A KRYLOV METHOD FOR THE DELAY EIGENVALUE PROBLEM

ELIAS JARLEBRING , KARL MEERBERGEN , AND WIM MICHIELS*

Abstract. The *Arnoldi method* is currently a very popular algorithm to solve large-scale eigenvalue problems. The main goal of this paper is to generalize the Arnoldi method to the characteristic equation of a *delay-differential equation* (DDE), here called a *delay eigenvalue problem* (DEP).

The DDE can equivalently be expressed with a linear infinite dimensional operator whose eigenvalues are the solutions to the DEP. We derive a new method by applying the Arnoldi method to the generalized eigenvalue problem (GEP) associated with a spectral discretization of the operator and by exploiting the structure. The result is a scheme where we expand a subspace not only in the traditional way done in the Arnoldi method. The subspace vectors are also expanded with one block of rows in each iteration. More importantly, the structure is such that if the Arnoldi method is started in an appropriate way, it has the (somewhat remarkable) property that it is in a sense independent of the number of discretization points. It is mathematically equivalent to an Arnoldi method with an infinite matrix, corresponding to the limit where we have an infinite number of discretization points.

We also show an equivalence with the Arnoldi method in an operator setting. It turns out that with an appropriately defined operator over a space equipped with scalar product with respect to which Chebyshev polynomials are orthonormal, the vectors in the Arnoldi iteration can be interpreted as the coefficients in a Chebyshev expansion of a function. The presented method yields the same Hessenberg matrix as the Arnoldi method applied to the operator.

1. Introduction. Consider the linear time-invariant differential equation with several discrete delays

$$\dot{x}(t) = A_0 x(t) + \sum_{i=1}^{m} A_i x(t-\tau_i), \tag{1.1}$$

where $A_0, \ldots, A_m \in \mathbb{C}^{n\times n}$ and $\tau_1, \ldots, \tau_m \in \mathbb{R}$. Without loss of generality, we will order the delays and let $0 =: \tau_0 < \cdots < \tau_m$. The differential equation (1.1) is known as a delay-differential equation and is often characterized and analyzed using the solutions of the associated characteristic equation

$$\det \Delta(\lambda) = 0, \tag{1.2}$$

where

$$\Delta(\lambda) = \lambda I - A_0 - \sum_{i=1}^{m} A_i e^{-\tau_i \lambda}.$$

We will call the problem of finding $\lambda \in \mathbb{C}$ such that (1.2) holds, the *delay eigenvalue problem* and the solutions $\lambda \in \mathbb{C}$ are called the characteristic roots or the eigenvalues of the DDE (1.1). The eigenvalues of (1.1) play the same important role for DDEs as eigenvalues play for matrices and ordinary differential equations (without delay). The eigenvalues can be used in numerous ways, e.g., to study stability or design systems with properties attractive for the application. See [MN07] for recent results on eigenvalues of delay-differential equations and their applications.

In this paper, we will present a numerical method for the DEP (1.2), which in a reliable way computes the solutions close to the origin. Note that due to the fact that (1.2) can be shifted by the substitution $\lambda \leftarrow \tilde{\lambda} - s$, the problem is equivalent

*K.U. Leuven, Department of Computer Science, 3001 Heverlee, Belgium, {Elias.Jarlebring,Karl.Meerbergen,Wim.Michiels}@cs.kuleuven.be

to finding solutions close to any arbitrary given point $s \in \mathbb{C}$, known as the shift. Also note that focusing on solutions close to the origin is a natural approach to study stability of (1.1), as the rightmost root which determines the stability of (1.1), is usually among the roots of smallest magnitude [MN07]. Our method is valid for arbitrary system matrices $A_0, \ldots, A_m$. We will however pay special attention to the case where $A_0, \ldots, A_m$ are large and sparse matrices.

In our method we will also assume that $\sum_{i=0}^{m} A_i$ is non-singular, i.e., that $\lambda = 0$ is not a solution to (1.2). Such an assumption is not unusual in eigenvalue solvers based on inverse iteration and it is not restrictive in practice, since, generically a small shift $s \neq 0$ will generate a problem for where $\tilde{\lambda} = 0$ is not a solution. For a deeper study on the choice of the shift s in a slightly different setting, see [MR96].

The Arnoldi method is a very popular algorithm for large standard and generalized eigenvalue problems. An essential component in the Arnoldi method is the repeated expansion of a subspace with a new basis vector. The presented algorithm is based on the Arnoldi method for the delay eigenvalue problem where the subspace is extended not only in the traditional way done in the Arnoldi method. The basis vectors are also extended with one block row in each iteration.

The method turns out to be very efficient and robust (and it is illustrated with the examples in Section 5). The reliability of the method can be understood from several properties and close relations with the Arnoldi method. This will be used to derive and understand the method.

An important property of (1.2) is that the solutions can be equivalently written as the eigenvalues of an infinite dimensional operator, denoted $\mathcal{A}$. A common approach to compute λ is to discretize this operator $\mathcal{A}$ and compute the eigenvalues of a matrix [BMV05, BMV09a]. We will start (in Section 2) from this approach and derive a variant of the discretization approach resulting in a special structure. Based on the pencil structure of this eigenvalue problem we derive (in Section 3) a method by projection on a Krylov subspace as in the Arnoldi method.

This construction is such that performing k iterations with the algorithm can be interpreted as a standard Arnoldi method applied to the discretized problem with N grid points, where the number of grid points N is arbitrary but larger than k. This somewhat remarkable property that the process is independent of $N \geq k$ makes the process dynamic in the sense that, unlike traditional discretization approaches, the number of grid points N does not have to be fixed before the iteration starts. In fact, the whole iteration is in this sense independent of N. It is therefore not surprising that the constructed algorithm is also equivalent to the Arnoldi method in a different setting. The operator $\mathcal{A}$ is a linear map of a function segment to a function segment. With an appropriately defined inner product, it is easy to conceptually construct the Arnoldi method in an infinite dimensional setting. We prove (in Section 4) that the presented method is mathematically equivalent to the Arnoldi method applied to an infinite operator with a special inner product.

Although there are numerous methods for the DEP, the presented method has several appealing properties not present in other methods. There are several methods based on discretization such as [BM00, Bre06, BMV05, BMV09a]. The software package DDE-BIFTOOL [ELR02, ELS01] also uses a discretization approach combined with an iterative method. It is a two-step method based on

1) estimating many roots with a discretization; and then
2) correcting the computed solutions with an iterative method.

In our approach there is no explicit need for a second step as sufficient accuracy can

be achieved directly by the iteration. Moreover, the grid of the discretization has to be fixed a priori in classical discretization approaches. See [VLR08] for a heuristic choice for the discretization used in DDE-BIFTOOL. There is also a software package called TRACE-DDE [BMV09b], which is an implementation of the discretization approach in [BMV05] and related works. Note that both software packages TRACE-DDE and DDE-BIFTOOL are based on computing the eigenvalues of a matrix of dimension $nN \times nN$ using the QR-method. Unlike these approaches which are based on discretization, our method only involves linear algebra operations with matrices of dimension $n \times n$, making it suitable for problems where $A_0, \ldots, A_m$ are large and possibly sparse. Our method also has the dynamic property in the sense that the number of discretization points is not fixed beforehand.

There are other approaches for the DEP, e.g., the method QPMR [VZ09] which is based on the coefficients of the characteristic equation and hence likely not very suitable for very large and sparse system matrices $A_0, \ldots, A_m$. Large and sparse matrices form an important class of problems for the presented approach. See [Jar08, Chapter 2] for more methods.

The DEP belongs to a class of problems called nonlinear eigenvalue problems. There are several general purpose methods for nonlinear eigenvalue problems; see [Ruh73, MV04]. There are, for instance, the Newton-type methods [Sch08, Neu85] and a nonlinear version of Jacobi-Davidson [BV04]. There is also a method which is inspired by the Arnoldi method in [Vos04]. We wish to stress that despite the similar name, the algorithm presented in this paper and the method in [Vos04] (which has been applied to the delay eigenvalue problem in [Jar08, Chapter 2]) have very little in common. In comparison to these methods, the presented method is expected to be more reliable since it inherits most properties of the Arnoldi method, e.g., robustness and simultaneous convergence to several eigenvalues. The block Newton method has recently been generalized to nonlinear eigenvalue problems [Kre09] and has been applied to the delay eigenvalue problem. The differences between the block Newton method and the Arnoldi method for standard eigenvalue problems seem to hold here as well. The Arnoldi method is often more efficient for very large systems since only the right-hand side and not the matrix of the linear system changes in each iteration. A decomposition (e.g. LU-decomposition) can be computed before the iteration starts and the linear system can be solved very efficiently. Moreover, in the Arnoldi method, it is not necessary to fix the number of wanted solutions before the iteration starts.

The polynomial eigenvalue problem (PEP) is an important nonlinear eigenvalue problem. There exist recent theory and numerical algorithms for polynomial eigenvalue problems; see, e.g., [MMMM06, MV04, Lan02]. In particular, it was recently shown in [ACL09] how to linearize a PEP if it is expressed in a Chebyshev polynomial basis. We will see in Section 4 that our approach has a relation with a Chebyshev expansion. The derivation of our approach is based on a discretization of an infinite dimensional operator and there is no explicit need for these results.

Finally, we note that the Arnoldi method presented here is inspired by the Arnoldi like methods for polynomial eigenvalue problems, in particular the quadratic eigenvalue problem in [BS05, Fre05, Mee08].

2. Discretization and companion-type formulation. Some theory available for DDEs is derived in a setting where the DDE is stated as an infinite-dimensional linear system. A common infinite dimensional representation will be briefly summarized in Section 2.1. The first step in the derivation of the main numerical method of this paper (given in Section 3) is based on a discretization of this infinite dimensional

representation. The discretization is different from other traditional discretizations available in the literature because the discretized matrix can be expressed with a companion like matrix and a block triangular matrix. The discretization and the associated manipulations are given in Section 2.2 and Section 2.3.

