Tensor-Krylov method for computing eigenvalues of parameter-dependent matrices

Koen Ruymbeek[a,*], Karl Meerbergen[a], Wim Michiels[a]

[a] *Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium*

Abstract

In this paper we extend the Residual Arnoldi method for calculating an extreme eigenvalue (e.g. largest real part, dominant,...) to the case where the matrices depend on parameters. The difference between this Arnoldi method and the classical Arnoldi algorithm is that in the former the residual is added to the subspace. We develop a Tensor-Krylov method that can be interpreted as an application of the Residual Arnoldi algorithm to a set of matrices obtained by evaluating a parameter-dependent matrix in parameter values on a grid. The subspace contains an approximate Krylov space for all these points. Instead of adding the residuals for all parameter values to the subspace we create a low-rank approximation of the matrix consisting of these residuals and add only the column space to the subspace. In order to keep the computations efficient, it is needed to limit the dimension of the subspace and to restart once the subspace has reached the prescribed maximal dimension. The novelty of this approach is twofold. Firstly, we observed that a large error in the low-rank approximations is allowed without slowing down the convergence, which implies that we can do more iterations before restarting. Secondly, we pay particular attention to the way the subspace is restarted, since classical restarting techniques give a too large subspace in our case. We motivate why it is good enough to just keep the approximation of the searched eigenvector. At the end of the paper we extend this algorithm to shift-and-invert Residual Arnoldi method to calculate the eigenvalue close to a shift σ for a specific parameter dependency. We provide theoretical results and report numerical experiments. The Matlab code is publicly available.

Keywords: Eigenvalues, Tensor-Krylov, Parameter-dependent matrices

2010 MSC: 65F15

1. Introduction

In this paper, we consider parameter-dependent eigenvalue problems

$$\mathbf{A}(\omega)\mathbf{x}(\omega) = \lambda(\omega)\mathbf{x}(\omega), \quad \omega \in \Omega, \quad \mathbf{A} : \Omega \to \mathbb{R}^{n\times n} : \omega \mapsto \sum_{i=1}^{m} f_i(\omega)\mathbf{A}_i \tag{1}$$

on a compact parameter set $\Omega \subset \mathbb{R}^d$ and functions $f_i : \Omega \to \mathbb{R}, i = 1, 2, \ldots, m$ where m is small relative to n. We look for an eigenpair $(\lambda, \mathbf{x})$ over Ω for which λ is extreme following some ordering, e.g. we sort by modulus if λ is the dominant eigenvalue, or by real part if λ is the rightmost eigenvalue in case of minimizing the *spectral abscissa*. We assume that the wanted eigenvalue converges for any ω in a few iterations of the Arnoldi algorithm. In practice, this means that λ should be a well separated extreme eigenvalue of $\mathbf{A}(\omega)$. Here we want to develop a numerical method to efficiently calculate λ for a large set of points in Ω as calculating this eigenvalue for each point individually using standard eigenvalue solvers becomes soon infeasible. In the proposed approach, we want to create a subspace that spans the eigenvectors associated to the wanted eigenvalue over the parameters. For this, we assume that the subspace spanned by all eigenvectors of the searched

*Corresponding author
Email address: koen.ruymbeek@kuleuven.be (Koen Ruymbeek)

eigenvalues is small, although the approximated eigenvalue can be non-smooth due to eigenvalue overtaking each other or due to interacting eigenvalues that induce a higher multiplicity at some parameter values. The eigenspaces corresponding to eigenvalues that overtake or interact with each other, are often much smoother and can be represented by low rank subspaces, this idea is used as motivation for block Newton methods [12]. Of course, if the eigenvector changes significantly with the parameters, the rank will increase. The fact whether low rank approximations are useful is problem dependent, but it is not necessarily related to possible kinks in the eigenvalue function over the parameter. The second assumption that we make, is that the set of sample points come from a tensor-product grid. At the end of the paper, we extend our method to a case for which the eigenvalue closest to a shift σ is wanted.

Applications can be found in the context of control of dynamical systems, see also Example 4 and 5. Here, an important first step consists of stabilizing the system. For this task, recently methods grounded in eigenvalue optimization have been proposed, which rely on minimizing directly the spectral abscissa (or spectral radius in the discrete-time setting) as a function of the controller parameters. Such an approach has for instance been adopted in the package HIFOO [9] for designing fixed-order controllers for linear time-invariant system and in the software TDS-STABIL [16] for linear time-delay system, which have been applied to several problem from engineering [21, 18]. Here, it should be noted that time-delay models give rise to nonlinear eigenvalue problems, however many methods for solving the delay eigenvalue problem are based on an approximation by a linear eigenvalue problem of larger dimension (obtained e.g. by a spectral discretization of the solution operator [16] or a rational approximation of the characteristic matrix [??]). Inherently to the adopted stabilization approach, the objective function is typically non-smooth in the local optima, and often even not-Lipschitz continuous, due to the interaction of several eigenvalue functions, and the optima are often characterized by multiple defective eigenvalues. If in a subsequent robustness analysis of the controlled system (e.g. to assess the effects of modeling errors), the spectral abscissa is examined as a function of uncertain parameters in the system model, efficient algorithms are needed that can handle the non-smooth nature of the spectral abscissa function, in particular around the nominal parameter values as parameter changes may cause the multiple eigenvalues to bifurcate into several branches. For this task, the adopted approach is particularly amendable since the approximations are inferred from (projected) eigenvalue problems that are able to accurate replicate the non-smooth behavior, unlike, e.g., methods based on a global approximation by a multi-variate polynomial.

If $\mathbf{A}(\omega)$ is symmetric for all $\omega \in \Omega$, a sampling-based subspace method [22] and the Successive Constraint method (SCM) [25] were recently proposed. In the latter, the desired eigenvalue is approximated by iteratively computing lower-and upper bounds for each parameter value. The extension to the non-symmetric case is not possible as the method relies on properties that solely holds for the symmetric case (e.g approximations via projection on subspaces are not bounded by the eigenvalues of the large eigenvalue problem and the wanted eigenvalue over the parameters may fail to be locally Lipschitz continuous). Another class of solvers are based on polynomial approximation, see for example [1]. This technique is also used for the related problem of calculating statistical moments if the parameter is stochastic, see [3]. The disadvantage of using polynomials is that they suffer from possible lack of smoothness of the desired eigenvalue function [8]. As the adopted approximation is still inferred from the solutions of an eigenvalue problem (obtained via the projection on a subspace), it does not suffer from possible local non-smoothness of the searched eigenvalue over the parameter.

The method that we propose is an example of a Tensor-Krylov method. In these methods, the iteration vectors in the Krylov method are represented by a low rank tensor. Tensor-Krylov techniques are already discussed for various problems, for example to solve a large system of non-linear equations [4, 5] and parametrized linear systems [13].

There are two important aspects in our method. First the iteration vector is represented by a tensor which is truncated to a low-rank representation, for reasons of storage and computational efficiency. Second, the Ritz pairs are obtained from

projection of $\mathbf{A}(\omega)$ on the sum of the Krylov spaces for all sample points. A practical issue is the growth of the dimension of this subspace as we add multiple vectors in each iteration. Therefore we often need to restart to keep the algorithm feasible. We gain in efficiency if we need to add less vectors to the subspace (and so introducing a larger error) and have a good way to restart the subspace. We demonstrate that adding fewer vectors in a careful way does not lead to slower convergence and we give an explanation. We give practical bounds for the introduced errors.

The plan of the paper is as follows. In Section 2 we discuss the Arnoldi method and analyse how errors are propagated in the algorithm and give convergence results. We explain the proposed algorithm in the one parameter case and discuss why it is an extension of the Residual Arnoldi algorithm in Section 3. Subsequently, we illustrate the convergence and error propagation with some numerical experiments. In Section 5, we revisit the method for multiple parameters and report a numerical experiment. We extend our method to shift-and-invert Residual Arnoldi when all matrices except one is of rank one and give some numerical examples. We finish this paper with some concluding remarks. The Matlab code can be downloaded following this link.

Throughout the paper, we denote vectors by small bold letters and matrices by capital bold letters. The subspace of $\mathbb{R}^n$ spanned by the columns of an orthonormal matrix $\mathbf{V}$ is denoted by the calligraphic letter $\mathcal{V}$. An index k may be added to refer to the iteration where this subspace is made. With calligraphic letters other than $\mathcal{V}$, we denote tensors of order strictly larger than two. Further, we denote by $\mathbf{I}$ the identity matrix of appropriate dimensions and by the function $\text{diag}(\mathbf{v})$ a diagonal matrix with the elements of the vector $\mathbf{v}$ on its diagonal. With $\|.\|$ we denote the Euclidean norm and we use $\|.\|_F$ to refer to the Frobenius norm.

We end this section by introducing the concept of reduced eigenvalue problem. Let $\mathcal{V} \subset \mathbb{R}^n$ be a subspace and $\mathbf{V}$ an orthonormal matrix where the columns form a basis for $\mathcal{V}$, then we call $\mathbf{V}^T\mathbf{A}\mathbf{V}$ the *reduced eigenvalue problem*. We denote by $\left(\lambda^{\mathcal{V}}, \mathbf{x}^{\mathcal{V}}\right)$ an eigenpair of this reduced eigenvalue problem and the corresponding Ritz vector by $\mathbf{y} = \mathbf{V}\mathbf{x}^{\mathcal{V}}$. A subindex may be used to indicate the iteration where the subspace is made. We define the corresponding residual as $\mathbf{r} = \mathbf{A}\mathbf{y} - \lambda^{\mathcal{V}}\mathbf{y}$. We further assume that an eigenvector has norm 1. All proposed algorithms in this paper are implemented in Matlab version R2018a. All experiments are performed on an Intel i5-6300U with 2.5 GHz and 8 GB RAM.

2. Arnoldi method

In this section we discuss the *Arnoldi method* and a variant on this, the Residual Arnoldi algorithm which both build up an orthonormal basis for a Krylov space. We explain how convergence is measured and give some results regarding its performance when the iterations are contaminated by error. The Arnoldi method is well-known in the literature in contrast to the Residual Arnoldi method. We motivate in this section why we prefer the Residual Arnoldi method to the Arnoldi method for our extension to parametrized eigenvalue problems. We conclude Section 2 by discussing how much error we can tolerate without harming convergence.

