Numerical Algebra, Control and Optimization
Vol. 14, No. 3, September 2024, pp. 526-546
doi:10.3934/naco.2023002

ACCURATE POLYNOMIAL FITTING AND EVALUATION VIA ARNOLDI

Lei-Hong Zhang[1], Yangfeng Su[2] and Ren-Cang Li[*3,4]

[1]School of Mathematical Sciences, Soochow University
Suzhou 215006, Jiangsu, China

[2]School of Mathematical Sciences, Fudan University
Shanghai 200433, China

[3]Department of Mathematics, University of Texas at Arlington
Arlington, TX 76019-0408, USA

[4]Department of Mathematics, Hong Kong Baptist University
Kowloon Tong, Kowloon, Hong Kong

(Communicated by Qiang Ye)

Abstract. Given sample data points $\{(x_j, f_j)\}_{j=1}^N$, in [Brubeck, Nakatsukasa, and Trefethen, *SIAM Review*, 63 (2021), pp. 405-415], an Arnoldi-based procedure is proposed to accurately evaluate the best fitting polynomial, in the least squares sense, at new nodes $\{s_j\}_{j=1}^M$, based on the Vandermonde basis. Numerical tests indicated that this procedure can in general achieve high accuracy. The main purpose of this paper is to perform a forward rounding error analysis in finite precision. Our result establishes sensitivity factors regarding the accuracy of the algorithm, and provides a theoretical justification for why the algorithm works. For least-squares approximation on an interval, we propose a variant of this Arnoldi-based evaluation by using the Chebyshev polynomial basis. Numerical tests are reported to demonstrate our forward rounding error analysis.

1. Introduction.

Approximation by polynomials and rational functions lies in the core of classical approximation theory. From the most general and basic tasks of designing subroutines for evaluating elementary functions and special functions, applications of approximation theory are across all spectra of sciences and engineering, including digital filters, eigenvalues and eigenvectors of matrices, model reduction and optimal control, numerical solution of PDEs (see e.g., [28, Chapter 23]) and other new areas (e.g., [4, 6, 12, 21, 29]).

Given a function $f(x)$ on a domain, a wealth of fundamental theories and efficient numerical algorithms for the best fitting polynomials in the L_2-norm as well as the L_∞-norm have been established (see, e.g., [20, 25, 28]). In the discrete situation, because f is either unknown or too complicated to deal with, sample data points

2020 *Mathematics Subject Classification.* Primary: 65G50, 41A05, 65D05; Secondary: 65D10.
Key words and phrases. Rounding error analysis, polynomial interpolation, Lanczos process, Arnoldi process, SOAR, least-squares, Vandermonde matrix.
Zhang is supported in part by the National Natural Science Foundation of China NSFC-12071332. Su is supported in part by by the National Natural Science Foundation of China NSFC-11971122. Li is supported in part by USA NSF DMS-1719620 and DMS-2009689.
*Corresponding author: Ren-Cang Li.

Algorithm 1 PFEvA: Polynomial fitting and evaluation via Arnoldi [5]

```
Input: interpolating points $\{(x_j, f_j)\}_{j=1}^N$, new nodes $\{s_j\}_{j=1}^M$ to be evaluated at, integer $0 \le n \le N$.
Output: values $y_j = \mathrm{p}_{n-1}(s_j)$ for $1 \le j \le M$ of the best fitting polynomial of degree $n-1$:
    $$\mathrm{p}_{n-1} = \operatorname*{argmin}_{\mathrm{p} \in \mathbb{P}_{n-1}} \|\boldsymbol{f} - \mathrm{p}_{n-1}(\boldsymbol{x})\|_2,$$
    where $\boldsymbol{x} = [x_1, \dots, x_N]^{\mathrm{T}}$ and $\boldsymbol{f} = [f_1, \dots, f_N]^{\mathrm{T}}$.
                         Construction stage
1: call the Arnoldi/Lanczos process with $X = \mathrm{diag}(\boldsymbol{x})$ on $Q\boldsymbol{e}_1 = \boldsymbol{1}_N$ to produce $XQ = QH + \beta_{n+1}\boldsymbol{q}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}$ such that[1] $Q^{\mathrm{H}}Q = N\,I_n$, where $H \in \mathbb{C}^{n\times n}$ is upper-Hessenberg;
2: compute $\boldsymbol{d} = \mathrm{argmin}_{\boldsymbol{z}} \|\boldsymbol{f} - Q\boldsymbol{z}\|_2$;
                         Evaluation stage
3: recursively compute $W$, one column at a time, such that $SW = WH + \beta_{n+1}\boldsymbol{w}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}$, where $\boldsymbol{s} = [s_1, \dots, s_M]^{\mathrm{T}}$, $S = \mathrm{diag}(\boldsymbol{s})$, and $W\boldsymbol{e}_1 = \boldsymbol{1}_n$;
4: return $\boldsymbol{y} = [y_1, \dots, y_M]^{\mathrm{T}} = W\boldsymbol{d}$.
```

$\{(x_j, f(x_j))\}_{j=1}^N$ are available, and least-squares approximation is a powerful tool to generate polynomials or rational functions that well approximate the underlying function $f(x)$ [25]. Often noise is inevitable and, as a result, the best fitting polynomial in the sense of minimizing the least squares error is a common and efficient approach for many practical situations.

Let $\mathrm{p}_{n-1}(x) \in \mathbb{P}_{n-1}$ be the best fitting polynomial of degree $n-1$ at given data points $\{(x_j, f_j)\}_{j=1}^N$, where $\mathbb{P}_{n-1}$ is the set of polynomials of degree no greater than $n-1$. There are numerous polynomial bases to express p_{n-1} in. The monomial basis $\{1, x, \dots, x^{n-1}\}$ is the most natural one, but unfortunately, in order to compute the associated coefficients, one needs to deal with a notoriously ill-conditioned Vandermonde linear system (see e.g., [2, 16, 17, 18]). However, in many applications, p_{n-1} is only a vehicle to produce values $\mathrm{p}_{n-1}(s_j) \approx f(s_j)$ at new nodes $\{s_j\}_{j=1}^M$, while the explicit form of p_{n-1} is not necessarily required. Recently, in [5], an effective Arnoldi/Lanczos-based procedure is proposed to evaluate $\{\mathrm{p}_{n-1}(s_j)\}_{j=1}^M$ without computing the monomial coefficients explicitly. It is in fact a realization of the on-the-fly construction of discrete orthogonal polynomials by Stieltjes orthogonalization. Algorithm 1 outlines the method in [5], where numerical examples are presented to demonstrate that the procedure in general computes $\{\mathrm{p}_{n-1}(s_j)\}_{j=1}^M$ more accurately than through first forming p_{n-1} in the monomial basis and then evaluating it at the new nodes.

The main purpose of this paper is to perform a forward rounding error analysis in finite precision arithmetic. In particular, we will establish sensitivity factors of Algorithm 1 and provide a theoretical justification for its effectiveness. As our second contribution, for least-squares approximation on an interval, we propose to extend this Arnoldi/Lanczos-based evaluation in Algorithm 1 by using the Chebyshev polynomials of the first kind as the polynomial basis, which is known to yield better accuracy [13]. Numerical tests are reported to verify our forward rounding error analysis, and also to illustrate the performance of two Arnoldi-based evaluation on an interval.

The rest of the paper is organized as follows. In section 2, we explain how Algorithm 1 works in exact arithmetic. In section 3, we perform a rounding error analysis for Algorithm 1. We then extend the approach to the case of using the Chebyshev polynomials as the polynomial basis in section 4 for the least-squares real approximation on an interval. Also in section 4, two implementations are presented, along with their rounding error analyses. Numerical demonstration of how good our rounding error analysis is for the three implementations is carried out in section 5. Finally conclusions are drawn in section 6.

Notation. Throughout this paper, $\mathbb{C}^{n\times m}$ is the set of all $n\times m$ complex matrices, $\mathbb{C}^n = \mathbb{C}^{n\times 1}$, and $\mathbb{C} = \mathbb{C}^1$. Similarly $\mathbb{R}^{n\times m}$, $\mathbb{R}^n$, and $\mathbb{R}$ are defined by replacing the word *complex* with *real*. Vectors are typeset in bold lower case letters. $I_n \equiv [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n] \in \mathbb{R}^{n\times n}$ is the $n \times n$ identity matrix, where $\boldsymbol{e}_i$ is its ith column. For $A \in \mathbb{C}^{m\times n}$, A^{T}, A^{H}, and $A^{\dagger}$ are the transpose, the complex conjugate transpose, and the Moore-Penrose inverse of A, respectively. $\mathcal{R}(A)$ represents the column space of A. For convenience, we adopt MATLAB notation: $A_{(i,j)}$ is the (i, j)th entry of A, and $A_{(k:\ell,i:j)}$ is the submatrix of A that consists of intersections from row k to row ℓ and column i to column j. $\|\cdot\|_2$ denotes the vector 2-norm or the matrix spectral norm, and and $\|\cdot\|_{\mathrm{F}}$ denotes the matrix Frobenius norms. The set of polynomials of degrees no higher than $n-1$ is denoted by $\mathbb{P}_{n-1}$ and the kth Krylov subspace generated by a matrix A on a vector $\boldsymbol{b}$ is defined as

$$\mathcal{K}_k(A, \boldsymbol{b}) = \mathcal{R}([\boldsymbol{b}, A\boldsymbol{b}, \ldots, A^{k-1}\boldsymbol{b}]).$$

2. Evaluation in exact arithmetic.

Throughout $\{x_j\}_{j=1}^N$ is reserved for the interpolating nodes, while $\{s_j\}_{j=1}^M$ are the nodes to be evaluated at. Let $\boldsymbol{x} = [x_j] \in \mathbb{C}^N$ be the vector formed by the interpolating nodes, and similarly $\boldsymbol{s} = [s_j] \in \mathbb{C}^M$ the vector formed by the evaluating nodes. Denote by

$$V_{\boldsymbol{x}} = [\boldsymbol{e}, X\boldsymbol{e}, \ldots, X^{n-1}\boldsymbol{e}] = \begin{bmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ & \ldots & \ldots & \\ 1 & x_N & \ldots & x_N^{n-1} \end{bmatrix} \tag{2.1}$$

the $N \times n$ rectangular Vandermonde matrix generated by the nodes $\{x_j\}_{j=1}^N$, where $X = \mathrm{diag}(\boldsymbol{x})$ is the diagonal matrix with the entries of $\boldsymbol{x}$ as the diagonal entries. Similarly, we define $V_{\boldsymbol{s}} \in \mathbb{C}^{M\times n}$. Before analyzing Algorithm 1 in finite precision arithmetic, we first provide a diagram to explain the process in Algorithm 1. The key is an unknown triangular matrix R that provides an implicit link between step 1 and step 3 of Algorithm 1, and it is this connection in R that ensures $y_j = \mathrm{p}_n(s_j)$ in theory. We point out that this connection is already stated in [5], but we provide much more detail here to aid our later rounding error analysis.

To understand the process in Figure 2.1, we first establish a relation between the QR decomposition of $V_{\boldsymbol{x}}$ and the Arnoldi/Lanczos process with X (see, e.g., [27, pp. 298] and [1, Lemma 2.2]). Theorem 2.1 is stated more generally in the complex domain. It holds true if all $\mathbb{C}$ is replaced by $\mathbb{R}$.

[1]This is done by slightly modifying the usual Arnoldi/Lanczos process by normalizing each new Arnoldi/Lanczos vector to have norm $\sqrt{N}$ instead of 1. We point out that Algorithm 1 as stated does not require that x_j and f_j be real. In the case when all x_j are real, it is the Lanczos process in the *construction stage* and, as a result, H is a real symmetric tridiagonal matrix.

$$\begin{tikzcd}
& V_{\boldsymbol{x}} = QR & V_{\boldsymbol{s}} = WR & \\
\boldsymbol{x} \arrow[r, "XQ = QH + \beta_{n+1}\boldsymbol{q}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}", "\text{step 1: Arnoldi}"'] & (Q, H) \arrow[u, dashed] \arrow[r] \arrow[d, "\text{step 2}"'] & W \arrow[u, dashed] \arrow[d, "\text{step 4}"] & (\boldsymbol{s}, H) \arrow[l, "SW = WH + \beta_{n+1}\boldsymbol{w}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}"', "\text{step 3}"] \\
& \boldsymbol{d} = Q^{\dagger}\boldsymbol{f} \arrow[r] & \boldsymbol{y} = W\boldsymbol{d} &
\end{tikzcd}$$

FIGURE 2.1. Diagram illustration of Algorithm 1 where H at step 3 is from step 1.