2.1. Infinite dimensional first order form. Let $X := \mathcal{C}([-\tau_m,\ 0],\ \mathbb{C}^n)$ be the Banach space of continuous functions mapping the interval $[-\tau_m,\ 0]$ onto $\mathbb{C}^m$ and equipped with the supremum norm. Consider the linear operator $\mathcal{A} : \mathcal{D}(\mathcal{A}) \subseteq X \to X$ defined by

$$\begin{aligned} &\mathcal{D}(\mathcal{A}) := \left\{\phi \in X :\ \tfrac{d\phi}{d\theta} \in X,\ \tfrac{d\phi}{d\theta}(0) = A_0\phi(0) + \textstyle\sum_{k=1}^{m} A_k\phi(-\tau_k)\right\}, \\ &(\mathcal{A}\ \phi)(\theta) := \tfrac{d\phi}{d\theta}(\theta),\ \theta \in [-\tau_m, 0]. \end{aligned} \tag{2.1}$$

The equation (1.1) can now be reformulated as an abstract ordinary differential equation over X

$$\frac{d}{dt} z_t = \mathcal{A} z_t. \tag{2.2}$$

See [HV93] for a detailed description of $\mathcal{A}$. The corresponding solutions of (1.1) and (2.2) are related by $z_t(\theta) \equiv x(t+\theta),\ \theta \in [-\tau_m,\ 0]$.

The properties of $\sigma(\mathcal{A})$, defined as the spectrum of the operator $\mathcal{A}$, are described in detail in [MN07, Chapter 1]. The operator only features a point spectrum and its spectrum is fully determined by the eigenvalue problem

$$\mathcal{A}\ z = \lambda z, \quad z \in X,\ z \neq 0. \tag{2.3}$$

The connections with the characteristic roots are as follows. The characteristic roots are the eigenvalues of the operator $\mathcal{A}$. Moreover, if $\lambda \in \sigma(\mathcal{A})$, then the corresponding eigenfunction takes the form

$$z(\theta) = v e^{\lambda\theta},\ \theta \in [-\tau_m,\ 0], \tag{2.4}$$

where $v \in \mathbb{C}^n \setminus \{0\}$ satisfies

$$\Delta(\lambda) v = 0. \tag{2.5}$$

Conversely, if the pair (v, λ) satisfies (2.5) and $v \neq 0$, then (2.4) is an eigenfunction of $\mathcal{A}$ corresponding to the eigenvalue λ.

We have summarized the following important relation. The characteristic roots appear as solutions of both the linear infinite-dimensional eigenvalue problem (2.3) and the finite-dimensional nonlinear eigenvalue problem (2.5). This connection plays an important role in many methods for computing characteristic roots and assessing their sensitivity [MN07].

2.2. Spectral discretization. A common way to numerically analyze the spectrum of an infinite-dimensional operator is to discretize it. Here, we will take an approach based on approximating $\mathcal{A}$ by a matrix using a *spectral discretization method* (see, e.g. [Tre00, BMV05]).

Given a positive integer N, we consider a mesh Ω_N of $N+1$ distinct points in the interval $[-\tau_m,\ 0]$:

$$\Omega_N = \{\theta_{N,i},\ i = 1, \ldots, N+1\},$$

where

$$-\tau_m \leq \theta_{N,1} < \ldots < \theta_{N,N} < \theta_{N,N+1} = 0.$$

Now consider the discretized problem where we have replaced X with the space X_N of discrete functions defined over the mesh Ω_N, i.e., any function $\phi \in X$ is discretized into a block vector $x^{(N)} = (x_1^{(N)T} \cdots \ x_{N+1}^{(N)\ T})^T \in X_N$ with components

$$x_i^{(N)} = \phi(\theta_{N,i}) \in \mathbb{C}^n, \quad i = 1, \ldots, N+1.$$

Let $\mathcal{P}_N x^{(N)}$, be the unique $\mathbb{C}^n$ valued interpolating polynomial of degree smaller or equal than N, satisfying

$$\left(\mathcal{P}_N x^{(N)}\right)(\theta_{N,i}) = x_i^{(N)}, \quad i = 1, \ldots, N+1.$$

In this way we can approximate the operator $\mathcal{A}$ defined by (2.1), with the matrix $\mathcal{A}_N : \ X_N \to X_N$, defined as

$$\begin{array}{lll} \left(\mathcal{A}_N \ x^{(N)}\right)_i = & \left(\mathcal{P}_N x^{(N)}\right)'(\theta_{N,i}), & i = 1, \ldots, N, \\ \left(\mathcal{A}_N \ x^{(N)}\right)_{N+1} = & A_0 \left(\mathcal{P}_N x^{(N)}\right)(0) + \sum_{i=1}^m A_i (\mathcal{P}_N x^{(N)})(-\tau_i). & \end{array}$$

Using the Lagrange representation

$$\mathcal{P}_N x^{(N)} = \sum_{k=1}^{N+1} l_{N,k} \ x_k^{(N)},$$

where the Lagrange polynomials $l_{N,k}$ such that, $l_{N,k}(\theta_{N,i}) = 1$ if $i = k$ and $l_{N,k}(\theta_{N,i}) = 0$ if $i \neq k$, we get an explicit form for the matrix $\mathcal{A}_N$,

$$\mathcal{A}_N = \begin{pmatrix} d_{1,1} & \ldots & d_{1,N+1} \\ \vdots & & \vdots \\ d_{N,1} & \ldots & d_{N,N+1} \\ a_1 & \ldots & a_{N+1} \end{pmatrix} \in \mathbb{R}^{(N+1)n \times (N+1)n}, \tag{2.6}$$

where

$$\begin{array}{lll} d_{i,k} & = & l'_{N,k}(\theta_{N,i}) I_n, \qquad\qquad i = 1, \ldots, N, \ k = 1, \ldots, N+1, \\ a_k & = & A_0 l_{N,k}(0) + \sum_{i=1}^m A_i l_{N,k}(-\tau_i), \qquad k = 1, \ldots, N+1. \end{array}$$

The numerical methods for computing characteristic roots, described in [BMV05, BMV06, BMV09a], are similarly based on solving a discretized linear eigenproblem

$$\mathcal{A}_N \ x^{(N)} = \lambda x^{(N)}, \ \lambda \in \mathbb{C}, \ x^{(N)} \in \mathbb{C}^{(N+1)n}, \ x^{(N)} \neq 0, \tag{2.7}$$

by constructing the matrix $\mathcal{A}_N$ and computing all its eigenvalues with the QR method.

With an appropriately chosen grid Ω_N in the discretization, the convergence of the individual eigenvalues of $\mathcal{A}_N$ to corresponding eigenvalues of $\mathcal{A}$ is fast. In [BMV05] it is proved that spectral accuracy (approximation error $O(N^{-N})$) is obtained with a grid consisting of (scaled and shifted) Chebyshev extremal points.

2.3. A companion-type reformulation. The matrix $\mathcal{A}_N$ in equation (2.6) does not have a simple apparent matrix structure. We will now see that the eigenvalues of $\mathcal{A}_N$ are identical to the eigenvalues of a generalized eigenvalue problem where one of the matrices is a block triangular matrix and the other is a companion like matrix, if we choose the grid points appropriately. The matrix structure will be exploited in Section 3.

We start with the observation that the discretized eigenvalue problem (2.7) can be directly obtained by requiring that there exists a polynomial of degree N,

$$(\mathcal{P}_N x^{(N)})(t) = \sum_{k=0}^{N} l_{N,k}(t)\ x_k^{(N)},$$

satisfying the conditions

$$(\mathcal{P}_N x^{(N)})'(\theta_{N,i}) = \lambda(\mathcal{P}_N x^{(N)})(\theta_{N,i}), \quad i = 1, \ldots, N, \tag{2.8}$$

$$A_0(\mathcal{P}_N x^{(N)})(0) + \sum_{i=1}^{m} A_i(\mathcal{P}_N x^{(N)})(-\tau_i) = \lambda(\mathcal{P}_N x^{(N)})(0). \tag{2.9}$$

Hence, an eigenvalue problem equivalent to (2.7) can be obtained by expressing $\mathcal{P}_N x^{(N)}$ in another basis and imposing the same conditions.

Consider the representation of $\mathcal{P}_N x^{(N)}$ in a basis of Chebyshev polynomials

$$(\mathcal{P}_N x^{(N)})(t) = \sum_{i=0}^{N} c_i T_i\left(2\frac{t}{\tau_m} + 1\right),$$

where T_i is the Chebyshev polynomial of the first kind and order i, and $c_i \in \mathbb{C}^N$ for $i = 0, \ldots, N$. By requiring that this polynomial satisfies the conditions (2.8)-(2.9) and by taking into that $T_i'(t) = iU_{i-1}(t)$, $i \geq 1$, where U_i is the Chebyshev polynomial of the second kind and order i, we obtain the following equivalent eigenvalue problem for (2.7):

(2.10)

$$\left(\lambda\left(\begin{array}{c|c} 1\ T_1(1)\ \cdots\ T_{N-1}(1) & T_N(1) \\ \hline \Gamma_1 & \Gamma_2 \end{array}\right) \otimes I_n - \left(\begin{array}{c|c} R_0 & R_1\ \cdots\ R_N \\ \hline 0 & U \otimes I_n \end{array}\right)\right) \begin{pmatrix} c_0 \\ \vdots \\ c_N \end{pmatrix} = 0,$$

where the submatrices are defined as

$$\Gamma_1 = \begin{pmatrix} 1 & T_1(\alpha_1) & \cdots & T_{N-1}(\alpha_1) \\ \vdots & & & \vdots \\ 1 & T_1(\alpha_N) & \cdots & T_{N-1}(\alpha_N) \end{pmatrix}, \quad \Gamma_2 = \begin{pmatrix} T_N(\alpha_1) \\ \vdots \\ T_N(\alpha_N) \end{pmatrix},$$

$$U = \frac{2}{\tau_m}\begin{pmatrix} U_0(\alpha_1) & 2U_1(\alpha_1) & \cdots & NU_{N-1}(\alpha_1) \\ \vdots & & & \vdots \\ U_0(\alpha_N) & 2U_1(\alpha_N) & \cdots & NU_{N-1}(\alpha_N) \end{pmatrix}, \tag{2.11}$$

$$R_i = A_0 T_i(1) + \sum_{k=1}^{m} A_k T_i\left(-2\frac{\tau_k}{\tau_m} + 1\right),\ i = 0, \ldots, N,$$

and

$$\alpha_i = 2\frac{\theta_i}{\tau_m} + 1, \quad i = 1, \ldots, N+1,$$

are the normalized grid-points, that is, scaled and shifted to the interval $[-1,\ 1]$.

