MATHEMATICS OF COMPUTATION
Volume 66, Number 218, April 1997, Pages 719–741
S 0025-5718(97)00845-4

ON NUMERICAL METHODS FOR DISCRETE LEAST-SQUARES APPROXIMATION BY TRIGONOMETRIC POLYNOMIALS

HEIKE FASSBENDER

ABSTRACT. Fast, efficient and reliable algorithms for discrete least-squares approximation of a real-valued function given at arbitrary distinct nodes in $[0, 2\pi)$ by trigonometric polynomials are presented. The algorithms are based on schemes for the solution of inverse unitary eigenproblems and require only $O(mn)$ arithmetic operations as compared to $O(mn^2)$ operations needed for algorithms that ignore the structure of the problem. An algorithm which solves this problem with real-valued data and real-valued solution using only real arithmetic is given. Numerical examples are presented that show that the proposed algorithms produce consistently accurate results that are often better than those obtained by general QR decomposition methods for the least-squares problem.

1. INTRODUCTION

A problem in signal processing is the approximation of a function known only at some measured points by a trigonometric function. A number of different models for representing the measured points as a finite superposition of sine- and cosine-oscillations are possible. One choice could be to compute the trigonometric interpolating function. Then several numerical algorithms are available ([4, 5, 15]). But in general a large number of measured points are given, such that this approach leads to a trigonometric polynomial with a lot of superimposed oscillations (and a large linear system to solve). In practical applications it is often sufficient to compute a trigonometric polynomial with only a small number of superimposed oscillations. A different, often chosen approach is the (fast) Fourier transform ([15]). In this case the frequencies of the sine- and cosine-oscillations have to be chosen equidistant. More freedom in the choice of the frequencies and the number of superposed oscillations gives the following approach. Given a set of m arbitrary distinct nodes $\{\theta_k\}_{k=1}^m$ in the interval $[0, 2\pi)$, a set of m positive weights $\{\omega_k^2\}_{k=1}^m$, and a real-valued function $f(\theta)$ whose values at the nodes θ_k are explicitly known. Then the trigonometric function

$$t(\theta) = a_0 + \sum_{j=1}^{\ell} (a_j \cos j\theta + b_j \sin j\theta), \qquad a_j, b_j \in \mathbb{R}, \tag{1}$$

Received by the editor March 29, 1995 and, in revised form, January 31, 1996.
1991 *Mathematics Subject Classification.* Primary 65D10, 42A10, 65F99.
Key words and phrases. Trigonometric approximation, unitary Hessenberg matrix, Schur parameter.

of order at most $\ell < m/2$ is sought that minimizes the discrete least-squares error

$$||f - t||_{\mathbb{R}} := \sqrt{\sum_{k=1}^{m} |f(\theta_k) - t(\theta_k)|^2 \omega_k^2}. \tag{2}$$

In general, m (the number of measured functional values) is much larger than $n = 2\ell + 1$ (the number of coefficients to be determined).

Standard algorithms for solving the approximation problem (2) require $O(mn^2)$ arithmetic operations. In this paper faster algorithms are presented which make use of the special structure of the problem (2). In [17] Reichel, Ammar, and Gragg reformulate the problem (2) as the following standard least-squares problem: Minimize

$$||DAc - Dg||_2 = \min, \tag{3}$$

where $D = diag(\omega_1, ..., \omega_m) \in \mathbb{C}^{m \times m}$ is a diagonal matrix with the given weights on the diagonal and A is a transposed Vandermonde matrix

$$A = \begin{pmatrix} 1 & z_1 & \cdots & z_1^{n-1} \\ 1 & z_2 & \cdots & z_2^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & z_m & \cdots & z_m^{n-1} \end{pmatrix} \in \mathbb{C}^{m \times n}$$

with $z_k = exp(i\theta_k)$. $g = [g(z_1), ..., g(z_m)]^T \in \mathbb{C}^m$ is a vector of the values of a complex function $g(z)$ and $c = [c_0, ..., c_{n-1}]^T \in \mathbb{C}^n$ is the solution vector. With the proper choice of n and g, it is easy to see that the coefficients of the trigonometric polynomial (1) that minimizes the error (2) can be read off of the least-squares solution $\widehat{c}$ of (3) (see [17]).