Theorem 2.1. *Given $\boldsymbol{q}_1 \in \mathbb{C}^N$, $A \in \mathbb{C}^{N\times N}$, and an integer $1 < n \le N$, let*

$$K = [\boldsymbol{q}_1, A\boldsymbol{q}_1, \dots, A^{n-1}\boldsymbol{q}_1] \in \mathbb{C}^{N\times n}. \tag{2.2}$$

Suppose $\operatorname{rank}(K) = n$.

(i) *If $K = QR$ with $Q \in \mathbb{C}^{N\times n}$ (not necessarily orthonormal) having $\boldsymbol{q}_1$ as its first column and upper triangular $R \in \mathbb{C}^{n\times n}$, then*

$$AQ = QH + \boldsymbol{q}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}, \tag{2.3a}$$

where

$$\boldsymbol{q}_{n+1} = \frac{[I_n - Q(Q^{\mathrm{H}}Q)^{-1}Q^{\mathrm{H}}]A^n\boldsymbol{q}_1}{r_{n,n}} \perp \mathcal{R}(K), \tag{2.3b}$$

$$H = R[\boldsymbol{e}_2, \dots, \boldsymbol{e}_n, \boldsymbol{v}]R^{-1} \quad \text{with} \quad \boldsymbol{v} = R^{-1}Q(Q^{\mathrm{H}}Q)^{-1}Q^{\mathrm{H}}A^n\boldsymbol{q}_1, \tag{2.3c}$$

and $r_{n,n}$ is the (n,n)th entry of R. In particular, H is upper-Hessenberg.

(ii) *Conversely, if (2.3a) holds for upper Hessenberg $H \in \mathbb{C}^{n\times n}$ and $Q \in \mathbb{C}^{N\times n}$ (not necessarily orthonormal), then $K = QR$ where*

$$R = [\boldsymbol{e}_1, H\boldsymbol{e}_1, \dots, H^{n-1}\boldsymbol{e}_1] \in \mathbb{C}^{n\times n} \tag{2.4}$$

is upper triangular.

Proof. For item (i), we note that $\operatorname{rank}(Q) = \operatorname{rank}(R) = n$ because $\operatorname{rank}(K) = n$. We have

$$AQR = AK = [A\boldsymbol{q}_1, \dots, A^{n-1}\boldsymbol{q}_1, A^n\boldsymbol{q}_1] = [K\boldsymbol{e}_2, \dots, K\boldsymbol{e}_n, A^n\boldsymbol{q}_1]. \tag{2.5}$$

Since $A^n\boldsymbol{q}_1 \in \mathcal{R}(K) \bigoplus \mathcal{R}(K)^{\perp}$ and $\operatorname{rank}(K) = n$, there are unique $\boldsymbol{v} \in \mathbb{C}^n$ and $\tilde{\boldsymbol{q}}_{n+1} \in \mathcal{R}(K)^{\perp}$ such that

$$A^n\boldsymbol{q}_1 = K\boldsymbol{v} + \tilde{\boldsymbol{q}}_{n+1}.$$

In fact, it can be seen that

$$\boldsymbol{v} = (K^{\mathrm{H}}K)^{-1}K^{\mathrm{H}}A^n\boldsymbol{q}_1 = R^{-1}(Q^{\mathrm{H}}Q)^{-1}Q^{\mathrm{H}}A^n\boldsymbol{q}_1, \;\; \tilde{\boldsymbol{q}}_{n+1} = A^n\boldsymbol{q}_1 - K\boldsymbol{v}.$$

Thus, we have from (2.5) that

$$AQR = K[\boldsymbol{e}_2, \dots, \boldsymbol{e}_n, \boldsymbol{v}] + \tilde{\boldsymbol{q}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}},$$

and, then, plugging in $K = QR$, we get

$$AQ = QR[\boldsymbol{e}_2, \dots, \boldsymbol{e}_n, \boldsymbol{v}]R^{-1} + \tilde{\boldsymbol{q}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}R^{-1},$$

yielding (2.3) upon noticing $\boldsymbol{e}_n^{\mathrm{T}} R^{-1} = \boldsymbol{e}_n / r_{n,n}$ and letting $\boldsymbol{q}_{n+1} = \tilde{\boldsymbol{q}}_{n+1}/r_{n,n}$. Finally, it can be verified that $H = R[\boldsymbol{e}_2, \dots, \boldsymbol{e}_n, \boldsymbol{v}]R^{-1}$ is upper Hessenberg because both R and R^{-1} are upper triangular.

For item (ii), we first note $K\boldsymbol{e}_1 = QR\boldsymbol{e}_1 = Q\boldsymbol{e}_1 = \boldsymbol{q}_1$. It follows from $n > 1$, (2.3a), and (2.4) that $\boldsymbol{e}_n^{\mathrm{T}}\boldsymbol{e}_1 = 0$ and

$$K\boldsymbol{e}_2 = A\boldsymbol{q}_1 = (AQ - \boldsymbol{q}_{n+1}\boldsymbol{e}_n^{\mathrm{T}})\boldsymbol{e}_1 = QH\boldsymbol{e}_1 = QR\boldsymbol{e}_2.$$

In summary, we have proved

$$A^{i-1}\boldsymbol{q}_1 = QH^{i-1}\boldsymbol{e}_1, \;\; K\boldsymbol{e}_i = QR\boldsymbol{e}_i \tag{2.6}$$

for $i = 1, 2$. Assume now that (2.6) holds for $i = k \le n-1$. We claim it will also hold for $i = k+1 \le n$. To this end, with the help of (2.3a), we have

$$\begin{aligned} A^k\boldsymbol{q}_1 &= A(A^{k-1}\boldsymbol{q}_1) = A(QH^{k-1}\boldsymbol{e}_1) = (AQ)H^{k-1}\boldsymbol{e}_1 \\ &= (QH + \boldsymbol{q}_{n+1}\boldsymbol{e}_n^{\mathrm{T}})H^{k-1}\boldsymbol{e}_1 \\ &= QH^k\boldsymbol{e}_1 + \boldsymbol{q}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}H^{k-1}\boldsymbol{e}_1 \\ &= QH^k\boldsymbol{e}_1, \end{aligned}$$

where we have used the fact $\boldsymbol{e}_n^{\mathrm{T}}H^j\boldsymbol{e}_1 = 0$ for $1 \le j < n-1$ because H^j is a left-banded matrix with left-bandwidth j. Now use (2.2) and (2.4) to get

$$K\boldsymbol{e}_{k+1} = A^k\boldsymbol{q}_1 = Q(H^k\boldsymbol{e}_1) = QR\boldsymbol{e}_{k+1}.$$

This completes the proof of (2.6) for $i = k+1 \le n$. By induction, (2.6) holds for all $1 \le i \le n$, yielding $K = QR$. $\square$

Remark 2.1. We emphasize that in Theorem 2.1, Q does not have to have mutually orthogonal columns, although in theory for Q in step 1 of Algorithm 1, $Q^{\mathrm{H}}Q = NI_N$. However, the columns of W in step 3 of Algorithm 1 does not have mutually orthogonal columns even in theory, but Theorem 2.1 still applies so long as $\operatorname{rank}(W) = n$ which is guaranteed because $\operatorname{rank}(V_{\boldsymbol{s}}) = n$.

Suppose for the moment that the entries of $\boldsymbol{f} = [f_1, \dots, f_N]^{\mathrm{T}}$ are values of a polynomial

$$\mathrm{p}_{n-1}(x) = [1, x, \dots, x^{n-1}]^{\mathrm{T}}\boldsymbol{a} = a_n x^{n-1} + a_{n-1}x^{n-2} + \cdots + a_1 \tag{2.7}$$

at the nodes $\{x_j\}_{j=1}^N$, where $\boldsymbol{a} \in \mathbb{C}^n$ is the coefficient vector, i.e., $f_j = \mathrm{p}_{n-1}(x_j)$ exactly for $j = 1, 2, \dots, N$. By Theorem 2.1, we know that in step 1 of Algorithm 1 there is a triangular matrix R as in (2.4) such that

$$V_{\boldsymbol{x}} = QR, \;\; \boldsymbol{f} = V_{\boldsymbol{x}}\boldsymbol{a} = Q(R\boldsymbol{a}) =: Q\boldsymbol{d}, \tag{2.8}$$

where $\boldsymbol{d} = R\boldsymbol{a}$. Observe that $\boldsymbol{f}$ has the coordinate vector $\boldsymbol{d}$ in the orthonormal basis composed of the columns of Q, modulo a constant scaling factor $\sqrt{N}$. By Theorem 2.1(ii), step 3 of Algorithm 1 implies that

$$V_{\boldsymbol{s}} = WR, \;\; \boldsymbol{y} = V_{\boldsymbol{s}}\boldsymbol{a} = W(R\boldsymbol{a}) = W\boldsymbol{d}. \tag{2.9}$$

Hence $y_j = \mathrm{p}_{n-1}(s_j)$ exactly in theory. It also reveals that what Algorithm 1 tries to do is to find a proper basis composed of the columns of W so that $\boldsymbol{y}$ has the same coordinate vector $\boldsymbol{d}$ in the basis as $\boldsymbol{f}$ has in (2.8), as oppose to use the columns of Vandermonde matrix $V_{\boldsymbol{s}}$ as the basis, which is notoriously ill-conditioned [2, 16, 17, 18]. For that reason, better accuracy in computed y_j should be expected.

3. **Evaluation in finite precision.**

We now establish our rounding error analysis for Algorithm 1. We assume that the nodes $\{x_j\}_{j=1}^N$ and $\{s_j\}_{j=1}^M$ are all real and floating point numbers in the working precision, in notation $x_j \in \texttt{FPN}$, $s_j \in \texttt{FPN}$, but the interpolating function values f_j, also real, may contain errors. But we point that our results in this section can be made valid for complex nodes and function values by a few corresponding modifications: 1) all occurrences of machine unit roundoff $\mathfrak{u}$ need to be replaced by $7\mathfrak{u}$ to account at most 7 flops in the real or imaginary part of $+$, $-$, $\times$, and $\div$ of two complex numbers; 2) tridiagonal matrices as a result of the Lanczos process, e.g., $\widehat{H}$ in Lemmas 3.1 and 3.4, become upper Hessenberg matrices as a result of the Arnoldi process; 3) error estimations involving $\widehat{H}$ will then depend on n (see Remark 3.1). For demonstrating the idea, we restrict our following analysis to the real case.

3.1. **Rounding error analysis in the construction stage.**

When $f(x)$ is not a polynomial of degree $n-1$, p_{n-1} is determined by the least squares fitting and as a result $\mathrm{p}_{n-1}(s_j)$ is most likely very different from $f(s_j)$. Since this paper concerns with how to accurately evaluate the interpolating/best fitting polynomial at new nodes, everything else being equal, for the purpose of error analysis and numerical demonstrations later, it makes sense to assume values $\{f_j\}$ are from some polynomial $f(x)$ of degree $n-1$ so that we can compare $\mathrm{p}_{n-1}(s_j)$ against $f(s_j)$. Even so, in general $f(x_j)$ cannot be represented exactly as in $\texttt{FPN}$ in the working precision.

Let $\texttt{FPN} \ni \hat{f}_j \approx f(x_j)$ and $f(x) = \sum_{i=0}^{n-1} a_i x^i$. If evaluated by Horner's rule for $f(x_j) \in \mathbb{R}$, we have [7, section 1.6]

$$|f(x_j) - \hat{f}_j| \le \mathfrak{u}\left(2(n-1)\sum_{i=0}^{n-1}|a_i x_j^i|\right) =: \epsilon_{f_j}^0, \tag{3.1}$$

where $\mathfrak{u}$ is the unit machine roundoff.

In view of this, in what follows, we model the input data as $\{(x_j, \hat{f}_j)\}_{j=1}^N$ where $|f(x_j) - \hat{f}_j| \le \epsilon_{f_j}^0$. Equivalently,

$$\hat{\boldsymbol{f}} = [\hat{f}_1, \dots, \hat{f}_N]^{\mathrm{T}} = \boldsymbol{f} + \boldsymbol{\epsilon}_f^0, \text{ with } \boldsymbol{\epsilon}_f^0 = [\epsilon_{f_1}^0, \dots, \epsilon_{f_N}^0]^{\mathrm{T}}. \tag{3.2}$$

With this setting, we attempt to establish the error $|y_j - \hat{y}_j|$ for the computed output $\hat{y}_j \in \texttt{FPN}$ of Algorithm 1 where $y_j = f(s_j)$.

