KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree project in Mathematics

Second cycle, 30 credits

Randomized sketching for the nonlinear Arnoldi method

A randomized method for nonlinear eigenproblems

VILHELM PETERSON LITHELL

Stockholm, Sweden 2024

Author

Vilhelm Peterson Lithell
<vipl@kth.se>
KTH Royal Institute of Technology

Supervisor

Professor Elias Jarlebring
Numerics, Optimization, and Systems
Department of Mathematics, KTH Royal Institute of Technology

Abstract

We present a novel application of randomized sketching to the important area of nonlinear eigenproblems. Our construction is motivated by the recent increase of interest in randomized methods within the numerical linear algebra community, and their effectiveness in accelerating computations, among other advantages over classical solvers. Initially, we provide some background on nonlinear eigenproblems and randomized sketching, with numerous references to literature for additional material. We present the construction of our main contribution, namely the sketched nonlinear Arnoldi (sNLAR) algorithm, a randomized analog of the successful nonlinear Arnoldi solver due to Voss, and we find support for our approach in the empirical success of recently proposed randomized methods for linear eigenvalue problems, among other applications. A number of numerical experiments also serve to justify the competetiveness of our proposed solver, with standard benchmarks, as well as a problem taylored specifically to our needs. We also mention some implementation details, providing the reader with the necessary material to recreate our experiments. Finally we discuss some issues that arise from our approach, and how to remedy them, in addition to an outlook on potential use cases and a strengthened theoretical understanding of our approach, and randomized solvers in general.

Sammanfattning

Vi presenterar en ny tillämpning av randomiserade metoder inom det viktiga området av ickelinjära egenvärdesproblem. Vår konstruktion motiveras av den aktuella ökningen av intresset för randomiserade metoder inom numerisk linjär algebra, och deras förmåga att accelerera beräkningar avsevärt, bland andra fördelar över klassiska lösare. Inledningsvis ges bakgrundsmaterial över ickelinjära egenvärdesproblem såväl som randomiserade metoder, med flertalet referenser till relevant literatur. Vi presenterar vårt huvudsakliga bidrag, sNLAR, en randomiserad version av den framgångsrika ickelinjära Arnoldi-algoritmen härledd av Voss, och vi motiverar vår konstruktion med de randomiserade metoder för linjära egenvärdesproblem, bland andra tillämpningar, som nyligen visat sig vara mycket effektiva. Flertalet numeriska experiment används som stöd för att vår metod är ett bra val bland alternativa metoder, där vi redogör för ett standardiserat problem med stort inflytande i området, samt ett problem som konstruerats specifikt för detta arbete. Vi redogör även för ett antal aspekter av implementeringen utav vår metod, med detaljer som tillåter läsaren att själv återskapa våra experiment. Slutligen diskuteras ett antal utmaningar med vår metod som måste hanteras, samt en överblick över möjliga tillämpningsområden och framtida teoretiskt arbete, både med avseende på vår metod, men även med avseende på utmaningar med randomiserade metoder i allmänhet.

Acknowledgements

I would like to begin by thanking my supervisor for this thesis, Professor Elias Jarlebring. I am grateful for the fun and interesting discussions we have had during the course of this project, and for your patience with my questions and ideas. Thank you for showing interest in my work.

I would also like to thank my parents for their unwavering support during these years at KTH, and for always encouraging me to pursue the things that make me happy. Without you, this would not have been possible. Thank you to my brother for keeping me company and making me laugh. Thank you Hanna for pushing me when I could not do so myself, and for always helping me when I needed it the most.

And finally, thank you to the friends I made at KTH for making even the most dreary course work a good time. I hope we have many good times yet to come.

Stockholm
May 2024

Contents

1 Introduction

Arguably one of the most exciting recent developments in numerical linear algebra is the use of randomized methods. Randomized methods have been developed for a suite of different applications such as least squares problems [31], linear systems [34], linear eigenvalue problems [25], matrix functions [11][30], and matrix decompositions [13]. Randomized methods promise to significantly accelerate computations, while retaining high accuracy, by means of randomized embeddings extracting the most important characteristics of a problem.

In this work, we present an application of the techniques of randomized numerical linear algebra to a standard method in solutions of nonlinear eigenproblems, namely the nonlinear Arnoldi (NLAR) algorithm. More specifically, we use the tools of randomized orthogonal projection methods to construct a randomized analog of the NLAR algorithm. In the literature, the type of randomization utilized in this work is often referred to as *sketching*. Our techniques will use ideas similar to previous work in the area, utilizing the so-called *sketch-and-solve* paradigm. This technique has been successfully applied to a number of different problems in numerical linear algebra. For instance, in [25] the sketch-and-solve paradigm is used to accelerate computations in GMRES for solving linear systems of equations, and for solving linear eigenvalue problems much faster than with standard methods. In [11] the sketch-and-solve paradigm is used in the context of FOM for matrix functions.

However, the literature on sketching for nonlinear eigenproblems is sparse, and to the knowledge of the author, this work is the first where these techniques are applied to orthogonal-projection-type methods. However, there have been some applications of these methods to this type of problem, for example in the context of rational approximation [10].

The NLAR algorithm was developed by Voss in [36], and has since become a standard tool in the solution of nonlinear eigenproblems [12]. NLAR is a projection-type algorithm, where the full problem is projected onto a subspace that is iteratively expanded during the course of the algorithm. This makes NLAR a prime candidate for the sketch-and-solve paradigm.

The main advatage of sketching, that is typically cited in the literature, is that this randomization step induces a significant reduction in dimensionality of the vectors and matrices involved in the computations, often unlocking previously infeasible techniques. This reduction in dimensionality, or *embedding*, also permits us to transfer expensive operation to lower dimensional *sketches* of the quantities of interest, reducing the amount of work that has to be done on potentially very large vectors and matrices.

We try to capture these advantages that have so far mostly been recorded for the linear case of eigenproblems, by transferring these techniques to the nonlinear case. Specifically, the main contributions of this thesis are:

1. A novel method for the solution of large and sparse nonlinear eigenvalue problems based on applying the sketch-and-solve paradigm to the nonlinear Arnoldi method, an approach that can also be generalized to other popular projection-type methods.

2. Application of various techniques from within randomized numerical linear algebra to provide an implementation of a robust and competetive randomized algorithm, including validating this implementation by standard benchmark problems.

The remainder of this work is organized as follows. In Section 2 we provide some background on nonlinear eigenproblems, establishing a number of important concepts central to the theory of these problems. We also mention in passing some techniques for solving these problem numerically, while spending a significant portion of the section investigating the main building block of our method, namely the NLAR method. Finally, we provide the necessary background in randomized subspace embedding for our proposed method. In Section 3, we construct the main contribution of this work, the sketched nonlinear Arnoldi algorithm (sNLAR). In this section, we also provide some suggestions concerning the implementation, and give various specializations for specific problem structures. Here, we also integrate standard techniques from the literature into our implementation. Section 4 provides a suite of numerical experiments supporting the effectiveness and competitiveness of our proposed method, where we will begin by investigating an example that has been constructed specifically for this work, and concluding by a thourough investigation into a standard large scale benchmark problem for nonlinear eigenproblems. Finally, Section 5 concludes the text by discussing some advatantages of our method, some shortcomings and how to remedy them, as well as some future work, both theoretical and practical.

2 Background

2.1 Nonlinear eigenproblems

We are concerned with the nonlinear eigenproblem (NEP). We begin by providing some background regarding NEPs. For a more thorough review of the nonlinear eigenproblem, both with regards to theory and methods, see for instance [12][14]. For a review of background material on matrix analysis, see [15] or [9]. We have the following fundamental definition.

Definition 2.1.1. Let $M(\lambda) \in \mathbb{C}^{n\times n}$ be a matrix whose elements depend analytically on the variable $\lambda \in D \subset \mathbb{C}$, where D is a compact simply connected subset of $\mathbb{C}$. Let $x, y \in \mathbb{C}^n$. Then

$$M(\lambda)x = 0\,, \quad \text{or} \quad y^* M(\lambda) = 0\,, \tag{2.1}$$

is the nonlinear eigenvalue problem. Here, λ is an *eigenvalue* and x and y are *right-* and *left eigenvectors*, respectively. $(\cdot)^*$ denotes the Hermitian transpose. A solution (λ, x) or (λ, y) to this problem will be called an *eigenpair* of the corresponding problem.

□

Remark 2.1.1. The requirement that $\lambda \in D$ for some compact subset D of $\mathbb{C}$ is assumed since we cannot in general expect to compute *all* eigenvalues of a NEP. Consider for instance the scalar NEP $\sin(\lambda)v = 0$, which has infinitely many solutions not restricted to any particular domain. Therefore we limit our attention to finding solutions lying only in some predetermined domain.

NEPs arise in a variety of applications. Common examples include time-delay systems giving rise to exponential nonlinearities, see [24], and the application of boundary conditions in discretizations of PDEs. See [17][22] for such applications resulting in square-root nonlinearities.

The problem (2.1) is the most general definition of the NEP, in the sense that the non-linearity is completely general with respect to the eigenvalue. There are also problems where the non-linearity is manifested in the eigenvector, but these will not be considered here. We also remark that this form of eigenproblem generalizes the familiar linear eigenproblem, which can be retrieved by simply choosing $M(\lambda) = A - \lambda I$, for some matrix A. Sometimes we will be dealing with more specialized forms of NEPs. We state the following specialized form of a NEP, which will be used throughout.

Definition 2.1.2. A NEP that is of the form

$$\left(\sum_{i=1}^{\ell} A_i f_i(\lambda)\right) x = 0\,, \tag{2.2}$$

where $A_i \in \mathbb{C}^{n\times n}$, f_i is analytic in $\lambda \in D \subset \mathbb{C}$, and $x \in \mathbb{C}^n$, for $i = 1, \ldots, \ell$, will be referred to as a *SPMF-NEP*, short for *Sum-of-Products-of-Matrices-and-Functions*.

□

This form of NEP is very common in application, and will receive considerable attention in this work. Some examples of SPMFs in application include modelling of turbulent flow, leading to exponential nonlinearities [27], and problems in fluid-solid vibration [37].

It is not immediately clear how the multiplicity of the eigenvalues should be treated. To this end, we introduce the following defintion, see [12] for details.

Definition 2.1.3. The *algebraic multiplicity* of an eigenvalue λ of (2.1) is the multiplicity of the root of $\det(M(z))$ evaluated at $z = \lambda$.