The solution $\widehat{c}$ can be computed by using the QR decomposition of DA. Since DA has full column rank, there is an $m \times m$ unitary matrix Q with orthonormal columns and an $m \times n$ upper triangular matrix R with positive diagonal elements such that

$$DA = QR = (Q_1|Q_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1,$$

where $Q_1 \in \mathbb{C}^{m \times n}$ has orthonormal columns and $R_1 \in \mathbb{C}^{n \times n}$ has positive diagonal elements. The solution of (3) is given by $\widehat{c} = R_1^{-1} Q_1^H Dg$. Algorithms that compute the QR decomposition of DA without using the special structure require $O(mn^2)$ arithmetic operations ([14]). Demeure [9] presents an $O(mn + n^2 + m)$ algorithm to compute the QR decomposition of a transposed Vandermonde matrix. This scheme explicitly uses $A^H A$.

In [17] Reichel, Ammar, and Gragg present an approach to compute the QR decomposition of DA that is based on computational aspects associated with the family of polynomials orthogonal with respect to an inner product on the unit circle. Such polynomials are known as Szegö polynomials. The following interpretation of the elements of Q_1 and R_1 in terms of Szegö polynomials can be given : Q_1 is determined by the values of the Szegö polynomials at the nodes z_k. R_1 expresses the power basis in terms of the orthonormal Szegö polynomials. Therefore, the columns of R_1^{-1} are the coefficients of the Szegö polynomials in the power basis. There exist algorithms for determining the values of the Szegö polynomials at nodes z_k ([17, 12]) which require $O(mn)$ arithmetic operations. The computation of the

columns of R_1^{-1} relies on the Szegö recursion and is closely related to the Levinson algorithm as $(DA)^T DA = R_1^T R_1$ is a Toeplitz matrix.

Observe that

$$\begin{aligned} DA &= \begin{pmatrix} \omega_1 & \omega_1 z_1 & \omega_1 z_1^2 & \cdots & \omega_1 z_1^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega_m & \omega_m z_m & \omega_m z_m^2 & \cdots & \omega_m z_m^{n-1} \end{pmatrix} \\ &= (q, \Lambda q, \Lambda^2 q, ..., \Lambda^{n-1} q) \\ &= \sigma_0 (q_0, \Lambda q_0, \Lambda^2 q_0, ..., \Lambda^{n-1} q_0) \end{aligned}$$

with $q := (\omega_1, ..., \omega_m)^T, \sigma_0 = ||q||_2, q_0 := \sigma_0^{-1} q$ and $\Lambda = diag(z_1, ..., z_m)$. Thus, the matrix DA is given by the first n columns of the Krylov matrix $K(\Lambda, q, m) = (q, \Lambda q, ..., \Lambda^{m-1} q)$. We may therefore use the following consequence of the Implicit Q Theorem to compute the desired QR decomposition. If there exists a unitary matrix U such that $U^H \Lambda U = H$ is a unitary upper Hessenberg matrix with positive subdiagonal elements, then the QR decomposition of $K(\Lambda, q_0, m)$ is given by UR with $R = K(H, e_1, m)$. The construction of such a unitary Hessenberg matrix from spectral data, here contained in Λ and q_0, is an inverse eigenproblem. Thus the best trigonometric approximation to f can be computed via solving this inverse eigenproblem. Because of the uniqueness of the here given QR decomposition of $K(\Lambda, q_0, m)$, it follows from the above given interpretation of the elements of Q_1 that the elements in U are the values of the Szegö polynomials at the nodes z_k. Thus solving the inverse unitary Hessenberg eigenvalue problem $U^H \Lambda U = H$ is equivalent to computing Szegö polynomials.