As our notation convention, in what follows, we will use $\hat{\xi} \in \texttt{FPN}$ to denote the computed value of its corresponding ξ. The following lemma summarizes an error analysis of the Arnoldi process in finite precision [11, 23, 24].

Lemma 3.1. *In finite precision, the Arnoldi process in Algorithm 1, for real $\{x_j\}_{j=1}^N$, produces $\widehat{Q}$, $\widehat{H}$, and $\hat{\boldsymbol{q}}_{n+1}$, satisfying*

$$X\widehat{Q} = \widehat{Q}\widehat{H} + \hat{\beta}_{n+1}\hat{\boldsymbol{q}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} + F, \tag{3.3}$$

where $\widehat{H} \in \mathbb{R}^{n\times n}$ is tridiagonal and $F \in \mathbb{R}^{N\times n}$ satisfies $\|F\|_{\mathrm{F}} \le \sqrt{n}\|X\|_2\mathfrak{u}$.

Based on this preliminary result and the connection in Theorem 2.1(ii), we can estimate the difference $V_{\boldsymbol{x}} - \widetilde{V}_{\boldsymbol{x}}$, where

$$\widetilde{V}_{\boldsymbol{x}} = \widehat{Q}\widetilde{R}, \text{ with } \widetilde{R} = [\boldsymbol{e}_1, \widehat{H}\boldsymbol{e}_1, \dots, \widehat{H}^{n-1}\boldsymbol{e}_1]. \tag{3.4}$$

Here we put tildes in $\widetilde{V}_{\boldsymbol{x}}$ and $\widetilde{R}$ instead of hats because there is no need to compute them, but rather their very existence is for the purpose of our error analysis. This convention will be adopted in what follows, too, i.e., a symbol with a tilde denote some quantity not needed during computations but for analysis only.

Lemma 3.2. *Let $\varDelta_{\boldsymbol{x}} := V_{\boldsymbol{x}} - \widetilde{V}_{\boldsymbol{x}} =: [\delta\boldsymbol{v}_1, \ldots, \delta\boldsymbol{v}_n]$. Then $\delta\boldsymbol{v}_1 = 0$, $\delta\boldsymbol{v}_2 = F\boldsymbol{e}_1$, and*

$$\delta\boldsymbol{v}_i = \left(X^{i-2}F + X^{i-3}F\widehat{H} + \cdots + XF\widehat{H}^{i-3} + F\widehat{H}^{i-2}\right)\boldsymbol{e}_1, \quad 3 \le i \le n, \tag{3.5a}$$

$$\|\delta\boldsymbol{v}_i\|_2 \le \left(\sqrt{n}\sum_{j=0}^{i-2}\|X\|_2^{j+1}\cdot\|\widehat{H}\|_2^{i-j-2}\right)\mathfrak{u} =: c_{X,i}\cdot\mathfrak{u}, \quad i \ge 2, \tag{3.5b}$$

$$\|\varDelta_{\boldsymbol{x}}\|_{\mathrm{F}} \le \sqrt{\sum_{i=2}^{n} c_{X,i}^2}\cdot\mathfrak{u} =: c_{v,\boldsymbol{x}}\cdot\mathfrak{u}. \tag{3.5c}$$

Proof. We will show (3.5a) only. Both (3.5b) and (3.5c) are consequences of (3.5a) and Lemma 3.1 together with $\|F\|_2 \le \|F\|_{\mathrm{F}}$. Note that (3.5a) is true for $i = 2, 3$. Assume now it also holds for i, and consider $\delta\boldsymbol{v}_{i+1}$. By (3.3),

$$\begin{aligned}\delta\boldsymbol{v}_{i+1} &= X^i\boldsymbol{e} - \widehat{Q}\widehat{H}^i\boldsymbol{e}_1 = X^i\boldsymbol{e} - (\widehat{Q}\widehat{H})\widehat{H}^{i-1}\boldsymbol{e}_1\\ &= X^i\boldsymbol{e} - (X\widehat{Q} - \hat{\beta}_{n+1}\hat{\boldsymbol{q}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} - F)\widehat{H}^{i-1}\boldsymbol{e}_1\\ &= X(X^{i-1}\boldsymbol{e} - \widehat{Q}\widehat{H}^{i-1}\boldsymbol{e}_1) + F\widehat{H}^{i-1}\boldsymbol{e}_1\\ &= X\delta\boldsymbol{v}_i + F\widehat{H}^{i-1}\boldsymbol{e}_1.\end{aligned}$$

The relation (3.5a) then follows by induction. $\square$

Now, we consider step 2 of Algorithm 1. With computed $\widehat{Q} \in$ `FPN`, output $\hat{\boldsymbol{d}} \in$ `FPN` is an approximation of $\widehat{Q}^{\dagger}\hat{\boldsymbol{f}}$, that is, $\hat{\boldsymbol{d}}$ is the computed solution to

$$\boldsymbol{d} = \operatorname*{argmin}_{\boldsymbol{z}\in\mathbb{R}^n}\|\hat{\boldsymbol{f}} - \widehat{Q}\boldsymbol{z}\|_2. \tag{3.6}$$

It is known (see, e.g., [3, Remark 2.4.8]) that any standard method for the least squares problem (3.6) is normwise backward stable. In particular, the computed $\hat{\boldsymbol{d}}$ is indeed the exact solution of a slightly perturbed one, namely,

$$\hat{\boldsymbol{d}} = \operatorname*{argmin}_{\boldsymbol{z}\in\mathbb{R}^n}\|(\hat{\boldsymbol{f}} + \delta\boldsymbol{f}) - (\widehat{Q} + \delta\widehat{Q})\boldsymbol{z}\|_2, \tag{3.7}$$

where

$$\|\delta\widehat{Q}\|_2 \le \mathfrak{u}\cdot c_{\mathrm{ls}}\sqrt{n}\|\widehat{Q}\|_2, \quad \|\delta\hat{\boldsymbol{f}}\|_2 \le \mathfrak{u}\cdot c_{\mathrm{ls}}\|\hat{\boldsymbol{f}}\|_2, \tag{3.8}$$

with $c_{\mathrm{ls}} = (6N - 3n + 41)n$ which often overestimates the error due to an artifact of analysis. We have the following key lemma.

Lemma 3.3. *Let $\hat{\boldsymbol{f}} = \boldsymbol{f} + \boldsymbol{\epsilon}_f^0$ where $\boldsymbol{f} = V_{\boldsymbol{x}}\boldsymbol{a}$, $\tilde{\boldsymbol{a}} = \widetilde{R}^{-1}\hat{\boldsymbol{d}}$ where $\widetilde{R}$ is given by (3.4), and $\boldsymbol{r}_a := \widetilde{R}(\boldsymbol{a} - \tilde{\boldsymbol{a}})$. Then*

$$\|\boldsymbol{r}_a\|_2 \le \frac{c_a\mathfrak{u} + 2\|\boldsymbol{\epsilon}_f^0\|_2}{\sigma_{\min}(\widehat{Q})}, \tag{3.9}$$

where $\sigma_{\min}(\widehat{Q})$ is the smallest singular value of $\widehat{Q}$,

$$c_{\boldsymbol{a}} = 2c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2 + c_{\mathrm{ls}}(\sqrt{n}\|\widehat{Q}\|_2(\|\boldsymbol{d}\|_2 + \|\hat{\boldsymbol{d}}\|_2) + 2\|\hat{\boldsymbol{f}}\|_2), \tag{3.10}$$

$c_{\mathrm{ls}} = (6N - 3n + 41)n$ as in (3.8), and $c_{v,\boldsymbol{x}}$ is given in Lemma 3.2.

Proof. Recall (3.7). We have

$$\begin{aligned}\xi_{\mathrm{ls}} := \min_{\boldsymbol{z}\in\mathbb{R}^n} \|(\hat{\boldsymbol{f}}+\delta\boldsymbol{f}) - (\widehat{Q}+\delta\widehat{Q})\boldsymbol{z}\|_2 &\le \|(\hat{\boldsymbol{f}}+\delta\boldsymbol{f}) - (\widehat{Q}+\delta\widehat{Q})\boldsymbol{d}\|_2 \\ &\le \|\hat{\boldsymbol{f}} - \widehat{Q}\boldsymbol{d}\|_2 + \|\delta\boldsymbol{f}\|_2 + \|\delta\widehat{Q}\|_2\|\boldsymbol{d}\|_2 \\ &\le \epsilon_d + \mathfrak{u}\cdot c_{\mathrm{ls}}(\sqrt{n}\|\widehat{Q}\|_2\cdot\|\boldsymbol{d}\|_2 + \|\hat{\boldsymbol{f}}\|_2).\end{aligned}$$

Notice that

$$\begin{aligned}\epsilon_d := \|\hat{\boldsymbol{f}} - \widehat{Q}\boldsymbol{d}\|_2 &= \min_{\boldsymbol{z}} \|\hat{\boldsymbol{f}} - \widehat{Q}\boldsymbol{z}\|_2 \\ &= \min_{\boldsymbol{z}} \|\boldsymbol{f} + \boldsymbol{\epsilon}_f^0 - \widehat{Q}\boldsymbol{z}\|_2 \\ &\le \|\boldsymbol{f} + \boldsymbol{\epsilon}_f^0 - \widehat{Q}\widetilde{R}\boldsymbol{a}\|_2 \qquad (\text{by choosing a particular } \boldsymbol{z} = \widetilde{R}\boldsymbol{a}) \\ &\le \|(V_{\boldsymbol{x}} - \widetilde{V}_{\boldsymbol{x}})\boldsymbol{a}\|_2 + \|\boldsymbol{\epsilon}_f^0\|_2 \quad (\widetilde{V}_{\boldsymbol{x}} = \widehat{Q}\widetilde{R} \text{ as in (3.4)}) \\ &\le \mathfrak{u}\cdot c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2 + \|\boldsymbol{\epsilon}_f^0\|_2. \qquad (\text{by Lemma 3.2})\end{aligned}$$

Therefore,

$$\xi_{\mathrm{ls}} \le \mathfrak{u}\cdot c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2 + \|\boldsymbol{\epsilon}_f^0\|_2 + \mathfrak{u}\cdot c_{\mathrm{ls}}(\sqrt{n}\|\widehat{Q}\|_2 + \|\hat{\boldsymbol{f}}\|_2). \tag{3.11}$$

Note from (3.11) and

$$\xi_{\mathrm{ls}} = \|(\hat{\boldsymbol{f}}+\delta\boldsymbol{f}) - (\widehat{Q}+\delta\widehat{Q})\hat{\boldsymbol{d}}\|_2 \ge \|\boldsymbol{f} - \widehat{Q}\hat{\boldsymbol{d}}\|_2 - (\|\boldsymbol{\epsilon}_f^0\|_2 + \|\delta\boldsymbol{f}\|_2 + \|\delta\widehat{Q}\|_2\|\hat{\boldsymbol{d}}\|_2)$$

that

$$\begin{aligned}\|\boldsymbol{f} - \widehat{Q}\hat{\boldsymbol{d}}\|_2 &\le \xi_{\mathrm{ls}} + \|\boldsymbol{\epsilon}_f^0\|_2 + \|\delta\boldsymbol{f}\|_2 + \|\delta\widehat{Q}\|_2\|\hat{\boldsymbol{d}}\|_2 \\ &\le \mathfrak{u}\cdot c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2 + 2\|\boldsymbol{\epsilon}_f^0\|_2 + \mathfrak{u}\cdot c_{\mathrm{ls}}(\sqrt{n}\|\widehat{Q}\|_2(\|\boldsymbol{d}\|_2 + \|\hat{\boldsymbol{d}}\|_2) + 2\|\hat{\boldsymbol{f}}\|_2).\end{aligned} \tag{3.12}$$

On the other hand, we have

$$\begin{aligned}\|\boldsymbol{f} - \widehat{Q}\hat{\boldsymbol{d}}\|_2 &= \|V_{\boldsymbol{x}}\boldsymbol{a} - \widetilde{V}_{\boldsymbol{x}}\tilde{\boldsymbol{a}}\|_2 \\ &= \|V_{\boldsymbol{x}}\boldsymbol{a} - \widetilde{V}_{\boldsymbol{x}}\boldsymbol{a} + \widetilde{V}_{\boldsymbol{x}}\boldsymbol{a} - \widetilde{V}_{\boldsymbol{x}}\tilde{\boldsymbol{a}}\|_2 \\ &\ge \|\widehat{Q}\boldsymbol{r}_a\|_2 - \|\varDelta_{\boldsymbol{x}}\boldsymbol{a}\|_2 \qquad (\widetilde{V}_{\boldsymbol{x}} = \widehat{Q}\widetilde{R} \text{ given in (3.4)}) \\ &\ge \|\widehat{Q}\boldsymbol{r}_a\|_2 - \mathfrak{u}\cdot c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2,\end{aligned}$$

which combining with (3.12) gives

$$\begin{aligned}\sigma_{\min}(\widehat{Q})\cdot\|\boldsymbol{r}_a\|_2 &\le \|\widehat{Q}\boldsymbol{r}_a\|_2 \\ &\le \mathfrak{u}\cdot 2c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2 + 2\|\boldsymbol{\epsilon}_f^0\|_2 + \mathfrak{u}\cdot c_{\mathrm{ls}}(\sqrt{n}\|\widehat{Q}\|_2(\|\boldsymbol{d}\|_2 + \|\hat{\boldsymbol{d}}\|_2) + 2\|\hat{\boldsymbol{f}}\|_2),\end{aligned}$$

as expected. ☐

For the factor $c_{\boldsymbol{a}}$ in (3.10), since in general $\boldsymbol{d} \approx \hat{\boldsymbol{d}}$, we have

$$c_{\boldsymbol{a}} \approx 2c_{v,\boldsymbol{x}}\|\boldsymbol{a}\|_2 + 2c_{\mathrm{ls}}(\sqrt{n}\|\widehat{Q}\|_2\cdot\|\hat{\boldsymbol{d}}\|_2 + \|\hat{\boldsymbol{f}}\|_2), \tag{3.13}$$

which can be computed with the data from Algorithm 1.