2.1. Description of the Arnoldi method

For our application, it is important that the algorithm is robust against introduced error. The reason is that we simultaneously apply a version of the Arnoldi algorithm on matrices obtained by evaluating $\mathbf{A}$ on a set of parameter values. The Arnoldi algorithm can be found in Algorithm 1. We note that in line 2 the continuation vector $\mathbf{t}$ is [1] as the initial subspace is one dimensional. To prevent that our subspace becomes too large, we make a low-rank approximation of the tensor consisting of all iteration vectors and so error is introduced in the algorithm. In the original Arnoldi algorithm [2], the adopted continuation vector is $\mathbf{t} = [0, \ldots, 0, 1]^T$, so that $\mathbf{V}\mathbf{t}$ is equal to the last column of $\mathbf{V}$. It is showed in [14] that in

Algorithm 1 Arnoldi algorithm

Input: Matrix $\mathbf{A}$, normalized vector $\mathbf{u} \in \mathbb{R}^n$

Output: Approximate eigenpair $(\hat{\lambda}_1, \hat{\mathbf{x}}_1)$ of $\mathbf{A}$

$\mathbf{w}^\perp = \mathbf{u}$
$\mathbf{t} = [1]$
for $k = 1$ to $n-1$ **do**
    Add $\mathbf{w}^\perp$ to the subspace $\mathcal{V}$
    $\mathbf{w} = \mathbf{AVt}$
    Orthogonalise $\mathbf{w}$ with respect to $\mathbf{V}$ and call this $\mathbf{w}^\perp$
    Calculate wanted eigenpair $\left(\lambda_1^{\mathcal{V}}, \mathbf{x}_1^{\mathcal{V}}\right)$ of $\mathbf{V}^T\mathbf{AV}$
    Break down if $\left(\lambda_1^{\mathcal{V}}, \mathbf{x}_1^{\mathcal{V}}\right)$ meets a certain convergence criterium
    Construct the continuation vector $\mathbf{t}$
end for
$\hat{\lambda}_1 = \lambda_1^{\mathcal{V}}$ and Ritz vector $\hat{\mathbf{x}}_1 = \mathbf{V}\mathbf{x}_1^{\mathcal{V}}$

case there is a significant error (numerical or/and intended) present in one of the vectors, the algorithm does not converge anymore. So we look for a continuation vector such that the Arnoldi algorithm converges even when the iteration vector is contaminated with a relatively large error. In the papers [29, 30] the authors answered the question what the best choice is for the continuation vector. For this, they searched for the continuation vector that maximizes

$$\begin{aligned} \cos\left(\theta(\mathbf{x}_1, \text{Range}\{\mathbf{V}, \mathbf{AVt}\})\right) &:= \max_{\mathbf{z} \in \text{Range}\{\mathbf{V}, \mathbf{AVt}\}} \cos(\theta(\mathbf{x}_1, \mathbf{z})) \\ &= \max_{\mathbf{z} \in \text{Range}\{\mathbf{V}, \mathbf{AVt}\}} \frac{\mathbf{x}_1^* \mathbf{z}}{\|\mathbf{x}_1\| \, \|\mathbf{z}\|}, \end{aligned} \tag{2}$$

so they look which continuation vector is such that the angle between the sought eigenvector and the extended subspace is minimized.

In [30], they proved that the vector that maximizes (2) is of the form

$$\mathbf{t} = \mathbf{R}^+ \mathbf{x}_1 + \mathbf{b} \tag{3}$$

with $\mathbf{b}$ a vector in the nullspace of $\mathbf{R} = \mathbf{AV} - \mathbf{V}\left(\mathbf{V}^T\mathbf{AV}\right)$ with $\mathbf{R}^+$ the Moore-Penrose pseudo-inverse of $\mathbf{R}$.

Although this result gives us insight in the optimal continuation vector, it is impossible for practical use as it depends on the searched eigenvector. The author of [30] argued that in practice, the eigenvector

$$\mathbf{t} = \mathbf{x}_1^{\mathcal{V}}, \tag{4}$$

associated with the desired eigenvalue of the reduced problem $\mathbf{V}^T\mathbf{AV}$ is a good practical choice for the continuation vector. In the remainder of this chapter, this is used as continuation vector, despite the fact that in the case of large non-Hermitian matrices, it may be better to use the *refined Ritz vector* [29].

In the literature, the Arnoldi algorithm with (4) as continuation vector is known as the *Residual Arnoldi algorithm (RAA)* [14, 15]. The name comes from the property that the vector with which we expand the subspace is the residual vector as follows from Lemma 1. It was already observed that this choice leads to a more robust version of the Arnoldi algorithm. This is further discussed in Section 2.2.

The following result is well-known in the literature [23, Proposition 6.8] and is frequently used in the analysis of Krylov methods although it holds for all subspace methods.

Lemma 1. *Let* $\left(\lambda^{\mathcal{V}}, \mathbf{y}\right)$ *be a Ritz vector of* $\mathbf{A}$ *with respect to the subspace* $\mathcal{V}$*, then the residual satisfies*

$$\mathbf{r} = \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\mathbf{A}\mathbf{y} \tag{5}$$

so the residual is the result of applying matrix $\mathbf{A}$ *on* $\mathbf{y}$ *orthogonalized with respect to* $\mathbf{V}$*.*

Proof 1. *We can express*

$$\begin{aligned}
\mathbf{r} &= \mathbf{A}\mathbf{y} - \lambda^{\mathcal{V}}\mathbf{y} \\
&= \mathbf{A}\mathbf{V}\mathbf{x}^{\mathcal{V}} - \mathbf{V}\lambda^{\mathcal{V}}\mathbf{x}^{\mathcal{V}} \\
&= \mathbf{A}\mathbf{V}\mathbf{x}^{\mathcal{V}} - \mathbf{V}\left(\mathbf{V}^T\mathbf{A}\mathbf{V}\right)\mathbf{x}^{\mathcal{V}} \\
&= \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\mathbf{A}\mathbf{V}\mathbf{x}^{\mathcal{V}}.
\end{aligned}$$

Lemma 1 implies that choosing (4) as continuation vector can simplify the Arnoldi algorithm (see Algorithm 1) provided the norm of the residual is used as convergence criterium. If the norm of the residual is used as convergence criterion, we can simplify RAA. The resulting algorithm is shown in Algorithm 2.

Algorithm 2 Residual Arnoldi Algorithm (RAA)

Input: Matrix $\mathbf{A}$, initial vector $\mathbf{u} \in \mathbb{R}^n$

Output: Approximate eigenpair $(\hat{\lambda}, \mathbf{x})$ of $\mathbf{A}$

1: $\mathbf{r} = \mathbf{A}\mathbf{u} - \lambda^{\mathbf{u}}\mathbf{u}$, with $\lambda^{\mathbf{u}}$ the Ritz value associated with the subspace .
2: **for** $k = 1$ to n **do**
3: Add $\mathbf{r}$ to the subspace $\mathcal{V}$
4: Calculate wanted eigenpair $\left(\lambda^{\mathcal{V}}, \mathbf{x}^{\mathcal{V}}\right)$ of $\mathbf{V}^T\mathbf{A}\mathbf{V}$
5: $\hat{\mathbf{x}} = \mathbf{V}\mathbf{x}^{\mathcal{V}}$
6: Calculate the residual $\mathbf{r}$ of $\left(\lambda^{\mathcal{V}}, \hat{\mathbf{x}}\right)$ via Lemma 1
7: Stop if $\|\mathbf{r}\|$ is small enough
8: **end for**
9: $\hat{\lambda} = \lambda^{\mathcal{V}}$ and Ritz vector $\hat{\mathbf{x}} = \mathbf{V}\mathbf{x}^{\mathcal{V}}$

2.2. Convergence analysis of RAA

In the next section we extend this method such that we simultaneously apply RAA on a set of matrices obtained by evaluating $\mathbf{A}$ for a large set of parameter values. For computational efficiency and feasibility, our algorithm will not perform all iteration steps in full precision. Therefore we analyse in this section how much error can be tolerated in each iteration without harming convergence.

The possible error $\mathbf{e}$ is modelled as an error on the residual $\mathbf{r}$, i.e on the vector that we want to add to the subspace. The error $\mathbf{e}$ consists of an error $\mathbf{e}_1$ on $\mathbf{x}_1^{\mathcal{V}}$ (before the orthogonalization on line 5 in RAA) and an error $\mathbf{e}_2$ on $\mathbf{r}$ (after the orthogonalization on line 6). We assume that $\mathbf{e}_2$ is already orthogonal to $\mathcal{V}$, because otherwise the added vector is no longer orthogonal to $\mathcal{V}$. To resume, the vector that we add to the subspace can be written as

$$\begin{aligned}
\hat{\mathbf{r}} &= \left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\mathbf{A}\mathbf{V}\left(\mathbf{x}^{\mathcal{V}} + \mathbf{e}_1\right) + \mathbf{e}_2 \\
&= \mathbf{r} + \underbrace{\underbrace{\left(\mathbf{I} - \mathbf{V}\mathbf{V}^T\right)\mathbf{A}\mathbf{V}\mathbf{e}_1}_{\mathbf{e}_1^{\perp}} + \mathbf{e}_2}_{\mathbf{e}}.
\end{aligned} \tag{6}$$

The convergence of this method was analysed in [14, 15]. The authors assume that in each iteration k the error $\mathbf{e}^{(k)}$ relative to the residual, is bounded by some number η, in the sense

$$\frac{\left\|\mathbf{e}^{(k)}\right\|}{\left\|\mathbf{r}^{(k)}\right\|} \leq \eta. \tag{7}$$

The main result on which the analysis relies, is that in iteration k there always exists $\mathbf{B}_k$ such that k iterations of the original Arnoldi method on the matrix

$$\hat{\mathbf{A}}_k := \mathbf{A} + \mathbf{B}_k,$$

in the absence of errors results in the same subspace as performing k iterations of RAA on matrix $\mathbf{A}$ where in each iteration an error $\mathbf{e}^{(k)}$ is made. In these papers, the authors make the assumption that the size of $\|\mathbf{B}_k\|$ is proportional to η, i.e. there exists a C_η such that

$$\|\mathbf{B}_k\| \leq \eta C_\eta, \quad \forall k.$$

The authors indicate that they cannot prove this assumption, although they have some empirical evidence that it is satisfied. Other assumptions were made on the separation of the spectrum of both $\mathbf{A}$ and $\hat{\mathbf{A}}$.

Theorem 1. *Assume that the above conditions are satisfied, let C_X be a constant depending on C_η and the spectrum of $\mathbf{A}$. If $2\eta C_X < 1$, then*

$$\left\|\hat{\mathbf{x}}^{(k)} - \mathbf{x}\right\| \leq \frac{\kappa_k}{1 - 2\eta C_X},$$

where κ_k is a bound for the difference between the approximation and the searched eigenvector when no error is made. This value goes to zero as k increases.

We refer to [15] for a proof and more details. In practice this bound is not (or hardly) computable, so a workable value of η needs to be used in practice.

If the method converges, then the question arises how it depends on η. Empirically, the authors of [14, 15] noticed that the speed of convergence for η lower than or equal to 10^{-3} stays unchanged but that for $1 \gg \eta > 10^{-3}$ the convergence can slow down a bit. For η around 1, we do not have convergence as the norm of the error is as large as the residual norm.

We now bound the norm of the errors $\mathbf{e}_1$ and $\mathbf{e}_2$ such that convergence is still preserved, i.e. such that condition (7) is fulfilled with $\eta < 10^{-3}$. Let

$$\frac{\|\mathbf{e}_1\|}{\|\mathbf{x}^{\mathcal{V}}\|} = \|\mathbf{e}_1\| \leq \eta_X \text{ and } \frac{\|\mathbf{e}_2\|}{\left\|\mathbf{r} + \mathbf{e}_1^{\perp}\right\|} \leq \eta_R \tag{8}$$

be tolerances for the relative error, then we obtain starting from (6)

$$\begin{aligned}
\frac{\|\mathbf{e}\|}{\|\mathbf{r}\|} &\leq \frac{\left\|\mathbf{e}_1^{\perp}\right\|}{\|\mathbf{r}\|} + \frac{\|\mathbf{e}_2\|}{\|\mathbf{r}\|} \\
&= \frac{\left\|(\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{A}\mathbf{V}\mathbf{e}_1\right\|}{\|\mathbf{r}\|} + \frac{\|\mathbf{e}_2\|}{\|\mathbf{r}\|} \\
&\leq \underbrace{\frac{\|\mathbf{A}\|\,\|\mathbf{e}_1\|}{\|\mathbf{r}\|}}_{\leq \eta/2} + \underbrace{\frac{\|\mathbf{e}_2\|}{\|\mathbf{r}\|}}_{\leq \eta/2}.
\end{aligned}$$

The aim is now to bound each term by $\eta/2$. Remark that bounding the first term means that the error propagation due to the matrix-vector product needs to be limited. From (8) it follows that

$$\eta_X \leq \frac{\eta\,\|\mathbf{r}\|}{2\,\|\mathbf{A}\|} \tag{9}$$

which means that η_X scales with the residual norm and the norm of the matrix $\mathbf{A}$. The lower the residual is, the lower η_X needs to be. In the example in Section 4, we demonstrate that the convergence stagnates if the residual norm is small and η_X is kept constant.