In order to simplify (2.10), we use the property

$$T_1(t) = \frac{1}{2}U_1(t), \quad T_i(t) = \frac{1}{2}U_i(t) - \frac{1}{2}U_{i-2}(t),\ i \geq 2. \tag{2.12}$$

This implies that

$$\Gamma_1 = UL_N, \tag{2.13}$$

where L_N is a band matrix, defined as

$$L_N = \frac{\tau_m}{4}\begin{pmatrix} 2 & 0 & -1 & & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & & \\ & & \frac{1}{3} & 0 & \ddots & \\ & & & \frac{1}{4} & \ddots & -\frac{1}{N-2} \\ & & & & \ddots & 0 \\ & & & & & \frac{1}{N} \end{pmatrix}. \tag{2.14}$$

If the grid points are such that U is nonsingular, we can use the relation (2.13) to transform (2.10) to the eigenvalue problem

(2.15)

$$\left(\lambda\left(\begin{array}{c|c} 1\ T_1(1)\ \cdots\ T_{N-1}(1) & T_N(1) \\ \hline L_N & U^{-1}\Gamma_2 \end{array}\right) \otimes I_n - \left(\begin{array}{c|c} R_0 & R_1\ \cdots\ R_N \\ \hline 0 & I_{Nn} \end{array}\right)\right)\begin{pmatrix} c_0 \\ \vdots \\ c_N \end{pmatrix} = 0.$$

An important property of the eigenvalue problem (2.15) is that all information about the grid Ω_N is *concentrated* in the column $U^{-1}\Gamma_2$. We now show that with an appropriately chosen Chebyshev type grid, the structure of L_N is continued in this column. For this, choose the nonzero grid points as scaled and shifted zeros of U_N, the latter given by

$$\alpha_i = -\cos\frac{\pi i}{N+1},\ i = 1, \ldots, N. \tag{2.16}$$

First, this choice implies that the matrix (2.11) is invertible. Second, from (2.12) it can be seen that the numbers (2.16) satisfy

$$T_N(\alpha_i) = -\frac{1}{2}U_{N-2}(\alpha_i),\ i = 1, \ldots, N, \quad N \geq 2,$$

which implies on its turn that

$$U^{-1}\Gamma_2 = \begin{pmatrix} 0 & \cdots & 0 & -\frac{\tau_m}{4(N-1)} & 0 \end{pmatrix}^T.$$

By combining the results above, we arrive at the following theorem.

THEOREM 2.1. *If the grid points in the spectral dicretization of the operator* (2.1) *are chosen as*

$$\theta_i = \frac{\tau_m}{2}(\alpha_i - 1), \quad \alpha_i = -\cos\frac{\pi i}{N+1}, \ i = 1, \ldots, N+1, \tag{2.17}$$

then the discretized eigenvalue problem (2.7) *is equivalent to*

$$(\lambda \Pi_N - \Sigma_N)\, c = 0, \ \lambda \in \mathbb{C}, \ c \in \mathbb{C}^{(N+1)n}, \ c \neq 0, \tag{2.18}$$

where

$$\Pi_N = \frac{\tau_m}{4} \begin{pmatrix} \frac{4}{\tau_m} & \frac{4}{\tau_m} & \frac{4}{\tau_m} & \cdots & & \cdots & \frac{4}{\tau_m} \\ 2 & 0 & -1 & & & & \\ & \frac{1}{2} & 0 & -\frac{1}{2} & & & \\ & & \frac{1}{3} & 0 & \ddots & & \\ & & & \frac{1}{4} & \ddots & -\frac{1}{N-2} & \\ & & & & \ddots & 0 & -\frac{1}{N-1} \\ & & & & & \frac{1}{N} & 0 \end{pmatrix} \otimes I_n \tag{2.19}$$

and

$$\Sigma_N = \begin{pmatrix} R_0 & R_1 & \cdots & R_N \\ & I_n & & \\ & & \ddots & \\ & & & I_n \end{pmatrix}, \tag{2.20}$$

with

$$R_i = A_0 + \sum_{k=1}^{m} A_k T_i \left(-2\frac{\tau_k}{\tau_m} + 1 \right), \ i = 0, \ldots, N.$$

REMARK 2.2 (The choice of discretization points). *A grid consisting of the points* α_i *as in* (2.17) *is very similar to a grid consisting of Chebyshev extremal points as in [BMV05], where the latter is defined as*

$$-\cos\frac{\pi(i-1)}{N}, \ i = 1, \ldots, N+1. \tag{2.21}$$

Note that the convergence theory of spectral discretizations is typically shown using reasoning with a potential function defined from a limit of the grid distribution. See, e.g., [Tre00, Theorem 5]. Since the discretization here (2.17) *and the grid in [BMV05], i.e.,* (2.21) *have the same asymptotic distribution, we expect the convergence properties to be the same. For instance, the eigenvalues of* $\mathcal{A}_N$ *exhibit spectral convergence to eigenvalues of* $\mathcal{A}$*. Moreover, the part of the spectrum of* $\mathcal{A}_N$ *which has not yet converged to corresponding eigenvalues of* $\mathcal{A}$*, is typically located to the* left *of the converged eigenvalues. This is an important property when assessing stability.*

We have chosen a new grid as this grid allows us to construct matrices with a particularly useful structure. The structure which will be used in the next section is that the matrices (2.19) *and* (2.20) *are such that* Π_{N_1} *and* Σ_{N_1} *are* submatrices *of* Π_{N_2} *and* Σ_{N_2} *whenever* $N_2 \geq N_1$*.*

3. A Krylov method for the DEP. The discretization in the previous section can be directly used to find approximations of the eigenvalues of (1.1) by computing the eigenvalues of the GEP (2.18), $(\lambda\Pi_N - \Sigma_N)x = 0$, with a general purpose eigenvalue solver. The Arnoldi method (first introduced in [Arn51]) is one popular general purpose eigenvalue solver. In a traditional approach for the DEP, it is common to fix N and apply the Arnoldi method to an eigenvalue problem, here the GEP (2.18). This has the drawback that the matrices Σ_N and Π_N are large if N is large. Another drawback is that N has to be fixed before the iteration starts. The choice of N is a trade-off between computation time and accuracy, as the error decreases with growing N and the computation time grows with increasing N. Unless we wish to solve several eigenvalue problems, N has chosen entirely based on the information about the problem available before the iteration is carried out.

We will adapt a version of the Arnoldi method to the GEP $(\lambda\Pi_N - \Sigma_N)x = 0$ and exploit the structure in such a way that the method has the (somewhat remarkable) property that it is, in a sense, independent of N. The constructed method is in this way a solution to the mentioned drawbacks and trade-offs of a traditional approach. It turns out that if we start the Arnoldi method corresponding to $\Pi_N^{-1}\Sigma_N$ in an appropriate way, the approximations after k iterations are identical to the approximations computed by k iterations of the Arnoldi method applied to the eigenvalue problem corresponding to *any* $N > k$. That is, it can be seen as carrying out an Arnoldi process associated with the limit $N \to \infty$. In the light of this we will show in Section 4 that the method is also equivalent to the Arnoldi method applied to the infinite-dimensional operator $\mathcal{A}^{-1}$.

We will use the natural limit interpretation of Σ_N and Π_N. Let $\mathrm{vec}(\mathbb{C}^{n\times\infty})$ denote the set of all ordered infinite sequences of vectors of length n exponentially convergent to zero. The natural interpretation of the limits of the operators Σ_∞ and Π_∞ is with this notation $\Sigma_\infty : \mathrm{vec}(\mathbb{C}^{n\times\infty}) \to \mathrm{vec}(\mathbb{C}^{n\times\infty})$ and $\Pi_\infty : \mathrm{vec}(\mathbb{C}^{n\times\infty}) \to \mathrm{vec}(\mathbb{C}^{n\times\infty})$. We will call an element of $\mathrm{vec}(\mathbb{C}^{n\times\infty})$ an infinite vector and Σ_∞ and Π_∞ infinite matrices. In many results and applications of the Arnoldi method, the method is implicitly equipped with the Euclidean scalar product. We will use natural extension of the Euclidean scalar product to infinite vectors, $x^*y := \sum_{i=0}^{\infty} x_i^* y_i$, where x_i, y_i are the elements of x, y respectively.

The Arnoldi method is a construction of an orthogonal basis of the set of linear combinations of a power sequence associated with matrix $A \in \mathbb{R}^{n\times n}$ and vector $b \in \mathbb{R}^n$,

$$\mathcal{K}_k(A, b) := \mathrm{span}\{b, Ab, \ldots, A^{k-1}b\}.$$

This subspace is called a *Krylov subspace*. The Arnoldi method approximates eigenvalues of A by the eigenvalues of $H_k = V_k^* A V_k$ (which are called the Ritz values) where the columns of $V_k \in \mathbb{C}^{n\times k}$ form an orthonormal basis of $\mathcal{K}_k(A, b)$. The eigenvalues of H_k converge to the extreme well-separated eigenvalues first [Saa92, Chapter VI]. In this paper we are interested in eigenvalues close to the origin. In many applications those are not very well-separated, so convergence is expected to be slow. However, convergence can be drastically improved by applying the Arnoldi method to A^{-1}, as the eigenvalues of A near zero become typically well-separated extreme eigenvalues of A^{-1}, and so, fast convergence is expected. For this reason we will apply the Arnoldi method to form $\mathcal{K}_k(A^{-1}, b)$ and call the inverse of the Ritz values the reciprocal Ritz values.

As in the definition of the Krylov subspace, the matrix vector product is an important component in the Arnoldi method. The underlying property which we will

use next is the matrix vector product associated with the infinite matrix $\Sigma_\infty^{-1}\Pi_\infty$. It turns out to be structured in such a way that it has a closed form for a special type of vector.