3.2. **Rounding error analysis in the evaluation stage.** We next analyze the rounding errors occurring in steps 3 and 4 of Algorithm 1 which involve new nodes $\{s_j\}_{j=1}^M$.

In step 3, we first compute $\widehat{W}$ with S and $\widehat{H}$ whose entries are floating point numbers. Similarly to Lemma 3.1, in this stage, we are concerned with the rounding errors during recursively calculating W accordingly to the equation $SW = WH + \beta_{n+1}\boldsymbol{w}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}$. For this purpose, we use the following standard models for vectors $\boldsymbol{v}$ and $\boldsymbol{z}$, scalar α, matrices A and B that are exactly representable in the working precision (see, e.g., [10, Section 2.7.8]).

$$\begin{aligned}
\mathrm{fl}(\boldsymbol{z}+\boldsymbol{v}) &= \boldsymbol{z}+\boldsymbol{v}+\boldsymbol{\delta},\ |\boldsymbol{\delta}| \le \mathfrak{u}\cdot(|\boldsymbol{z}|+|\boldsymbol{v}|),\\
\mathrm{fl}(\alpha A) &= \alpha A + E_1,\ |E_1| \le \mathfrak{u}\cdot|\alpha A|,\\
\mathrm{fl}(AB) &= AB + E_2,\ |E_2| \le \mathfrak{u}\cdot n|A|\cdot|B| + O(\mathfrak{u}^2),
\end{aligned}$$

where $|\cdot|$ on a vector/matrix is interpreted as taking entrywise absolute value.

Lemma 3.4. *Given diagonal $S \in \mathtt{FPN}$, $\hat{\beta}_{n+1} \in \mathtt{FPN}$ and tridiagonal $\widehat{H} \equiv [\hat{h}_{ij}] \in \mathtt{FPN}$ with $\hat{h}_{i,i+1} \ne 0$ for $1 \le i \le n$, computed $\widehat{W} \equiv [\hat{\boldsymbol{w}}_1,\dots,\hat{\boldsymbol{w}}_n]$ satisfies*

$$S\widehat{W} = \widehat{W}\widehat{H} + \hat{\beta}_{n+1}\hat{\boldsymbol{w}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} + G, \tag{3.14}$$

where $G = [\boldsymbol{g}_1, \boldsymbol{g}_2, \dots, \boldsymbol{g}_n]$ satisfies $\boldsymbol{g}_1 = 0$, and $\|\boldsymbol{g}_i\|_2 \le c_{w,i}\cdot\mathfrak{u} + O(\mathfrak{u}^2)$ and

$$c_{w,i} = 3\left\| |S|\cdot|\hat{\boldsymbol{w}}_i| + |\hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i}| + |\hat{h}_{i,i}\hat{\boldsymbol{w}}_i| \right\|_2, \quad 2 \le i \le n. \tag{3.15}$$

In particular, $\|G\|_{\mathrm{F}} \le c_w\cdot\mathfrak{u} + O(\mathfrak{u}^2)$, where $c_w = \sqrt{\sum_{i=2}^n c_{w,i}^2}$.

Proof. For $i = 1$, we always have $\hat{\boldsymbol{w}}_1 = \boldsymbol{e}_1$. For $i > 1$, it follows from the recursive relation $S\boldsymbol{w}_i = \boldsymbol{w}_{i-1}h_{i-1,i} + h_{i,i}\boldsymbol{w}_i + h_{i+1,i}\boldsymbol{w}_{i+1}$ that

$$\begin{aligned}
\hat{\boldsymbol{w}}_{i+1} &= \mathrm{fl}\left(\frac{S\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i} - \hat{h}_{i,i}\hat{\boldsymbol{w}}_i}{\hat{h}_{i+1,i}}\right) \qquad (3.16)\\
&= \frac{S\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i} - \hat{h}_{i,i}\hat{\boldsymbol{w}}_i + E_1 + E_2 + E_3 + E_4}{\hat{h}_{i+1,i}} + E_5,
\end{aligned}$$

where (by the diagonal matrix S)

$$\begin{aligned}
|E_1| &\le \mathfrak{u}|S|\cdot|\hat{\boldsymbol{w}}_i|,\\
|E_2| &\le \mathfrak{u}|\hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i}|,\\
|E_3| &\le \mathfrak{u}|\hat{h}_{i,i}\hat{\boldsymbol{w}}_i|,\\
|E_4| &\le \mathfrak{u}(|S|\cdot|\hat{\boldsymbol{w}}_i| + |\hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i}| + |\hat{h}_{i,i}\hat{\boldsymbol{w}}_i|) + O(\mathfrak{u}^2),\\
|E_5| &\le \mathfrak{u}\left|\frac{S\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i} - \hat{h}_{i,i}\hat{\boldsymbol{w}}_i}{\hat{h}_{i+1,i}}\right| + O(\mathfrak{u}^2).
\end{aligned}$$

Thus,

$$\begin{aligned}
|\boldsymbol{g}_i| &= |S\hat{\boldsymbol{w}}_i - \hat{\boldsymbol{w}}_{i-1}\hat{h}_{i-1,i} - \hat{h}_{i,i}\hat{\boldsymbol{w}}_i - \hat{h}_{i+1,i}\hat{\boldsymbol{w}}_{i+1}|\\
&\le |E_1| + |E_2| + |E_3| + |E_4| + |\hat{h}_{i+1,i}|\cdot|E_5| + O(\mathfrak{u}^2)\\
&\le c_{w,i}\cdot\mathfrak{u} + O(\mathfrak{u}^2),
\end{aligned}$$

where $c_{w,i}$ is given by (3.15). ∎

Remark 3.1. The proof of Lemma 3.4 can be easily modified to cover the case when $\widehat{H}$ is an upper Hessenberg matrix. It starts by modifying the numerator in (3.16) to $S\hat{\boldsymbol{w}}_i - \sum_{j=1}^{i} \hat{\boldsymbol{w}}_j \hat{h}_{ji}$ and the rest of the proof accordingly.

Lastly, we analyze the forward rounding errors in computing $\boldsymbol{y} = V_{\boldsymbol{s}}\boldsymbol{a}$:

$$\boldsymbol{\epsilon_y} = \boldsymbol{y} - \mathrm{fl}(\widehat{W}\hat{\boldsymbol{d}}) = V_{\boldsymbol{s}}\boldsymbol{a} - \widehat{W}\hat{\boldsymbol{d}} + E, \tag{3.17}$$

where $|E| \le \mathfrak{u}2n|\widehat{W}| \cdot |\hat{\boldsymbol{d}}|$. Noting in Lemma 3.3, we have defined $\tilde{\boldsymbol{a}} = \widetilde{R}^{-1}\hat{\boldsymbol{d}}$, and thus

$$\begin{aligned}
\|\boldsymbol{\epsilon_y}\|_2 &= \|V_{\boldsymbol{s}}\boldsymbol{a} - \widehat{W}\hat{\boldsymbol{d}} + E\|_2 \\
&\le \|V_{\boldsymbol{s}}\boldsymbol{a} - \widehat{W}\hat{\boldsymbol{d}}\|_2 + \|E\|_2 \\
&= \|V_{\boldsymbol{s}}\boldsymbol{a} - \widetilde{V}_{\boldsymbol{s}}\boldsymbol{a} + \widetilde{V}_{\boldsymbol{s}}\boldsymbol{a} - \widetilde{V}_{\boldsymbol{s}}\tilde{\boldsymbol{a}}\|_2 + \|E\|_2 \qquad (\widetilde{V}_{\boldsymbol{s}} = \widehat{W}\widetilde{R} \text{ with } \widetilde{R} \text{ as in (3.4)}) \\
&\le \|(V_{\boldsymbol{s}} - \widetilde{V}_{\boldsymbol{s}})\boldsymbol{a}\|_2 + \|\widetilde{V}_{\boldsymbol{s}}(\boldsymbol{a} - \tilde{\boldsymbol{a}})\|_2 + \|E\|_2 \\
&\le \|\varDelta_{\boldsymbol{s}}\|_2 \cdot \|\boldsymbol{a}\|_2 + \|\widehat{W}\widetilde{R}(\boldsymbol{a} - \tilde{\boldsymbol{a}})\|_2 + \|E\|_2 \qquad (\varDelta_{\boldsymbol{s}} := V_{\boldsymbol{s}} - \widetilde{V}_{\boldsymbol{s}}) \\
&\le \|\varDelta_{\boldsymbol{s}}\|_2 \cdot \|\boldsymbol{a}\|_2 + \|\widehat{W}\|_2 \cdot \|\widetilde{R}(\boldsymbol{a} - \tilde{\boldsymbol{a}})\|_2 + \|E\|_2 \\
&= \|\varDelta_{\boldsymbol{s}}\|_2 \cdot \|\boldsymbol{a}\|_2 + \|\widehat{W}\|_2 \cdot \|\boldsymbol{r}_a\|_2 + \mathfrak{u} \cdot 2n\|\widehat{W}\|_2 \cdot \|\hat{\boldsymbol{d}}\|_2 \qquad (\|\boldsymbol{r}_a\|_2 \text{ as in (3.9)}).
\end{aligned} \tag{3.18}$$

In view of (3.18), we can finally have an upper bound for $\|\boldsymbol{\epsilon_y}\|_2$ if we can bound $\|\varDelta_{\boldsymbol{s}}\|_2$. This is done in the following lemma.

Lemma 3.5. *For $\widetilde{V}_{\boldsymbol{s}} = \widehat{W}\widetilde{R}$ with $\widetilde{R} = [\boldsymbol{e}_1, \widehat{H}\boldsymbol{e}_1, \ldots, \widehat{H}^{n-1}\boldsymbol{e}_1]$ given in (3.4), let*

$$\varDelta_{\boldsymbol{s}} := V_{\boldsymbol{s}} - \widetilde{V}_{\boldsymbol{s}} \equiv [\delta\boldsymbol{u}_1, \ldots, \delta\boldsymbol{u}_n].$$

We have $\delta\boldsymbol{u}_1 = 0$, $\delta\boldsymbol{u}_2 = G\boldsymbol{e}_1$ and

$$\delta\boldsymbol{u}_i = \left(S^{i-2}G + S^{i-3}G\widehat{H} + S^{i-4}G\widehat{H}^2 + \cdots + SG\widehat{H}^{i-3} + G\widehat{H}^{i-2}\right)\boldsymbol{e}_1, \; 3 \le i \le n, \tag{3.19}$$

where G is given in (3.14). As a result,

$$\|\delta\boldsymbol{u}_i\|_2 \le \left(c_w \cdot \sum_{j=0}^{i-2} \|S\|_2^j \cdot \|\widehat{H}\|_2^{i-j-2}\right)\mathfrak{u} + O(\mathfrak{u}^2) =: c_{S,i} \cdot \mathfrak{u} + O(\mathfrak{u}^2), \quad i \ge 2, \tag{3.20}$$

and $\|\varDelta_{\boldsymbol{s}}\|_{\mathrm{F}} \le \mathfrak{u} \cdot c_{v,\boldsymbol{s}} + O(\mathfrak{u}^2)$ where c_w is given in Lemma 3.4 and $c_{v,\boldsymbol{s}} = \sqrt{\sum_{i=2}^{n} c_{S,i}^2}$.