Unitary Hessenberg matrices have special properties which allow the development of efficient algorithms for this class of matrices. Any $n \times n$ unitary Hessenberg matrix with positive subdiagonal elements can be uniquely parametrized by n complex parameters, that is

$$H = G_1(\gamma_1) G_2(\gamma_2) \cdots G_n(\gamma_n)$$

for certain complex-valued parameters $|\gamma_k| < 1, 1 \leq k < n$, and $|\gamma_n| = 1$. Here $G_k(\gamma_k)$ denotes the $n \times n$ Givens reflector in the $(k, k+1)$ plane

$$G_k = G_k(\gamma_k) = diag(I_{k-1}, \begin{bmatrix} -\gamma_k & \sigma_k \\ \sigma_k & \overline{\gamma_k} \end{bmatrix}, I_{n-k-1})$$

with $\gamma_k \in \mathbb{C}, \sigma_k \in \mathbb{R}^+$, $|\gamma_k|^2 + \sigma_k^2 = 1$, and

$$G_n(\gamma_n) = diag(I_{n-1}, -\gamma_n)$$

with $\gamma_n \in \mathbb{C}, |\gamma_n| = 1$. The nontrivial entries γ_k are called *Schur parameters* and the σ_k are called *complementary Schur parameters.* Ammar, Gragg, and Reichel make use of this parametrization in [2] by developing an efficient and reliable algorithm (IUQR-algorithm) for solving the inverse unitary Hessenberg eigenvalue problem. The algorithm manipulates the n complex parameters instead of the n^2 matrix elements. An adaption of the IUQR scheme to the computation of the vector $c' = Q_1^H Dg$ can be given, which requires $O(mn)$ arithmetic operations. After computing the vector c', the least-squares solution $\widehat{c} = R_1^{-1} c'$ of (3) can be obtained using an algorithm closely related to the Levinson algorithm. Reichel, Ammar, and Gragg present in [17] an $O(n^2)$ algorithm to compute $R_1^{-1} b$ for an arbitrary vector $b \in \mathbb{C}^n$.

The algorithms proposed by Reichel, Ammar, and Gragg in [17] construct the least-squares solution $\widehat{c}$ of (3) in $O(mn + n^2)$ arithmetic operations. The coefficients of the optimal trigonometric polynomial t of (2) can be recovered from $\widehat{c}$. This representation of t is convenient if we desire to integrate or differentiate the polynomial or if we wish to evaluate it at many equidistant points on a circle with a center at the origin. If we, on the other hand, only desire to evaluate t at a few points, then we can use the representation of t in terms of Szegö polynomials. For details see [17].

In [3] Van Barel and Bultheel generalize the method by Ammar, Gragg, and Reichel to solve a discrete linearized rational least-squares approximation on the unit circle. Further generalizations are given by Bultheel and Van Barel in [6].

In [16] Newbery presents an algorithm for least-squares approximation by trigonometric polynomials which is closely related to the computation of Szegö polynomials. This $O(n^2)$ algorithm and its connection to the algorithms presented here is discussed in [11, 13].

The method proposed by Reichel, Ammar, and Gragg to solve the real-valued approximation problem (2) computes the real-valued solution using complex arithmetic by solving an inverse unitary Hessenberg eigenvalue problem $U^H \Lambda U = H$, where a unitary Hessenberg matrix is constructed from spectral data. Now $H = G_1(\gamma_1)G_2(\gamma_2)\cdots G_n(\gamma_n)$ can be transformed to $G_o G_e^H$ by a unitary similarity transformation (see [1]), where