THEOREM 3.1 (Matrix vector product). *Suppose* $\sum_{i=0}^m A_i$ *is non-singular. Let* $\Sigma_\infty, \Pi_\infty$ *be as in Theorem 2.1 and* $Y \in \mathbb{C}^{n\times k}$. *Then*

$$\Sigma_\infty^{-1}\Pi_\infty \mathrm{vec}(Y, 0, \ldots) = \mathrm{vec}(\hat{x}, Z, 0, \ldots)$$

and

$$Z = YL_k^T, \tag{3.1}$$

where $L_k \in \mathbb{R}^{k\times k}$ *is given by* (2.14) *and*

$$\hat{x} = \left(\sum_{j=0}^m A_j\right)^{-1}\left(\sum_{i=0}^{k-1} y_i - A_0\sum_{i=0}^{k-1} z_i - \sum_{j=1}^m A_j\left(\sum_{i=0}^{k-1} T_{i+1}(1-2\frac{\tau_j}{\tau_m})z_i\right)\right). \tag{3.2}$$

Proof. The proof is based on forming the limits for $\Sigma_\infty \mathrm{vec}(\hat{x}, Z, 0, \ldots)$ and $\Pi_\infty \mathrm{vec}(Y, 0, \ldots)$ and noting that the result yields (3.1) and (3.2). Note that from (2.20) we have that

$$\Sigma_\infty \mathrm{vec}(\hat{x}, Z, 0, \ldots) \;=\; \begin{pmatrix} R_0 & R_1 & \cdots \\ & I_n & \\ & & \ddots \end{pmatrix} \mathrm{vec}(\hat{x}, Z, 0, \ldots) \;=\; \mathrm{vec}(y, Z, 0, \ldots),$$

where $y = R_0\hat{x} + R_1 z_0 + \cdots + R_k z_{k-1}$. Let $v_k^T = (1, \ldots, 1)^T \in \mathbb{R}^k$. We now have from (2.19) and rules of vectorization and Kronecker products that

$$\Pi_\infty \mathrm{vec}(Y, 0, \ldots) = \left(\begin{pmatrix} v_\infty^T \\ L_\infty \end{pmatrix} \otimes I_n\right) \mathrm{vec}(Y, 0, \ldots) = \\ \mathrm{vec}\left((Y, 0, \ldots)v_\infty, (Y, 0, \ldots)L_\infty^T,\right) = \mathrm{vec}\left(Yv_k, YL_k^T, 0, \ldots\right).$$

The proof is completed by solving $y = Yv_k$. ∎

3.1. Algorithm. Now consider the power sequence for the operator $\Sigma_\infty^{-1}\Pi_\infty$ started with $\mathrm{vec}(w, 0, \cdots)$, $w \in \mathbb{R}^n$. From Theorem 3.1 we see that the non-zero part of the infinite vector grows by one vector (of length n) in each iteration such that at the jth step, the resulting infinite vector is $\mathrm{vec}(Y, 0, \ldots)$ where $Y \in \mathbb{R}^{n\times(j+1)}$.

The Arnoldi method builds the Krylov sequence vector by vector, where in addition, the vectors are orthogonalized. In step k, the orthogonalization is a linear combination of the $k+1$st vector and the previously computed k vectors. Hence, the orthogonalization at the kth iteration does not change the general structure of the $k+1$st vector.

This allows us to construct a scheme similar to the Arnoldi method where we dynamically increase the size of the basis vectors. Let V_k be the matrix consisting of the basis vectors and $v_{ij} \in \mathbb{C}^n$ the vector corresponding to block element i, j. The dependency tree of the basis vectors is given in Figure 3.1, where the gray arrows represent the computation of the first component $\hat{x}$.

The algorithm given in Algorithm 1 is (from the reasoning above) mathematically equivalent to the Arnoldi method applied to $\Sigma_\infty^{-1}\Pi_\infty$, as well as the Arnoldi method

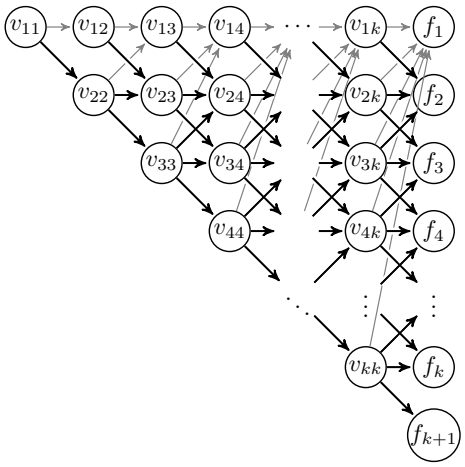


FIGURE 3.1. *Dependency tree of the basis matrix V_k. The column added in Step 11 in Algorithm 1 has been denoted $f := (f_1^T, \ldots, f_{k+1}^T)^T := (v_{k+1,1}^T, \ldots, v_{k+1,k+1}^T)^T$.*

applied to the matrix $\Sigma_N^{-1}\Pi_N$, where N is larger than the total number of iteration steps taken. We use notation common for Arnoldi iterations; we let $\underline{H}_k \in \mathbb{C}^{(k+1)\times k}$ denote the dynamically constructed rectangular Hessenberg matrix and $H_k \in \mathbb{C}^{k\times k}$ the corresponding $k \times k$ upper part.

Algorithm 1 A Krylov method for the DEP

Require: $x_0 \in \mathbb{C}^n$ and time-delay system (1.1)

1: Let $v_1 = x_0/\|x_0\|_2$, $V_1 = v_1$, $k = 1$, $\underline{H}_0$ =empty matrix,
2: Factorize $\sum_{i=0}^m A_i$
3: **for** $k = 1, 2, \ldots$ until converged **do**
4: Let $\mathrm{vec}(Y) = v_k$
5: Compute Z according to (3.1) with sparse L_k
6: Compute $\hat{x}$ according to (3.2) using the factorization computed in Step 2
7: Expand V_k with one block row (zeros)
8: Let $w_k := \mathrm{vec}(\hat{x}, Z)$, compute $h_k = V_k^* w_k$ and then $\hat{w}_k = w_k - V_k h_k$
9: Compute $\beta_k = \|\hat{w}_k\|_2$ and let $v_{k+1} = \hat{w}_k/\beta_k$
10: Let $\underline{H}_k = \begin{pmatrix} \underline{H}_{k-1} & h_k \\ 0 & \beta_k \end{pmatrix} \in \mathbb{C}^{(k+1)\times k}$
11: Expand V_k into $V_{k+1} = [V_k, v_{k+1}]$
12: **end for**
13: Compute the eigenvalues μ from the Hessenberg matrix H_k
14: Return approximations $1/\mu$

3.2. Implementation details. The efficiency and robustness of Algorithm 1 can only be guaranteed if the important implementational issues are addressed. We will apply some techniques used in standard implementations of the Arnoldi method and some techniques which are adapted for this problem.

As is usually done in eigenvalue computations using the Arnoldi method, $\sum_{k=0}^m A_k$ is factorized by a sparse direct solver and then each Arnoldi step requires a backward solve with the factors for computing $(\sum_{k=0}^m A_k)^{-1} y$. Examples of such direct solvers

are [ADLK01, SG04, DEG+99, Dav04]. In Step 8, the vector w_k should be orthogonalized against V. We use iterative reorthogonalization as in ARPACK [LSY98].

The derivation of the method is based on the assumption that $A_0 + \cdots + A_m$ is nonsingular, i.e., that $\lambda = 0$ is not a solution to the characteristic equation (1.2). This can be easily verified in practice since for most types of factorizations, e.g., the LU-factorization, it can be directly established if the matrix is singular. If we establish (in Step 2) that the $A_0 + \cdots + A_m$ is singular then the problem can be shifted (with a small shift) such that the shifted problem is nonsingular.

The Ritz vectors associated with Step 13 in Algorithm 1 are $u = V_k z$ where z is an eigenvector of the $k \times k$ Hessenberg matrix H_k. Note that the eigenvalues μ of H_k are approximations to eigenvalues of $\Sigma_N^{-1}\Pi_N$, so $\lambda = \mu^{-1}$ are the corresponding approximate eigenvalues of (1.1). Note that $u \in \mathbb{C}^{nk}$ and that an eigenpair of a time-delay system can be represented by the eigenvalue $\lambda \in \mathbb{C}$ and a (short) eigenvector $v \in \mathbb{C}^n$. A user is typically only interested in approximations of v and not approximations of u. For this reason we will now discuss adequate ways to extract $v \in \mathbb{C}^n$ from $u \in \mathbb{C}^{nk}$. This will also be used to derive stopping criteria.

Note that the eigenvectors of $\Sigma_N^{-1}\Pi_N$ approach in the limit the structure $w = c \otimes v$. Given a Ritz vector $u \in \mathbb{C}^{nk}$ we will construct the vector $v \in \mathbb{C}^n$ from the first n components of u. This can be motivated by the following observation in Figure 3.1. Note that the vector $v_{k-p,k}$ does not depend on the nodes in the right upper triangle with sides of length $p-1$ in the graph. In fact, $v_{k-p,k}$ is a linear combination of $v_{1,i}$, $i = 1, \ldots, p+1$. Hence, the quality of the vector $v_{k-p,k}$ cannot be expected to be much better than the first p iterations. With inductive reasoning, we conclude that the first block of n rows of V contains the information with the highest quality, in the sense that all other vectors are linear combinations of previously computed information. This is similar to the reasoning for quadratic eigenvalue problem in [Mee08, Section 4.2.1].

Another natural way to extract an approximation of v is by using the singular vector associated with the dominant singular value of the matrix $U \in \mathbb{C}^{n \times k}$, where $\mathrm{vec}(U) = u$, since the dominant singular value corresponds to a best rank-one approximation of U. In general, our numerical experiments are not conclusive and indicate only a very minor difference between the two approaches. We propose to use the former approach, since it is cheaper.