□

We highlight the following interesting difference from the linear eigenvalue problem, with an example due to [12]. The algebraic multiplicity of an *isolated* eigenvalue of (2.1) must be finite [12], but it need not necessarily be bounded by the problem size. This is in contrast to the linear setting. Consider the scalar NEP, i.e. a problem size of $n = 1$, defined by $M(\lambda) = \lambda^{m+1}, m \in \mathbb{N}$. This problem has an eigenvalue $\lambda = 0$ of algebraic multiplicity $m+1$. The algebraic multiplicity of an eigenvalue might not always be as easy to determine as this example seems to imply.

Continuing, as with linear problems, it is also relevant to consider the *geometric multiplicity* of an eigenvalue. The relationship between the algebraic and geometric multiplicity often has implications for how difficult the problem is to solve numerically, and many convergence results rely on assuming a particular relationship between these two quantities. It should be fairly evident that the right eigenvectors of (2.1) must be elements of the kernel of $M(\lambda)$, i.e. $x \in \ker(M(\lambda))$. Then, the geometric multiplicity is defined as the dimension of this space, $\dim(\ker(M(\lambda)))$.

Definition 2.1.4. The geometric multiplicity of an eigenvalue λ of (2.1) is the dimension of the kernel of $M(\lambda)$, $\dim(\ker(M(\lambda)))$.

□

An eigenvalue will be called *semi-simple* if the geometric multiplicty equals the algebraic multiplicity, and *simple* if its algebraic multiplicity is unity.

Next we relate some perturbation results that will also be of interest in our numerical experiments. We will begin by considering the *sensitivity* of an eigenvalue. Our focus will be centered on establishing the so-called *eigenvalue condition number*, and we will use this tool to assess the performance of our methods in our numerical experiments. It should also be noted that while we will not consider *pseudospectral* stability-analysis, this is another standard and very powerful tool for gaining insight into the stability characteristics of a problem. For more information on this type of analysis, the reader is referred to the book by Threfethen, Lloyd, and Embree [35].

Notice that the form of NEP in Definition 2.1.2 is actually more general than presented there, in the sense that every NEP can be considered in a *split* form, and thus assume the form of a SPMF. More precisely, every NEP can be expressed as a SPMF with at most n^2 terms, if n is assumed to be the problem size. The function defining the NEP then takes the explicit form

$$M(\lambda) = f_1(\lambda)C_1 + \cdots + f_\ell(\lambda)C_\ell \, , \tag{2.3}$$

where we may have one function corresponding to every entry in the matrix. Here, C_i for $i = 1, \ldots, \ell$ are coefficient matrices of the same size as the original problem, i.e. $C_i \in \mathbb{C}^{n\times n}$ for $i = 1, \ldots, \ell$, and ℓ is at most n^2.

As a simple example, consider the problem defined by

$$M(\lambda) = \begin{pmatrix} a(\lambda) & b(\lambda) \\ c(\lambda) & d(\lambda) \end{pmatrix} \, , \tag{2.4}$$

and notice that this problem can also be written as the SPMF-NEP

$$M(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} a(\lambda) + \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} b(\lambda) + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} c(\lambda) + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} d(\lambda) \,, \tag{2.5}$$

thus assuming the desired form, and consisting of n^2 terms.

Following [12], we will consider a perturbation of the problem (2.1), expressed on the form (2.3), that is determined by

$$\Delta M(\lambda) = f_1(\lambda)\Delta C_1 + \cdots + f_\ell(\lambda)\Delta C_\ell \,. \tag{2.6}$$

Assuming that λ is an eigenvalue of the problem, and x, y are its right and left eigenvectors, respectively, we are interested in the sensitivity of λ. More precisely, we want to know how much the eigenvalue changes, expressed in the quantity $\Delta\lambda$, when the problem is perturbed by (2.6). To quantify this sensitivty, we introduce the eigenvalue condition number, also called the *normwise* condition number.

Definition 2.1.5. The normwise condition number of an eigenvalue λ of (2.1) is defined by

$$\begin{aligned}\kappa(\lambda, M) = \limsup_{\varepsilon\to 0} \Bigg(\frac{|\Delta\lambda|}{\varepsilon|\lambda|} \ : \ (M(\lambda+\Delta\lambda) + \Delta M(\lambda+\Delta\lambda))\,(v+\Delta v) = 0\,, \\ \|\Delta C_j\|_2 \le \varepsilon\alpha_j,\ j = 1,\dots,\ell \Bigg)\,,\end{aligned} \tag{2.7}$$

where $\alpha_j \in \mathbb{R}$ are scalars.

□

Notice that this definition contains some parameters that we are free to choose. Specifically, the parameters α_j, $j = 1,\dots,\ell$, allow us to choose how we measure the perturbation. For instance, we could take $\alpha_j = 1$ for every j, which would mean we are measuring the perturbation in an absolute sense. Other choices result in different ways of controlling the perturbation, see [12].

While the above form is a satisfactory definition of the normwise condition number, it is not very pleasant to work with in practice. Instead, [12] suggests a different but equivalent form.

Theorem 2.1.1. *Let the normwise condition number $\kappa(\lambda, M)$ be defined by (2.7). Then the normwise condition number permits the alternate form*

$$\kappa(\lambda, M) = \frac{\left(\sum_{j=1}^{\ell} \alpha_j |f_j(\lambda)|\right) \|x\|_2 \, \|y\|_2}{|\lambda| y^* M'(\lambda) x}\,, \tag{2.8}$$

where x, y are right and left eigenvectors respectively, corresponding to the eigenvalue λ. $\alpha_j \in \mathbb{R}$ are scalars.

□

For a proof of Theorem 2.1.1, the reader is referred to [12]. The eigenvalue condition number multiplied by the machine precision is in practice for many methods a lower bound on the smallest attainable error in the eigenvalue in question. Hence, this expression if often of practical importance to us, and can sometimes explain seemingly stagnated convergence. We will make use of this condition number later, when presenting some numerical experiments on our method. We will finally remark that for many problems, the dominating part of the expression (2.8) is not the function evaluation, nor the various eigenvalue quantities involved in this expression, but rather, the expression

$$\kappa(\lambda, M) \approx \frac{1}{y^* M'(\lambda) x} \tag{2.9}$$

is often a satisfactory estimate on the condition number, and unless explicitly mentioned, this is the expression we will use henceforth. This includes our numerical experiments.

2.2 Nonlinear projection methods

One class of methods for solving NEPs that will be of particular interest to us are the *nonlinear projection methods*. This class of methods include several classic methods for the solution of NEPs, and have found wide applicability and success in practice. In this section we will provide a review of a general framework for these methods, setting the stage for the forthcoming section in which we will provide a special case that will form the foundation for the proposed method.

This section will however not give a comprehensive review of methods for NEPs, as the literature on the subject is vast, and a full summary of these methods is beyond the scope of this text. For a more encompassing picture of the current state of this research area, the reader is referred to, for instance, the review paper [12], or [14].

When we refer to projection-type methods, we intend methods that solve a series of smaller, or *projected*, nonlinear eigenproblems whose solutions provide approximations to the solution of the original, large scale, problem. It is important to remark that these methods are primarily suited for solving *large and sparse* problems. Sparsity is a common characteristic of many problems that may be of interest to us, and can often render direct methods infeasible at scale, since they frequently do not account for this sparsity. While we will not provide a formal definition of sparse-ness, it is often fairly easy to know when to choose a sparse solver, as opposed to a *dense* solver. Many times, the choice of a sparse solver over a dense one is self-evident.

However, these types of methods do still make use of dense solvers, and they are an integral step in iteratively producing approximations to the solution of the original NEP. The purpose of this dense solver will be to solve the sequence of smaller problems produced in the course of the computation, since the construction of these generally do not preserve any sparsity structure present in the larger problem, making the use of dense solvers the better choice. To make things explicit, the nonlinear projection methods that will be of interest to us can on a high level be divided into the following steps.

1. Select a suitable projection subspace $\mathcal{V} \subset \mathbb{C}^n$.
2. Compute an approximate eigenpair $(\widehat{\lambda}, \widehat{x})$ satisfying the Galerkin condition
$$\widehat{x} \in \mathcal{V} \quad \text{and} \quad M(\widehat{\lambda})\widehat{x} \perp \mathcal{V}\,. \tag{2.10}$$
3. If the approximations are sufficently accurate, stop. Otherwise, return to the previous step with a suitably augmented search space $\mathcal{V}$.

Now, a number of questions immediately arise, but first, some comments. Initially, we should remark for clarity that we will be searching for approximations in the space $\mathcal{V}$. We will interchangably refer to this space as *the projection space*, *the search space*, and when it is obvious from context, simply *the subspace*. Continuing, we also remark that the space $\mathcal{V}$ should be easy to construct, but also have the ability to generate approximations of high quality, two goals that are often in discord. Then, first of all, how can we select a subspace $\mathcal{V}$ that has sufficent approximation power to achieve high enough accuracy in approximation? Furthermore, in what fashion should we augment the search space $\mathcal{V}$ such that it will keep generating better and better approximations? Also, how do we formulate the conditions from step 2. in practice?

To begin the construction of our general projection method, and to begin to answer these questions, suppose V is a matrix whose columns constitute an orthonormal basis for our search space $\mathcal{V}$. Then the so-called *Ritz vector* $\widehat{x}$ can be expressed as $\widehat{x} = V\widehat{y}$ for some $\widehat{y}$. With the Ritz vector expressed on this form, and with the help of the basis matrix V, the Galerkin condition in step 2. above can be reformulated as solving a *projected* problem. Specifically, the condition is equivalent to requiring that

$$V^* M(\widehat{\lambda}) V \widehat{y} = 0\,, \tag{2.11}$$

where V^* denotes the Hermitian transpose of V. The reader will notice that this projected problem is in turn a NEP, and that the size of this problem will be the same as the dimension of the subspace $\mathcal{V}$. Therefore, as long as the dimension of the search space is significantly smaller than the problem size, generating approximations to the full problem from (2.11) will involve solving a problem that is much smaller than the original NEP. This is the main idea behind projection-type methods, namely to solve a series of smaller (and hopefully easier) problems that each give us an approximation of a solution to the original problem. Of course, the quality of these approximations will heavily depend on the specific subspace employed, and some choices of this space might not provide approximations of usable quality what so ever. The choice of $\mathcal{V}$ is the pivotal ingredient of these methods. Also, as mentioned earlier, the projected problem will in general not conserve any structure that may have been present in the original problem. In particular, any sparsity structure is almost always lost.

Remark 2.2.1. Notice that $M(\widehat{\lambda})\widehat{x} = M(\widehat{\lambda})V\widehat{y}$ is the residual associated with the Ritz pair $(\widehat{\lambda}, \widehat{x})$. The condition (2.10) requires the residual to be orthogonal to the span of the columns of V, i.e. to the search space $\mathcal{V}$. In this sense, we ensure our approximation is *pseudo-optimal*, given our search space. This type of approximation strategy is very common in numerical analysis, and forms the foundations for e.g. the finite element method. See for instance [21], for an introduction.