Proof. It can be proved in the same way as that for Lemma 3.2 upon replacing X and F by S and G, respectively. □

Finally, we are now able to bound the forward error $\|\boldsymbol{\epsilon_y}\|_2$ in (3.17).

Theorem 3.1. *In finite precision, computed vector $\hat{\boldsymbol{y}}$ by Algorithm 1 satisfies*

$$\|\boldsymbol{\epsilon_y}\|_2 = \|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2 \le \mathfrak{u} \cdot \left(c_{v,\boldsymbol{s}}\|\boldsymbol{a}\|_2 + \|\widehat{W}\|_2 \left[\frac{c_{\boldsymbol{a}} + 2\|\boldsymbol{\epsilon}_f^0\|_2/\mathfrak{u}}{\sigma_{\min}(\widehat{Q})} + 2n\|\hat{\boldsymbol{d}}\|_2\right]\right) + O(\mathfrak{u}^2), \tag{3.21}$$

where $c_{v,\boldsymbol{s}}$ and $c_{\boldsymbol{a}}$ are given in Lemma 3.5 and (3.10), respectively, $\boldsymbol{a}$ in (2.7) is the coefficient vector of the underlying polynomial in the monomial basis, $\boldsymbol{\epsilon}_f^0$ is given in (3.2), and $(\widehat{W}, \widehat{Q}, \hat{\boldsymbol{d}})$ are the computed quantities of $(W, Q, \boldsymbol{d})$.

Proof. It is a consequence of (3.18), Lemmas 3.3 and 3.5. $\square$

Remark 3.2. (1) It is noticed in (3.21) that no condition number of either Vandermonde matrix $V_{\boldsymbol{x}}$ or $V_{\boldsymbol{s}}$ is involved. By taking a closer look at the factor $c_{\boldsymbol{a}}$ in (3.10), it turns out the condition number $\kappa_2(\widehat{Q}) := \|\widehat{Q}\|_2\|\widehat{Q}^{\dagger}\|_2$ plays a role. As $\widehat{Q}$ is the computed "orthonormal matrix", Q with $Q^{\mathrm{H}}Q = NI_N$, of the Krylov subspace $\mathcal{K}_n(X, \boldsymbol{e})$, the orthogonality in $\widehat{Q}$ can be good initially, and gradually becomes worse as some of Ritz values (i.e., the eigenvalues of $\widehat{H}$) converge to ones of the true eigenvalues of X, i.e., some x_j [11, 23, 24], but here very rough orthogonality suffices as only the condition number $\kappa_2(\widehat{Q})$ is concerned, e.g., $\widehat{Q}^{\mathrm{H}}\widehat{Q} - NI_n \approx 10^{-2}$ still gives $\kappa_2(\widehat{Q}) \approx 1.02$.

(2) In (3.21), the factor $\frac{\|\boldsymbol{\epsilon}_f^0\|_2}{\mathfrak{u}}$ also plays a role. As we have mentioned in subsection 3.1, when the underlying $f(x) = \mathrm{p}_{n-1}(x)$, by (3.1), it holds that

$$\frac{|\epsilon_{f_j}^0|}{\mathfrak{u}} \le 2(n-1)\sum_{i=0}^{n-1}|a_i x_j^i|. \tag{3.22}$$

(3) The quantity $\|\boldsymbol{a}\|_2$ being not too large is important.

(4) Overall, (3.21) reveals that when Algorithm 1 can in general deliver y_j accurately.

4. **Chebyshev polynomial basis.**

For the case of approximation on an interval, in this section, we will present a variant of Algorithm 1 using a well-known orthogonal polynomial basis, namely, the Chebyshev polynomials of the 1st kind after a proper transformation.

The original Chebyshev polynomials of the 1st kind are defined as, for $m \ge 0$,

$$T_m(t) = \begin{cases} \cos(m \arccos t), & \text{for } |t| \le 1, \\ \frac{1}{2}\left(t + \sqrt{t^2-1}\right)^m + \frac{1}{2}\left(t - \sqrt{t^2-1}\right)^m, & \text{for } |t| \ge 1. \end{cases} \tag{4.1}$$

They satisfy recursively

$$\begin{aligned} &T_0(t) = 1,\ T_1(t) = t, \\ &T_{m+1}(t) = 2t\,T_m(t) - T_{m-1}(t) \quad \text{for } m \ge 1. \end{aligned}$$

Suppose the nodes $\{x_j\}_{j=1}^N$ and $\{s_i\}_{i=1}^M$ fall in the interval $[a,b]$, i.e., $x_j, s_i \in [a,b]$ for $1 \le j \le N$ and $1 \le i \le M$. We can perform a linear transformation

$$t(x) = \frac{x}{\omega} + \tau, \text{ with } \omega = \frac{b-a}{2} \text{ and } \tau = -\frac{b+a}{b-a}, \tag{4.2}$$

to translate $x \in [a,b]$ into $t \in [-1,1]$, and use the translated Chebyshev polynomials[2] in x defined by

$$\mathscr{T}_m(x;\omega,\tau) = T_m(x/\omega + \tau)$$

to span the fitting polynomial on $[a,b]$, namely

$$\mathbb{P}_{n-1}[x] = \left\{\sum_{i=0}^{n-1} a_i \mathscr{T}_i(x;\omega,\tau) \,:\, a_i \in \mathbb{R}\ \forall i\right\}.$$

[2]Although we present our extension in this section for real nodes $\{x_j\}$. It can be further extended to complex nodes that are enclosed in some ellipse for which transformed Chebyshev polynomials (see [26, Section 6.11.2], [9] and [14]) can be designed as previously done for solutions of linear systems of equations [26].

By the recursive relation on T_m, we have

$$\mathscr{T}_0(x;\omega,\tau)=1,\ \mathscr{T}_1(x;\omega,\tau)=x/\omega+\tau, \tag{4.3a}$$

$$\mathscr{T}_{m+1}(x;\omega,\tau)=2(x/\omega+\tau)\,\mathscr{T}_m(x;\omega,\tau)-\mathscr{T}_{m-1}(x;\omega,\tau)\quad\text{for } m\geq 1. \tag{4.3b}$$

The mth Chebyshev-Vandermonde matrix of order N with nodes $\{x_j\}_{j=1}^N$ is defined as

$$[\mathscr{T}_0(\boldsymbol{x};\omega,\tau),\mathscr{T}_1(\boldsymbol{x};\omega,\tau),\dots,\mathscr{T}_m(\boldsymbol{x};\omega,\tau)]\in\mathbb{R}^{N\times(m+1)}.$$

Recall $X=\operatorname{diag}(\boldsymbol{x})$ and $S=\operatorname{diag}(\boldsymbol{s})$. Define

$$\mathscr{X}=\begin{bmatrix}(2/\omega)X+2\tau I_N & -I_N\\ I_N & 0\end{bmatrix},\quad \boldsymbol{u}=\begin{bmatrix}\mathscr{T}_1(\boldsymbol{x};\omega,\tau)\\ \mathscr{T}_0(\boldsymbol{x};\omega,\tau)\end{bmatrix}, \tag{4.4a}$$

$$\mathscr{S}=\begin{bmatrix}(2/\omega)S+2\tau I_M & -I_M\\ I_M & 0\end{bmatrix}. \tag{4.4b}$$

It can be verified that

$$[\boldsymbol{u},\mathscr{X}\boldsymbol{u},\dots,\mathscr{X}^{n-1}\boldsymbol{u}]=\begin{bmatrix}\mathscr{T}_1(\boldsymbol{x};\omega,\tau) & \mathscr{T}_2(\boldsymbol{x};\omega,\tau) & \dots & \mathscr{T}_n(\boldsymbol{x};\omega,\tau)\\ \mathscr{T}_0(\boldsymbol{x};\omega,\tau) & \mathscr{T}_1(\boldsymbol{x};\omega,\tau) & \dots & \mathscr{T}_{n-1}(\boldsymbol{x};\omega,\tau)\end{bmatrix}=:\mathscr{K} \tag{4.5}$$

whose bottom half

$$\mathscr{V}_{\boldsymbol{x}}=[\mathscr{T}_0(\boldsymbol{x};\omega,\tau),\mathscr{T}_1(\boldsymbol{x};\omega,\tau),\dots,\mathscr{T}_{n-1}(\boldsymbol{x};\omega,\tau)]\in\mathbb{R}^{N\times n} \tag{4.6}$$

is the $(n-1)$st Chebyshev-Vandermonde matrix of order N with nodes $\{x_j\}_{j=1}^N$.

Using the basis $\{\mathscr{T}_i(x;\omega,\tau)\}_{i=0}^{n-1}$, we express the fitting polynomial p_{n-1} by

$$\mathrm{p}_{n-1}(x)=\sum_{i=0}^{n-1}a_i\mathscr{T}_i(x;\omega,\tau), \tag{4.7}$$

and thus $\boldsymbol{f}\approx\mathscr{V}_{\boldsymbol{x}}\boldsymbol{a}$. At new nodes $\{s_i\}$, we will have to evaluate

$$\boldsymbol{y}=[\mathrm{p}_{n-1}(s_1),\dots,\mathrm{p}_{n-1}(s_M)]^{\mathrm{T}}=\mathscr{V}_{\boldsymbol{s}}\boldsymbol{a}, \tag{4.8}$$

where $\mathscr{V}_{\boldsymbol{s}}=[\mathscr{T}_0(\boldsymbol{s};\omega,\tau),\mathscr{T}_1(\boldsymbol{s};\omega,\tau),\dots,\mathscr{T}_{n-1}(\boldsymbol{s};\omega,\tau)]\in\mathbb{R}^{M\times n}$. The evaluation can be done in the same way as in Algorithm 1; that is, we first compute a well-condition basis Q for $\mathscr{V}_{\boldsymbol{x}}$ and obtain a QR-type decomposition $\mathscr{V}_{\boldsymbol{x}}=QR$, from which a QR-type decomposition $\mathscr{V}_{\boldsymbol{s}}=WR$ can be computed for new Chebyshev-Vandermande matrix $\mathscr{V}_{\boldsymbol{s}}$. For this purpose, we have two different implementations.