REMARK 3.2 (Residuals). *The termination criterion in the standard Arnoldi method is typically an expression involving the residual. In the setting of Algorithm 1 there are two natural ways to define residuals. There is the residual*

$$r := \Sigma_N^{-1}\Pi_N u - \lambda^{-1} u \in \mathbb{C}^{nN}$$

and the (short) residual $\hat{r} := \Delta(\lambda)v \in \mathbb{C}^n$. *The norm of the residual* r *is cheaply available as a by-product of the Arnoldi iteration as for the standard Arnoldi method: let* $H_k z = \lambda^{-1} z$ *with* $\|z\|_2 = 1$, *then* $\|r\|_2 = h_{k+1,k}|e_k^T z|$. *It is however more natural to have termination criteria involving* $\|\hat{r}\|$ *since from the residual norm it is easy to derive a backward error. Unfortunately, even though* v *can easily be extracted from* u *(as is mentioned above) the computation* $\Delta(\lambda)v$ *is too expensive to evaluate in each iteration for each eigenvector candidate. In the examples section we will (for illustrative purposes) use a fixed number of iterations, but in a general purpose implementation we propose to use a heuristic combination, where the cheap residual norm* $\|r\|$ *is used until it is sufficiently small and in a post-processing step, the residual norms* $\|\hat{r}\|$ *can be used to check the result. The residual* $\|r\|_2$ *will be further interpreted in Remark 4.6.*

REMARK 3.3 (Reducing memory requirements). *The storage of the Arnoldi vectors, i.e., V_k, is of order $O(k^2 n)$, and may become prohibitive in some cases. As for the polynomial eigenvalue problem, it is possible to exploit the special structure illustrated in Figure 3.1 to reduce the cost to $O(kn)$. This is the same complexity as for standard Arnoldi. See the similar approach for polynomial eigenvalue problems [BS05], [Fre05] and [Mee08]. Note that attention should be payed to issues of numerical stability [Mee08].*

4. Equivalence with an infinite dimensional operator setting. The original problem to find λ is already a standard eigenvalue problem in the sense that λ is an eigenvalue of the infinite dimensional operator $\mathcal{A}$. Since $\mathcal{A}$ is a linear operator, one can consider the Arnoldi method applied to $\mathcal{A}^{-1}$ in an abstract setting, such that the Arnoldi method constructs a Krylov subspace of functions, i.e.,

$$\mathcal{K}_k(\mathcal{A}^{-1}, \varphi) := \text{span}\{\varphi, \mathcal{A}^{-1}\varphi, \ldots, \mathcal{A}^{-(k-1)}\varphi\}, \tag{4.1}$$

and projects on it. In this section we will see that Algorithm 1 has a complete interpretation in this setting if a scalar product is appropriately defined. The vector v_k in Algorithm 1 turns out to play the same role as the coefficients in the Chebyshev expansion. The Krylov subspace (4.1) is constructed for the inverse of $\mathcal{A}$. The inverse is explicitly given as follows.

PROPOSITION 4.1 (The inverse of $\mathcal{A}$). *The inverse of $\mathcal{A} : X \to X$ exists iff $A_0 + \sum_{i=1}^m A_i$ is nonsingular. Moreover, it is explicitly given as*

$$\begin{array}{rcl} \mathcal{D}(\mathcal{A}^{-1}) & = & X \\ \left(\mathcal{A}^{-1}\ \phi\right)(\theta) & = & \int_0^\theta \phi(s)ds + C(\phi),\ \theta \in [-\tau_m,\ 0], \qquad \phi \in \mathcal{D}(\mathcal{A}^{-1}), \end{array} \tag{4.2}$$

where the constant $C(\phi)$ satisfies

$$C(\phi) = \left(A_0 + \sum_{i=1}^m A_i\right)^{-1} \left[\phi(0) - \sum_{i=1}^m A_i \int_0^{-\tau_i} \phi(s)ds\right]. \tag{4.3}$$

Proof. First assume $A_0 + \sum_{i=1}^m A_i$ is nonsingular and note that if $\phi \in X$ then ϕ is continuous and bounded on the closed interval $[-\tau_m, 0]$. Hence, the integrals in (4.2) and (4.3) exist and (4.2) defines an operator, which we first denote by $\mathcal{T}$. It can be easily verified that $\mathcal{T}\mathcal{A}\phi = \phi$ when $\phi \in \mathcal{D}(\mathcal{A})$ and $\mathcal{A}\mathcal{T}\phi = \phi$ when $\phi \in \mathcal{D}(\mathcal{T})$. Hence, $\mathcal{T} = \mathcal{A}^{-1}$. It remains to show that the inverse is not uniquely defined if $A_0 + \sum_{i=1}^m A_i$ is singular. Let $v \in \mathbb{C}^n \backslash \{0\}$ be a null vector of $A_0 + \sum_{i=1}^m A_i$ and consider a constant function $\varphi(t) := v$. From the definition (2.1) we have that $\mathcal{A}\varphi = 0$ and the inverse is not uniquely defined. ∎

4.1. Action and Krylov subspace equivalence. The key to the functional setting duality of this section is that we consider a scaled and shifted Chebyshev expansion of entire functions. Consider the expansion of two entire functions ψ and ϕ in series of scaled Chebyshev polynomials,

$$\begin{array}{rcl} \phi(t) & = & \sum_{i=0}^\infty c_i T_i\left(2\frac{t}{\tau_m} + 1\right) \\ \psi(t) & = & \sum_{i=0}^\infty d_i T_i\left(2\frac{t}{\tau_m} + 1\right), \quad t \in [-\tau_m,\ 0]. \end{array} \tag{4.4}$$

We will now see that the operation $\psi = \mathcal{A}^{-1}\phi$ can be expressed as a mapping of the coefficients, $c_0, c_1, \ldots$ and $d_0, d_1, \ldots$. This mapping turns out to reduce to the matrix

vector product in Theorem 3.1. Suppose $\psi = \mathcal{A}^{-1}\phi$, then

$$\psi \in \mathcal{D}(\mathcal{A}), \tag{4.5}$$

$$\mathcal{A}\psi = \psi' = \phi. \tag{4.6}$$

From the fact that the derivative of a Chebyshev polynomial of the first kind can be expressed as a Chebyshev polynomial of the second kind, we note that

$$\psi'(t) \quad = \quad \sum_{i=1}^{\infty} \frac{2d_i i}{\tau_m} U_{i-1}\left(2\frac{t}{\tau_m}+1\right).$$

Moreover, the relation between Chebyshev polynomials of the first kind and Chebyshev polynomials of the second kind, i.e., property (2.12), yields

$$\begin{aligned}\phi(t) &= c_0 U_0\left(2\tfrac{t}{\tau_m}+1\right) + \tfrac{1}{2}c_1 U_1\left(2\tfrac{t}{\tau_m}+1\right) + \sum_{i=2}^{\infty}\tfrac{c_i}{2}\left(U_i\left(2\tfrac{t}{\tau_m}+1\right) - U_{i-2}\left(2\tfrac{t}{\tau_m}+1\right)\right)\\ &= c_0 U_0\left(2\tfrac{t}{\tau_m}+1\right) + \tfrac{1}{2}c_1 U_1\left(2\tfrac{t}{\tau_m}+1\right)\\ &\quad + \sum_{i=3}^{\infty}\tfrac{c_{i-1}}{2}U_{i-1}\left(2\tfrac{t}{\tau_m}+1\right) - \sum_{i=1}^{\infty}\tfrac{c_{i+1}}{2}U_{i-1}\left(2\tfrac{t}{\tau_m}+1\right).\end{aligned}$$

By matching coefficients in (4.6) we obtain the following recurrence relation for the coefficients,

$$d_i = \begin{cases} \frac{\tau_m}{4}(2c_0 - c_2) & i = 1,\\ \frac{\tau_m}{4}\frac{c_{i-1}-c_{i+1}}{i} & i \geq 2.\end{cases} \tag{4.7}$$

From (4.5) and (4.6) we get

$$\phi(0) = A_0\psi(0) + \sum_{k=1}^{m} A_k\psi(-\tau_k).$$

Hence,

$$\sum_{i=0}^{\infty} c_i T_i(1) = \sum_{i=0}^{\infty}\sum_{k=0}^{m} A_k T_i\left(-2\frac{\tau_k}{\tau_m}+1\right) d_i = \sum_{i=0}^{\infty} R_i d_i. \tag{4.8}$$

By combining the results above and the fact that the Chebyshev coefficients of entire functions decay exponentially [Tre00, Theorem 1] (as they are the Fourier coefficients of an entire function) we have proved the following relation.

THEOREM 4.2 (Action equivalence). *Consider two entire functions ϕ and ψ and the associated Chebyshev expansion* (4.4). *Denote $c = (c_0^T, c_1^T, \ldots)^T, d = (d_0^T, d_1^T, \ldots)^T \in \mathrm{vec}(\mathbb{C}^{n\times\infty})$. Suppose $\sum_{i=0}^{m} A_i$ is non-singular. Then the following two statements are equivalent*

i) $\psi = \mathcal{A}^{-1}\phi$

ii) $d = \Sigma_\infty^{-1}\Pi_\infty c$ *where c, d fulfill* (4.7)-(4.8).

Moreover, if $c = (c_0^T, \ldots, c_{k-1}^T, 0, \ldots, 0)^T = \mathrm{vec}(Y, 0, \ldots, 0)$, $Y \in \mathbb{R}^{n\times k}$ *then* $d = \mathrm{vec}(\hat{x}, Z, 0, \ldots, 0)$ *where $\hat{x}$ and Z are the formulas in Theorem 3.1.*

REMARK 4.3 (Krylov subspace equivalence). *The equivalence between $\mathcal{A}^{-1}$ and $\Sigma_\infty^{-1}\Pi_\infty$ in Theorem 4.2 propagates to an equivalence between the corresponding Krylov subspaces. Let $\phi_0(t) = x_0$ be a constant function. It now follows from the fact that Theorem 4.2 implies*

$$\phi \in \mathcal{K}_k(\mathcal{A}^{-1}, \phi_0),$$

if and only if

$$\mathrm{vec}(c_0, c_1, \ldots, c_{k-1}) \in \mathcal{K}_k(\Sigma_k^{-1}\Pi_k, \mathrm{vec}(x_0, 0, \ldots, 0)),$$

where $c_0, \ldots, c_{k-1}$ are the Chebyshev coefficients of ϕ, i.e., (4.4).

4.2. Orthogonalization equivalence. We saw in Section 4.1 that the matrix vector operation associated with $\Sigma_\infty^{-1}\Pi_\infty$ is equivalent to the operation $\mathcal{A}^{-1}$, in the sense that $\mathcal{A}$ applied to a function corresponds to a map between Chebyshev coefficients of two functions. The associated Krylov subspaces are also equivalent.