$$G_o = G_1(\gamma_1)G_3(\gamma_3)\cdots G_{2[(n+1)/2]-1}(\gamma_{2[(n+1)/2]-1}) = \begin{pmatrix} -\gamma_1 & \sigma_1 & & & \\ \sigma_1 & \overline{\gamma_1} & & & \\ & & -\gamma_3 & \sigma_3 & \\ & & \sigma_3 & \overline{\gamma_3} & \\ & & & & \ddots \end{pmatrix}$$

is the product of the odd numbered elementary reflectors and

$$G_e^H = G_2(\gamma_2)G_4(\gamma_4)\cdots G_{2[n/2]}(\gamma_{2[n/2]}) = \begin{pmatrix} 1 & & & \\ & -\gamma_2 & \sigma_2 & \\ & \sigma_2 & \overline{\gamma_2} & \\ & & & \ddots \end{pmatrix}$$

is the product of the even numbered elementary reflectors. Here $[x] = \max\{i \in \mathbb{N} | i \leq x\}$. G_o, G_e are block diagonal matrices with block size at most two. Thus the inverse unitary Hessenberg eigenvalue problem $U^H \Lambda U = H$ is equivalent to an inverse eigenvalue problem $Q^H(\Lambda - \lambda I)QG_e = G_o - \lambda G_e$, where a *Schur parameter pencil* is constructed from spectral data.

In this paper numerical methods for the trigonometric approximation are discussed which rely on this inverse eigenvalue problem for Schur parameter pencils. Especially, an algorithm is developed which requires $O(mn)$ arithmetic operations to solve the real-valued approximation problem (2) using only real arithmetic. The following approach for solving the approximation problem (2) is considered: (2) is reformulated to a real-valued least-squares problem $||D\widehat{f} - D\widetilde{A}\widetilde{t}||_2$ where $D \in \mathbb{R}^{m\times m}, \widetilde{A} \in \mathbb{R}^{m\times n}, \widehat{f} \in \mathbb{R}^m, \widetilde{t} \in \mathbb{R}^n$. This least-squares problem will be solved via an QR decomposition of $D\widetilde{A}$. As $D\widetilde{A}$ is a real $m \times n$ matrix with full column rank, there exists a unique "skinny" real QR decomposition $\widetilde{Q}_1\widetilde{R}_1$ of $D\widetilde{A}$ where $\widetilde{Q}_1 \in \mathbb{R}^{m\times n}$ has orthonormal columns and $\widetilde{R}_1 \in \mathbb{R}^{n\times n}$ is upper triangular

with positive diagonal entries [14, Theorem 5.2.2]. First it is shown that the Q-factor of the QR decomposition of $D\widetilde{A}$ is the unitary matrix Q which transforms $\Lambda = diag(z_1, ..., z_m)$ to Schur parameter pencil form $G_o - \lambda G_e$. The R-factor of the desired QR decomposition is a modified Krylov matrix based on $G_o G_e^H$. The computation of R implicitly yields the Cholesky factorization of a bordered block-Toeplitz-plus-block-Hankel matrix with 2×2 blocks. An algorithm for inverting the upper square subblock of R is given.

In Sections 3 and 4 algorithms for computing $\widetilde{Q}_1$ and $\widetilde{R}_1$ are developed, which use only real arithmetic and require merely $O(mn)$ arithmetic operations. For that purpose the effect of the transformation matrix $\widetilde{Q}_1$ on the real and imaginary part of $\Lambda = diag(z_1, ..., z_m) = diag(\cos\theta_1, ..., \cos\theta_m) + i\ diag(\sin\theta_1, ..., \sin\theta_m)$ is considered.

Numerical results are given in Section 5. We will see that the proposed algorithms produce consistently accurate results that are often better than those obtained by general QR decomposition methods for the least-squares problem.