Using $\left\|\mathbf{r}+\mathbf{e}_1^{\perp}\right\| \leq \|\mathbf{r}\| + \left\|\mathbf{e}_1^{\perp}\right\|$, we obtain

$$\begin{aligned}\frac{\|\mathbf{e}_2\|}{\|\mathbf{r}\|} &= \frac{\|\mathbf{e}_2\|}{\left\|\mathbf{r}+\mathbf{e}_1^{\perp}\right\|}\frac{\left\|\mathbf{r}+\mathbf{e}_1^{\perp}\right\|}{\|\mathbf{r}\|} \\ &\leq \frac{\|\mathbf{e}_2\|}{\left\|\mathbf{r}+\mathbf{e}_1^{\perp}\right\|}\frac{\|\mathbf{r}\|+\left\|\mathbf{e}_1^{\perp}\right\|}{\|\mathbf{r}\|} \\ &\leq \frac{\|\mathbf{e}_2\|}{\left\|\mathbf{r}+\mathbf{e}_1^{\perp}\right\|}(1+\eta/2).\end{aligned}$$

As the left-hand side needs to be smaller than $\eta/2$, we impose that

$$\eta_R \leq \frac{\eta}{2+\eta}. \tag{10}$$

3. Tensor-Arnoldi method for one parameter

As was already mentioned, the idea behind our method/algorithm is to apply RAA on a large set of matrices $\mathbf{A}(\omega_i), i = 1,\ldots,n_1$ at the same time. In this section we discuss in more detail how our developed algorithm works and how the results from the previous section are incorporated in the algorithm. We restrict ourselves to the case where the parameter space is one dimensional. The extension to more parameters is the subject of Section 5. We describe first the algorithm in a general form, then we go into more detail on some specific steps of the algorithm.

We first discretise Ω into points $\omega_1,\ldots,\omega_{n_1}$. We want to compute the desired eigenvalue λ_1 at these points. For this, we consider the linear operator $\mathcal{A}$ such that

$$\mathcal{A}\mathbf{x} = \begin{bmatrix}\mathbf{A}(\omega_1) & & \\ & \ddots & \\ & & \mathbf{A}(\omega_{n_1})\end{bmatrix}\begin{bmatrix}\mathbf{x}(\omega_1)\\ \mathbf{x}(\omega_2)\\ \vdots \\ \mathbf{x}(\omega_{n_1})\end{bmatrix} = \begin{bmatrix}\mathbf{A}(\omega_1)\mathbf{x}(\omega_1)\\ \vdots \\ \mathbf{A}(\omega_{n_1})\mathbf{x}(\omega_{n_1})\end{bmatrix} \in \mathbb{R}^{nn_1}. \tag{11}$$

If $\mathbf{x}$ is reordered as matrix $\mathbf{X} = [\mathbf{x}(\omega_1), \mathbf{x}(\omega_2), \ldots, \mathbf{x}(\omega_{n_1})] \in \mathbb{C}^{n\times n_1}$, then (11) is equivalent to an operator $\mathcal{A}(\mathbf{X})$ such that $\mathrm{vec}(\mathcal{A}(\mathbf{X})) = \mathcal{A}\mathbf{x}$. Conceptually, our method boils down to applying RAA to (11) which boils down to applying RAA to each $\mathbf{A}(\omega_i)$ individually.

In order to efficiently compute $\mathcal{A}(\mathbf{X})$ we need the expansion in (1). If a decomposition $\mathbf{X} = \mathbf{U}\mathbf{Z}$ with $\mathbf{U} \in \mathbb{R}^{n\times n_u}$ and $\mathbf{Z} \in \mathbb{C}^{n_u\times n_1}$ is given, then following [13] we obtain

$$\mathcal{A}(\mathbf{X}) = \underbrace{[\mathbf{A}_1\mathbf{U}, \mathbf{A}_2\mathbf{U}, \ldots, \mathbf{A}_m\mathbf{U}]}_{\mathbf{W}_U\in\mathbb{R}^{n\times n_u m}} \underbrace{\begin{bmatrix}\mathbf{Z}\cdot\mathrm{diag}([f_1(\omega_1), f_1(\omega_2), \ldots, f_1(\omega_{n_1})]) \\ \mathbf{Z}\cdot\mathrm{diag}([f_2(\omega_1), f_2(\omega_2), \ldots, f_2(\omega_{n_1})]) \\ \vdots \\ \mathbf{Z}\cdot\mathrm{diag}([f_m(\omega_1), f_m(\omega_2), \ldots, f_m(\omega_{n_1})])\end{bmatrix}}_{\mathbf{W}_Z\in\mathbb{C}^{n_u m\times n_1}}. \tag{12}$$

Remark that

$$\mathbf{Z}\cdot\mathrm{diag}([f_i(\omega_1),\ldots,f_i(\omega_{n_1})]) = [f_i(\omega_1)\mathbf{Z}(:,1),\ldots,f_i(\omega_{n_1})\mathbf{Z}(:,n_1)],$$

which means that the column j of $\mathbf{Z}$ is multiplied with the function f_i evaluated in ω_j.

Let $\mathbf{X}^{\mathcal{V}}$ be the matrix whose ith column is the eigenvector $\mathbf{x}^{\mathcal{V}}(\omega_i)$ of the projected eigenvalue problem $\mathbf{V}^T\mathbf{A}(\omega_i)\mathbf{V}$. We first make a low-rank decomposition $\mathbf{UZ}$ of $\mathbf{X}^{\mathcal{V}}$, then we apply $\mathcal{A}$ on $\mathbf{VX}^{\mathcal{V}} = (\mathbf{VU})\mathbf{Z}$, i.e.

$$\mathbf{W}_U\mathbf{W}_Z = \mathcal{A}(\mathbf{VX}^{\mathcal{V}}) = \mathcal{A}((\mathbf{VU})\mathbf{Z}), \tag{13}$$

using (12) and we orthogonalise the result with respect to $\mathcal{V}$ to obtain an approximate residual for each parameter value, see Lemma 1. This is analogous to lines 5 and 6 in Algorithm 2). It is not feasible to add vectors for all parameter values, so we first orthogonalise (13) with respect to $\mathcal{V}$, then we make a low-rank approximation of the resulting matrix and we add a basis for its column space to $\mathcal{V}$. For the low-rank algorithm the method described in [13] is used. We finish an iteration by calculating new eigenvalues and eigenvectors. This procedure is repeated until all residuals are below a certain tolerance ϵ. The algorithm can be found in Algorithm 3 and is called the Parametrized Residual Arnoldi Algorithm (PRAA).

3.1. Restarting the subspace

A problem is that we add multiple vectors to the subspace and as the dimension of the subspace is prescribed, this means we need to restart the subspace often. The techniques for restarting known in the literature try to delete the part of the subspace that corresponds to an unwanted part of the spectrum but still keep a significant part of the subspace, see [26]. The reason is that not only the Ritz vector associated with the wanted eigenvalue, but also some other Ritz vectors whose eigenvalues are close to $\lambda^{\mathcal{V}}$ help improving the convergence to the wanted eigenvalue. This is called *thick restarting*.

In our case, eigenvectors depend on parameters which means that for just representing the wanted eigenvalue for each parameter value, we already need quite a large subspace. If several eigenvectors per parameter value are kept in the basis, the restarted basis is relatively large, so that fewer iterations can be performed to obtain a basis of a given size. It is known from [22, 24] that the partial derivative of an eigenvector, which is a measure for the change of the eigenvector, depends the most on the eigenvectors whose eigenvalues are the closest to $\lambda(\omega)$. As we estimate the eigenvector over the parameter space, we therefore also estimate these eigenvectors.

For the calculation of $\mathbf{V}^T\mathbf{A}(\omega_i)\mathbf{V}, i = 1, \ldots, n_1$, we first calculate $\mathbf{V}^T\mathbf{A}_i\mathbf{V}, i = 1, \ldots, m$ and sum these smaller matrices appropriately. The operation $\mathbf{A}_i\mathbf{V}, i = 1, \ldots m$ is not expensive as these $\mathbf{A}_i$ is usually sparse, but the inner product $\mathbf{V}^T(\mathbf{A}_i\mathbf{V})$ need $2n^2n_v^2$ operations with n_v the dimension of $\mathbf{V}$. As $\mathbf{V}$ is built iteratively, results from the previous iteration can be used if the subspace was not restarted. The lower η_X and η_R, the higher the number of these inner products. In the example in the Section 4, we count the number of these inner products and compare them for different values of η_X and η_R.

4. Error analysis

In Section 2.2 we conclude that in RAA, η_X need to be chosen relative to the current residual and the tolerance η_R can be as high as $\dfrac{10^{-3}}{2+10^{-3}}$ without breaking down the convergence. PRAA is equivalent to applying simultaneously RAA on a set of matrices obtained by evaluating $\mathbf{A}$ in a large set of parameter values.

In this section we show with an example why it is even favourable to choose η_R relatively high, as we need fewer iterations and fewer expensive matrix-vector products. The example is described in Example 1.

Example 1. *Consider the convection-diffusion differential equation*

$$f = c_1u_x + c_2u_y + d_{11}u_{xx} + 2d_{12}u_{xy} + d_{22}u_{yy} \tag{14}$$

Algorithm 3 Parametrized Residual Arnoldi Algorithm (PRAA)

Input: $\mathbf{A}(\omega)$, orthonormal matrix $\mathbf{U} \in \mathbb{R}^{n\times p}$, tolerance ϵ, tolerances η_X and η_R for the two decompositions, maximal dimension of subspace $n^{\mathcal{V}}$

Output: Approximate eigenpair $(\hat{\lambda}(\omega_i), \mathbf{y}(\omega_i)), i = 1, 2, \ldots, n_1$

$\hat{\mathbf{R}} = [1, 1, \ldots, 1]$
while $\left\|\hat{\mathbf{R}}\right\|_F / \sqrt{n_1} > \epsilon$ **do**
  Add $\mathbf{U}$ to the subspace $\mathcal{V}$
  Calculate eigenpair $\left(\lambda^{\mathcal{V}}(\omega_i), \mathbf{x}^{\mathcal{V}}(\omega_i)\right)$ of $\mathbf{V}^T\mathbf{A}(\omega_i)\mathbf{V}, i = 1, 2, \ldots, n_1$
  $\mathbf{X}^{\mathcal{V}} = [\mathbf{x}^{\mathcal{V}}(\omega_1), \mathbf{x}^{\mathcal{V}}(\omega_2), \ldots, \mathbf{x}^{\mathcal{V}}(\omega_{n_1})]$
  if dimension $\mathcal{V} > n^{\mathcal{V}}$ **then**
    $[\mathbf{U}, \mathbf{Z}] = \text{lowrank}(\mathbf{X}^{\mathcal{V}}, 10^{-12})$
    $\mathbf{U} = \mathbf{V}\mathbf{U}$
    Make $\mathcal{V}$ empty
  else
    $[\mathbf{U}, \mathbf{Z}] = \text{lowrank}(\mathbf{X}^{\mathcal{V}}, \eta_X)$
    $[\mathbf{W}_U, \mathbf{W}_Z] = \mathcal{A}((\mathbf{V}\mathbf{U})\,\mathbf{Z})$ using (12)
    $\mathbf{W}_U^{\perp} = (\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{W}_U$ (Orthogonalize $\mathbf{W}_U$ w.r.t $\mathbf{V}$)
    Residual $\hat{\mathbf{R}} := [\hat{\mathbf{r}}(\omega_1), \hat{\mathbf{r}}(\omega_2), \ldots, \hat{\mathbf{r}}(\omega_{n_1})] = \mathbf{W}_U^{\perp}\mathbf{W}_Z$
    $[\mathbf{U}, \sim] = \text{lowrank}(\mathbf{W}_U^{\perp}\mathbf{W}_Z, \eta_R)$;
  end if
end while
$\hat{\lambda}(\omega_i) = \lambda^{\mathcal{V}}(\omega_i)$ and Ritz vector $\mathbf{y}(\omega_i) = \mathbf{V}\mathbf{X}^{\mathcal{V}}(\omega_i), i = 1, 2, \ldots, n_1$

for $(x, y) \in \Xi = (0, 1) \times (0, 1)$. The real function u has the argument (x, y). The constants c_1 and c_2 are associated with the convection part and the constants d_{11}, d_{12} and d_{22} are associated with the diffusion part of the equation. It is always assumed that matrix