4.1. **Chebyshev with Arnoldi.**

Apply the Arnoldi process (see, e.g., [7, 11, 23]) with $\mathscr{X}$ on $[\boldsymbol{x}^{\mathrm{T}},\boldsymbol{1}_N^{\mathrm{T}}]^{\mathrm{T}}$, assuming no breakdown occurs, to get

$$\mathscr{X}U=UH+\beta_{n+1}\boldsymbol{u}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}\quad\text{with}\quad U\boldsymbol{e}_1=[\boldsymbol{x}^{\mathrm{T}},\boldsymbol{1}_N^{\mathrm{T}}]^{\mathrm{T}}, \tag{4.9}$$

such that[3] $U^{\mathrm{T}}U=\gamma_0^2 I_{2N}$ where $\gamma_0=\|[\boldsymbol{x}^{\mathrm{T}},\boldsymbol{1}_N^{\mathrm{T}}]\|_2$. By (4.5), we know that the columns of $Q:=U_{(N+1:2N,:)}$ form a basis of $\mathcal{R}(\mathscr{V}_{\boldsymbol{x}})$ for $\mathscr{V}_{\boldsymbol{x}}$ as in (4.6). Now, according to Theorem 2.1, $\mathscr{K}=UR$ where $R=[\boldsymbol{e}_1,H\boldsymbol{e}_1,\dots,H^{n-1}\boldsymbol{e}_1]\in\mathbb{R}^{2N\times n}$ is an upper triangular matrix. This yields $\mathscr{V}_{\boldsymbol{x}}=QR$. Similarly, by Theorem 2.1 again, we can construct matrix $Z\in\mathbb{R}^{2M\times n}$ based on the relation

$$\mathscr{S}Z=ZH+\beta_{n+1}\boldsymbol{z}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}\quad\text{with}\quad Z\boldsymbol{e}_1=[\boldsymbol{s}^{\mathrm{T}},\boldsymbol{e}^{\mathrm{T}}]^{\mathrm{T}}.$$

[3]This is done by slightly modifying the usual Arnoldi process in normalizing each new Arnoldi vector to have norm γ_0 instead of 1.

```
Algorithm 2 PFECvA: Polynomial fitting and evaluation with Chebyshev via Arnoldi
Input: interpolating points $\{(x_j, f_j)\}_{j=1}^N$, new nodes $\{s_j\}_{j=1}^M$ to be evaluated at,
    integer $0 \le n \le N$.
Output: values $y_j = \mathrm{p}_{n-1}(s_j)$ for $1 \le j \le M$ of the best fitting polynomial of
    degree $n-1$:
        $\mathrm{p}_{n-1} = \underset{\mathrm{p}\in\mathbb{P}_{n-1}}{\operatorname{argmin}} \|\boldsymbol{f} - \mathrm{p}_{n-1}(\boldsymbol{x})\|_2,$
    where $\boldsymbol{x} = [x_1, \ldots, x_N]^{\mathrm{T}}$ and $\boldsymbol{f} = [f_1, \ldots, f_N]^{\mathrm{T}}$.
                        Construction stage
1: call the Arnoldi or SOAR [1] process with $\mathscr{X}$ on $U\boldsymbol{e}_1 = [\boldsymbol{x}^{\mathrm{T}}, \mathbf{1}_N^{\mathrm{T}}]^{\mathrm{T}}$ to produce
   $\mathscr{X}U = UH + \beta_{n+1}\boldsymbol{u}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}$, where $H \in \mathbb{R}^{n\times n}$ is upper-Hessenberg;
2: compute $\boldsymbol{d} = \operatorname{argmin}_{\boldsymbol{z}} \|\boldsymbol{f} - Q\boldsymbol{z}\|_2$, where $Q = U_{(N+1:2N,:)}$;
                        Evaluation stage
3: recursively compute $Z$, one column at a time, such that $\mathscr{S}Z = ZH +$
   $\beta_{n+1}\boldsymbol{z}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}$, where $\boldsymbol{s} = [s_1, \ldots, s_M]^{\mathrm{T}}$ and $Z\boldsymbol{e}_1 = [\boldsymbol{s}^{\mathrm{T}}, \mathbf{1}_M^{\mathrm{T}}]^{\mathrm{T}}$;
4: return $\boldsymbol{y} = [y_1, \ldots, y_M]^{\mathrm{T}} = W\boldsymbol{d}$, where $W = Z_{(M+1:2M,:)}$.
```

$$\begin{tikzcd}
 & \mathscr{V}_{\boldsymbol{x}} = QR & \mathscr{V}_{\boldsymbol{s}} = WR & \\
\boldsymbol{x} \arrow[r, "\mathscr{X}U = UH + \beta_{n+1}\boldsymbol{u}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}", "\substack{\text{step 1: Arnoldi/SOAR} \\ Q = U_{(N+1:2N,:)}}"'] & (Q, H) \arrow[u, dashed] \arrow[r] \arrow[d, "\text{step 2}"] & W \arrow[u, dashed] \arrow[d, "\text{step 4}"] & (\boldsymbol{s}, H) \arrow[l, "\mathscr{S}Z = ZH + \beta_{n+1}\boldsymbol{z}_{n+1}\boldsymbol{e}_n^{\mathrm{T}}"', "\text{step 3.}\quad W = Z_{(N+1:2N,:)}"] \\
 & \boldsymbol{d} = Q^{\dagger}\boldsymbol{f} \arrow[r] & \boldsymbol{y} = W\boldsymbol{d} &
\end{tikzcd}$$

FIGURE 4.1. Diagram illustration of Algorithm 2 where H at step 3 is from step 1.

Consequently, we have $\mathscr{V}_{\boldsymbol{s}} = WR$ where $W = Z_{(M+1:2M,:)}$. Analogously to (2.8) and (2.9), the least squares problem

$$\mathrm{p}_{n-1} = \underset{\mathrm{p_{n-1}}\in\mathbb{P}_{n-1}}{\operatorname{argmin}} \|\boldsymbol{f} - \mathrm{p_{n-1}}(\boldsymbol{x})\|_2, \tag{4.10}$$

gives a polynomial whose coefficient vector $\boldsymbol{a}$ in the Chebyshev polynomial basis is related to $\boldsymbol{d} = Q^{\dagger}\boldsymbol{f} = R\boldsymbol{a}$. Thus, $\boldsymbol{y} = \mathscr{V}_{\boldsymbol{s}}\boldsymbol{a} = W(R\boldsymbol{a}) =: W\boldsymbol{d}$. The overall algorithmic procedure is summarized in Algorithm 2, where the SOAR process to be explain in the next subsection is also mentioned as another way to yield an equation of the same form as (4.9) but with a different orthogonality constraint. A diagram illustration of Algorithm 2 is given in Figure 4.1.

In finite precision arithmetic, we can follow the same argument for establishing Theorem 3.1. Due to similarity and artificial overestimation of constant factors in terms of some low degree polynomials in n, N, and M in the upper bound, we will not duplicate the effort in detailing those factors, but rather state critical sensitivity factors that dictate the accuracy of computed $\boldsymbol{y}$ in (4.8). We will use again the notation convention: $\hat{\xi}$ to denote the computed quantity of ξ. Again for the Arnoldi process (see, e.g., [22, Section 3.2] [15, Section 7.1]), a counterpart of

(3.3) is

$$\mathscr{X}\widehat{U} = \widehat{U}\widehat{H} + \hat{\beta}_{n+1}\hat{\boldsymbol{u}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} + F, \ \|F\|_2 = O(\|\mathscr{X}\|_2)\mathfrak{u}.$$

This leads to a counterpart of Lemma 3.2, roughly,

$$\|\varDelta_x\|_{\mathrm{F}} = \|\mathscr{V}_{\boldsymbol{x}} - \widetilde{\mathscr{V}}_{\boldsymbol{x}}\|_{\mathrm{F}} = O(\|\mathscr{X}^{n-1}\|_2, \|\widehat{H}^{n-1}\|_2, \|\mathscr{X}\|_2, \|\widehat{H}\|_2)\mathfrak{u},$$

where $\widetilde{\mathscr{V}}_{\boldsymbol{x}} = \widehat{Q}[\boldsymbol{e}_1, \widehat{H}\boldsymbol{e}_1, \dots, \widehat{H}^{n-1}\boldsymbol{e}_1]$. A counterpart of (3.14) reads as

$$\mathscr{S}\widehat{Z} = \widehat{Z}\widehat{H} + \hat{\beta}_{n+1}\hat{\boldsymbol{z}}_{n+1}\boldsymbol{e}_n^{\mathrm{T}} + G,$$

where $\|G\|_{\mathrm{F}} = O(\|\mathscr{S}\|_2, \|\widehat{H}\|_2, \|\widehat{Z}\|_2)\mathfrak{u}$. Thus, for the difference

$$\varDelta_s = \mathscr{V}_{\boldsymbol{s}} - \widetilde{\mathscr{V}}_{\boldsymbol{s}} = \mathscr{V}_{\boldsymbol{s}} - \hat{W}[\boldsymbol{e}_1, \widehat{H}\boldsymbol{e}_1, \dots, \widehat{H}^{n-1}\boldsymbol{e}_1],$$

similarly to Lemma 3.5, we have

$$\|\varDelta_s\|_2 = \|\mathscr{V}_{\boldsymbol{s}} - \widetilde{\mathscr{V}}_{\boldsymbol{s}}\|_2 = O(\|\mathscr{S}^{n-1}\|_2, \|\widehat{H}^{n-1}\|_2, \|\mathscr{S}\|_2, \|\widehat{H}\|_2, \|\widehat{Z}\|_2)\mathfrak{u}.$$

Consequently, the accuracy in computed $\boldsymbol{y}$ is governed by

$$\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2 \le \mathfrak{u}\cdot\vartheta\left(\|\mathscr{X}^n\|_2, \|\mathscr{S}^n\|_2, \|\widehat{H}^n\|_2, \|\widehat{Z}\|_2, \frac{\|\boldsymbol{\epsilon}_f^0\|_2/\mathfrak{u}}{\sigma_{\min}(\widehat{Q})}, \|\boldsymbol{a}\|_2, \|\hat{\boldsymbol{d}}\|_2, \|\hat{\boldsymbol{f}}\|_2, \kappa_2(\widehat{Q})\right) + O(\mathfrak{u}^2), \tag{4.11}$$

where ϑ is a low degree polynomial of n, N, and M.

Compared with (3.21), the new way of evaluating $\boldsymbol{y}$ is affected by $\|\mathscr{X}^n\|_2$, $\|\mathscr{S}^n\|_2$ and $\kappa_2(\widehat{Q})$. Fortunately, Theorem 4.1 below reveals that $\|\mathscr{X}^n\|_2$, $\|\mathscr{S}^n\|_2$ can grows at most linearly with respect to n, whereas previously $\|X^n\|_2$, $\|S^n\|_2$ grows exponentially at the rate $(\max_j |x_j|)^n$ and $(\max_j |s_j|)^n$, respectively.

Lemma 4.1. *For $t \in [-1,1]$ and integer $n \ge 0$,*

$$\left\| \begin{bmatrix} 2t & -1 \\ 1 & 0 \end{bmatrix}^n \right\|_2 \le 2n+1. \tag{4.12}$$

Proof. The eigenvalues of $J(t) := \begin{bmatrix} 2t & -1 \\ 1 & 0 \end{bmatrix}$ are $\lambda_\pm = t \pm \iota\sqrt{1-t^2} = e^{\pm\iota\theta}$ where $\iota = \sqrt{-1}$ and $\cos\theta = t$. Furthermore, if $|t| < 1$, then $J(t)$ has the following eigendecomposition:

$$\begin{aligned} J(t) &= \begin{bmatrix} e^{\iota\theta} & e^{-\iota\theta} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} e^{\iota\theta} & \\ & e^{-\iota\theta} \end{bmatrix} \begin{bmatrix} e^{\iota\theta} & e^{-\iota\theta} \\ 1 & 1 \end{bmatrix}^{-1} \\ &= \begin{bmatrix} e^{\iota\theta} & e^{-\iota\theta} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} e^{\iota\theta} & \\ & e^{-\iota\theta} \end{bmatrix} \left(\frac{1}{2\iota\sin\theta} \begin{bmatrix} 1 & -e^{-\iota\theta} \\ -1 & e^{\iota\theta} \end{bmatrix} \right). \end{aligned}$$

Thus for any interger $n \ge 0$,

$$\begin{aligned} [J(t)]^n &= \frac{1}{2\iota\sin\theta} \begin{bmatrix} e^{\iota\theta} & e^{-\iota\theta} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} e^{\iota n\theta} & \\ & e^{-\iota n\theta} \end{bmatrix} \begin{bmatrix} 1 & -e^{-\iota\theta} \\ -1 & e^{\iota\theta} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\sin(n+1)\theta}{\sin\theta} & -\frac{\sin n\theta}{\sin\theta} \\ \frac{\sin n\theta}{\sin\theta} & -\frac{\sin(n-1)\theta}{\sin\theta} \end{bmatrix}. \end{aligned}$$

Noticing that

$$\left|\frac{\sin(n+1)\theta}{\sin\theta}\right| = \left|\frac{\sin n\theta}{\sin\theta}\cos\theta + \cos n\theta\right| \le \left|\frac{\sin n\theta}{\sin\theta}\right| + 1 \le \cdots \le n+1,$$

we have

$$\|[J(t)]^n\|_2 \le \sqrt{\|[J(t)]^n\|_1 \|[J(t)]^n\|_\infty} \le 2n+1.$$

Finally, letting $t \to 1$ or $t \to -1$ leads to (4.12) for $t = \pm 1$. □

Theorem 4.1. *For $x_i \in [a,b]$, $i = 1,2,\dots,N$, $\mathscr{X}$ given in (4.4) satisfies*

$$\|\mathscr{X}^n\|_2 \le 2n+1.$$

Proof. For $\mathscr{X}$ of (4.4), there is a permutation matrix P so that

$$P\mathscr{X}P^{\mathrm{T}} = \mathrm{diag}(J_1,\dots,J_N),$$

where $J_i = \begin{bmatrix} 2t_i & -1 \\ 1 & 0 \end{bmatrix}$ and $t_i = x_i/\omega + \tau$. It can be verified that $t_i \in [-1,1]$ for all i. Hence by Lemma 4.1, we have $\|J_i^n\|_2 \le 2n+1$ for all i and

$$\|\mathscr{X}^n\|_2 = \|(P\mathscr{X}P^{\mathrm{T}})^n\|_2 = \max_i \|J_i^n\|_2 \le 2n+1,$$

as was to be shown. □

4.2. **Chebyshev with SOAR.** We next introduce the SOAR process [1, 19] to generate a basis matrix of $\mathcal{R}(\mathscr{V}_{\boldsymbol{x}})$ in (4.6):

$$\mathcal{R}(\mathscr{V}_{\boldsymbol{x}}) = \mathcal{R}([\mathscr{T}_0(\boldsymbol{x};\omega,\tau), \mathscr{T}_1(\boldsymbol{x};\omega,\tau), \dots, \mathscr{T}_{n-1}(\boldsymbol{x};\omega,\tau)]).$$