The Arnoldi method is a way to project on a Krylov subspace. In order to define the projection and compute the elements of H_k, we need to define a scalar product. In Algorithm 1 we use the natural way to define a scalar product, the Euclidean scalar product on the Chebyshev coefficients. In order to define a projection equivalent to Algorithm 1 in a consistent way, we define a scalar product in the function setting as

$$\langle \phi, \psi \rangle := c^* d = \sum_{i=0}^{\infty} c_i^* d_i, \tag{4.9}$$

where ϕ, ψ, c and d are as in Theorem 4.2. We combine this with Theorem 4.2 to conclude that the Hessenberg matrix generated in Algorithm 1 and the Hessenberg matrix generated by the standard Arnoldi method applied to $\mathcal{A}^{-1}$ with the scalar product (4.9) are equal.

THEOREM 4.4 (Hessenberg equivalence). *Let ϕ, ψ, c and d be as in Theorem 4.2. The Hessenberg matrix computed (with exact arithmetic) in Algorithm 1 is identical to the Hessenberg matrix of the Arnoldi method applied to $\mathcal{A}^{-1}$ with the scalar product* (4.9) *and the constant starting vector $\varphi(t) = x_0$.*

The definition (4.9) involves the coefficients of a Chebyshev expansion. We will now see that this definition can be reformulated to an explicit expression with weighted integrals involving the functions ϕ and ψ. First note that Chebyshev polynomials are orthogonal (but not orthonormal) in the following sense,

$$\frac{2}{\tau_m}\int_{-\tau_m}^{0} \frac{T_i(\frac{2}{\tau_m}t+1)T_j(\frac{2}{\tau_m}t+1)}{\sqrt{1-(\frac{2}{\tau_m}t+1)^2}}\,dt = \int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}}\,dx = \begin{cases} 0, & i \neq j, \\ \pi, & i = j = 0, \\ \pi/2 & i = j \neq 0. \end{cases}$$

In order to simplify the notation, we introduce the functional

$$I(f) := \frac{2}{\tau_m}\int_{-\tau_m}^{0} \frac{f(t)}{\sqrt{1-(\frac{2}{\tau_m}t+1)^2}}\,dt.$$

We will now show that

$$\langle \phi, \psi \rangle = \frac{2}{\pi} I(\phi^*\psi) - \frac{1}{\pi^2} I(\phi^*) I(\psi), \tag{4.10}$$

where $(\phi^*\psi)(t) = \phi(t)^*\psi(t)$, by inserting the expansion of ϕ and ψ, i.e., (4.4), into (4.10). Note that from the orthogonality of Chebyshev polynomials we have that

$$I(\phi^*\psi) = \sum_{i,j=0}^{\infty} c_i^* d_j \frac{2}{\tau_m}\int_{-\tau_m}^{0} \frac{T_i(\frac{2}{\tau_m}t+1)T_j(\frac{2}{\tau_m}t+1)}{\sqrt{1-(\frac{2}{\tau_m}t+1)^2}}\,dt = \frac{\pi}{2}\Big(\sum_{i=0}^{\infty} c_i^* d_i + c_0^* d_0\Big),$$

and from the fact that $T_0(x) = 1$,

$$I(\phi^*) = \sum_{i=0}^{\infty} c_i^* \frac{2}{\tau_m}\int_{-\tau_m}^{0} \frac{T_i(\frac{2}{\tau_m}t+1)T_0(\frac{2}{\tau_m}t+1)}{\sqrt{1-(\frac{2}{\tau_m}t+1)^2}}\,dt = \pi c_0^*.$$

Analogously $I(\psi) = \pi d_0$. We have shown that $\langle \phi, \psi \rangle = \sum_{i=0}^{\infty} c_i^* d_i$.

REMARK 4.5 (Computation with functions). *From the reasoning above we see that Algorithm 1 can be interpreted as the Arnoldi method applied to $\mathcal{A}^{-1}$ with the scalar product* (4.10) *for functions on $\mathcal{C}([-\tau_m,\ 0],\ \mathbb{C}^n)$ with a constant starting function, where the computation is carried out by mapping Chebyshev coefficients. We note that the representation of functions with Chebyshev coefficients and associated manipulations are also done in the software package* `chebfun` *[BT04].*

REMARK 4.6 (Residual equivalence). *Note that there is a complete duality between $\Sigma_\infty^{-1}\Pi_\infty$ and $\mathcal{A}^{-1}$. A direct consequence is that the residual norm, which can be used as a stopping criterion as described in Remark 3.2, has an interpretation as the norm of function residual with respect to the norm induced by the scalar product* (4.9). *That is, $\|(\Sigma_\infty^{-1}\Pi_\infty - \mu)\tilde{u}\|_2 = \|(\Sigma_N^{-1}\Pi_N - \mu)u\|_2 = h_{k+1,k}|e_k^T z| = \|\mathcal{A}^{-1}\phi - \mu\phi\|_c := \sqrt{\langle \mathcal{A}^{-1}\phi - \mu\phi, \mathcal{A}^{-1}\phi - \mu\phi \rangle}$.*

4.3. The block Arnoldi method and full spectral discretization. In the setting of functions, the definition of the scalar product (4.9) and the corresponding function representation (4.10) seem artificial in the sense that one cannot easily identify a property why this definition is better than any other definition of a scalar product.

In fact, in earlier works [JMM10] we derived a scheme similar to Algorithm 1 by using a Taylor expansion instead of a Chebyshev discretization. It is not difficult to show that the Taylor approach also can be interpreted in a function setting, where the scalar product is defined such that the monomials are orthogonal, i.e., $\langle \phi, \psi \rangle_T := (1/2\pi) \int_0^{2\pi} \phi(e^{i\theta})^* \psi(e^{i\theta})\, d\theta$.

In this paper, the attractive convergence properties of Algorithm 1 are motivated by the connection with the discretization scheme. Discretization schemes similar to what we present in Section 2 have been used in the literature and are known to be efficient for the delay eigenvalue problem.

We will now see that Algorithm 1 is not only the Arnoldi method applied to the discretized problem. The block version of Algorithm 1 produces the same approximation as the full discretization approach, i.e., computing the eigenvalues of the generalized eigenvalue problem associated with Π_N, Σ_N.

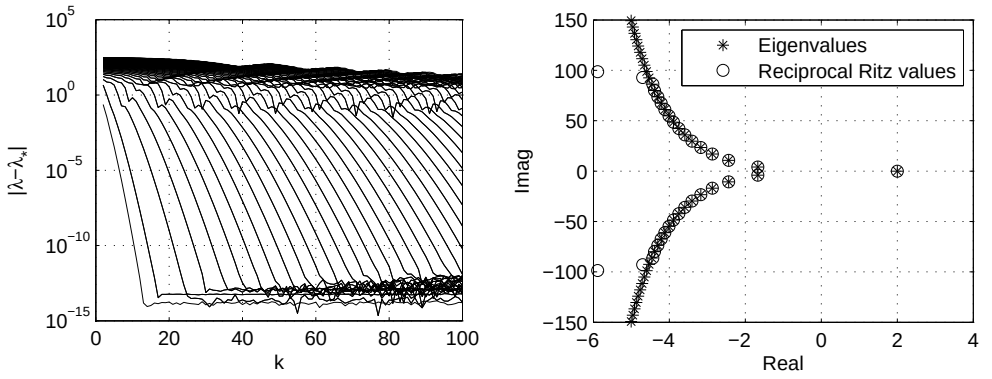
The block Arnoldi method is a variant of the Arnoldi method, where each vector is replaced by a number of vectors, which are kept orthogonal in a block sense. The block Arnoldi method is described in, e.g., [BDD+00]. It is straightforward to construct a block version of Algorithm 1. In the following result we see that this construction is equivalent to the full spectral discretization approach, if we choose the block size n, i.e., equal to the dimension of the system.

THEOREM 4.7. *Let $V_{[k]} = (V_1, \ldots, V_k)$, $V_i^T = (W_{i,1}^T, \ldots, W_{i,i}^T, 0, \ldots)$ where $W_{i,j} \in \mathbb{R}^{n\times n}$. Suppose $V_{[k]}$ is orthogonal, i.e., $V_{[k]}^* V_{[k]} = I \in \mathbb{R}^{nk\times nk}$, then*

$$H = V_{[k]}^* \Sigma_\infty^{-1} \Pi_\infty V_{[k]} \sim \Sigma_{k-1}^{-1} \Pi_{k-1} \sim \mathcal{A}_{k-1}^{-1}.$$

In words, when performing k steps of the block Arnoldi method, the resulting reciprocal Ritz values are the same approximations as those from [BMV05] (but with the grid points (2.16)) where N discretization points are used.

5. Numerical Examples.



(a) Absolute error

(b) Eigenvalues and reciprocal Ritz values ($k = 50$)

FIGURE 5.1. *Convergence and eigenvalue approximations for the example in Section 5.1 using Algorithm 1.*

5.1. A scalar example. We first illustrate several properties of the method by means of an academic example. Consider the scalar DDE (also studied in [Bre06])

$$\dot{x}(t) = (2 - e^{-2})x(t) + x(t-1).$$

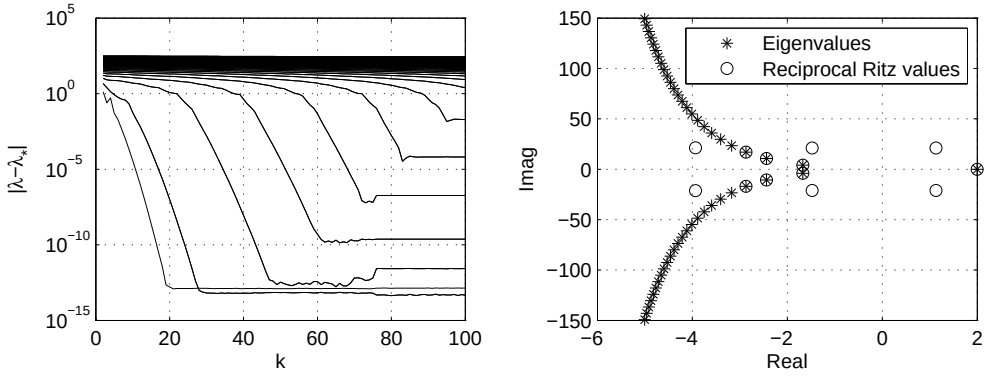
The convergence of the reciprocal Ritz values is shown in Figure 5.1. There are two different ways to interpret the error between the reciprocal Ritz values and the characteristic roots of the time-delay system.