Since the projected problem (2.11) is in general a dense NEP, and solving this problem can in itself be nontrivial. For our purposes, the specific method to do this is not of great importance, and any number of standard choices suffice for this purpose. The summary [12] provides various alternatives.

Let us deal with the remaining questions that we indicated above. Initially, how does one choose a suitable search space, $\mathcal{V}$, and how do we alter it if our approximations are not satisfactory? In practice it is unrealistic to expect our initial choice of search space to be of sufficient quality to generate good approximations, and hence the initial subspace is of less importance than the way in which we augment or expand it. Hence, these problems are really one and the same, since the way in which we choose to expand $\mathcal{V}$ will largely determine the characteristics of it.

This fact notwithstanding, the choice of our initial search space does in large determine the convergence speed of our method, and this choice can be central to the success of any projection-type method. A number of approaches have been suggested for this problem. Many of them rely on either a priori information about the spectrum of a specific NEP we are solving, or by obtaining approximate knowledge of it by means of, for example, a linearization. For instance, the authors in [22] suggest a way to start the method proposed in their work, a method which we will return to below. Essentially, the authors are solving a specific NEP related to electromagnetic accelerator modelling, and suggest using a linearized version of this problem, using the eigenpairs from this linear problem as a starting point. We will return to this specific NEP in Section 4.

On a more general note, it is of course always beneficial to use any a priori information about the distribution of eigenvalues, and any strategy that gives even approximate knowledge of the spectrum of a problem will naturally help us immensely. In other cases however, such approximate strategies might not be viable, for a variety of reasons, and no prior knowledge about the characteristics of the problem can be utilized. This is not a problem that is unique to projection methods, but to all NEP solvers, and some information about the problem at hand will continue to be a major part in effectively solving NEPs.

Hence, we conclude that the issue of choosing a good initial approximation is highly problem specific, and we will consider this problem secondary to the more relevant one, namely how to augment the search space during the course of the algorithm. This question is one that distinguishes different projection methods from one another, and a number of viable choices are available. One specific example among many is the *Jacobi-Davidson* method for NEPs. See [8] for details. Although we will not spend any time reviewing this method here, the reader is encouraged to consult this reference with the purpose of highlighting that there is no *best* choice of how to augment our search space, and many different approaches are indeed good ones. The one, very successful, choice we will consider in this work is detailed presently, but the reader should keep the framework of the nonlinear projection methods from earlier in mind, and remember that the choice of search space we will work with for the remainder of this text is just one choice among many. It is likely that the proposed method that we will consider in Section 3 would also be applicable to other projection-type methods, but this is not something we have investigated further in this text.

2.3 The nonlinear Arnoldi method

We will now proceed with the *nonlinear Arnoldi algorithm*, henceforth NLAR, developed by Voss in [36]. See also [22] for a generalization that avoids complex arithmetic.

The method falls in the category of projection methods, discussed in the previous section. Importantly, the three main steps of orthogonal projection methods presented in this previous section are also present in NLAR. In particular, NLAR also features the central step of a projection onto our search space. As shown in the previous section, this projection manifests itself in solving the so-called projected problem (2.11). In the preceeding section we remarked that the main way in which we distinguish between different projection-type methods is the way in which we expand our search space, and this is of course also true for NLAR. The reader will recall from the discussion in the previous section that this step is also the main feature determining the characteristics of the search space, denoted $\mathcal{V}$.

As established, the computation proceeds iteratively, expanding the search space during the course of the algorithm. We recall that we are trying to solve a NEP, i.e. we want to find an eigenpair (λ, x) that solves the problem $M(\lambda)x = 0$. NLAR will focus on one eigenvalue at a time, and in every iteration expand the search space in a direction that is believed to have a high approximation potential for the eigenpair that is its current focus.

We begin by considering this central issue of how to determine a new search direction. Following [36], we will consider two different approaches to this problem. Both of these approaches will build upon previously well established methods for solving NEPs, but will of course be modified to suit our purposes. As we will see, some choices of subspace expansion are more suitable than others.

Initially, we consider one variant of *inverse iteration.* Several types of inverse iteration provide effective methods for the NEP, see for instance [32]. Suppose for the moment that we have an approximate eigenpair $(\widehat{\lambda}, \widehat{x})$, to the wanted eigenpair (λ, x). Then, the form of inverse iteration we will be considering suggests that we choose a new search direction according to

$$v = M(\widehat{\lambda})^{-1} M'(\widehat{\lambda}) \widehat{x} \,. \tag{2.12}$$

Now, inverse iteration is known to converge exceedingly quickly. For instance, it converges cubically for Hermitian problems [36]. However, it suffers from having to solve a potentially very large linear system every iteration. In addition, the matrices in this system are not constant throughout the course of the algorithm, but change as the approximations of the eigenpair are updated. We could instead solve the system with a fixed shift σ, i.e. compute $M(\sigma)^{-1}M'(\widehat{\lambda})\widehat{x}$, where the shift could be changed during the computations, but this will lead to incorrect convergence [36].

Algorithm 1 The nonlinear Arnoldi method for $M(\lambda)x = 0$

```
 1: **input:** pole $\sigma$, initial basis vector $v$, $\|v\| = 1$
 2: **output:** eigenpair approximations
 3: $k = 1$, $m = 0$, $V = v$
 4: **while** $m <$ number of wanted eigenvalues and $k <$ max iterations **do**
 5:     solve projected problem $V^* M(\mu) V y = 0$
 6:     compute Ritz vector $u = Vy$ and residual $r_k = M(\mu)u$
 7:     **if** $r_k <$ tol **then**
 8:         save eigenpair $(\mu, u)$
 9:         $m = m + 1$
10:         choose approximations $\mu$ and $u$ to next eigenpair
11:         compute residual $r = M(\mu)u$
12:     **end if**
13:     $v = M(\sigma)^{-1} M(\mu) u$
14:     orthogonalize $v$ against span$\{V\}$
15:     $v = v / \|v\|$
16:     expand basis $V = [V, v]$
17:     $k = k + 1$
18: **end while**
19: **return** computed eigenpairs
```

To remedy the shortcomings of inverse iteration, we shift our attention to the so-called *residual inverse iteration* of Neumaier [26]. A full description of the algorithm can be found in [36][26], but here we will limit ourselves to considering the search space expansion. The residual inverse iteration suggests we expand the search space by

$$v = M(\sigma)^{-1} M(\widehat{\lambda}) \widehat{x} \,, \tag{2.13}$$

for a fixed shift σ. We emphasize the difference from (2.12). Notice that we have replaced the inverse evaluated at $\widehat{\lambda}$ with the inverse evaluated at the fixed value σ, as well as having replaced $M'(\widehat{\lambda})$ by $M(\widehat{\lambda})$. It should be noted that σ can be changed during the course of the algorithm as we move from one eigenvalue to another. We also note that (2.13) is not strictly the update used in the residual inverse iteration, but is equivalent for our projected problem. See [36] for a short discussion on why this search space update makes the algorithm equivalent to the Arnoldi update in the linear case.

We also remark that as suggested in [36], we can replace $M(\sigma)^{-1} \approx P$, and instead compute the new search direction $v = PM(\widehat{\lambda})\widehat{x}$, if the solution of the linear system is deemed too expensive. This preconditioner could be any number of standard choices. One issue that is naturally of interest is the convergence properties of the residual inverse iteration. For completeness, we provide a such a characterization of the convergence, due to [36] and [26].

Theorem 2.3.1. *Let M in Definition 2.1.1 be sufficiently smooth, with a simple eigenvalue λ, and let x be the corresponding eigenvector, normalized by a fixed vector $e \in \mathbb{C}^n$, i.e. $e^*x = 1$. Then the residual inverse iteration converges for all choices of σ sufficiently close to λ, and it holds that*

$$\begin{aligned} \frac{\|\widehat{x}_{\ell+1} - x\|}{\|\widehat{x}_\ell - x\|} &= \mathcal{O}\left(|\sigma - \lambda|\right) \;, \quad \textit{and} \\ \left|\widehat{\lambda}_{\ell+1} - \lambda\right| &= \mathcal{O}\left(\|\widehat{x}_\ell - x\|\right) \;, \end{aligned} \tag{2.14}$$

where $\left(\widehat{\lambda}_\ell, \widehat{x}_\ell\right)$ is the approximate eigenpair in iteration ℓ.

□

It is noted in [36] that the rate of convergence of the inverse residual iteration is expected, with reference to Theorem 2.3.1, to depend on the the distance between the current approximation of the wanted eigenvalue and the shift σ. Hence, it is advisable to change the shift during the computations, to ensure we always have an acceptable rate of convergence.

We have now established an approach to choose a new search direction, and thereby have a strategy for expanding our search space. We are then ready to state the full algorithm. The nonlinear Arnoldi algorithm can be found in Algorithm 1.

The reader will recognize that the main elements of the framework given in Section 2.2 are indeed present in Algorithm 1. Now, in [36], Voss provides some additional discussion on various important elements that are of practical importance when considering the use of NLAR. For one, a significant portion of [36] is dedicated to a discussion of how one can efficiently solve the projected problem (2.11). Voss suggests the use of the so-called *safeguarded iteration*. This method builds upon the theory of *Rayleigh functionals*, an extention of the familiar Rayleigh quotient. While this method is certainly a viable option for solving the projected problem, we will not consider it further in this text, since solving the projected problem is largely a practical problem, and will not affect our main contribution significantly. As mentioned earlier, any number of standard choices of dense NEP solvers will usually serve our purposes in this regard. See [14] or [12] for exposition on Rayleigh quotients and Rayleigh functionals, and the original paper by Voss for a discussion on the safeguarded iteration [36].

Continuing our review of NLAR, Voss also gives several suggestions of various optimizations one could reasonably utilize for problems with certain structure. While we will use some of these suggestions in the implementation of our numerical experiments, and in some cases extend them to work better with the current method, we will leave most of them untouched. The suggestions that we will use, will however be reserved for the implementation details of our main contribution, and the reader is asked to reference Section 3.2 for our discussion on these. In addition to this, [36] will of course be an excellent source for implementation details not mentioned in this work.

2.4 Random subspace embeddings

We are now ready to proceed with examining the essential tool for this work, namely *randomized subspace embeddings*. Much of this exposition can be found in more detail, and with greater scope, in a number of excellent survey papers on the subject of randomized numerical linear algebra. For instance, [23] provides a thorough introduction to the subject, with treatment of algorithms ranging from randomized trace estimation to machine learning applications. [20]

provides additional theory on subspace embeddings, in addition to a wide variety of topics in the area. Notably, they also include discussion on randomized methods for tensor computations. See the references in [23] for a wider selection of surveys on the subject.