The recursive formulas in (4.3), the structure of $\mathscr{X}$ in (4.4), and the matrix in (4.5) together make SOAR, an Arnoldi-type procedure, a natural way to do so. There are a couple of variations. In [19], a backward stable and memory-efficiency procedure called TOAR is proposed. For our purpose, since N and n are generally not large, the original version of SOAR [1] suffices. It recursively computes U one column at a time according to (4.9). Denote by

$$U = \begin{bmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_n \\ \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \end{bmatrix}, \quad H = [h_{ij}]. \tag{4.13}$$

What follows is a modified SOAR in that, besides (4.9), it is also required that

$$[\boldsymbol{q}_1, \boldsymbol{q}_2, \dots, \boldsymbol{q}_n]^{\mathrm{T}} [\boldsymbol{q}_1, \boldsymbol{q}_2, \dots, \boldsymbol{q}_n] = N I_n.$$

It goes as follows. Initially, the first column of U is set to $[\boldsymbol{x}^{\mathrm{T}}, \boldsymbol{1}_N^{\mathrm{T}}]^{\mathrm{T}}$. After k steps, we have computed the first k columns of U and H. By (4.9), we have

$$\mathscr{X} \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{q}_k \end{bmatrix} = \sum_{j=1}^{k} h_{j,k} \begin{bmatrix} \boldsymbol{p}_j \\ \boldsymbol{q}_j \end{bmatrix} + h_{k+1,k} \begin{bmatrix} \boldsymbol{p}_{k+1} \\ \boldsymbol{q}_{k+1} \end{bmatrix},$$

or equivalently

$$h_{k+1,k}\boldsymbol{q}_{k+1} = \boldsymbol{p}_k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{q}_j, \tag{4.14a}$$

$$h_{k+1,k}\boldsymbol{p}_{k+1} = [(2/\omega)X + 2\tau I_N]\boldsymbol{p}_k - \boldsymbol{q}_k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{p}_j. \tag{4.14b}$$

Noting the orthogonality among $\boldsymbol{q}_j$: $\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{q}_j = 0$ for $i \ne j$ and $\boldsymbol{q}_i^{\mathrm{T}}\boldsymbol{q}_i = N$, we have from (4.14a)

$$h_{j,k} = \frac{1}{N}\boldsymbol{q}_j^{\mathrm{T}}\boldsymbol{p}_k \quad \text{for } 1 \le j \le k,$$

$$\tilde{\boldsymbol{q}}_{k+1} = \boldsymbol{p}_k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{q}_j, \quad h_{k+1,k} = \frac{\|\tilde{\boldsymbol{q}}_{k+1}\|_2}{\sqrt{N}}, \quad \boldsymbol{q}_{k+1} = \frac{1}{h_{k+1,k}}\tilde{\boldsymbol{q}}_{k+1},$$

and then from (4.14b)

$$\tilde{\boldsymbol{p}}_{k+1} = [(2/\omega)X + 2\tau I_N]\boldsymbol{p}_k - \boldsymbol{q}_k - \sum_{j=1}^{k} h_{j,k}\boldsymbol{p}_j, \quad \boldsymbol{p}_{k+1} = \frac{1}{h_{k+1,k}}\tilde{\boldsymbol{p}}_{k+1},$$

provided $h_{k+1,k} > 0$. The complete procedure is summarized in Algorithm 3, and the corresponding polynomial evaluation method is presented in Algorithm 2 together with the diagram illustration in Figure 4.1.

Algorithm 3 The SOAR process [1]

```
Input: $X$, $\omega$, $\tau$;
Output: $U$ and $H$ as in (4.13).
 1: Let $\boldsymbol{q}_1 = \mathbf{1}_N \in \mathbb{R}^N$ and $\boldsymbol{p}_1 = \boldsymbol{x}$;
 2: for $k = 1, 2, \ldots, n$ do
 3:     $\tilde{\boldsymbol{p}} = [(2/\omega)X + 2\tau I]\boldsymbol{p}_k - \boldsymbol{q}_k$, $\tilde{\boldsymbol{q}} = \boldsymbol{p}_k$;
 4:     for $i = 1, 2, \ldots, k$ do
 5:         % modified Gram-Schmidt
 6:         $h_{i,k} = \boldsymbol{p}_k^{\mathrm{T}}\boldsymbol{q}_i/N$;
 7:         $\tilde{\boldsymbol{p}} = \tilde{\boldsymbol{p}} - h_{i,k}\boldsymbol{p}_i$, $\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{q}} - h_{i,k}\boldsymbol{q}_i$;
 8:     end for
 9:     $h_{k+1,k} = \|\tilde{\boldsymbol{q}}\|_2/\sqrt{N}$;
10:     stop if $h_{k+1,k} = 0$;
11:     $\boldsymbol{p}_{k+1} = \tilde{\boldsymbol{p}}/h_{k+1,k}$, $\boldsymbol{q}_{k+1} = \tilde{\boldsymbol{q}}/h_{k+1,k}$;
12: end for
```

Omitting all detail, in finite precision, as before, we can have an error estimation:

$$\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2 \le \mathfrak{u} \cdot \vartheta\left(\|\widehat{H}^n\|_2, \|\widehat{Z}\|_2, \frac{\|\epsilon_f^0\|_2}{\mathfrak{u}}, \|\boldsymbol{a}\|_2, \|\hat{\boldsymbol{d}}\|_2, \|\hat{\boldsymbol{f}}\|_2\right) + O(\mathfrak{u}^2), \tag{4.15}$$

where ϑ is a low degree polynomial of n, N, and M.

5. Numerical experiments.

In this section, we will conduct numerical tests for two purposes: 1) demonstrate the sharpness of the upper bound ϵ_{up} in (3.21), for approximating $f(s_j)$ in finite precision arithmetic, and 2) evaluate the benefit of using the Chebyshev polynomial basis. Our numerical tests are carried out in MATLAB 2018Ra on a MacBook Pro with 16 GB 3733 MHz LPDDR4X. The IEEE double precision is used with the unit machine roundoff $\mathfrak{u} = 2^{-53} \approx 1.1 \times 10^{-16}$.

To guarantee that we can evaluate $f(x)$ accurately in the working precision for the purpose of testing, we choose the underlying function $f(x)$ as a polynomial in the product form with zeros at given $\{\alpha_i\}_{i=1}^{n-1}$, i.e.,

$$f(x) = (x - \alpha_1)\cdots(x - \alpha_{n-1}), \ \ \alpha_i \in \mathtt{FPN}. \tag{5.1}$$

For any $x \in \mathtt{FPN}$, we can be assured that the error $\boldsymbol{\epsilon}_f^0$ defined in (3.2) is of $O(\mathfrak{u})$, and thus $\frac{\|\boldsymbol{\epsilon}_f^0\|_2}{\mathfrak{u}}$ in (3.21) is $O(1)$. In fact, for any $x \in \mathtt{FPN}$, if $(n-1)\mathfrak{u} \le 0.01$, then

[13, Lemma 3.4]

$$\mathrm{fl}(f(x)) = (x-\alpha_1)\cdots(x-\alpha_{n-1})\cdot\prod_{i=1}^{n-1}(1+\delta_i) := f(x)\,(1+1.01(n-1)\delta),$$

where $|\delta_i| \le \mathfrak{u}$ for $i = 1, 2, \dots, n-1$ and $|\delta| \le \mathfrak{u}$. Therefore, in our numerical tests, we simply use $\mathrm{fl}(f(x))$ as the value $f(x)$ at x.

Example 5.1. We set the interval $[a, b] = [-1, 1]$ and choose the roots $\{\alpha_i\}_{i=1}^{n-1}$ of $f(x)$ in (5.1) uniformly distributed in $[-1, 1]$. Furthermore, we let $n = 11$ and 21. The interpolating data points are $\{(x_j, \hat{f}_j)\}_{j=1}^{129}$, where nodes $x_j = -1 + \frac{j-1}{64} \in \mathtt{FPN}$ and $\hat{f}_j = \mathrm{fl}(f(x_j))$ for $1 \le j \le 129$. The new nodes are $s_i = -1 + \frac{i-1}{128} \in \mathtt{FPN}$ for $i = 1, 2, \dots, 257$.

We generated randomly 100 such testing polynomials $f(x)$ of (5.1). For each testing polynomial, in Figure 5.1, we plot the absolute error $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$ as well as upper bound ϵ_{up} in (3.21), where each entry y_i of $\boldsymbol{y}$ is computed as $y_i = \mathrm{fl}(f(s_i))$ while $\hat{y}_i$ of $\hat{\boldsymbol{y}}$ is $p_{n-1}(s_i)$ computed by Algorithm 1.

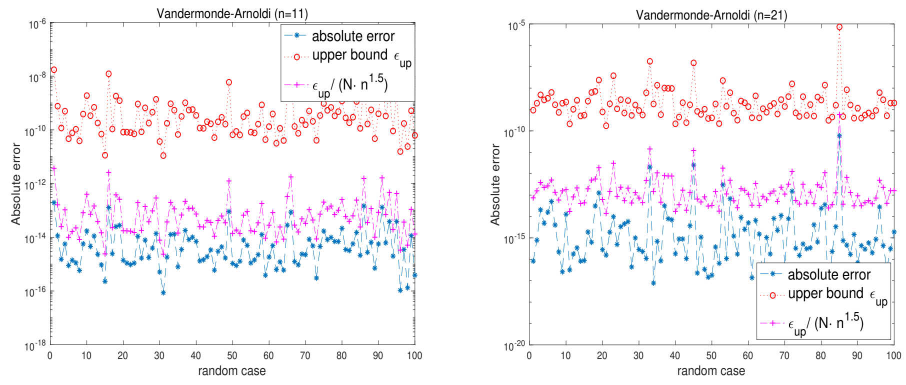

FIGURE 5.1. Absolute error $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$ *vs.* upper bound ϵ_{up} in (3.21) and empirical one $\frac{\epsilon_{\mathrm{up}}}{N\sqrt{n^3}}$.

We observed from Figure 5.1 that upper bound ϵ_{up} in (3.21) overestimates the true error $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$, not surprisingly. This is because the rounding error analysis in section 3 reflects the worst-case scenario, as most error analyses usually do. For example, for the least squares problem (3.6), a factor $c_{\mathrm{ls}} = (6N - 3n + 41)n = O(Nn)$ shows up and contributes a magnifying factor about $O(Nn^{3/2})$ to $c_{\boldsymbol{a}}$ in (3.10) in the final upper bound in (3.21). Interestingly, numerical results in Figure 5.1 suggests that $\epsilon_{\mathrm{up}}/[Nn^{3/2}]$ would be more indicative than ϵ_{up} itself.