1. In Section 4, and in particular in Section 4.1 and Section 4.2, we have seen that Algorithm 1 is equivalent to a standard Arnoldi method, applied to the infinite-dimensional operator $\mathcal{A}^{-1}$. In this interpretation the observed error is due to the Arnoldi process.
2. Since the example is scalar, the Arnoldi method and the block Arnoldi method with blocks of width n are equivalent. Hence, one can alternatively interpret the results in the light of Theorem 4.7 in Section 4.3: the computed Ritz values for a given value of k correspond to the eigenvalues of $\mathcal{A}_{k-1}^{-1}$. Accordingly, the error between the reciprocal of the Ritz values and the characteristic roots can be interpreted as the error due to an approximation of $\mathcal{A}$ by $\mathcal{A}_{k-1}$.

In Figure 5.1 we observe geometric convergence. This observation is consistent with the two interpretations above. More precisely, it is consistent with the convergence theory for the standard Arnoldi method as well as the convergence theory for the spectral discretization method. The spectral method is expected to converge exponentially [Tre00]. The generic situation for the Arnoldi method is that the angle between the Krylov subspace and the associated eigenvector converges geometrically [Saa92, Section VI.7].

With Figure 5.2 we wish to illustrate the impact and importance of the underlying approximation type. The plot shows the convergence for the method based on a Taylor approximation in [JMM10], which also has an interpretation as the Arnoldi method on a function; see Section 4.3. In comparison to Algorithm 1 very few eigenvalues are captured with [JMM10]. This is true despite the fact that the convergence to

each individual eigenvalue is exponential. Moreover, we observe that high accuracy is not achieved for eigenvalues of larger modulus. Further analysis shows that the reciprocal Ritz values have a large condition number, such that high accuracy can not be expected.



(a) Absolute error

(b) Eigenvalues and reciprocal Ritz values ($k = 50$)

FIGURE 5.2. *Convergence and eigenvalue approximations for the example in Section 5.1 using the Taylor approach in [JMM10]. The remaining (not-shown) reciprocal Ritz values are distributed circular fashion outside the range of the plot.*

5.2. A large-scale problem. The numerical methods for sparse standard eigenvalue problems have turned out to be very useful because many applications are discretizations of partial differential equations (PDEs) which are (by construction) typically large and sparse. In order to illustrate that the presented method is particularly well suited for such problems, we consider a PDE with a delayed term. See [Wu96] for many phenomena modeled as PDEs with delay and see [BMV09a] for possibilities to combine a spatial discretization of the PDE with a discretization of the operator. Consider

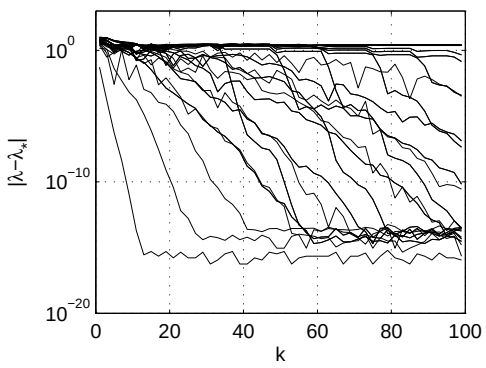
$$\frac{\partial v(x,t)}{\partial t} = \frac{\partial^2 v(x,t)}{\partial x^2} + a_0(x)v(x,t) + a_1(x)v(\pi - x, t-1).$$

with $a_0(x) = -2\sin(x)$, $a_1(x) = 2\sin(x)$ and $v_x(0,t) = v_x(\pi,t) = 0$.

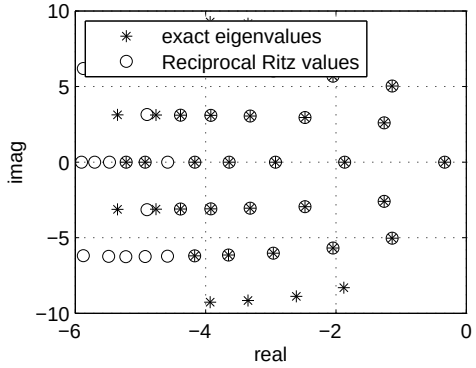
We discretize the PDE and construct a DDE by approximating the second derivative in space with central difference. This is a variant of a PDE considered in [BMV09a], where we have modified the delayed term. In the original formulation, the matrices are tridiagonal, which is not representative for our purposes. The convergence is visualized in Figure 5.3. For a system of size $n = 5000$, we carry out 100 iterations of Algorithm 1, in a total CPU time $16.2s$.

We see in Figure 5.3c that the iteration converges first to the extreme eigenvalues of the inverted spectrum (which are well isolated). This is the behavior we would expect with the standard Arnoldi method and confirms the equivalence between Algorithm 1 with an infinite dimensional Arnoldi method shown in Section 4.

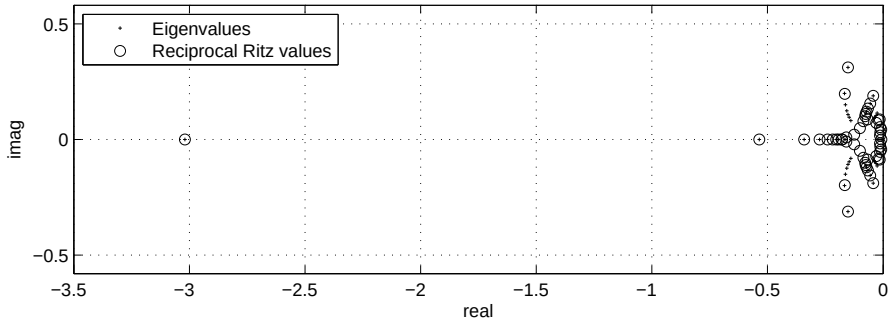
The general purpose software package DDE-BIFTOOL [ELR02, ELS01] as well as the more recent software package TRACE-DDE [BMV09b] are based on constructing



(a) Absolute error



(b) Characteristic roots and reciprocal of Ritz values ($k = 50$)



(c) The inverted spectrum

FIGURE 5.3. *Convergence and eigenvalue approximations for the example in Section 5.2.*

a large full matrix and computing its' eigenvalues using the QR-method. In our case, $n = 5000$, and constructing a larger full matrix with such an approach is not computationally feasable. We will here use an adaption of the software package [BMV09b] where we construct large *sparse* matrices and compute some eigenvalues using the Matlab command `eigs`. This approach, which associated matrix is in fact the spectral discretization discussed in Section 2.2 (with the grid (2.21)), will for brevity here be referred as *spectral + eigs*.

We report results of numerical simluations for spectral+eigs and Algorithm 1 in Table 5.1. The first column shows the number of eigenvalues which have an absolute error less than 10^{-6} and the second column the total CPU time for the simulation. The table can be interpreted as follows. If we are interested in only 10 eigenvalues (to an accuracy of 10^{-6}), Algorithm 1 is slightly faster whereas if we are interested in 20 eigenvalues spectral+eigs is slightly faster. Hence, for this particular example, finding 10 or 20 eigenvalues, the two approaches have roughly the same CPU time, under the condition that optimal values of N and k are known.

	Spectral+eigs								Algorithm 1						
N	10	11	12	13	14	15	16	k	40	50	70	75	80	100	
$\#\lambda$	5	8	11	20	29	47	75		8	11	17	20	22	27	
CPU	1.3	2.2	3.8	6.9	12.2	25.9	68.4		1.9	3.1	6.8	8.0	9.4	16.2	

TABLE 5.1

The number of accurate eigenvalues and CPU time for a direct spectral discretization approach and Algorithm 1. The CPU time is given in seconds and $\#\lambda$ denotes the number of eigenvalues which are more accurate than 10^{-6}.

Note also that for this particular example, $(A_0 + A_1)^{-1}x$ can be computed very efficiently by solving a linear system. In our implementation, we use the factorization implemented in the Matlab function `[L,U,P,Q]=lu(A0+A1)`. In fact, the CPU time consumption in the orthogonalization part was completely dominating (99% of the total computation time), since the automatic reordering implemented in the LU-factorization can exploit the structure of the matrices. Similar properties hold for the memory requirements.

An important property of Algorithm 1 is the dynamic iterative feature. It is easier to find a good value for the number of iterations k in Algorithm 1, than it is to find a good number of discretization points N in spectral+eigs. This is due to the fact that the iteration can be inspected and continued if deemed insufficient. The corresponding situation for spectral+eigs requires recomputation, since if we for some value N do not find a sufficient number of eigenvalues, the whole computation has to be restarted with a larger value of N.

We would additionally like to stress that the computational comparison is in a sense somewhat unfair to the disadvantage of Algorithm 1. The function `eigs` is based on the software package ARPACK [LSY98] which is an efficient compiled code. This issue is not present in the example that follows. Moreover, due to the advanced reordering schemes in the software for the LU-factorization, the very structured $nN \times nN$ matrix in spectral+eigs can be computed almost as efficiently as the LU-factorization of $A_0 + A_1 \in \mathbb{R}^{n\times n}$ used in Algorithm 1.

5.3. A DDE with random matrices. In the previous example we saw that the factorizations and matrix vector products could, for that example, be carried out very efficiently, both for Algorithm 1 and for spectral+eigs. The structured matrices A_0 and A_1 (and the discretized matrix) were such that a very efficient factorization could be computed. We will now, in order to illustrate a case where such an exploitation is not possible, consider a random sparse DDE with a single delay where both matrices are generated with the command `sprandn(n,n,0.005)` and $n = 4000$. The factorization of random matrices generated in this way is very computationally demanding.

We also mentioned in the previous section that a comparison involving the command `eigs` is not entirely fair (to the disadvantage of Algorithm 1) since `eigs` is based on compiled and optimized code. In this example we wish to compare numerical methods in such a way that the implementation aspects of the code play a minor role. To this end we carry out Algorithm 1 and a direct spectral discretization approach (as in the previous example) combined with the standard Arnoldi method with 100 iterations. Unlike the previous example, we combine the direct spectral approach with our own implementation of the (standard) Arnoldi method in Matlab, such that orthogonalization and other implementation aspects can be done very similar to the way done in Algorithm 1. We here call this construction *spectral+Arnoldi*.