Randomized subspace embedding involves the application of a randomly drawn matrix S. In the literature, this application is often referred to as *sketching*, and this is the terminology we will adopt for this work.

Definition 2.4.1. (Subspace embedding). Let $V \subset \mathbb{C}^n$ be a subset of complex Euclidean space, and let $S \in \mathbb{C}^{s \times n}$ be a linear map. Let $\varepsilon \in [0, 1]$. Then S is a *subspace embedding* for V with *distortion* ε if for every $x, y \in V$ we have the following estimate on inner products over V

$$\langle x, y \rangle - \varepsilon \|x\| \|y\| \leq \langle Sx, Sy \rangle \leq \langle x, y \rangle + \varepsilon \|x\| \|y\| \ . \tag{2.15}$$

□

Remark 2.4.1. In application, it is assumed that $s \ll n$, so that the map S induces a reduction in dimension, while approximately retaining the "geometry" of the subspace being embedded. For this to be true, it will of course also be desirable to have $\varepsilon \ll 1$. Intuitively, a larger embedding dimension would permit us to choose the embedding distortion smaller, and a smaller embedding size would allow us to do less computational work. It turns out however, that these are competing choices.

The reason for introducing this subspace embedding is that we hope to work with a reduced, or *sketched*, version of our (presumably large) problem. If we can additionally construct a subspace embedding that preserves important qualities of our problem, we hope to achieve similar results as for the full problem, while doing work only on quantities of smaller dimensionality. At the moment it is not entirely clear how such an embedding would be constructed, or if such an embedding even exists in general.

While the proof is beyond the scope of this text, such an embedding does indeed exist. In their now classic paper [19], Johnson and Lindenstrauss established that points in higher dimensional space can be embedded into lower dimensional space, with a relatively small distortion. They originally showed this result in the context of the geometry of Banach spaces, and the reader is encouraged to consult this reference for further details on their work.

With reference to Definition 2.4.1, we expect to require some knowledge about the specific structure of the subspace V we are trying to sketch. In application however, this is not always known a priori, and this is certainly not the case for our purposes, where V will eventually take the form of a search space that is iteratively expanded. As in [23], we additionally note some other desirable qualities of S. Firstly, we would like S to be easy to construct and not too expensive to store. Furthermore, from a computational perspective, we would like the map S to be easy and fast to apply to our subspace. For instance, if S happens to take the form of a full matrix, the cost of repeatedly sketching very large vectors may become prohibitive, and hinder the performance gains of the method. It may additionally be deemed to expensive to store explicitly if memory is scarce.

The solution to these problems is to draw the sketching matrix from a probability distribution. Many such distributions have been proposed, among them Gaussian matrices [23], Rademacher matrices [1], and sub-sampled randomized Fourier and trigonometric transforms [13]. Such matrices drawn at random are often called *oblivious* subspace embeddings, since they are agnostic as to the specific space being handled.

Definition 2.4.2. The matrix S, taking values in $\mathbb{C}^{s \times n}$, is an (ε, δ) oblivious subspace embedding for any fixed subspace $V \subset \mathbb{C}^n$ if it is a subspace embedding with distortion ε for V with probability at least $1 - \delta$, where $\delta \in [0, 1]$.

□

It is important to emphasize that the matrix S is *drawn at random* from a underlying distribution, some examples of which where mentioned above. We also emphasize that an oblivious subspace embedding can be constructed without knowledge of the space V that we wish to embedd, with the exception of its dimension. In this way, we require no prior knowledge of the space in question, but the challenge of constructing an embedding complying with Definition 2.4.2 still remains. To this end, we continue by way of presenting some explicit constructions of oblivious subspace embeddings, that will be employed in our method.

Remark 2.4.2. The theoretical background underpinning the validity of the oblivious subspace embeddings employed in this work is, while present, somewhat hazy in comparison with the theory for other (less structured) approaches to subspace emebedding. However, the techniques used in this work have the same, or better, practical performance as that of many of the approaches that have more rigorous theoretical guarantees. For this reason, we will be sparse in our theoretical handling of these techniques, and refer the reader to empirical investigation, and the theory applicable for other types of less structured embeddings. For a discussion on theoretical guarantees for e.g. Gaussian embeddings, see [13].

To begin our construction of a practical randomized subspace embedding, we provide some intuition about their rationale. Much of this reasoning is due to [13] and [2]. One of the most important features of any performant random subspace embedding is an initial *mixing* step of the coordinates of, say, a vector that we wish to embedd. To be explicit, Definition 2.4.2 is also applicable to single vectors of a space, and it is in this capacity that the embedding will be most frequently referenced. The random embedding should in this sense homogenize the coordinates in such a way that each coordinate carries about the same amount of *energy* after the mixing step. In this way, we can then sample the mixed vector randomly and hope that the initial mixing has distributed the characteristics of the vector such that our random sampling will be of low variance, and provide a good stand in for the original information in the vector. It is paramount to note that this embedded vector should of course be of lower dimension than the original vector we are embedding. Various strategies for random subspace embedding differ in the way they do the initial mixing, and an overview of different methods is provided in [13].

In the present work, we will employ a specific embeddings strategy that is part of a larger family of such embeddings, namely *subsampled randomized trigonometric transforms* (SRTTs). The reason for employing trigonometric transforms may seem unclear, but is made concrete in [2]. We will not attempt to give a thourough analysis of the reasoning behind this, and the reader is referred to this resource for exposition on the importance of this choice for the performance of the method, as well as an interesting connection with the Heisenberg uncertainty principle. Now, to concretize our choice of random embedding, let $D \in \mathbb{C}^{s\times n}$ be the diagonal projector obtained by uniformly selecting s rows of the identity matrix I at random, or equivalently D is the matrix that unifornly selects s rows of its input (in the sense of right multiplication) at random. In the terminology of [13], D is a *random restriction.* Continuing, we let $E \in \mathbb{C}^{n\times n}$ be a diagonal matrix whose entries are indepenantly Rademacher-distributed. A random variable is Rademacher if it is uniformly distributed over the set $\{-1, 1\}$. Finally, we take $F \in \mathbb{C}^{n\times n}$ to be a discrete trigonometric transform matrix. Specifically, in this work we have employed a discrete cosine transform. Then, our random subspace embedding matrix $S \in \mathbb{C}^{s\times n}$ is given by

$$S = \sqrt{\frac{n}{s}} DFE \,.$$

For completeness, we provide a definition of our SRTT.

Definition 2.4.3. A subsampled randomized trigonometric transform is given by

$$S = \sqrt{\frac{n}{s}} DFE \,, \tag{2.16}$$

where $D \in \mathbb{C}^{s \times n}$ is a diagonal projector randomly selecting s rows of its input, $E \in \mathbb{C}^{n \times n}$ is a diagonal Rademacher matrix, and $F \in \mathbb{C}^{n \times n}$ is a discrete trigonometric transform matrix.

□

Remark 2.4.3. By far the most computationally expensive part in the construction of our random embedding, S, is the discrete trigonometric transform. For a naive approach, a trigonometric transform has quadratic complexity, which quickly becomes detrimental to the performance for large problems. Our method will therefore naturally make use of *fast* transforms (e.g. the fast Fourier transform, fast cosine transform, etc.), which typically have $\mathcal{O}\left(n \log(n)\right)$ complexity.

Now, it still remains to determine how large we need to choose our embedding dimension, s, to ensure we get a satisfactory distortion of our space after the embedding. Obviously, this will depend on the dimension of the original space we are embedding. In the literature, the embedding dimension s is known as the *sketching parameter*, and this is the terminology we will adopt. With reference to [19], as well as [25][13], the optimal scaling for the sketching parameter approximately follows the law

$$s \approx \frac{n}{\varepsilon^2} \,, \tag{2.17}$$

if we wish to embed a n-dimensional space, with distortion ε. In practice, at least in our application, the dimension of the space n will correspond to the maximum search depth in our method, since the space V in Definition 2.4.1 corresponds to our search space. Hence, it is important that our method gives acceptable performance for moderate distortions, say $\varepsilon = 1/\sqrt{2}$ (in the sense of expectation), since the scaling of the sketching parameter is rather poor in relation to the desired distortion. With this in mind, it is customary to choose the sketching parameter to be on the same order as the maximum search depth, in the context of iterative methods. Typically we choose two, four, or six times the maximum search depth as our sketching parameter. This usually gives good performance, while retaining the advantage gained by the reduction in dimension produced by the embedding.

3 The sketched nonlinear Arnoldi method

In this section we present the derivation of our main result, the *sketched nonlinear Arnoldi algorithm.* We will begin by motivating our method by comparison with the classical nonlinear Arnoldi algorithm, presented in the previous chapter. Having established our main approach, we also point out some implementation details that have been considered in this work, and the possibility of some that have not been implemented here.

3.1 Derivation of the sketched nonlinear Arnoldi algorithm

We begin by again considering NLAR in the framework presented in Section 2.2 and 2.3. Recall that one of the main steps in NLAR is solving the projected problem

$$V^* M(\widehat{\lambda}) V \widehat{y} = 0 \,. \tag{3.1}$$

Again, we notice that the quantity $M(\widehat{\lambda})V\widehat{y}$ is the residual associated with the Ritz pair $(\widehat{\lambda}, V\widehat{y})$. As usual, V denotes the matrix whose columns constitute a basis for the search space, $\mathcal{V}$. As discussed previously, (3.1) can then be interpreted as requiring the residual to be orthogonal to the span of the search space.

Having established the concept of oblivious subspace embeddings, we are now ready to combnine this with the construction of NLAR in order to motivate our proposed approach. Recall that the purpose of an oblivious subspace embedding is to compress a subspace into one that is of smaller size, while still approximately retaining the geometry of this subspace. The *geometry* is here to be understood as approximately conserving inner products between vectors, or collections of vectors. The reader is encouraged to again refer to Definition 2.4.2. In particular, othogonality conditions will also be approximately conserved. This encourages us to use random sketching as a tool to approximately capture Galerkin conditions, while significantly reducing the dimension of the vectors that any algorithm requiring such a condition must operate on.