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} \\ d_{12} & d_{22} \end{bmatrix}$$

is positive semi-definite. For the discretisation of Ξ we use second-order central finite difference formulas and we assume Dirichlet boundary conditions. In this way, we obtain a matrix $\mathbf{A}$ depending on the values c_1,c_2,d_{11},d_{12} and d_{22} such that

$$\mathbf{A}(c_1, c_2, d_{11}, d_{22}, d_{12}) = c_1\mathbf{A}_1 + c_2\mathbf{A}_2 + d_{11}\mathbf{A}_3 + d_{22}\mathbf{A}_4 + d_{12}\mathbf{A}_5. \tag{15}$$

For this first example we take $c_2 = 1, d_{11} = 1.1, d_{22} = 1, d_{12} = 1$ and c_1 takes values in the interval $\Omega = [-2.5, 2.5]$. We discretize this interval in 100 points. This means we can rewrite (14) as

$$\mathbf{A}(c_1) = \mathbf{A}_0 + c_1\mathbf{A}_1,$$

with $\mathbf{A}_0 = c_2\mathbf{A}_2 + d_{11}\mathbf{A}_3 + d_{22}\mathbf{A}_4 + d_{12}\mathbf{A}_5$. The desired eigenvalue is the eigenvalue with largest real part, in this case the same as the eigenvalue closest to zero. The region Ξ is discretised into 100 points in each dimension, so the resulting matrices $\mathbf{A}_i$ are of size $10^4 \times 10^4$. The results can be found in Figure 1 and Table 1. Remark that we need many iterations but tests with individual parameter values reveal that also the Arnoldi algorithm needs many iterations. We choose an example with slow convergence to be able to see clearly what effect different values for η_X and η_R have on the convergence.

At line 11 on the one hand, and at line 15 of PRAA on the other, we make low-rank approximations of the matrix consisting of the eigenvector resp. the approximate residual and so introducing errors for each parameter value $\mathbf{E}_1 := [\mathbf{e}_1(\omega_1), \mathbf{e}_1(\omega_2), \ldots, \mathbf{e}_1(\omega_{n_1})]$ and $\mathbf{E}_2 := [\mathbf{e}_2(\omega_1), \mathbf{e}_2(\omega_2), \ldots, \mathbf{e}_2(\omega_{n_1})]$. We first discuss the error $\mathbf{E}_1$. The low-rank approximation is such that the relative error is bounded by the tolerance η_X, i.e

$$\frac{\|\mathbf{E}_1\|_F}{\|\mathbf{X}^{\mathcal{V}}\|_F} \leq \eta_X. \tag{16}$$

Since $\left\|\mathbf{X}^{\mathcal{V}}\right\|_F = \sqrt{n_1}$, it follows that $\|\mathbf{E}_1\|_F \leq \eta_X \sqrt{n_1}$. From this, it can be seen that

$$\|\mathbf{e}_1(\omega_i)\|^2 \leq \eta_X^2 n_1, \quad i = 1, 2, \ldots, n_1,$$

but assuming the error is uniformly distributed over the sample points, we can state that

$$\|\mathbf{e}_1(\omega_i)\| \lessapprox \eta_X. \tag{17}$$

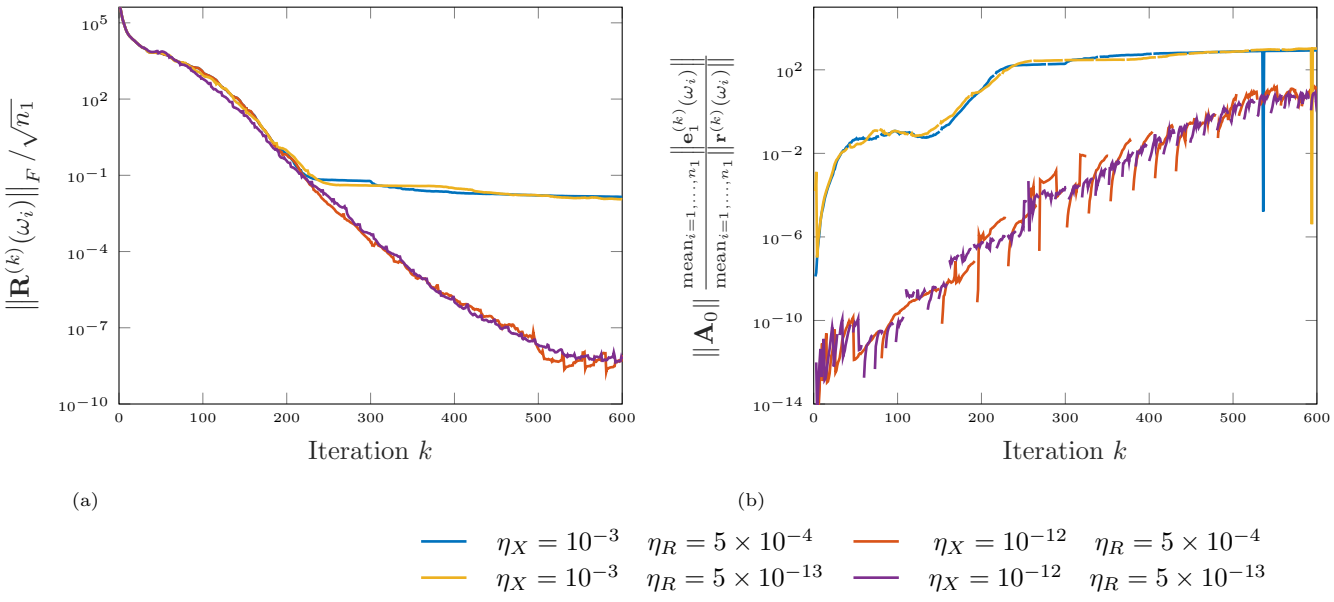


Figure 1: Results of PRAA for different tolerances η_X and η_R with maximal subspace dimension 150 for Example 1. In (a) we plot the Frobenius norm of the residual, scaled with the number of sample points. In (b) the average first error $\mathbf{e}_1$ relative to the average residual, scaled with the norm of $\mathbf{A}_0$ is shown. As long as the value is below or around 10^{-3}, the algorithm converges as if there were no error, see the criterium in (9). If the value is large (> 1) convergence is lost. The almost vertical lines in (b) are present just after iterations where we needed to restart the subspace.

Figure 1 (a) shows the average residual over the parameter and we observe that for $\eta_X = 10^{-3}$ the convergence slows down after 200 iterations. This is related to the average relative error $\mathbf{e}_1$ plotted in Figure 1 (b), as this becomes too high such that condition (9) is not fulfilled. Analogously to (17) we can prove that

$$\frac{\|\mathbf{e}_2(\omega_i)\|}{\left\|\mathbf{r}(\omega_i) + \mathbf{e}_1^{\perp}\right\|} \lessapprox \eta_R, \quad i = 1, 2, \ldots, n_1.$$

As seen in (10), we only need to make sure that $\eta_R \leq \dfrac{\eta}{2+\eta}$, where η can be chosen as high as 10^{-3}. This explains why in Figure 1 (a) choosing $\eta_R = \dfrac{10^{-3}}{2+10^{-3}}$ does not harm convergence. Furthermore, it seems even that convergence is faster for higher η_R. This phenomenon is intriguing and cannot be explained only by a relative error plot. The higher the tolerance η_R, the fewer vectors are added to the subspace. If per iteration less vectors are added, the constructed subspace reaches its

maximal dimension in a later iteration. It follows that we can do more iterations within the same subspace so we need less restarts. Commonly, the convergence of Arnoldi-type algorithms slows down with each restart [19], explaining why a higher tolerance η_R (upto some limit) leads to faster convergence

In Table 1, we see statistics about the computational cost. As expected, we observe that the number of restarts increases as the tolerances decrease and that the average size of the subspace is independent of the chosen tolerances. As an indication of the increased computational cost when taking the tolerances smaller, we count the number of inner products when creating the reduced eigenvalue problem (line 4 in PRAA). A more detailed explanation of this was given at the end of Section 3.

In the experiments in Figure 1 both η_X and η_R are kept constant. We observe that if η_X is chosen high, that the convergence is disrupted when the norm of the error $\mathbf{e}_1$ becomes high relative to the residual norm. The solution for this is to choose the tolerance η_X in iteration k relative to the residual in the previous iteration. In Figure 2, we compare results when holding η_X constant and when we adapt its value relatively with respect to the residual. We see that we obtain similar results. This is explained by Figure 2 (b) as we see that the error $\mathbf{e}_1$ relative to the residual remains constant.

	$\eta_X = 10^{-3}$ $\eta_R = 5 \times 10^{-4}$	$\eta_X = 10^{-12}$ $\eta_R = 5 \times 10^{-4}$	$\eta_X = 10^{-3}$ $\eta_R = 5 \times 10^{-13}$	$\eta_X = 10^{-12}$ $\eta_R = 5 \times 10^{-13}$
Number of restarts	7	16	15	38
Avg. size of subspace	79.5233	78.7383	79.545	81.0283
Avg. number of inner products for reduced eigenvalue problem	691.74	1429.9167	1272.9133	3434.36

Table 1: Statistics associated to the results shown in Figure 1.

The conclusion is that we obtain the best results when taking η_R high and taking η_X relative to the residual. In this way the norm of both errors $\mathbf{e}_1$ and $\mathbf{e}_2$ stays constant with respect to the residual, which preserves convergence if this constant is low enough ($< 10^{-3}$), see Section 4.

5. More than one parameter

In the previous sections, we assumed that we only have one parameter to ease the presentation. In this section, we extend our algorithm to the case of multiple parameters. This means that instead of matrices filled with a vector for each parameter value, we now have multidimensional arrays, i.e. tensors, where the mode-1 fibres are filled with vectors for each parameter value. As the storage requirement grows exponentially with the dimension, we need a suitable tensor-format. As a consequence, all steps in PRAA need to have a tensor counterpart. In this section, we first introduce some notation, then we discuss the chosen tensor-format and explain how all operations are adapted.

We assume that Ω is a Cartesian product of intervals in $\mathbb{R}$, so

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_d = [a^1, b^1] \times [a^2, b^2] \times \ldots \times [a^d, b^d].$$

Each dimension $i, i = 1, 2, \ldots, d$ is discretised into n_i points, making a total of $n_0 = n_1 n_2 \ldots n_d$ sample points where we want to approximate the desired eigenvalue. A point ω on this grid is denoted as $(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_d})$ for $i_j = 1, 2, \ldots, n_j, j = 1, 2, \ldots, d$.