Example 5.2. In this example, we will evaluate the effects of $\|\widehat{W}\|_2$ and $\|\boldsymbol{a}\|_2$ on error $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|$, where $\widehat{W}$ is the computed basis matrix associated with new nodes in $\boldsymbol{s}$. We let $f(x) = T_{n-1}(x)$, the $(n-1)$st Chebyshev polynomial (4.1) of the 1st kind in $[-1, 1]$, i.e., $\alpha_j = \cos\left(\frac{2j-1}{2(n-1)}\pi\right)$ for $1 \le j \le n-1$ in (5.1). The same nodes $\{x_j\}_{j=1}^{129}$ and $\{s_j\}_{j=1}^{257}$ as in Example 5.1 are used and we call MATLAB's `chebypoly` [8] in package `chebfun` to generate $\{f(x_j)\}_{j=1}^{129}$ and $\{f(s_j)\}_{j=1}^{257}$. We observed that as degree n increases, so does $\|\widehat{W}\|_2$. In Table 5.1, we report $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$, $\frac{\epsilon_{\mathrm{up}}}{Nn^{3/2}}$ together

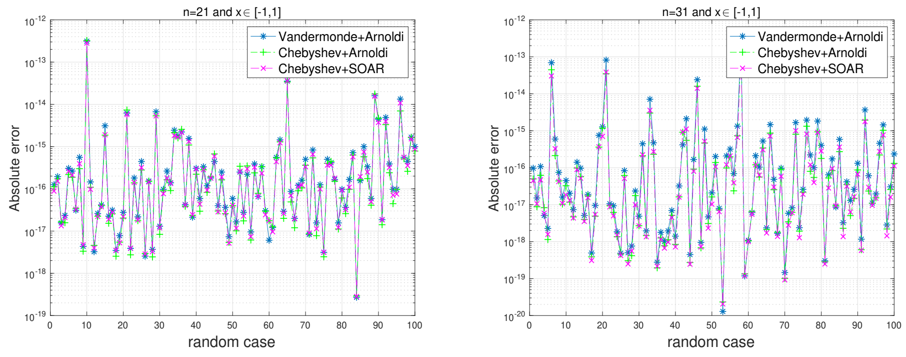

FIGURE 5.2. Absolute error $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$ for three methods on $f(x)$ in (5.1).

with $\|\boldsymbol{a}\|_2$, $\|\widehat{W}\|_2$ and the condition number $\kappa_2(\widehat{Q})$. Empirically, $\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$ and $\mathtt{u}\,\|\widehat{W}\|_2$ and $\mathtt{u}\,\|\boldsymbol{a}\|_2$ are of the same order of magnitude.

TABLE 5.1. Absolute error, upper bound, $\|\widehat{W}\|_2$ and $\|\boldsymbol{a}\|_2$ in Example 5.2.

n	$\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_\infty$	$\frac{\epsilon_{\rm up}}{N\sqrt{n^3}}$	$\|\widehat{W}\|_2$	$\|\boldsymbol{a}\|_2$	$\kappa_2(\widehat{Q})$
31	1.2712e-14	1.1918e-13	2.2414e+01	1.1751e+02	1.0000e+00
41	3.1530e-14	2.0783e-12	8.7119e+01	7.1896e+02	1.0000e+00
51	5.5622e-13	2.4895e-10	1.5549e+03	4.4686e+03	1.0000e+00
61	1.3901e-11	1.1939e-07	8.9303e+04	2.8054e+04	1.0000e+00

Example 5.3. This example is to evaluate our new ways based on the Chebyshev polynomial basis. In the setting of Example 5.2, we did two implementations of Algorithm 2 that differ in their step 1: calling the Arnoldi process or SOAR. In Figure 5.2, we compare the two ways with Algorithm 2 for $n = 11$ and $n = 21$, each on 100 randomly generated polynomials of form (5.1). It turns out that they all deliver roughly the same accuracy in evaluating the best fitting polynomial at $\{s_i\}_{i=1}^M$, but the methods based on the Chebyshev polynomial basis hold an edge in providing slightly more accurate evaluations.

We next perform tests on general intervals. In Figure 5.3, we consider interval [10,12] with: randomly choosing roots α_j uniformly on the interval, interpolating nodes $x_j = 10 + \frac{j-1}{64} \in \mathtt{FPN}$ for $j = 1, 2, \dots, 129$, and nodes $s_i = 10 + \frac{i-1}{128} \in \mathtt{FPN}$ for $i = 1, 2, \dots, 257$ to evaluate at. Once again, numerical results demonstrate that the methods based on the Chebyshev polynomial basis can achieve slightly more accurate evaluations.

Finally, let $f(x) = T_{n-1}(x)$ given in (4.1). Following Example 5.2 and using MATLAB's `chebypoly`, for each degree $n \in [11 : 5 : 31]$, we generate data points $\{(x_j, \hat{f}_j = \mathrm{fl}(T_{n-1}(x_j)))\}_{j=1}^N$ on the intervals $[1, 2]$ or $[2, 4]$, respectively, where $\{x_j\}$ are equally spaced with stepsize 2^{-6}. The three methods then are tested on the new equidistant nodes $\{s_j\}$ with stepsize 2^{-7} on the respective intervals. The results in Figure 5.4 clearly show that using the translated Chebyshev polynomial basis is superior to the monomial basis.

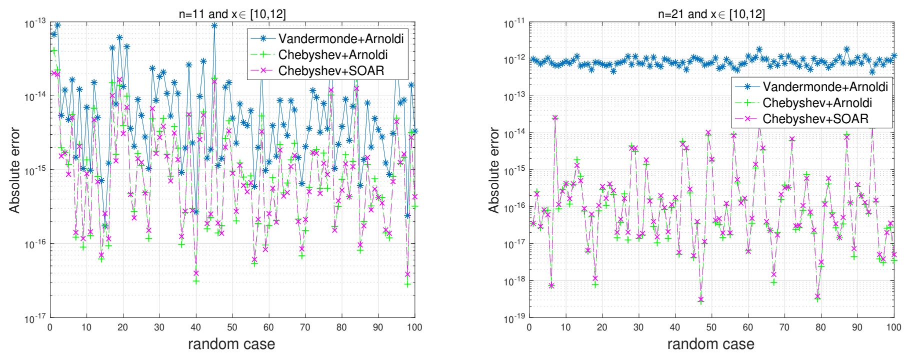

FIGURE 5.3. Absolute error $\|\boldsymbol{y}-\hat{\boldsymbol{y}}\|_\infty$ for three methods on $f(x)$ in (5.1).

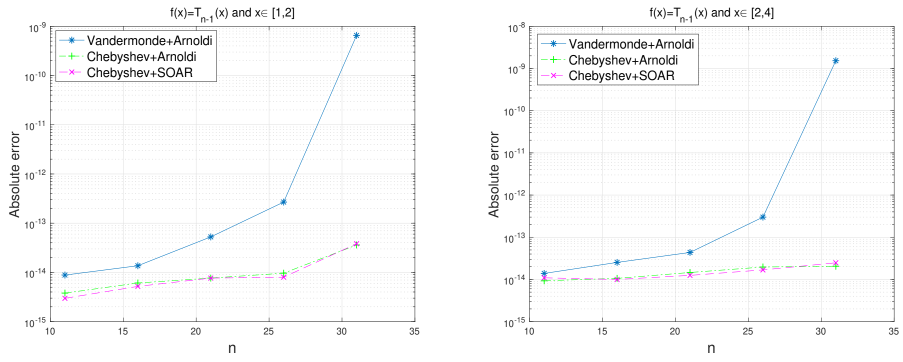

FIGURE 5.4. Absolute error $\|\boldsymbol{y}-\hat{\boldsymbol{y}}\|_\infty$ for three methods on $f(x) = T_{n-1}(x)$ given in (4.1).

6. **Conclusion.**

We have provided a theoretical justification for the effectiveness of Algorithm 1 in the presence of rounding errors. Our analysis reveals an implicit link R which acts as the bridge between the fitting stage: steps 1 and 2 of the algorithm, and the evaluating stage: steps 3 and 4. It plays a unique role to bypass a Vandermonde system in determining the best fitting polynomial and later evaluating the polynomial at nodes $\{s_j\}_{j=1}^M$ and thereby successfully avoid the notorious ill-conditionedness of a Vandermonde matrix. A detailed error analysis is given in Theorem 3.1 that is helpful to better understand the behavior of Algorithm 1 in finite precision. Inspired by Algorithm 1, for the least-squares approximation on an interval, we propose to use an orthogonal polynomial basis instead of the monomial basis. For that, we showcased the (translated) Chebyshev polynomials of the first kind in Algorithm 2. Finally, we performed several numerical tests. The numerical results reflect the behavior of the algorithms in finite precision arithmetic, as revealed in Theorem 3.1 and other similar estimates; for the approximation in real case on an interval, we also demonstrate the slightly more accurate evaluation of the (translated) Chebyshev polynomials, especially when some of the interpolating nodes have much greater magnitudes than 1.

Although our analysis is restricted to real nodes and real function values, it can be extended to complex nodes and complex function values as we commented at the beginning of section 3.

Acknowledgements. The authors would like to thank Prof. Lloyd N. Trefethen for useful comments and suggestions that have improved the presentation of this paper. They also thank the anonymous referees for their useful comments and suggestions.

REFERENCES

[1] Z. Bai and Y. Su, SOAR: A second-order Arnoldi method for the solution of the quadratic eigenvalue problem, *SIAM J. Matrix Anal. Appl.*, **26** (2005), 640-659.

[2] B. Beckermann, The condition number of real Vandermonde, Krylov and positive definite Hankel matrices, *Numer. Math.*, **85** (2000), 553-577.

[3] A. Björk, *Numerical Methods For Least Squares Problems*, SIAM, Philadelphia, 1996.

[4] N. Boulle, Y. Nakatsukasa and A. Townsend, Rational neural networks, in *Neural Information Processing Systems (NeurIPS)*, 2020.

[5] P. D. Brubeck, Y. Nakatsukasa and L. N. Trefethen, Vandermonde with Arnoldi, *SIAM Rev.*, **63** (2021), 405-415.

[6] A. Deczky, Equiripple and minimax (Chebyshev) approximations for recursive digital filters, *IEEE Trans. Acoust., Speech, Signal Processing*, **22** (1974), 98-111.

[7] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[8] T. A. Driscoll, N. Hale and L. N. Trefethen, *Chebfun User's Guide*, Pafnuty Publications, Oxford, 2014, See also `www.chebfun.org`.

[9] B. Fischer and R. Freund, On the constrained Chebyshev approximation problem on ellipses, *J. Approx. Theory*, **62** (1990), 297-315.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4^{th} edition, Johns Hopkins University Press, Baltimore, Maryland, 2013.

[11] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[12] M. H. Gutknecht, J. O. Smith and L. N. Trefethen, The Caratheodory-Féjer method for recursive, digital filter design, *IEEE Trans. Acoust., Speech, Signal Processing*, **ASSP-31** (1983), 1417-1426.

[13] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2^{nd} edition, SIAM, Philadephia, USA, 2002.

[14] W. D. Joubert and G. F. Carey, *Parallelizable restarted iterative methods for nonsymmetric linear systems. Part 1: Theory*, Technical report, Center for Numerical Analysis CNA-251, University of Texas at Austin, 1991.

[15] R. B. Lehoucq, *Analysis and Implementation of an Implicitly Restarted Arnoldi Iteration*, PhD Thesis, Rice University, Houston, Texas, 1995.

[16] R.-C. Li, Asymptotically optimal lower bounds for the condition number of a real Vandermonde matrix, *SIAM J. Matrix Anal. Appl.*, **28** (2006), 829-844.

[17] R.-C. Li, Lower bounds for the condition number of a real confluent Vandermonde matrix, *Math. Comp.*, **75** (2006), 1987-1995.

[18] R.-C. Li, Vandermonde matrices with Chebyshev nodes, *Linear Algebra Appl.*, **428** (2008), 1803-1832.

[19] D. Lu, Y. Su and Z. Bai, Stability analysis of the two-level orthogonal Arnoldi procedure, *SIAM J. Matrix Anal. Appl.*, **37** (2016), 195-214.

[20] G. Meinardus, *Approximation of Functions: Theory and Numerical Methods*, Translated by L. L. Schumaker, Berlin Heidelberg, Springer, 1967.

[21] Y. Nakatsukasa and R. W. Freund, Computing fundamental matrix decompositions accurately via the matrix sign function in two iterations: The power of Zolotarev's functions, *SIAM Rev.*, **58** (2016), 461-493.

[22] C. C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, PhD Thesis, University of London, 1971.

[23] C. C. Paige, Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix, *J. Inst. for Math. Appl.*, **18** (1976), 341-349.

[24] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[25] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge U. Press, 1981.

[26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2^{nd} edition, SIAM, Philadelphia, 2003.

[27] G. W. Stewart, *Matrix Algorithms, Vol. II: Eigensystems*, SIAM, Philadelphia, 2001.

[28] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*, SIAM, 2019.

[29] L. N. Trefethen, Y. Nakatsukasa and J. A. C. Weideman, Exponential node clustering at singularities for rational approximation, quadrature, and PDEs, *Numer. Math.*, **147** (2021), 227-254.

Received September 2022; final revision December 2022; early access January 2023.