Explicitly, we propose to replace the Galerkin-type condition (3.1) by the sketched Galerkin condition

$$(SV)^*(SM(\widehat{\lambda})V\widehat{y}) = 0 \,, \tag{3.2}$$

that is, requiring that the sketched residual be orthogonal to the sketched span of the search space. This is the projected problem we will solve in each iteration of our method, where S is an $s \times n$ oblivious subspace embedding as defined in Definition 2.4.2. Naturally, we assume $s \ll n$, so that some significant dimensionality reduction is actually taking place. Some comments on this approach are in order. Initially, this way of replacing the strict Galerkin condition in various solvers is commonplace in the setting of the sketch-and-solve paradigm [11][34][25], and with these works as empirical support for requiring (3.2), this method is expected to work well. It should however be noted that the listed works all work in the linear setting, but the motivation for employing this strategy is the same in the linear case as in the nonlinear one. On a more heuristic note, with reference to Definition 2.4.1, and our previous discussion, the sketched condition is expected to approximately capture the relationship between the span of the search space and the residual, in the sense of approximately preserving inner products, and hence we expect to produce approximations to eigenpairs of similar quality to those generated by (3.1). More explicitly, as long as the true residual is orthogonal to the search space, the sketched condition should approximately capture this relationship. Finally, as in [11], we remark that if $S = I$, then the sketched problem (3.2) reduces to the classical condition (3.1).

On a high level, this is the core element of the proposed method. However, the benefits of this approach may not be immeadiately clear, and we will devote the forthcoming section to addressing the gains that can be made with the randomized algorithm that can not be had with the classical version. While we will withhold most of these details for now, it is important to remark on one crucial point, namely on the matter of orthogonalization. The orthogonalization of the search space basis, V, is imperative to the effectiveness of NLAR, cf. Algorithm 1.

While NLAR does not make any explicit assumption on the orthogonality of the basis, it is a very important element of the method in practice. Loss of orthogonality is a central problem in numerical linear algebra, and various accelerated methods of orthogonalization is an active area of research. See for instance [4] for an application of random sketching in this context. The absence of an orthogonalization step can often make a method completely break down.

What is important to notice is that while these principles also apply to sNLAR, much of the work of orthogonalization can be transferred to take place only on sketched vectors, i.e. ones of very small dimension. As a result of this, when handling the orthogonalization of the non-sketched basis, V, it is customary within the sketch-and-solve paradigm to make use of so-called *truncated orthogonalization*, i.e. only orthogonalizing a small number of vectors against each other, typically the latest few vectors produced by the method. This strategy has also recently been combined with a tool often referred to as *basis whitening*, where in the condition number of the sketched basis, which is typically large as a result of the truncated orthogonalization, is reduced by means of a QR-decomposition. Both of these strategies will be clarified shortly, but it is important for understanding the advantages of the sNLAR to keep these aspects in mind.

3.2 Implementation details

We will now deal with some implementation details that are crucial for obtaining a competetive algorithm.

The previous section made reference to an alternative to full orthogonalization of the search space basis, V, namely by using a truncated orthogonalization. This approach has been used to good success in a number of different works on random sketching in various contexts, see [25][34][11] for some examples. [34] also discusses some alternative basis construction techniques. In these works, the orthogonalization is to be understood in the context of the Arnoldi iteration [3]. The Arnoldi iteration with truncated orthogonalization is therefore often called the truncated Arnoldi's method for constructing a basis for, in these cases, a Krylov space.

In this orthogonalization strategy, we only orthogonalize the latest vector produced, by whatever subspace expansion strategy is used, against a small number of the previous vectors, typically $4-6$ of the latest basis vectors, although some authors, and the author of this text, have found that this number can often be made as small as 2 without significantly impacting the performance of the method.

One might rightfully conjecture that the condition number of the search space basis would increase very rapidly under this scheme, soon rendering the basis unusable. This issue is common to all truncated orthogonalization strategies, and various approaches to combat it have been proposed. One approach suggested in [25] employs the singular value decomposition of the basis before solving an associated generalized eigenvalue problem as a means of stabilizaing the basis, in the sense of its condition number. However, by far the most common and effective technique that has been proposed was first reported in [31] in the context of solving overdetermined least squares problems via random sketching. In this article, the authors present a preconditioning technique that has since been used in the works suggested above to control the rapid growth of the condition number of the search space basis.

Many authors have since adopted the term *basis whitening* as referring to this preconditioning technique. Explicitly, let $SV = QR$ be a thin QR-decomposition of the sketched basis. Then the basis whitening consists of performing the replacements

$$SV \longleftarrow Q \qquad V \longleftarrow VR^{-1} \,. \tag{3.3}$$

Notice that the first replacement is in essence an orthogonalization of the sketched basis. As discussed above, we employ a truncated orthogonalization approach with respect to the non-sketched basis, i.e. V, but with the basis whitening, the orthogonalization is in a sense transferred to the sketched basis instead. Since the sketched basis is typically of significantly lower dimension than the original problem, for medium to large problems often around $1-2\%$ the problem order, this approach frequently drastically cuts down on the time spent in orthogonalization. The second replacement is the preconditioning step mentioned above. As the non-sketched basis, V, will only be partially orthogonalized each iteration the condition number will rapidly increase. The right-side application of R^{-1} will control this condition number, and the condition number after this application will satisfy the inequality

$$\kappa_2(VR^{-1}) \leq \frac{1+\varepsilon}{1-\varepsilon} \,, \tag{3.4}$$

where ε is the distortion induced under the subspace embedding S, cf. Definition 2.4.2, [25][31]. Now, different authors apply this basis whitening step with different frequency. Obviously, the basis whitening is not free, and we would like to apply it as seldom as possible. For this reason, some authors advocate for the usage of condition number diagnostics that can be obtained cheaply by means of the subspace embedding, see for instance the suggestions made in [25]. The naive approach however, and perhaps the most robust one, is performing the basis whitening every iteration, thus controlling the basis condition number very effectively. This is the approach we have adopted in this work, and implementing any savings in computation with regards to this step will be left as future work.

Algorithm 2 The sketched nonlinear Arnoldi method for $M(\lambda)x = 0$

```
1: input: sketching matrix $S$, truncation length $t$, pole $\sigma$, initial basis vector $v$, $\|v\| = 1$
2: output: eigenpair approximations
3: $k = 1$, $m = 0$, $V = v$, $SV = Sv$
4: while $m <$ number of wanted eigenvalues and $k <$ max iterations do
5:     compute thin $QR$-decomposition, $SV = QR$
6:     whiten basis $SV \leftarrow Q$, $V \leftarrow VR^{-1}$
7:     solve projected problem $(SV)^*(SM(\mu)Vy) = 0$
8:     compute Ritz vector $u = Vy$ and residual $r_k = M(\mu)u$
9:     if $r_k <$ tol then
10:        save eigenpair $(\mu, u)$
11:        $m = m + 1$
12:        choose approximations $\mu$ and $u$ to next eigenpair
13:        compute residual $r = M(\mu)u$
14:    end if
15:    $v = M(\sigma)^{-1}M(\mu)u$
16:    orthogonalize $v$ against span$\{V_{:,k-t+1:k}\}$
17:    $v = v/\|v\|$
18:    expand basis $V = [V,\ v]$
19:    expand sketched basis $SV = [SV,\ Sv]$
20:    $k = k + 1$
21: end while
22: return computed eigenpairs
```

We are now ready to state the first practical version of the algorithm. It can be found in Algorithm 2.

Having established a way to deal with the consequences of the truncated orthogonalization, and thus constructing a practically viable method, we will mention one aspect of the implementation that was also suggested in [36], and adapted here for the use in our proposed method. Recall that many NEPs are on the form of a SPMF-NEP, cf. Definition 2.1.2. For this particular structure, the construction of the projected problem can be formulated in a way that makes very efficent use of the projected problem constructed in the previous iteration, saving some computation. In particular, we notice that when projected, every term in (2.2) will simply be augmented by a left multiplication by V^*, and a right multiplication by V. Since multiplication by a scalar function is commutative, this operation will only affect the matrix factor of every term, and hence it is only this matrix that needs to be altered from iteration to iteration. Explicitly, the projection of a SPMF will take the following form, where we have denoted by V_k the full basis in iteration k. We have

$$V_k^* M(\lambda) V_k = \sum_{i=1}^{\ell} V_k^* A_i V_k f_i(\lambda) := \sum_{i=1}^{\ell} A_{i,k} f_i(\lambda)\ . \tag{3.5}$$

Now, as noted above, we only need to consider how to alter the matrices from iteration to iteration, and the functions can safely be ignored. Suppose therefore that we have access to the problem matrices from the previous iteration $k-1$, i.e. $A_{i,k-1}$ for $i = 1, \dots, \ell$, and we would like to construct the matrices corresonding to the current iteration, k.

Following [36], we notice that the matrices can be expanded in the following way

$$A_{i,k} = \begin{pmatrix} A_{i,k-1} & V_{k-1}^* A_i v_k \\ v_k^* A_i V_{k-1} & v_k^* A_i v_k \end{pmatrix} . \tag{3.6}$$

Here, v_k is the latest basis vector produced, i.e. $V_k = [V_{k-1}, v_k]$. One element of this approach that is important to remark on is that the matvecs involved in this expansion cannot be recovered from the previous projected problem, but must to a large part be recomputed each iteration, unless we want to explicitly store the reduced matrices $A_i V$ for each term of our NEP. If the problem is particularly large, or the memory limitations particularly stringent, this might not be feasible. However, as we will see, the equivalent expansion strategy for our proposed method do not suffer from this limitation to the same extent, thanks to the subspace embedding strategy.

Then, let us consider this strategy for the sketched Galerkin condition (2.11). In this case, and assuming we are still dealing with a SPMF-NEP, our problem now assumes the form

$$(SV_k)^*(SM(\lambda)V_k) = \sum_{i=1}^{\ell} (SV_k)^*(SA_iV_k) f_i(\lambda) , \tag{3.7}$$

and in similarity with the form (3.6), we can epand the projected problem in the next iteration by expanding the problem matrices by one row and one column. Explicitly, we have the termwise expansion formula

$$(SV_k)^*(SA_iV_k) = \begin{pmatrix} (SV_{k-1})^*(SA_iV_{k-1}) & (SV_{k-1})^*(SA_iv_k) \\ (Sv_k)^*(SA_iV_{k-1}) & (Sv_k)^*(SA_iv_k) \end{pmatrix} , \tag{3.8}$$

with $i = 1, \ldots, \ell$. Now, notice that as with the non-sketched projected problem, the upper left block matrix is the projected problem from the previous iteration. Furthermore, the expansion strategy (3.8) shares another quality with the classical approach, namely that the matvec involved in computing this expansions can not be retrieved from the previous projected problem. However, for the sketched method, this is slightly more problematic than for the classical method. To see this, notice for instance that the expression SA_iV_{k-1} needs to be evaluated each iteration, i.e. we need to sketch the full search space basis times the problem matrix, for every term in the NEP. This is not free and presents a considerable overhead in the computation, cf. Section 2.4. The need to repeatedly sketch a potentially very large matrix can quickly overwhelm the performance of the method, and render it inferior, in terms of comlexity, to the classical method.