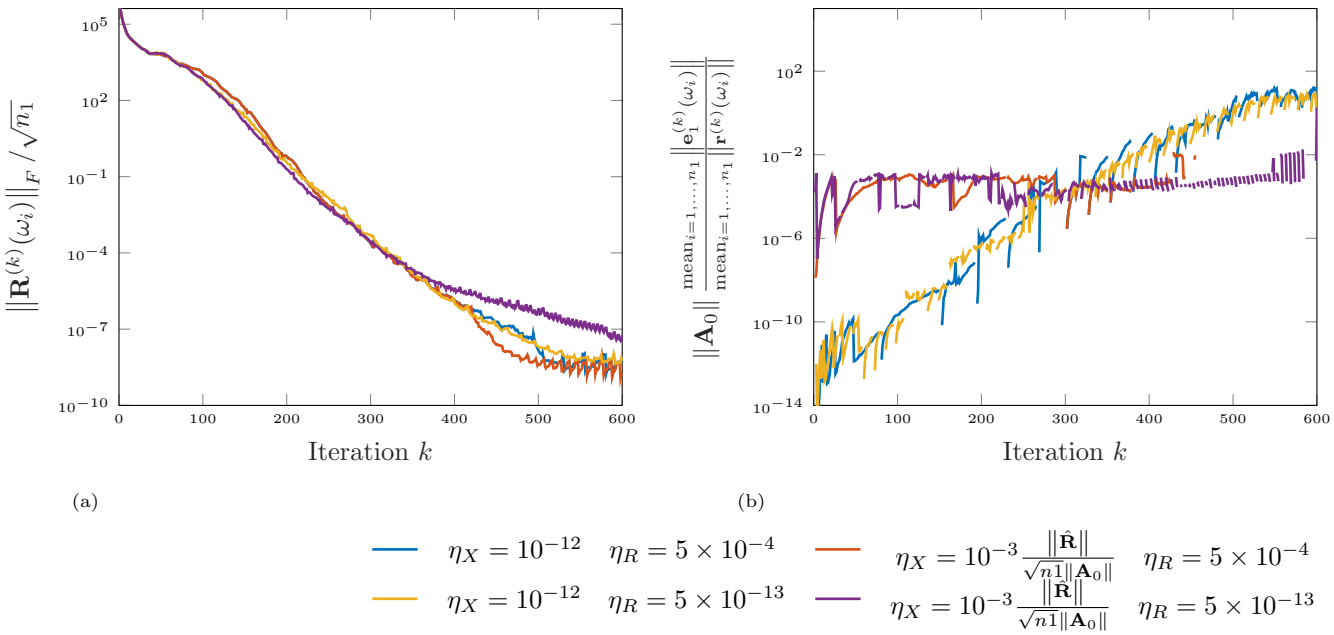


Figure 2: Results of PRAA for different tolerances η_X and η_R with maximal subspace dimension 150 for Example 1. We compare the results when taking a constant low tolerance η_X with the case that η_X is relative to the residual. In (a) we plot the Frobenius norm of the residual, scaled with the number of sample. We see that in all four cases the algorithm converges. In (b), the average error $\mathbf{e}_1$ relative to the average residual, scaled with the norm of $\mathbf{A}_0$ is shown. As expected, the ratio between $\|\mathbf{e}_1\|$ and the residual increases if η_X is kept constant. If we choose η_X relative to the residual, we see that the ratio between $\|\mathbf{e}_1\|$ and the residual stays constant.

All tensor equivalents are the calligraphic version of the letters used in the one parameter case. For example, the eigenvectors are stored in tensor $\mathcal{X} \in \mathbb{C}^{n \times n_1 \times \ldots \times n_d}$ where

$$\mathcal{X}(:, i_1, i_2, \ldots, i_d) = \mathbf{x}(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_d}).$$

We define tensor $\mathcal{F} \in \mathbb{R}^{m \times n_1 \times n_2 \times \ldots \times n_d}$ as the tensor with all function values, so

$$\mathcal{F}(i_0, i_1, \ldots, i_d) = f_{i_0}\left(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_d}\right)$$

for a point $\omega = \left(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_d}\right)$. If $d = 1$, this is the matrix

$$\mathbf{F} = \begin{bmatrix} f_1(\omega_1) & f_1(\omega_2) & \ldots & f_1(\omega_{n_1}) \\ f_2(\omega_1) & f_2(\omega_2) & \ldots & f_2(\omega_{n_1}) \\ \vdots & & & \\ f_m(\omega_1) & f_m(\omega_2) & \ldots & f_m(\omega_{n_1}) \end{bmatrix}.$$

We first state some basic operations that can be performed on tensors, then we describe how to adapt PRAA. A detailed overview on tensors can be found in [11].

Definition 1. 1. *The mode-i matriciziation or unfolding of a tensor $\mathcal{Z} \in \mathbb{C}^{n \times n_1 \times \ldots \times n_d}$ is a matrix $\mathbf{Z}_{(i)} \in \mathbb{C}^{n_i \times (n n_1 \ldots n_{i-1} n_{i+1} \ldots n_d)}$ where the columns are the mode-i vectors. The order in which they are placed in $\mathbf{Z}_{(i)}$ is chosen to be canonical.*

2. *The mode-i multiplication of a matrix $\mathbf{M} \in \mathbb{R}^{k \times n_i}$ with tensor $\mathcal{Z}$ is defined as*

$$\mathcal{Y} = (\mathbf{I}, \ldots, \mathbf{I}, \mathbf{M}, \mathbf{I}, \ldots, \mathbf{I}) \cdot \mathcal{Z} \Leftrightarrow \mathbf{Y}_{(i)} = \mathbf{M}\mathbf{Z}_{(i)}$$

where $\mathbf{M}$ is at the $i+1$th position in the tuple.

3. *The* Hadamard product *between two tensors* $\mathcal{U}$ *and* $\mathcal{W}$ *of the same dimension, is obtained by the elementwise multiplication and is denoted by* $\mathcal{U}.{}_H\mathcal{W}$.

If $\mathbf{U}$ is the column basis for the tensor $\mathcal{X}$, then we can write

$$\mathcal{X} := (\mathbf{U}, \mathbf{I}, \mathbf{I}, \ldots, \mathbf{I}) \,.\mathcal{Z} \tag{18}$$

with $\mathbf{U} \in \mathbb{R}^{n \times k}$ and $\mathcal{Z} = \mathbb{R}^{k \times n_1 \times n_2 \times \ldots \times n_d}$, where $\mathbf{U}$ is a basis for $\mathbf{X}_{(0)}$.

We look now for a suitable tensor representation such that all the computations in the iteration can be done in an efficient way. A strong requirement for the decomposition of $\mathcal{X}^{\mathcal{V}}$ besides the storage requirement is that the following operations in PRAA can be computed efficiently within this decomposition:

1. Calculating a real basis for the column space, even if the tensor contains complex numbers (lines 7, 11 and 15 in PRAA (Algorithm 3)).
2. Approximating a tensor up to a given tolerance (lines 7, 11 and 15), in the same sense as in (16).
3. Computing a decomposition of $\mathcal{W}_Z$ given the decomposition of $\mathcal{Z}$ (line 12).

The Tensor-Train format (TT-format) satisfies all the above conditions. Before we illustrate this we briefly recall the TT-format. A more detailed overview can be found in [20]. A tensor $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_m}$ is in TT-format if

$$\mathcal{Y}(i_1, i_2, \ldots, i_m) = \mathbf{G}_1^Y(i_1)\mathbf{G}_2^Y(i_2)\ldots\mathbf{G}_m^Y(i_m) \tag{19}$$

with $\mathbf{G}_j^Y(i_j)$ a $r_{j-1}^Y \times r_j^Y$ matrix. The length r_{j-1}^Y and width r_j^Y of these matrices $r_i^Y, i = 0, \ldots, m$ are called the *TT-ranks*. It can immediately be observed that $r_0^Y = r_m^Y = 1$. We call $r_i^Y, i = 1, 2, \ldots, m-1$ the *interior* ranks. The storage of the TT-representation is lower than mnr^2 with $r_i^Y \leq r$ and $n_i \leq n$, which is for r small significantly smaller than the storage of the full tensor. The matrices $\mathbf{G}_i^Y$ can be stored in a tensor $\mathcal{G}_j^Y \in \mathbb{R}^{r_{j-1}^Y \times n_j \times r_j^Y}, j = 1, \ldots, m$ as $\mathcal{G}_j^Y(:, i_j, :) = \mathbf{G}^Y(i_j), j = 1, 2, \ldots, n_j$.

For computing a TT-decomposition of a tensor, we perform for each mode consecutively a matricization, an SVD and a reshape, see [20, Algorithm 1]. Approximating a tensor for a given tolerance in the same sense as (16) can also be done in the TT-decomposition, see [20, Theorem 2.2].

5.1. Adaptations to the algorithm

We start by explaining how we calculate a TT-decomposition of $\mathcal{W}_Z$ (the equivalent of $\mathbf{W}_Z$ in (12)), directly from the decomposition of $\mathcal{Z}$. To see this, we recall the case of $d = 1$. Let $\mathbf{1}_q$ be a vector with all ones of length q. We first note that

$$\mathbf{Z} \cdot \mathrm{diag}([f_i(\omega_1), f_i(\omega_2), \ldots, f_i(\omega_{n_1})]) = \mathbf{Z}.{}_H(\mathbf{1}_{n_u}[f_i(\omega_1), f_i(\omega_2), \ldots, f_i(\omega_{n_1})])$$

from which it follows that

$$\mathbf{W}_Z = ((\mathbf{1}_{n_u} \otimes \mathbf{I}_m)\mathbf{F}) \,.{}_H \left((\mathbf{I}_{n_u} \otimes \mathbf{1}_m)\mathbf{Z}\right).$$

This means that in case of multiple parameters, $\mathcal{W}_Z$ can be generalised to

$$\mathcal{W}_Z = (\mathbf{1}_{n_u} \otimes \mathbf{I}_m, \mathbf{I}, \ldots, \mathbf{I}) \,.\mathcal{F}.{}_H \left(\mathbf{1}_{n_u} \otimes \mathbf{1}_m, \mathbf{I}, \ldots, \mathbf{I}\right).\mathcal{Z}.$$

From [20], we obtain that

$$\begin{cases} \mathbf{G}_{j+1}^{W_Z}(i_j) &= \mathbf{G}_{j+1}^F(i_j) \otimes \mathbf{G}_{j+1}^Z(i_j) \qquad j = 1, 2, \ldots, d, i_j = 1, 2, \ldots, n_j, \\ \mathbf{G}_1^{W_Z} &= \mathbf{G}_1^F \otimes \mathbf{G}_1^Z. \end{cases} \tag{20}$$

We see that the TT-representation of $\mathcal{W}_Z$ depends on the TT-representation of $\mathcal{F}$ and $\mathcal{Z}$. In the next paragraph we discuss how to make a TT-representation of $\mathcal{F}$. Note that the TT-decomposition is not unique. Moreover, it is possible that the ranks of the decomposition in (20) of the tensor $\mathcal{W}_Z$ are overestimated. Calculating a TT-decomposition with a given tolerance is discussed in [20]. At lines 7, 11 and 15 of Algorithm 3, we want to approximate a possibly complex tensor $\mathcal{X}^{\mathcal{V}}$ rounded to a given tolerance and keeping the basis $\mathbf{U}$ for the first mode real-valued. We can perform these operations by the TT-rounding algorithm presented in [20].