		CPU time			
	$\#\lambda : \lvert\lambda - \lambda_*\rvert \leq 10^{-6}$	LU	Mat.vec.	Orth.	Total
Spectral + Arnoldi:					
$N = 4$	27	16.4s	7.5s	0.5s	25s
$N = 5$	52	16.2s	7.3s	0.7s	25s
$N = 7$	52	16.7s	7.3s	1.0s	26s
$N = 10$	52	17.7s	6.0s	1.6s	27s
$N = 12$	52	17.4s	6.1s	1.9s	28s
$N = 15$	52	15.5s	6.3s	2.3s	27s
$N = 20$	52	14.8s	7.6s	3.4s	30s
Algorithm 1:	52	8.7s	4.9s	2.8s	16.9s

TABLE 5.2

Computation time and number of accurate eigenvalues for several runs of the direct spectral approach and Algorithm 1 applied to the example with random matrices in Section 5.3. The number of Arnoldi iterations is fixed to 100.

A comparison of spectral+Arnoldi and Algorithm 1 is given in Table 5.2. We see that 100 iterations of Algorithm 1 yields better results or is more efficient than 100 iterations of spectral+Arnoldi, since it can be carried out in $16.9s$ and the approach spectral+Arnoldi is either slower or does not find the same number of eigenvalues.

We wish to point out some additional remarkable properties in Table 5.2. The CPU time for spectral+Arnoldi grows very slowly (and not monotonically) in N. This is due to the fact that the structure can be automatically exploited in the factorization. In fact, the number of nonzero elements of the LU-decomposition also grows very slowly and irregularly with N. It is also not even monotone.

Moreover, for spectral+Arnoldi, increasing N does eventually not yield more eigenvalues. In order to find more eigenvalues with spectral+Arnoldi one has to increase the number of Arnoldi iterations. Determining whether it is necessary to increase the number of Arnoldi iterations or the number of discretization points N can be a difficult problem. This problem is not present in Algorithm 1 as there is only (iteration) parameter k, and iterating further yields more eigenvalues. For instance, with 110 iterations we find 58 eigenvalues with a total CPU time of $18s$, i.e., six additional eigenvalues by only an additional computation cost of less than two seconds.

6. Conclusions and outlook. The approach of this paper is in a sense very natural. It is known from the literature that spectral discretization methods tend to be efficient for the DEP. The main computational part of a discretization approach is to solve a large eigenvalue problem. The Arnoldi method is typically very efficient for large standard and generalized eigenvalue problems. Our construction is natural in the sense that we combine an efficient discretization method (a spectral discretization) with an efficient eigenvalue solver (the Arnoldi method) and exploit the non-zero pattern in the iteration vectors and the connection with an infinite dimensional operator.

Although the approach is very natural, several issues related to the Arnoldi method appear difficult to extend in a natural way. We will now list some techniques and theory for the standard Arnoldi method which appear to extend easily and some which appear to be more involved.

Algorithm 1 can conceptually be fitted with explicit or implicit restarting (as in e.g. [Sor92, LS96]) after k iterations by restarting the iteration with a vector of length kn. However, the reduction of processing time and memory would not be as dramatic as the standard case since the starting vector would be of length kn. There

are different approaches to convergence theory of the Arnoldi method. Some of the convergence theory in [Saa92] is expressed in terms of angles between subspaces. The scalar product in Section 4 induces an angle definition, and it is to expect that at least some theory in [Saa92] is applicable with the appropriate angle definition. There is also theory based on potential theory [Kui06].

In this paper we assumed we are looking for eigenvalues close to the origin. Note that this assumption is not a restriction since the matrices $A_0, \ldots, A_m$ can be shifted and scaled such that an arbitrary point is shifted to the origin. Changing the shift throughout the iteration in the sense of rational Krylov [Ruh98] seems somewhat involved.

Acknowledgment. This article present results of the Belgian Programme on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture, the Optimization in Engineering Centre OPTEC of the K.U.Leuven, and the project STRT1-09/33 of the K.U.Leuven Research Foundation.

REFERENCES

[ACL09] A. Amiraslani, R. Corless, and P. Lancaster. Linearization of matrix polynomials expressed in polynomial bases. *IMA J. Numer. Anal.*, 29(1):141–157, 2009.

[ADLK01] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41, 2001. `http://graal.ens-lyon.fr/MUMPS/`.

[Arn51] W. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Q. appl. Math.*, 9:17–29, 1951.

[BDD+00] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. A. van der Vorst, editors. *Templates for the solution of algebraic eigenvalue problems. A practical guide*. SIAM, Society for Industrial and Applied Mathematics, 2000.

[BM00] A. Bellen and S. Maset. Numerical solution of constant coefficient linear delay differential equations as abstract Cauchy problems. *Numer. Math.*, 84(3):351–374, 2000.

[BMV05] D. Breda, S. Maset, and R. Vermiglio. Pseudospectral differencing methods for characteristic roots of delay differential equations. *SIAM J. Sci. Comput.*, 27(2):482–495, 2005.

[BMV06] 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.

[BMV09a] D. Breda, S. Maset, and R. Vermiglio. Numerical approximation of characteristic values of partial retarded functional differential equations. *Numer. Math.*, 113(2):181–242, 2009.

[BMV09b] D. Breda, S. Maset, and R. Vermiglio. TRACE-DDE: a tool for robust analysis and characteristic equations for delay differential equations. In *Topics in time-delay systems*, volume 388 of *Lecture Notes in Control and Information Sciences*, pages 145–155. Springer, 2009.

[Bre06] D. Breda. Solution operator approximations for characteristic roots of delay differential equations. *Appl. Numer. Math.*, 56:305–317, 2006.

[BS05] 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.

[BT04] Z. Battles and L. N. Trefethen. An extension of MATLAB to continuous functions and operators. *SIAM J. Sci. Comput.*, 25(5):1743–1770, 2004.

[BV04] T. Betcke and H. Voss. A Jacobi-Davidson type projection method for nonlinear eigenvalue problems. *Future Generation Computer Systems*, 20(3):363–372, 2004.

[Dav04] T. A. Davis. Algorithm 832: UMFPACK V4.3 – an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, 2004.

[DEG+99] J. Demmel, S. Eisenstat, J. R. Gilbert, X.-G. Li, and J. W. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Anal. Appl.*, 20(3):720–755, 1999.

[ELR02] K. Engelborghs, T. Luzyanina, and D. Roose. Numerical bifurcation analysis of

delay differential equations using DDE-BIFTOOL. *ACM Trans. Math. Softw.*, 28(1):1–24, 2002.

[ELS01] K. Engelborghs, T. Luzyanina, and G. Samaey. DDE-BIFTOOL v. 2.00: a Matlab package for bifurcation analysis of delay differential equations. Technical report, K.U.Leuven, Leuven, Belgium, 2001.

[Fre05] R. W. Freund. Subspaces associated with higher-order linear dynamical systems. *BIT*, 45:495–516, 2005.

[HV93] J. Hale and S. M. Verduyn Lunel. *Introduction to functional differential equations.* Springer-Verlag, 1993.

[Jar08] E. Jarlebring. *The spectrum of delay-differential equations: numerical methods, stability and perturbation.* PhD thesis, TU Braunschweig, 2008.

[JMM10] E. Jarlebring, K. Meerbergen, and W. Michiels. An Arnoldi method with structured starting vectors for the delay eigenvalue problem. In *Proceedings of the 9th IFAC workshop on time-delay systems, Prague*, 2010. accepted.

[Kre09] D. Kressner. A block Newton method for nonlinear eigenvalue problems. *Numer. Math.*, 114(2):355–372, 2009.

[Kui06] A. B. Kuijlaars. Convergence analysis of Krylov subspace iterations with methods from potential theory. *SIAM Rev.*, 48(1):3–40, 2006.

[Lan02] P. Lancaster. *Lambda-matrices and vibrating systems.* Mineola, NY: Dover Publications, 2002.

[LS96] R. Lehoucq and D. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17(4):789–821, 1996.

[LSY98] 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.

[Mee08] K. Meerbergen. The quadratic Arnoldi method for the solution of the quadratic eigenvalue problem. *SIAM J. Matrix Anal. Appl.*, 30(4):1463–1482, 2008.

[MMMM06] S. Mackey, N. Mackey, C. Mehl, and V. Mehrmann. Vector spaces of linearizations for matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 28:971–1004, 2006.

[MN07] 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.

[MR96] K. Meerbergen and D. Roose. Matrix transformations for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems. *IMA Journal on Numerical Analysis*, 16:297–346, 1996.

[MV04] V. Mehrmann and H. Voss. Nonlinear eigenvalue problems: A challenge for modern eigenvalue methods. *GAMM Mitteilungen*, 27:121–152, 2004.

[Neu85] A. Neumaier. Residual inverse iteration for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 22:914–923, 1985.

[Ruh73] A. Ruhe. Algorithms for the nonlinear eigenvalue problem. *SIAM J. Numer. Anal.*, 10:674–689, 1973.

[Ruh98] A. Ruhe. Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils. *SIAM J. Sci. Comput.*, 19(5):1535–1551, 1998.

[Saa92] Y. Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, 1992.

[Sch08] K. Schreiber. *Nonlinear Eigenvalue Problems: Newton-type Methods and Nonlinear Rayleigh Functionals.* PhD thesis, TU Berlin, 2008.

[SG04] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[Sor92] D. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[Tre00] L. N. Trefethen. *Spectral Methods in MATLAB.* SIAM Publications, Philadelphia, 2000.

[VLR08] K. Verheyden, T. Luzyanina, and D. Roose. Efficient computation of characteristic roots of delay differential equations using LMS methods. *J. Comput. Appl. Math.*, 214(1):209–226, 2008. doi: 10.1016/j.cam.2007.02.02.

[Vos04] H. Voss. An Arnoldi method for nonlinear eigenvalue problems. *BIT*, 44:387 – 401, 2004.

[VZ09] T. Vyhlídal and P. Zítek. Mapping based algorithm for large-scale computation of quasi-polynomial zeros. *IEEE Trans. Autom. Control*, 54(1):171–177, 2009.

[Wu96] J. Wu. *Theory and applications of partial functional differential equations.* Applied Mathematical Sciences. 119. New York, NY: Springer., 1996.