Now, we remarked that for the classical method, this shortcoming of the approach cannot really be remedied without explicitly storing the matrices A_iV_{k-1}, for $i = 1, \ldots, \ell$, something that might not always be feasible. However, this is not as much of a concern for our proposed method, since we only need to store the matrices SA_iV_{k-1}, which will be of significantly smaller dimension than A_iV_{k-1}, and in general only be of moderate size. Hence, it is feasible to explicitly store these sketched matrices, and they can also be iteratively expanded during the course of the algorithm. This means that the approach (3.8) only needs to sketch one additional vector per iteration, as opposed to repeating previous computations. This alternative method does induce some additional memory costs as a trade-off, but as mentioned this will not be limiting in most applications due to the small size of the matrices involved.

Another aspect that might then be of concern is the way in which we iteratively expand the matrices SA_iV, $i = 1, \ldots, \ell$, during the course of the execution of the algorithm. More precisely, these matrices will suffer from the same rapid growth of condition number as the matrics V and SV, unless accounted for. One approach is to apply the preconditioner from the basis whitening (3.3) to SA_iV each iteration and for each i, but we have found it sufficient in practice to simply compute SA_iv_k *after* the basis whitening step, where v_k is the latest basis vector computed. We have observed good stability characteristics and performance of this approach, and this is the strategy we will adopt for the remainder of this text. However, if the resources are available, application of the preconditioner R^{-1} might be more robust, cf. the use in [11] or [25]. Further investigation of this will be left for future work.

Algorithm 3 The sketched nonlinear Arnoldi method for $\left(\sum_{i=1}^{\ell} A_i f_i(\lambda)\right) x = 0$

```
 1: input: sketching matrix $S$, truncation length $t$, pole $\sigma$, initial basis vector $v$, $\|v\| = 1$
 2: output: eigenpair approximations
 3: $k = 1$, $m = 0$, $V = v$, $SV = Sv$, $SA_iV = SA_iv$ for $i = 1, \dots, \ell$
 4: while $m <$ number of wanted eigenvalues and $k <$ max iterations do
 5:     compute thin $QR$-decomposition, $SV = QR$
 6:     whiten basis $SV \leftarrow Q$, $V \leftarrow VR^{-1}$
 7:     $SA_iV = [SA_iV, SA_iv]$ for $i = 1, \dots, \ell$
 8:     expand projected problem according to (3.8)
 9:     solve projected problem $(SV)^*(SM(\mu)Vy) = 0$
10:     compute Ritz vector $u = Vy$ and residual $r_k = M(\mu)u$
11:     if $r_k <$ tol then
12:         save eigenpair $(\mu, u)$
13:         $m = m + 1$
14:         choose approximations $\mu$ and $u$ to next eigenpair
15:         compute residual $r = M(\mu)u$
16:     end if
17:     $v = M(\sigma)^{-1}M(\mu)u$
18:     orthogonalize $v$ against span$\{V_{:,k-t+1:k}\}$
19:     $v = v/\|v\|$
20:     expand basis $V = [V,\ v]$
21:     expand sketched basis $SV = [SV,\ Sv]$
22:     $k = k + 1$
23: end while
24: return computed eigenpairs
```

For completeness, we explicitly provide the algorithm specialized for SPMF-NEPs. See Algorithm 3.

We will also note an additional approach that could present an alternate way of reducing the cost of sketching when expanding SPMF-NEPs. Since matrix multiplication is associative, one could conceivably sketch the problem matrices before beginning the computations, thus circumventing the issue of having to repeatedly sketch the entire basis, instead reducing the problem to a regular matvec. This is not strategy that we have explored, and is left as a suggestion for future work.

4 Numerical experiments

This section considers some numerical experiments to characterize the performance of sNLAR. Initially we consider a NEP that is a perturbation of a linear problem with known spectrum, with the aim of validating the performance of our method compared to the classical approach. We also provide a larger problem showcasing the competetiveness of our approach.

4.1 A nonlinearly perturbed linear problem

To begin, we consider a NEP that is a perturbation of a linear problem. This type of problem is common in application. For instance, in [36], this type of NEP arises in the modeling of a nonproportionally damped vibrating structure, modeled with the finite element method. Another example with a connection to the modelling of vibrating structures, also resulting in a nonlinearly perturbed problem, can be found in [33]. More specifically, we will consider a low rank perturbation, i.e. a problem on the form

$$\left(A - \lambda I + \frac{uw^T}{\sigma - \lambda}\right) v = 0\,, \tag{4.1}$$

where $A \in \mathbb{C}^{n\times n}$ is a matrix with known spectrum, I is the identity, and $u, w \in \mathbb{C}^n$ and $\sigma \in \mathbb{C}$ are chosen arbitrarily, but in a way that does not make the problem unreasonable. Furthermore, the spectrum of the linear part of the problem is chosen such that we expect to be able to control the rate of convergence of the method. In particular, we would like to validate that the sNLAR-approximant is close to the approximations computed with NLAR. We expect sNLAR to work well when NLAR does for the same reasons elucidated in [13], and the purpose of this problem is to gain insight into this assumption. Since NLAR is equivalent to Arnoldi's method for linear eigenproblems [36], we can expect that if the nonlinear perturbation in (4.1) is sufficiently small (in the sense of some matrix norm) NLAR would have similar convergence characteristics to the linear Arnoldi method. It is known that the linear Arnoldi method readily converges to extreme eigenvalues, and that the rate of convergence is dependent on the distance of the extreme eigenvalues to the rest of the spectrum. Then, to control the rate of convergence in this example, we will let the spectrum of the linear part of the problem consist if a disc of eigenvalues, with one extremal eigenvalue. The problem is constructed such that the matrix for the linear part is large and sparse, with a dominant ridge structure. The construction of this matrix employed techniques similar to the ones presented in [6].

We solve this problem with the sketching parameter set to two times the maximum number of iterations, and the orthogonalization truncation length set to 2. The maximum search depth is set to 50. The results from this problem can be seen in Figure 4.1.

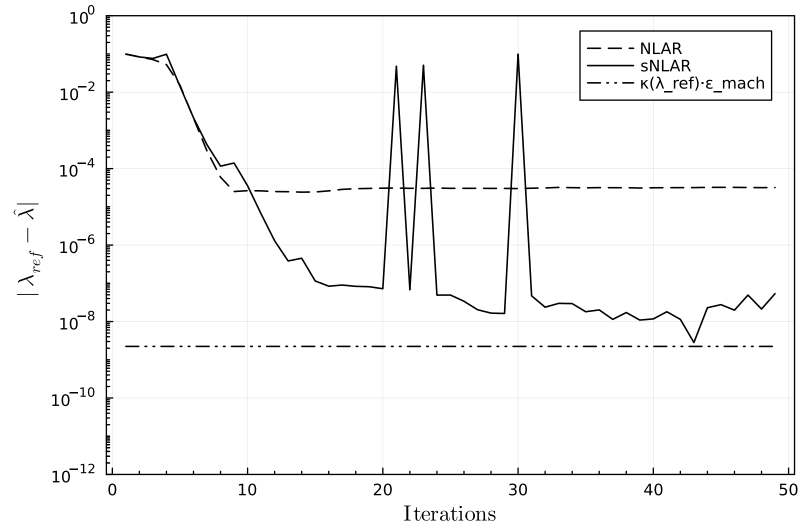

Figure 4.1: Convergence of sNLAR and NLAR on the perturbed problem (4.1)*. For sNLAR, the sketching parameter is set to two times the maximum search depth, and the truncation length is set to* 2*. The problem size is* $n = 1500$*. The figure also shows the eigenvalue condition number scaled by the machine precision for the outlier eigenvalue, where we have used the expression* (2.9)*. We see that sNLAR converges down to the condition number of the eigenvalue, while NLAR does not manage to converge fully.*

From Figure 4.1, we see that sNLAR successfully finds the outlier eigenvalue, while NLAR fails to do this to a satisfactory accuracy. Notice that the error is bounded from below by the normwise condition number of the eigenvalue multiplied by the machine precision. This is to be expected, and in practice the condition number of an eigenvalue often acts as a lower bound on the minimum obtainable error for many methods, as discussed in Section 2. We observe distinct spikes in the convergence of the randomized method. These spikes seem to be inherent to many sketched methods, especially when the sketching parameter is set relatively low, see [34] for a discussion on this phenomenon, as well as our discussion in Section 5.

This example shows that sNLAR converges as expected, closely mirroring the convergence of the classical method, and in this case appears to outperform it.

4.2 An example from electromagnetic accelerator modelling

We will now consider a more ambitious problem, that has become a standard benchmark in the literature. It was first introduced in [22], and has since become known as the *gun*-problem. The problem stems from a finite-element discretization of Maxwell's equations in the context of accelerator modelling. For the derivation and physical details, see for instance [22][16]. For our purposes however, it will suffice to consider the large and sparse NEP that results from this discretization. Let our problem be defined by the function

$$M(\lambda) = A_0 - \lambda A_1 + i\sqrt{\lambda - \sigma_1^2}A_2 + i\sqrt{\lambda - \sigma_2^2}A_3\,. \tag{4.2}$$

The reader will note that the problem contains square roots, and these are defined as the principal branch of the complex square root. Further, we will choose $\sigma_1 = 0$ and $\sigma_2 = 108.8774$, to comply with the standard form of this benchmark. Here, A_0, A_1, A_2, and A_3 are large and sparse, with dimension $n = 9,956$. This problem is readily available in the problem collection [7].

As in [18], we note that the region of interest is away from the branch points $\lambda = 0$ and $\lambda = \sigma_2^2$. That is, we are interested in $\mathrm{Re}\,\lambda > \sigma_1^2$. It is common to shift and scale the problem to transform the region of interest to be approximately within unit length. I.e. we consider the transformed problem $\widehat{\lambda} = \alpha\lambda + \beta$, where $\alpha = 300^2 - 200^2$, and the shift is chosen to be $\beta = 250^2$, cf. [18][22].

