a constant function but a polynomial. The direct implementation of [31] hence reduces the dimension of the subspace, but there is still a growth in the size of the matrix of basis vectors. A potential solution to this growth is to include basis functions which are not polynomials, e.g., exponential functions. Note that the framework (in particular Lemma 2) allows the use of general basis functions. Some further techniques used in ARPACK [18] such as locking and purging seem to carry over but deserve further attention.

The resources required for the orthogonalization is substantial and even dominating in the example in Section 7.2. 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 is cheaper, but it is in general not a scalar product but only a semidefinite bilinear form. In related methods, e.g., [3], this type of orthogonalization is combined with the solving of a projected small nonlinear eigenvalue problem instead of computing the eigenvalues of a Hessenberg matrix.

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 [19] in Fig. 4.

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. C. T. Baker and R. Lehoucq. Preconditioning constrained eigenvalue problems. *Linear Algebra Appl.*, 431:396–408, 2009.
5. Z. Battles and L. N. Trefethen. An extension of MATLAB to continuous functions and operators. *SIAM J. Sci. Comput.*, 25(5):1743–1770, 2004.
6. 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.
7. W. J. Beyn. An integral method for solving nonlinear eigenvalue problems. Technical report, Bielefeld University, 2010.
8. 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.
9. F. Chatelin. *Spectral approximation of linear operators*. Academic Press, New York, USA, 1983.
10. T. A. Driscoll. Automatic spectral collocation for integral, integro-differential, and integrally reformulated differential equations. *J. Comput. Phys.*, 229(17):5980–5998, 2010.
11. 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.
12. I. Gohberg, P. Lancaster, and L. Rodman. *Matrix polynomials*. Academic press, 1982.
13. J. Hale and S. M. Verduyn Lunel. *Introduction to functional differential equations*. Springer-Verlag, 1993.
14. 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.
15. 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.
16. D. Kressner. A block Newton method for nonlinear eigenvalue problems. *Numer. Math.*, 114(2):355–372, 2009.
17. D. Kressner, C. Schröder, and D. S. Watkins. Implicit QR algorithms for palindromic and even eigenvalue problems. *Numer. Algorithms*, 51(2):209–238, 2009.
18. 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.
19. 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.

20. 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.
21. K. Meerbergen. The Lanczos method with semi-definite inner product. *BIT*, 41(5):1069–1078, 2001.
22. K. Meerbergen. The quadratic Arnoldi method for the solution of the quadratic eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 30(4):1463–1482, 2008.
23. V. Mehrmann and H. Voss. Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods. *GAMM Mitteilungen*, 27:121–152, 2004.
24. V. Mehrmann and D. Watkins. Structure-preserving methods for computing eigenpairs of large sparse skew-hamiltonian/hamiltonian pencils. *SIAM J. Sci. Comput.*, 22(6):1905–1925, 2001.
25. W. Michiels and S.-I. Niculescu. *Stability and Stabilization of Time-Delay Systems: An Eigenvalue-Based Approach.* Advances in Design and Control 12. SIAM Publications, Philadelphia, 2007.
26. A. Neumaier. Residual inverse iteration for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 22:914–923, 1985.
27. B. N. Parlett and H. Chen. Use of indefinite pencils for computing damped natural modes. *Linear Algebra Appl.*, 140:53–88, 1990.
28. G. Peters and J. Wilkinson. Inverse iterations, ill-conditioned equations and Newton's method. *SIAM Rev.*, 21:339–360, 1979.
29. A. Ruhe. Algorithms for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 10:674–689, 1973.
30. 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.
31. D. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.
32. G. W. Stewart. On the semidefinite B-Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 31(3):1458–1468, 2009.
33. Y. Su and Z. Bai. Solving rational eigenvalue problems via linearization. Technical report, Department of Computer Science and Mathematics, University of California, Davis, 2008.
34. L. N. Trefethen. *Spectral Methods in MATLAB.* SIAM Publications, Philadelphia, 2000.
35. 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.
36. H. Voss. An Arnoldi method for nonlinear eigenvalue problems. *BIT*, 44:387 – 401, 2004.