2. A REAL-VALUED APPROACH

New, fast algorithms to solve the discrete least-squares approximation are developed, particularly algorithms which solve this problem with real-valued data and real-valued solution in $O(mn)$ arithmetic operations using only real arithmetic. Instead of the approach used by Reichel, Ammar, and Gragg the following real-valued linear least-squares problem is considered. Since

$$\underbrace{\begin{pmatrix} 1 & \sin\theta_1 & \cos\theta_1 & \cdots & \sin\ell\theta_1 & \cos\ell\theta_1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & \sin\theta_m & \cos\theta_m & \cdots & \sin\ell\theta_m & \cos\ell\theta_m \end{pmatrix}}_{\widetilde{A}} \underbrace{\begin{pmatrix} a_0 \\ b_1 \\ a_1 \\ \vdots \\ b_\ell \\ a_\ell \end{pmatrix}}_{\widetilde{t}} = \underbrace{\begin{pmatrix} t(\theta_1) \\ \vdots \\ t(\theta_m) \end{pmatrix}}_{\widehat{t},}$$

it follows with $D = diag(\omega_1, ..., \omega_m)$ and $\widehat{f} = (f(\theta_1), ..., f(\theta_m))^T$ that

$$||f - t||_{\mathbb{R}} = ||D(\widehat{f} - \widehat{t})||_2 = ||D\widehat{f} - D\widetilde{A}\widetilde{t}||_2. \tag{4}$$

As proven in [11] the matrix $(D\widetilde{A})^T(D\widetilde{A})$ belonging to the normal equations corresponding to (4),

$$(D\widetilde{A})^T(D\widetilde{A})\widetilde{t} = (D\widetilde{A})^T D\widehat{f},$$

is a bordered block-Toeplitz-plus-block-Hankel-matrix with 2×2 blocks. In particular

$$(D\widetilde{A})^T D\widetilde{A} = \begin{pmatrix} x_{11} & x_1^T & \cdots & x_\ell^T \\ x_1 & & & \\ \vdots & & T + H & \\ x_\ell & & & \end{pmatrix}$$

with the symmetric block-Toeplitz-matrix

$$T = \begin{pmatrix} A_0 & A_1 & A_2 & A_3 & \cdots & \cdots & A_{\ell-1} \\ A_1^T & A_0 & A_1 & A_2 & \cdots & \cdots & A_{\ell-2} \\ A_2^T & A_1^T & A_0 & A_1 & \cdots & \cdots & A_{\ell-3} \\ A_3^T & A_2^T & A_1^T & A_0 & \ddots & & A_{\ell-4} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & A_0 & A_1 \\ A_{\ell-1}^T & A_{\ell-2}^T & A_{\ell-3}^T & A_{\ell-4}^T & \cdots & A_1^T & A_0 \end{pmatrix} \in \mathbb{R}^{2\ell \times 2\ell}$$

and the symmetric block-Hankel-matrix

$$H = \begin{pmatrix} B_0 & B_1 & B_2 & \cdots & B_{\ell-2} & B_{\ell-1} \\ B_1 & B_2 & B_3 & \cdots & B_{\ell-1} & B_\ell \\ B_2 & B_3 & B_4 & \cdots & B_\ell & B_{\ell+1} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ B_{\ell-2} & B_{\ell-1} & B_\ell & \cdots & B_{2\ell-4} & B_{2\ell-3} \\ B_{\ell-1} & B_\ell & B_{\ell+1} & \cdots & B_{2\ell-3} & B_{2\ell-2} \end{pmatrix} \in \mathbb{R}^{2\ell \times 2\ell}$$

where

$$2A_0 = I,$$

$$2A_j = \begin{pmatrix} \sum_{p=1}^m \omega_p^2 \cos j\theta_p & -\sum_{p=1}^m \omega_p^2 \sin j\theta_p \\ \sum_{p=1}^m \omega_p^2 \sin j\theta_p & \sum_{p=1}^m \omega_p^2 \cos j\theta_p \end{pmatrix}, \quad j = 1, 2, ..., \ell - 1,$$