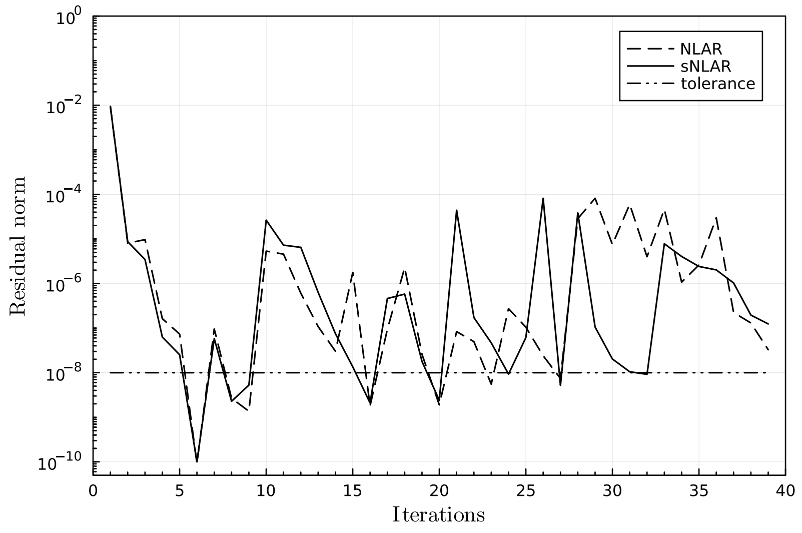

Figure 4.2: *Convergence history of sNLAR and NLAR. Eigenvalues are computed to a tolerance of* 10^{-8}. *Each time an eigenvalue is found, the method moves on to the next one, producing the spikey appearance in the plot. Notice that the convergence history of sNLAR follows that of NLAR closely, although sNLAR sometimes requires a few more iterations to converge. Notice also that NLAR appears to stagnate during the computation of the eighth eigenvalue, while sNLAR does not. However, this may be an artefact of how the error is measured. Both methods are started identically. The sketching parameter was chosen as 4 times the maximum number of iterations, and we choose the maximium number of iterations to be* 40, *i.e. sNLAR is operating on vectors of size* 160. *sNLAR uses an orthogonalization truncation length of* 4.

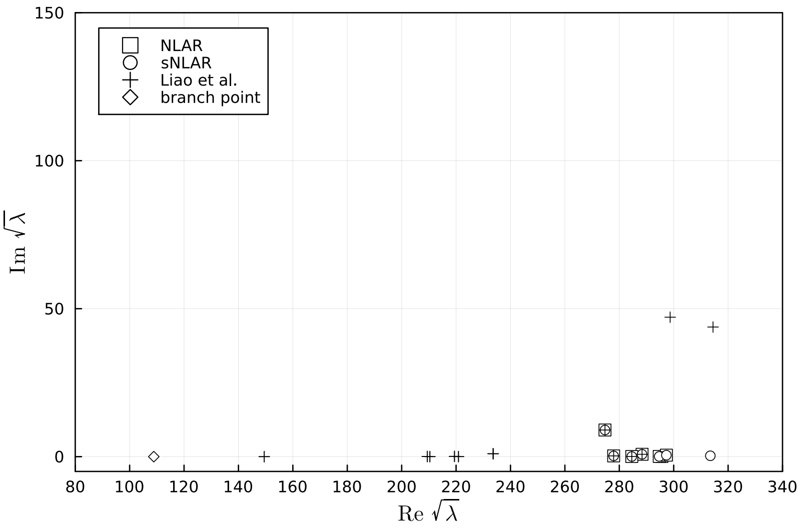

Figure 4.3: *Computed spectrum from sNLAR, NLAR, and from [22], in square-root scale. Notice that sNLAR succeeds in computing all eight eigenvalues in the desired cluster, while NLAR manages to compute seven of them. Also notice that we find some eigenvalues of the cluster not reported by Liao et al.*

Since the convergence of our method depends heavily on the distance between the shift, σ, and the eigenvalue currently being approximated, cf. Theorem 2.3.1, we will focus only on finding smaller groups of eigenvalues, as opposed to finding all eigenvalues within the region of interest. Notice that the shift σ in the methods is independant of the shift *of the problem*, α. Rather, after we have transformed our problem to be within unit magnitude, our shift σ will typically be between 0 and 1, controlling where in the region of interest we would like to search for eigenvalues. We will adopt this strategy since the eigenvalues of our problem are fairly clustered, see [18]. Were we to try and find all eigenvalues in the region, we would quickly observe stagnated convergence after the initial group of eigs has converged. Hence, the computational task is as follows; we want to find all eigenvalues in the cluster of eight eigenvalues in the right-most part of the region of interest, cf. [18], to a tolerance of 10^{-8}. We compare the performance of sNLAR with that of NLAR. The convergence of both methods for this task can be seen in Figure 4.2, and Figure 4.3 shows the computed spectrum of the problem in comparison with the spectrum computed in [22]. Cf. also the spectrum computed in [18].

We see that sNLAR is successful in computing the desired part of the spectrum. Figure 4.3 also shows that we manage to compute a few eigenvalues that where not captured by the method in [22]. They are however included in the spectrum computed in [18]. From the convergence history of sNLAR, we see that it follows the classical method closely, only sometimes requiring a few more iteratins to converge. It also appears as if the initial approximations when starting the work on a new eigenvalue are sometimes of lower quality in sNLAR. However, as with the classical method, the convergence is exceedingly rapid after this, and so this does not represent a significant shortcoming of the method.

Another aspect of the converge that is interesting to note is that for this problem, the spikes in the error that are so charachteristic to linear randomly sketched methods [34][11], seem to be absent. Even though the sketching parameter is relatively small, four times the maximum search depth for this example, the convergence appears very smooth. For context, this means that sNLAR operates on vectors of size 160, i.e. vectors that are around $1-2\%$ the original size of the problem. This example highlights the massive reduction in dimension that is possible with this class of method. However, it is also possible that the lack of irregular convergence is simply a result of the extremely rapid convergence, and that the charachteristic spikes in the convergence would once more materialize in problems where the convergence speed is more moderate.

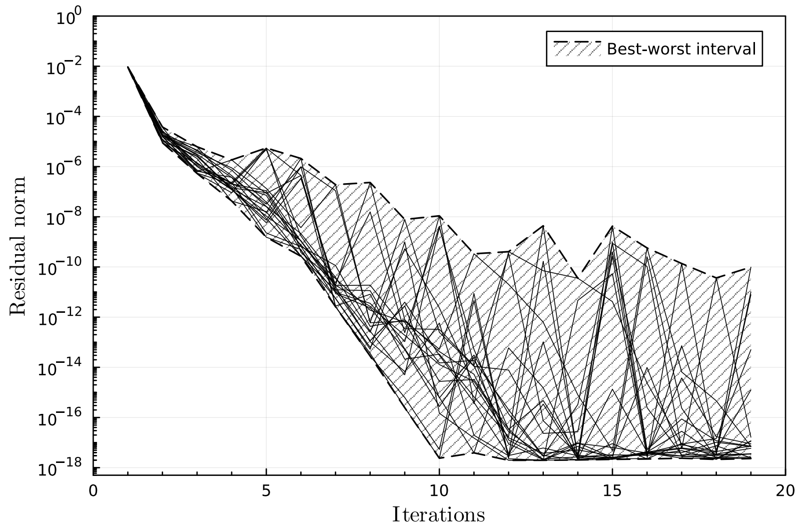

Figure 4.4: *Convergence history over 20 runs of sNLAR, with the objective of finding 1 eigenvalue in the "gun" benchmark. The sketching paramter is set to be four times the maximum number of iterations, and the truncation length in the orthogonalization is chosen to be* 4*. The maximum number of iterations is taken to be* 20*. The randomization is seeded differently each run, but the shift σ is the same each run. The figure also records the combined best-worst interval over all the runs.*

Continuing, we will take the opportunity to use this example as the basis for validating the robustness of the current method. More precisely, since the method contains randomization as one of its core elements, it is important to understand how this affects the consistency of the algorithm. I.e. how much do the results change over a larger sample of runs. To try and resolve this question, we will focus on computing only one eigenvalue of the same benchmark problem as above, but we will repeat the computation a number of times, with differently seeded randomization each time. In particular, we will perform 20 runs of the method, with the tolerance set so low that we record the entire convergence history of the method while computing this one eigenvalue. The parameters of sNLAR are set to the same values as in the example above, with the exception of setting the maximum number of iterations to 20, to not record an unnecessary amount of history once the method has already converged to machine precision.

The results from these computations are related in Figure 4.4. Comparing the convergence characteristics with those of Figure 4.2, we see that the convergence is very similar in the intial part of the computations, but that significant spikes in the deacy of the residual error begin to appear after around eight or so iterations. This explains why these spikes were mostly absent from the earlier convergence plot.

At an initial inspection, it might appear as if these spikes completely dominate the convergence of the method, and that they might be highly detrimental to the robustness of sNLAR.

While they absolutely affect the usage of the method, and in some cases they do warrant some careful treatment, this initial impression is largely an artefact of the way that these error-curves are presented here. While the convergence does suffer from these occasional spikes, it is not sporadic to the point of making the method unreliable. Many of the spikes occur only once the error is approaching machine precision, and the general trend indicates that the presence of a spike in the error does not inhibit further decrease in the error, or even indicate stagnated convergence in continued computation Furthermore, in the presence of a robust stopping condition, perhaps an extension of the type of condition suggested in [11], these spikes really do not become a problem unless very stringent tolerance conditions are imposed. The spikes rarely last more than a handful of iterations, and the method quickly recovers from these anomalies, continuing to generate high-quality approximations.

In conclusion, these experiments seem to indicate that the method is fairly robust, certainly robust enough for consistently generating high quality approximations of the desired eigenpairs, and the characteristic error-spikes present in linear methods, e.g. as shown in [34][11], also manifests themselves in this nonlinear application of the randomized sketching framework.

5 Conclusions and outlook

We have presented a novel method, namely an application of the sketch-and-solve paradigm to nonlinear eigenproblems. Our method is built upon the NLAR method due to Voss [36], but modifies this method by introducing a random subspace embedding, allowing us to significantly lower the dimensionality of the problem.

We have also verified our approach by a number of numerical experiments, indicating good performance, and indeed comparable performance to the classical NLAR method. In addition, our approach has been motivated by various analog methods for the linear eigenproblem, cf. Section 3.

A number of comments are due. Firstly, we mention some shortcomings of sNLAR, in addition to some advantages over the classical method. Initially, we note that for the problems we have investigated, we have not observed speed-ups, in terms of computation time, to the extent that has been observed for some linear problems, see for instance [25]. In [25], the authors report speed-ups of a factor of up to 100. The main reason for this extreme performance advantage over classical methods is that their randomized method, in the same manner as our proposed method, utilizes a truncated orthogonalization strategy. In linear problems, the orthogonalization is often the dominating step in the computation, and hence very large reductions in computation are available for any strategy minimizing this. Traditional methods for reducing the time spent in orthogonalization include *restarting*, see for example [9], but as we have seen, the sketching framework offers an attractive alternative to this approach.