If all functions $f_i, i = 1, 2, \ldots, m$, are separable, then there is no need to first calculate a TT-decomposition of $\mathcal{F}$. We can get a decomposition directly from the separability of the functions f_i. As the functions f_i are separable there are functions $g^i_{i_j}(\omega_{i_j})$ such that $f_i(\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_d}) = g^i_{i_1}(\omega_{i_1}) g^i_{i_2}(\omega_{i_2}) \ldots g^i_{i_d}(\omega_{i_d})$. It can be seen that a possible TT-decomposition is of the form

$$\begin{cases} \mathbf{G}^F_1 = \mathbf{I}_m \\ \mathbf{G}^F_{j+1}(i_j) = \operatorname{diag}([g^j_1(\omega_{i_j}), g^j_2(\omega_{i_j}), \ldots, g^j_m(\omega_{i_j})]) \qquad j = 1, 2, \ldots, d-1, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i_j = 1, 2, \ldots, n_j, \\ \mathbf{G}^F_{d+1}(i_d) = [g^d_1(\omega_{i_d}), g^d_2(\omega_{i_d}), \ldots, g^d_m(\omega_{i_d})]^T \qquad j = d, i_d = 1, 2, \ldots, n_d. \end{cases} \tag{21}$$

In this case, all interior ranks are m. This means that the interior ranks of the tensor $\mathcal{W}_Z$ are maximal $mr^Z_i, i = 0, 1, \ldots, d$. Note that also here the ranks of the decomposition are possibly overestimated. The lower are the ranks of the tensor $\mathcal{Z}$, the more efficient the other operations are. To have this, it is important how the eigenvectors are stored in the tensor $\mathcal{X}^{\mathcal{V}}$. Eigenvectors are only unique up to a complex constant, but for the ranks of the tensor $\mathcal{X}^{\mathcal{V}}$ it makes a difference which constant is used. We illustrate this in Example 2. The best we can do is to compare the eigenvector $\mathbf{x}(\omega)$ for a point ω with a certain vector $\mathbf{a}$, and store instead of $\mathbf{x}(\omega)$ the vector $\dfrac{(\mathbf{a}^H \mathbf{x}(\omega))}{\|\mathbf{a}^H \mathbf{x}(\omega)\|} \mathbf{x}(\omega)$. Remark that if $\mathbf{a}$ and the eigenvector have no complex elements then this boils down to multiplying with 1 or -1. A possible choice for $\mathbf{a}$ is to take the eigenvector corresponding to a sample point.

Example 2. *Let $\mathbf{A}(\omega_1, \omega_2)$ be a 1×1-matrix, depending on two parameters ω_1 and ω_2 in the parameter space $[a_1, b_1] \times [a_2, b_2]$. This means that the eigenvector is equal to -1 or 1. We discretise both intervals into three points. The tensor $\mathcal{X} \in \mathbb{R}^{1 \times 3 \times 3}$ representing the eigenvector over the parameters is*

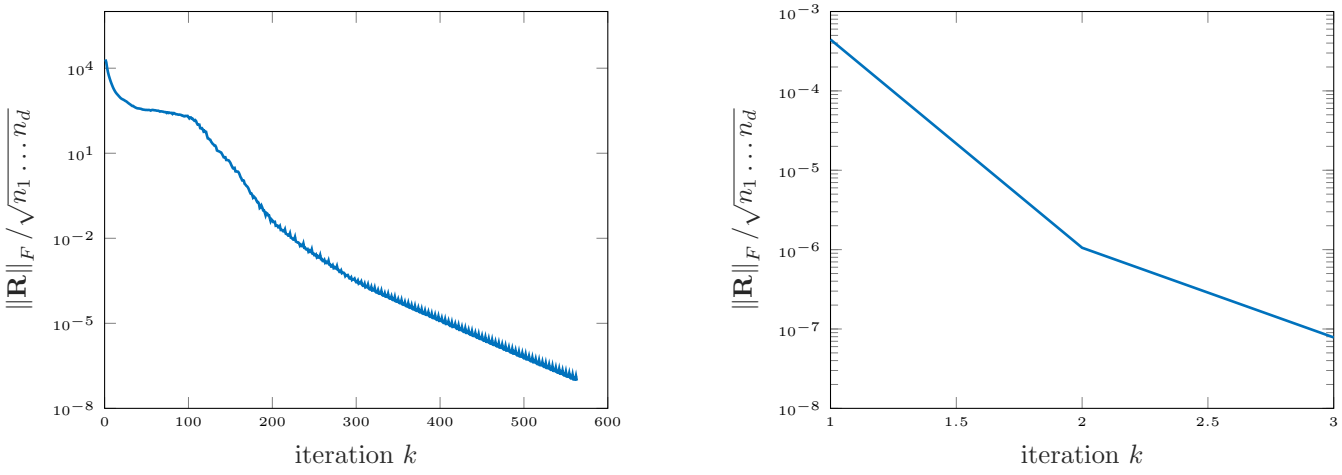
$$\mathcal{X}(1, :, :) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix},$$

so its rank is one. If for one of the parameter values we use as eigenvector -1 instead of 1, then

$$\mathcal{X}(1, :, :) = \begin{bmatrix} -1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix},$$

so the rank is two. This means that the ranks of the decomposition depend on the normalization of the eigenvector.

If there are not many points in each dimension, it might be better not to use tensors but to reshape the tensor $\mathcal{X}^{\mathcal{V}}$ as a matrix and to apply PRAA as described in Algorithm 3. The advantage of using a decomposition of the tensor is to reduce the storage and to reduce the number of operations, but if the number of parameter values is low in each dimension, then we cannot gain from the TT-representation as the cost of decomposing is too high in comparison with the possible gain in the remaining steps of the algorithm.



(a) (b)

Figure 3: Results of PRAA with maximal subspace dimension 100 for Example 3. In (a), we see the convergence when taking $n_1 = n_2 = n_3 = 10$. We see that we need quite a large number of iterations but tests showed us that using Arnoldi for all parameter points separately needs approximately the same number of iterations. If we take the resulting subspace $\mathcal{V}$ and we use this as the initial subspace for $n_1 = n_2 = n_3 = 30$, we see that we just need a few iterations

5.2. Numerical Experiments

In this section we extend Example 1 to multiple parameters and give the convergence results. The code can be downloaded following this link.

Example 3. *Consider the convection-diffusion differential equation* (14) *in Example 1. We consider the case where* $d_{12} = 0$*, we vary the variables* c_1 *and* c_2 *both in the interval* $[4, 6]$ *and we let the variable* $d := d_{11} = d_{22}$ *vary in the interval* $[-1.1, -0.9]$*. In this way we can rewrite* (15) *into the form*

$$\mathbf{A}(c_1, c_2, d) = c_1\mathbf{A}_1 + c_2\mathbf{A}_2 + d\mathbf{A}_3.$$

The desired eigenvalue is the eigenvalue with the smallest real part, in this case the same as the eigenvalue closest to zero. The matrices are of size $10^3 \times 10^3$ *and we choose* $n_1 = n_2 = n_3 = 30$ *resulting in a total of* $n_0 = 27 \times 10^3$ *parameter values. As tolerances we choose* $\eta_R = \dfrac{10^{-3}}{2 + 10^{-3}}$ *and* $\eta_X^{(k)} = 10^{-3} \left\|\hat{\mathbf{R}}^{(k)}\right\|_F / \sqrt{n_0}$

We first solve the problem discretised in each dimension by 10 interior discretisation points. We see the result in Figure 3 (a). We observe that we need many iterations for convergence to the solution which is not surprising as using Arnoldi's method for all parameter points separately needs approximately the same number of iterations. The obtained subspace $\mathcal{V}$ is then used as the initial subspace when discretising each dimension in 30. We see in Figure 3 (b) that we only need a few iterations.

6. Parametrized shift-inverted Residual Arnoldi method

In each iteration of RAA, we applied the operator $\mathcal{A}$ (see (11)) or its tensor counterpart to Ritz vectors contaminated with error. In this way we are able to compute eigenvalues which are extreme in some sense. Users are often interested in the eigenvalue nearest σ. We make here a variant of PRAA to make a global approximation of the eigenvalue closest to σ. For this, the shift-inverted Residual Arnoldi method can be used. We assume for now that $\sigma = 0$.

In this case we need to be able to efficiently compute linear systems with $\mathbf{A}(\omega)$ for all parameter samples. The inverse of $\mathbf{A}(\omega)$ can in general not be written in the form (1). However if all matrices except one are of rank one, i.e

$$\mathbf{A}(\omega) = f_0(\omega)\mathbf{A}_0 + \sum_{i=1}^{m} f_i(\omega)\mathbf{A}_i \tag{22}$$

with $\mathbf{A}_i = \mathbf{u}_i\mathbf{q}_i^T, i = 1, \ldots, m$ and $\mathbf{u}_i, \mathbf{q}_i \in \mathbb{R}^n$, it can be proven via the Sherman-Morrison-Woodbury formula [10, 28] that for a vector $\mathbf{q}$, it holds that

$$\mathbf{A}(\omega)^{-1}\mathbf{q} \in \text{span}(\mathbf{A}_0^{-1}\mathbf{q}, \mathbf{A}_0^{-1}\mathbf{u}_1, \mathbf{A}_0^{-1}\mathbf{u}_2, \ldots, \mathbf{A}_0^{-1}\mathbf{u}_m). \tag{23}$$

We see that there is only one basis vector of the subspace that depends on $\mathbf{q}$, while other vectors are the same for any $\mathbf{q}$ and for all parameter samples. Hence, one could keep all other vectors during all iterations in subspace $\mathcal{V}$. We note that from a computational point of view, we require that m is small with respect to the dimension.

It is possible to directly apply PRAA on $\mathbf{A}(\omega)^{-1}$, however, in [14, 15], the authors proposed an adaptation of RA. In exact arithmetic, if we apply the RA algorithm on $\mathbf{A}^{-1}$, we add the residual vector $\mathbf{r}_{A^{-1}} = \mathbf{A}^{-1}\mathbf{y} - \frac{1}{\lambda}\mathbf{y}$ with $(\lambda, \mathbf{y})$ a Ritz-pair of $\mathbf{A}$. The authors proposed to first calculate the residual $\mathbf{r} = \mathbf{A}\mathbf{y} - \lambda\mathbf{y}$, to solve the system

$$\mathbf{A}\mathbf{u} = \mathbf{r}. \tag{24}$$

and to add the resulting $\mathbf{u}$ to the subspace. The authors in [14, 15] proved that $\mathbf{u}$ is parallel to $\mathbf{r}_{A^{-1}}$.

As the authors of [14, 15] claim that also here the error proportional to the residual can be as high as 10^{-3} without harming the convergence, it is more appropriate to calculate the vector via (24). Their method was called the Shift-and-invert Residual Arnoldi algorithm and similarly to PRAA, we extend this method to the parametric case. Instead of (24) we only need to calculate

$$\mathbf{A}_0\mathbf{u}(\omega) = \mathbf{r}(\omega)$$

up to a relative tolerance η_R, as we keep the other basis vectors in (23) in the subspace $\mathcal{V}$ in all iterations. The tolerance for the relative error when solving (24) is denoted with η_R and the tolerance for the low-rank approximation of the eigenvector over the parameter, is denoted with η_X.

The algorithm can be found in Algorithm 4 and is called Parametrized Shift-Inverted Residual Algorithm (PSIRAA). It takes the additional assumption that $f_0(\omega) \equiv 1$. Note that if $\sigma = 0$ and f_0 has no zeros in Ω, this assumption can be made without losing generality. In case $\sigma \neq 0$, matrix $\mathbf{A}_0$ in (22) needs to be replaced by

$$\mathbf{A}_0 - \sigma\mathbf{I}$$

in a preliminary step.

6.1. Numerical experiments

The following examples arise from spectrum-based stabilization methods for time-delay systems [7, 17]. Time-delay stability can be inferred from the spectral abscissa (real part of the rightmost eigenvalue) of an eigenvalue problem, discretizing the infinitesimal generator operator by the collocation approach [6]. Spectrum-based stabilization is based on minimizing the spectral abscissa as a function of controller parameter and favors situations where the spectral abscissa is not smooth, since minima of the spectral abscissa are often characterized by defective rightmost eigenvalues [8]. In this context, our novel approach may lead to a significant contribution in probabilistic spectrum-based stabilization methods [7] since it permits to tackle larger scale problems with higher stochastic dimension.

Algorithm 4 Parametrized Shift-Inverted Residual Arnoldi Algorithm (PSIRAA). For ease of notation, we assume here that the parameter space is one dimensional.

Input: $\mathbf{A}(\omega)$, orthonormal matrix $\mathbf{U} \in \mathbb{R}^{n\times p}$, tolerance ϵ, tolerances η_X and η_R for the two decompositions, maximal dimension of subspace $n^{\mathcal{V}}$

Output: Approximate eigenpair $(\hat{\lambda}(\omega_i), \mathbf{y}(\omega_i)), i = 1, 2, \ldots, n_1$

Solve m linear systems: $\mathbf{A}_0\mathbf{v}_i = \mathbf{u}_i$ (with $\mathbf{A}_i = \mathbf{u}_i\mathbf{q}_i^T, i = 1, \ldots, m$)
$\mathcal{V} = \text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ with orthonormal basis $\mathbf{V}$
$\hat{\mathbf{R}} = [1, 1, \ldots, 1]$
while $\left\|\hat{\mathbf{R}}\right\|_F / \sqrt{n_1} > \epsilon$ **do**
    Add $\mathbf{U}$ to the subspace $\mathcal{V}$
    Calculate eigenpair $\left(\lambda^{\mathcal{V}}(\omega_i), \mathbf{x}^{\mathcal{V}}(\omega_i)\right)$ of $\mathbf{V}^T\mathbf{A}(\omega_i)\mathbf{V}, i = 1, 2, \ldots, n_1$
    $\mathbf{X}^{\mathcal{V}} = [\mathbf{x}^{\mathcal{V}}(\omega_1), \mathbf{x}^{\mathcal{V}}(\omega_2), \ldots, \mathbf{x}^{\mathcal{V}}(\omega_{n_1})]$
    if dimension $\mathcal{V} > n^{\mathcal{V}}$ **then**
        $[\mathbf{U}, \mathbf{Z}] = \text{lowrank}(\mathbf{X}^{\mathcal{V}}, 10^{-12})$
        $\mathbf{U} = \mathbf{V}\mathbf{U}$
        $\mathcal{V} = \text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ with orthonormal basis $\mathbf{V}$
    else
        $[\hat{\mathbf{U}}, \mathbf{Z}] = \text{lowrank}(\mathbf{X}^{\mathcal{V}}, \eta_X)$
        $[\mathbf{W}_U, \mathbf{W}_Z] = \mathcal{A}\left((\mathbf{V}\hat{\mathbf{U}})\mathbf{Z}\right)$ using (12)
        $\mathbf{W}_U^{\perp} = (\mathbf{I} - \mathbf{V}\mathbf{V}^T)\mathbf{W}_U$ (Orthogonalize $\mathbf{W}_U$ w.r.t $\mathbf{V}$)
        Residual $\hat{\mathbf{R}} := [\hat{\mathbf{r}}(\omega_1), \hat{\mathbf{r}}(\omega_2), \ldots, \hat{\mathbf{r}}(\omega_{n_1})] = \mathbf{W}_U^{\perp}\mathbf{W}_Z$
        $[\hat{\mathbf{U}}, \sim] = \text{lowrank}(\mathbf{W}_U^{\perp}\mathbf{W}_Z, \eta_R)$;
        Solve linear system $\mathbf{A}_0\mathbf{U} = \hat{\mathbf{U}}$
    end if
end while
$\hat{\lambda}(\omega_i) = \lambda^{\mathcal{V}}(\omega_i)$ and Ritz vector $\mathbf{y}(\omega_i) = \mathbf{V}\mathbf{X}^{\mathcal{V}}(\omega_i), i = 1, 2, \ldots, n_1$

Example 4. *The spectral abscissa function, associated with the oscillator with feedback delay system [8, Example 2], is approximated by the Infinitesimal Generator approach [6] as the spectral abscissa of the following parametrized matrix:*

$$\mathbf{A}(\omega_1, \omega_2; K_1, K_2) = \mathbf{A}_0(K_1, K_2) - \omega_1^2\mathbf{A}_1 + -2\omega_1\omega_2\mathbf{A}_2, \quad \text{with } \omega_1 \in [0.9, 1.1], \omega_2 \in [0.1, 0.2]. \tag{25}$$

The non-singular matrix $\mathbf{A}_0$ *and the rank-1 matrices* $\mathbf{A}_1$, $\mathbf{A}_2$ *have dimension* 10002×10002 *and are given by:*

$$\mathbf{A}_0(K_1, K_2) = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & K_1 & K_2 \\ & & & \mathbf{D}_M \otimes \mathbf{I}_2 & & & \end{pmatrix}$$

$$\mathbf{A}_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}^T, \quad \mathbf{A}_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}^T,$$

where $\mathbf{D}_M \in \mathbb{R}^{M \times M}$ *is the differentiation matrix [27, Chapter 6] obtained with* $M = 5000$ *Chebyshev nodes as collocation points. The controller parameters* K_1 *and* K_2 *are selected so that the spectral abscissa function presents different behaviors on the parametric domain* $(\omega_1, \omega_2) \in [0.9, 1.1] \times [0.1, 0.2]$ *as illustrated in Figures 4 (a) and (b): for* $(K_1, K_2) = [0.2, 0.2]$ *the spectral abscissa smoothly behaves, while for* $(K_1, K_2) = [0.6179, -0.0072]$ *the spectral abscissa presents non-Lipschitz continuous points, due to multiple non-semi-simple rightmost eigenvalues. We refer to [8] for further information on the example.*

We use $\eta_R = \dfrac{10^{-3}}{2 + 10^{-3}}$ *fixed,* $\eta_X^{(k)} = 10^{-3}/2 \left\| \hat{\mathbf{R}}^{(k)} \right\|_F / \sqrt{n_0}$ *relative to the residual as explained in Section 4 and 50 as maximal dimension of the subspace.. The parameter intervals are both discretised into 30 points, so we have a total of* $n_0 = 900$ *points for which the spectral abscissa needs to be found. The spectral abscissa is plotted in Figure 4 (a)-(b). The convergence results are shown in Figure 4 (c)-(d).*

Example 5. *We analyse the spectral abscissa function associated with the heat transfer set up with two static feedback controllers* $\mathbf{K}$*, designed by probabilistic stabilization method considering the temperature on the left cooler and both coolers are affected by uncertainty [7, Section 5.3]. The spectral abscissa function is determined by the eigenvalues of the following* 10040×10040 *parametrized matrix*

$$\mathbf{A}(\omega_1, \omega_2; \mathbf{K}) = \mathbf{A}_0(\mathbf{K}) + \frac{1 - \omega_1}{\omega_1} \mathbf{A}_1 + \frac{1 - \omega_2}{\omega_2} \mathbf{A}_2,$$
$$\text{with } \omega_1 \in [13.5, 16.5], \omega_2 \in [15.3, 18.7].$$

The two static controllers $\mathbf{K}$*, designed in order to minimize the mean of the spectral abscissa, modelling the uncertainty on the temperatures by independent uniform random variables [7, Section 5.3], lead to different behaviors of the spectral abscissa. In Figure 5 (a) resp. (b), the spectral abscissa for* $\mathbf{K} = [0.5105, -9.1810 \times 10^{-2}]$ *resp.* $\mathbf{K} = [0.45043, -0.1631]$ *is used. The parameter intervals are both discretised into 30 points, so we have a total of* $n_0 = 900$ *parameter values for which the spectral abscissa needs to be found. We use* $\eta_R = \dfrac{10^{-3}}{2 + 10^{-3}}$ *fixed,* $\eta_X^{(k)} = \dfrac{10^{-3}}{2} \left\| \hat{\mathbf{R}}^{(k)} \right\|_F / \sqrt{n_0}$ *relative to the residual as explained in Section 4 and 100 as maximal dimension of the subspace. The convergence results are shown in Figure 5 (c)-(d).*

We see in Figures 4 (c)-(d) and 5 (c)-(d) that in both examples only a few iterations were needed. In Figure 4 (b) and 5 (a)-(b) it can be observed that the spectral abscissa over the parameter is not differentiable as in some points (we only count eigenvalues with imaginary part equal or greater than zero) there are multiple eigenvalues for which the real part collides. Such a kink is the result of eigenvalues with the same real part crossing each other or of interacting eigenvalues. We see that nevertheless the dimension of searched subspace stays small. This can be explained as follows. In order to accurately

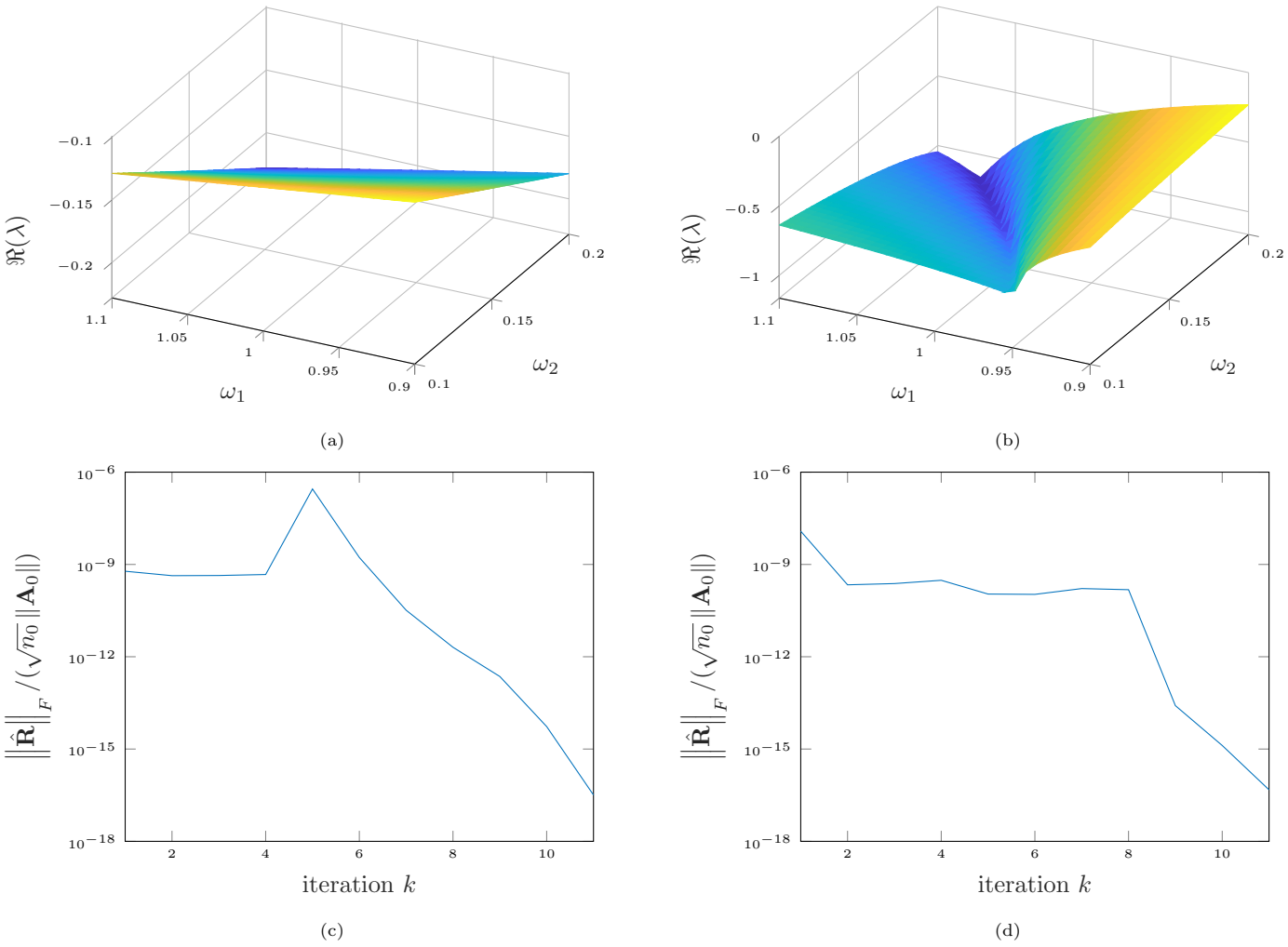


Figure 4: The spectral abscissa over the parameters for the two different cases in Example 4. We see in (a) that the spectral abscissa varies smoothly over the parameters, in (b) this is not the case anymore. The convergence results for the two different cases in Example 4 are plotted in (c) and (d). We see that in the two cases the algorithm converges within eleven iterations.