Now, the reason we do not see comparable speed-ups for our problems in mainly due to this dominance of the orthogonalization being absent. More precisely, the solution of the projected problem, and the solution of the linear system when solving for the next search direction, are by far the most computationally intensive steps of our method. The main appeal of sketched methods is that they significantly reduce the time spent orthogonalizing the search space basis, and if this step is not a significant contributor to the over all complexity, we simply do not obtain reductions in complexity comparable to many linear problems.

However, there are other benefits to be had when employing a sketched method such as sNLAR. In particular, such methods unlock opportunities for implementation in modern highly parallel environments. In these environments, orthogonalization induces a global syncronization step, and since communication between processes is the main bottleneck in these modern arcitechtures, any reduction in the amount of orthogonalization needed for the algorithm to be performant is highly attractive. See for instance [5][29] for discussions on the role of orthogonalization in modern computing environments. Since our method heavily cuts down on the amount of orthogonalization that needs to be performed, it could find use in cases such as this, since the performance in terms of accuracy appears to be comparable to the classical method, from our experiments. Since the dominating steps in terms of computation time, are present in both NLAR and sNLAR, the methods achieve similar performance in this regard.

Finally, with regards to the gains made form reduced orthogonalization, we remark that for some problems we might see more significant reduction in computation time. The NEPs we have chosen to use as demonstration in this work, while fairly large, dwindle in comparison with other problems when it somes to shear size. Furthermore, we have employed the *infinite Arnoldi method* [18] for the solution of the projected problem. We have chosen to use this method mainly due to its very reliable stability characteristics, and tendency to be very robust in finding a large number of eigenvalues, see [18]. However, the method comes with some overhead, and other methods might be more suitable if computation time is of greater importance, for instance a Newton-type solver, which typically boasts quadratic convergence and less overhead. The combination of using a medium scale problem, together with the chosen inner solver, may have contributed to making the time spent in othogonalization even more insignificant when compared to the other elements of the algorithm. Hence, when dealing with very large problems where orthogonalization becomes more influential, in combination with using a different solver for the projected problem, we might see a more significant reduction in the computation time.

There are a number of questions about sNLAR that still need answering. The convergence characteristics of our method have not been investigated theoretically in this work. One reason for this is that the convergence of NLAR is still not well understood theoretically. For this reason, it will likely be very difficult to gain understanding about our proposed method without first gaining some grip on the classical method. However, while a full understanding of sNLAR might be intractable for the moment, it would still be interesting to construct a bound on the distance from the sNLAR approximant to that of NLAR. This approach has been adopted for various application when a full understanding of the theoretical aspects of a randomized method might not be feasible. For instance, [11] and [34] both employ this approach. It is possible that a construction similar to the one in [34] might be a fruitful approach for such a characterization of the sNLAR approximant. If a bound to the NLAR approximant could be established, the empirical evidence for the performance of NLAR would also speak in favor of sNLAR. Since NLAR has enjoyed years of practical use to good success, this would be strong support for sNLAR.

Another aspect of the convergence of sNLAR that might be of concern, and seems to be almost universal among sketched methods, are the occasionally very large spikes in the convergence that can be observed in Figure 4.4. See also the references provided above for other examples. These spikes are still not well understood, partly because they seldom are of practical concern and can often be dealt with effectively. Some theoretical work has been done on the issue, but the phenomenon is still not fully understood. In [34], the authors observe that the spikes in the convergence correlates strongly with a difference in the Ritz values generated in the classical and randomized methods, respectively, but they make no further investigations into this observation. Perhaps this could be one possibly way to characterize the convergence of these sketched methods.

Other possible theoretical approaches include [28], where the authors establish a theoretical framework for sketched Krylov-type methods, in the context of matrix functions. Perhaps some of this machinery could be leveraged in the context of our proposed method.

In conclusion, we have presented a novel method for the solution of large and sparse NEPs, demonstrated good performance, with comparable characteristics to the nonlinear Arnoldi method, and while many theoretical questions remain, our experiments have demonstrated the practical potential of our approach. sNLAR also has the possibility of finding use in various settings within numerical linear algebra, for instance in highly parallel environments, making it an attractive alternative for the solution of NEPs.

References

[1] Dimitris Achlioptas. "Database-friendly random projections: Johnson-Lindenstrauss with binary coins". *Journal of Computer and System Sciences* 66.4 (2003), pp. 671–687.

[2] Nir Ailon and Bernard Chazelle. "The fast Johnson–Lindenstrauss transform and approximate nearest neighbors". *SIAM Journal on Computing* 39.1 (2009), pp. 302–322.

[3] W. E. Arnoldi. "The principle of minimized iterations in the solution of the matrix eigenvalue problem". *Quarterly of Applied Mathematics* 9.1 (1951), pp. 17–29.

[4] Oleg Balabanov and Laura Grigori. "Randomized Gram–Schmidt process with application to GMRES". *SIAM Journal on Scientific Computing* 44.3 (2022), A1450–A1474.

[5] Grey Ballard et al. "Minimizing communication in linear algebra". arxiv:0905.2485 (2009).

[6] Michele Benzi and Paola Boito. "Decay properties for functions of matrices over C^*-algebras". *Linear Algebra and its Applications* 456 (2014), pp. 174–198.

[7] Timo Betcke et al. "NLEVP: a collection of nonlinear eigenvalue problems". *ACM Transactions on Mathematical Software* 39.2 (2013), pp. 1–28.

[8] Ernest R. Davidson. "The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices". *Journal of Computational Physics* 17 (1975), pp. 87–94.

[9] Gene H. Golub and Charles F. Van Loan. *Matrix computations.* Fourth edition. The Johns Hopkins University Press, 2013.

[10] Stefan Güttel, Daniel Kressner, and Bart Vandereycken. "Randomized sketching of nonlinear eigenvalue problems". arXiv:2211.12175 (2022).

[11] Stefan Güttel and Marcel Schweitzer. "Randomized sketching for Krylov approximations of large-scale matrix functions". *SIAM Journal on Matrix Analysis and Applications* 44.3 (2023), pp. 1073–1095.

[12] Stefan Güttel and Françoise Tisseur. "The nonlinear eigenvalue problem". *Acta Numerica* 26 (2017), pp. 1–94.

[13] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions". *SIAM Review* 53.2 (2011), pp. 217–288.

[14] Leslie Hogben, ed. *Handbook of Linear Algebra.* Chapman and Hall/CRC, 2013.

[15] Roger A. Horn and Charles R. Johnson. *Matrix analysis.* Second edition. Cambridge University Press, 2017.

[16] H. Igarashi, Y. Sugawara, and T. Honma. "A numerical computation of external Q of resonant cavities". *IEEE Transactions on Magnetics* 31.3 (1995), pp. 1642–1645.

[17] Elias Jarlebring, Giampaolo Mele, and Olof Runborg. "The waveguide eigenvalue problem and the tensor infinite Arnoldi method". *SIAM Journal on Scientific Computing* 39.3 (2017), A1062–A1088.

[18] Elias Jarlebring, Wim Michiels, and Karl Meerbergen. "A linear eigenvalue algorithm for the nonlinear eigenvalue problem". *Numerische Mathematik* 122.1 (2012), pp. 169–195.

[19] William B. Johnson and Joram Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space". 26 (1984). Ed. by Richard Beals et al., pp. 189–206.

[20] Ravindran Kannan and Santosh Vempala. "Randomized algorithms in numerical linear algebra". *Acta Numerica* 26 (2017), pp. 95–135.

[21] Mats G. Larson and Fredrik Bengzon. *The finite element method: theory, implementation, and applications.* Springer Berlin Heidelberg, 2013.

[22] Ben-Shan Liao et al. "Nonlinear Rayleigh-Ritz iterative method for solving large scale nonlinear eigenvalue problems". *Taiwanese Journal of Mathematics* 14.3 (2010).

[23] Per-Gunnar Martinsson and Joel A. Tropp. "Randomized numerical linear algebra: foundations and algorithms". *Acta Numerica* 29 (2020), pp. 403–572.

[24] Wim Michiels and Silviu-Iulian Niculescu. *Stability and stabilization of time-delay systems: an eigenvalue-based approach.* Society for Industrial & Applied Mathematics, 2008.

[25] Yuji Nakatsukasa and Joel A. Tropp. "Fast & accurate randomized algorithms for linear systems and eigenvalue problems". arXiv:2111.00113 (2022).

[26] A. Neumaier. "Residual Inverse Iteration for the Nonlinear Eigenvalue Problem". *SIAM Journal on Numerical Analysis* 22.5 (1985), pp. 914–923.

[27] Kouhei Ooi et al. "Solution of a nonlinear eigenvalue problem using signed singular values". *East Asian Journal on Applied Mathematics* 7.4 (2017), pp. 799–809.

[28] Davide Palitta, Marcel Schweitzer, and Valeria Simoncini. "Sketched and truncated polynomial Krylov methods: Evaluation of matrix functions". arxiv:2306.06481 (2023).

[29] Bernard Philippe and Lothar Reichel. "On the generation of Krylov subspace bases". *Applied Numerical Mathematics* 62 (2012), pp. 1171–1186.

[30] Geoff Pleiss et al. "Fast Matrix Square Roots with Applications to Gaussian Processes and Bayesian Optimization" (2020), arXiv:2006.11267.

[31] Vladimir Rokhlin and Mark Tygert. "A fast randomized algorithm for overdetermined linear least-squares regression". *Proceedings of the National Academy of Sciences* 105.36 (2008), pp. 13212–13217.

[32] Axel Ruhe. "Algorithms for the nonlinear eigenvalue problem". *SIAM Journal on Numerical Analysis* 10.4 (1973), pp. 674–689.

[33] Sergey I. Solov'ëv. "Preconditioned iterative methods for a class of nonlinear eigenvalue problems". *Linear Algebra and its Applications* 415.1 (2006), pp. 210–229.

[34] Edouard Timsit, Laura Grigori, and Oleg Balabanov. "Randomized orthogonal projection methods for Krylov subspace solvers". arXiv:2302.07466 (Mar. 10, 2023).

[35] Lloyd N. Trefethen and Mark Embree. *Spectra and pseudospectra: the behavior of nonnormal matrices and operators.* Princeton University Press, 2020.

[36] H. Voss. "An Arnoldi method for nonlinear eigenvalue problems". *BIT Numerical Mathematics* 44.2 (2004), pp. 387–401.

[37] Heinrich Voss. ”A maxmin principle for nonlinear eigenvalue problems with application to a rational spectral problem in fluid-solid vibration”. *Applications of Mathematics* 48.6 (2003), pp. 607–622.