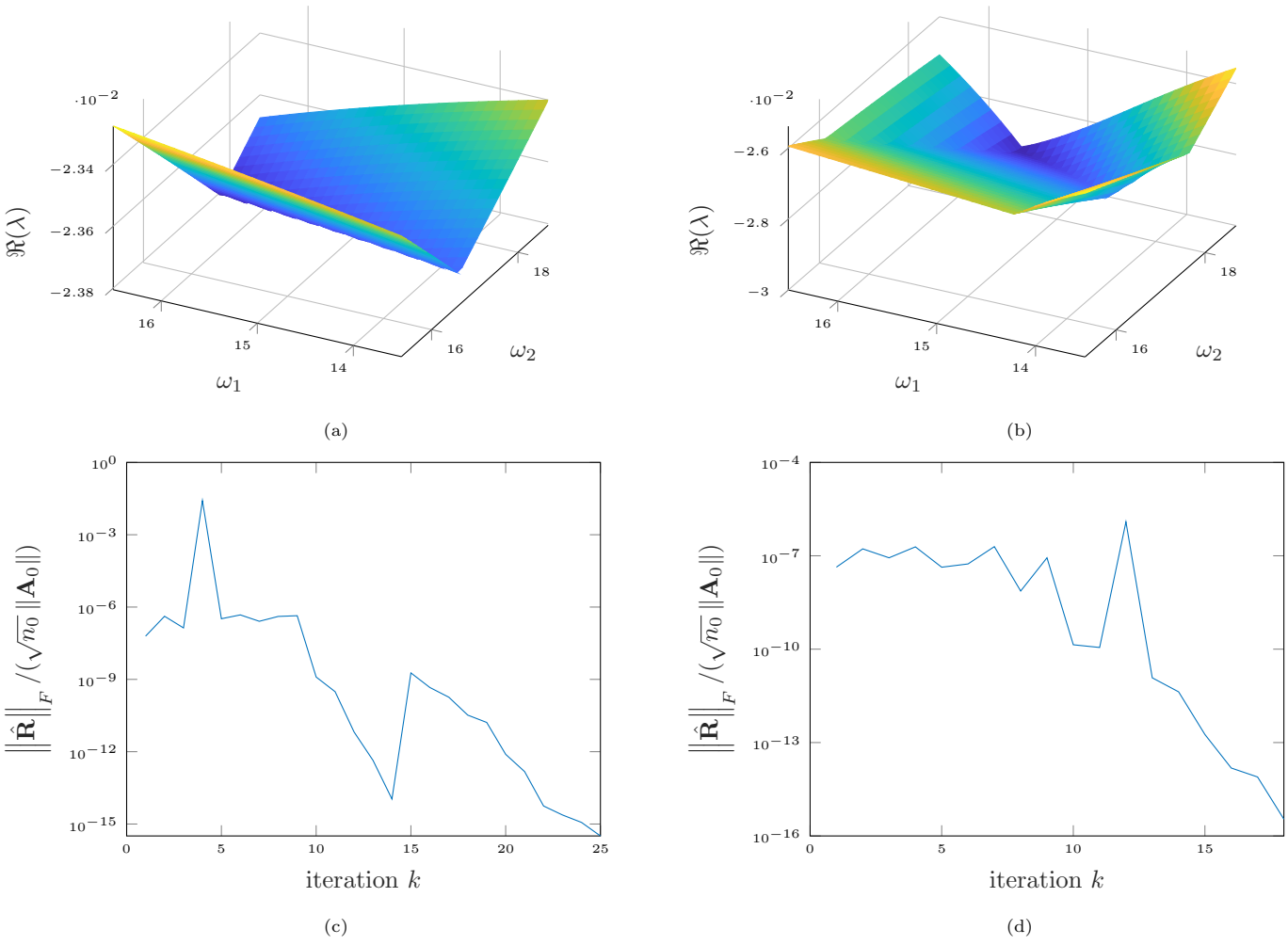


Figure 5: In (a)-(b), The spectral abscissa over the parameters for the two different cases in Example 5. The convergence results for the two different cases in Example 5 can be seen in (c)-(d).

capture such a crossing or interaction in the approximation, it is sufficient that the eigenspace corresponding to the two (or more) eigenvalues can be well approximated by the subspace on which we project. Even if the interaction is characterized by a multiple defective eigenvalue, the generalized eigenspaces remain in general smooth functions of the parameters.

We illustrated with numerical experiments that a higher η_R even improves performance, since a higher tolerance leads to a slower increase of the subspace dimension and consecutively fewer restarts. At the end of Section 6, we made the extension to multiple parameters and to shift-inverted Arnoldi for a specific range of problems. A possible extension is to make a version of the parametric shift-inverted Arnoldi method when more than one matrix has rank higher than one. The difficulty in this problem is that we need to be able to solve parametrized linear systems efficiently. Another possibility for further research is the calculation of statistical moments of certain eigenvalues over the parameter.

Acknowledgements

The authors thank the referees for their helpful remarks. We also thank Nick Vannieuwenhoven for some fruitful discussions about the used tensor format and we thank Luca Fenzi for the examples used in Section 6.1. This work was supported by the project KU Leuven Research Council grant C14/17/072 and by the project G092721N of the Research Foundation-Flanders (FWO - Vlaanderen). This work does not have any conflicts of interest.

References

[1] Andreev, R., Schwab, C., 2012. Sparse Tensor Approximation of Parametric Eigenvalue Problems, Springer, Berlin, Heidelberg.

[2] Arnoldi, W.E., 1951. The principle of minimized iterations in the solution of the matrix eigenvalue problem. Quarterly of Applied Mathematics doi:10.1090/qam/42792.

[3] Benner, P., Onwunta, A., Stoll, M., 2019. A Low-Rank Inexact Newton-Krylov Method for Stochastic Eigenvalue Problems. Computational Methods in Applied Mathematics doi:10.1515/cmam-2018-0030.

[4] Bouaricha, A., 1996. Tensor-Krylov methods for large nonlinear equations. Computational Optimization and Applications doi:10.1007/BF00248265.

[5] Bouaricha, A., Schnabel, R.B., 1998. Tensor methods for large sparse systems of nonlinear equations. Mathematical Programming, Series B doi:10.1007/BF01580076.

[6] Breda, D., Maset, S., Vermiglio, R., 2015. Stability of linear delay differential equations: A numerical approach with MATLAB, in: Briefs in Control, Automation and Robotics. Springer.

[7] Fenzi, L., Michiels, W., 2017. Robust stability optimization for linear delay systems in a probabilistic framework. Linear Algebra and Its Applications doi:10.1016/j.laa.2017.03.020.

[8] Fenzi, L., Michiels, W., 2019. Polynomial (chaos) approximation of maximum eigenvalue functions. Numerical Algorithms doi:10.1007/s11075-018-00648-9.

[9] Gumussoy, S., Henrion, D., Millstone, M., Overton, M.L., 2009. Multiobjective robust control with HIFOO 2.0. IFAC Proceedings Volumes 42, 144–149.

[10] Hager, W.W., 1989. Updating the inverse of a matrix. SIAM Review doi:10.1137/1031049.

[11] Kolda, T.G., Bader, B.W., 2009. Tensor Decompositions and Applications. SIAM Review doi:10.1137/07070111X.

[12] Kressner, D., 2009. A block Newton method for nonlinear eigenvalue problems. Numer. Math 114, 355–372. doi:10.1007/s00211-009-0259-x.

[13] Kressner, D., Tobler, C., 2011. Low-Rank Tensor Krylov Subspace Methods for Parametrized Linear Systems. SIAM Journal on Matrix Analysis and Applications doi:10.1137/100799010.

[14] Lee, C.R., 2007. Residual Arnoldi Methods : Theory, Package and Experiments. Ph.D. thesis. University of Maryland.

[15] Lee, C.R., Stewart, G., 2007. Analysis of the residual Arnoldi method. Technical Report. University of Maryland.

[16] Michiels, W., 2010. Spectrum based stability analysis and stabilization of systems described by delay differential algebraic equations. TW Reports .

[17] Michiels, W., 2011. Spectrum-based stability analysis and stabilisation of systems described by delay differential algebraic equations. IET Control Theory and Applications doi:10.1049/iet-cta.2010.0752.

[18] Michiels, W., Vyhlídal, T., Zítek, P., 2010. Control design for time-delay systems based on quasi-direct pole placement. Journal of Process Control doi:10.1016/j.jprocont.2009.11.004.

[19] Morgan, R.B., 1996. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. Mathematics of Computation 65, 1213–1230.

[20] Oseledets, I.V., 2011. Tensor-Train Decomposition. SIAM Journal on Scientific Computing doi:10.1137/090752286.

[21] Pilbauer, D., Michiels, W., Bušek, J., Osta, D., Vyhlidal, T., 2018. Control design and experimental validation for flexible multi-body systems pre-compensated by inverse shapers. Systems & Control Letters 113, 93–100.

[22] Ruymbeek, K., Meerbergen, K., Michiels, W., 2019. Calculating the minimal/maximal eigenvalue of symmetric parametrized matrices using projection. Numerical Linear Algebra with Applications 26.

[23] Saad, Y., 2011. Numerical Methods for Large Eigenvalue Problems. doi:10.1137/1.9781611970739.

[24] Seyranian, A.P., Mailybaev, A.A., 2003. Multiparameter Stability Theory with Mechanical Applications. volume 13. World Scientific.

[25] Sirkovic, P., 2016. Low-rank methods for parameter-dependent eigenvalue problems and matrix equations. Ph.D. thesis. EPFL.

[26] Sorensen, D.C., 1992. Implicit Application of Polynomial Filters in a k -Step Arnoldi Method. SIAM Journal on Matrix Analysis and Applications doi:10.1137/0613025.

[27] Trefethen, L.N., 2000. Spectral Methods in MATLAB. doi:10.1137/1.9780898719598.

[28] Woodbury, M., 1950. Inverting modified matrices. Technical Report.

[29] Wu, G., Zhang, L., 2014. On expansion of search subspaces for large non-Hermitian eigenproblems. Linear Algebra and Its Applications doi:10.1016/j.laa.2014.04.021.

[30] Ye, Q., 2008. Optimal expansion of subspaces for eigenvector approximations. Linear Algebra and Its Applications doi:10.1016/j.laa.2007.08.021.