$$2B_j = \begin{pmatrix} -\sum_{p=1}^m \omega_p^2 \cos(j+2)\theta_p & \sum_{p=1}^m \omega_p^2 \sin(j+2)\theta_p \\ \sum_{p=1}^m \omega_p^2 \sin(j+2)\theta_p & \sum_{p=1}^m \omega_p^2 \cos(j+2)\theta_p \end{pmatrix}, \quad j = 0, 1, ..., 2\ell - 3,$$

$$x_{11} = \sum_{p=1}^m \omega_p^2, \qquad x_j^T = (\sum_{p=1}^m \omega_p^2 \sin j\theta_p, \sum_{p=1}^m \omega_p^2 \cos j\theta_p), \quad j = 2, 3, ..., \ell.$$

The minimum norm solution $\widetilde{t}$ of the linear least-squares problem (4) can be computed by using the QR decomposition of $D\widetilde{A}$. Since $D\widetilde{A}$ has full column rank $n = 2\ell+1$, there exists an $m \times m$ orthogonal matrix Q and an $m \times n$ upper triangular matrix R with positive diagonal elements such that

$$D\widetilde{A} = QR = (Q_1|Q_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1,$$

where $R_1 \in \mathbb{R}^{n \times n}$ has positive diagonal elements and $Q_1 \in \mathbb{R}^{m \times n}$ has orthonormal columns. The minimum norm solution $\widetilde{t}$ of (4) is therefore given by

$$\widetilde{t} = R_1^{-1} Q_1^H D\widehat{f}.$$

Moreover

$$(D\widetilde{A})^T D\widetilde{A} = R_1^T R_1.$$

Thus R_1 is the Cholesky factor of $(D\widetilde{A})^T D\widetilde{A}$. Implicitly we have to compute the Cholesky factorization of the special bordered block-Toeplitz-plus-block-Hankel-matrix $(D\widetilde{A})^T D\widetilde{A}$.

Algorithms that compute the QR decomposition of $D\widetilde{A}$ without using its structure require $O(mn^2)$ arithmetic operations. In this paper we present algorithms

for the solution of the least-squares problem (4) that implicitly compute the QR decomposition of $D\widetilde{A}$ and require only $O(mn)$ arithmetic operations.

Let $\Lambda = diag(e^{i\theta_1}, ..., e^{i\theta_m})$, then because of $\cos\theta = \frac{1}{2}(e^{i\theta} + e^{-i\theta})$ and $\sin\theta = \frac{1}{2i}(e^{i\theta} - e^{-i\theta})$ it follows with $q = (\omega_1, ..., \omega_m)^T$ that

$$D\widetilde{A} = \frac{1}{2}[(\Lambda^0+(\Lambda^H)^0)q, -i(\Lambda-\Lambda^H)q, (\Lambda+\Lambda^H)q, ..., -i(\Lambda^\ell-(\Lambda^H)^\ell)q, (\Lambda^\ell+(\Lambda^H)^\ell)q].$$

Let $\kappa(\Lambda, q, \ell)$ be a *modified Krylov matrix*

$$\kappa(\Lambda, q, \ell) = [q, \Lambda q, \Lambda^H q, \Lambda^2 q, (\Lambda^H)^2 q, ..., \Lambda^\ell q, (\Lambda^H)^\ell q] \in \mathbb{C}^{m\times(2\ell+1)},$$

whose columns are the vectors of the Krylov sequence $\{q, \Lambda^H q, (\Lambda^H)^2 q, ..., (\Lambda^H)^\ell q\}$ based on Λ^H interleaved with the vectors of the Krylov sequence $\{\Lambda q, \Lambda^2 q, ..., \Lambda^\ell q\}$ based on Λ.

Then we have

$$D\widetilde{A} = \frac{1}{2}\kappa(\Lambda, q, \ell)\begin{pmatrix} 2 & & & & & & \\ & -i & 1 & & & & \\ & i & 1 & & & & \\ & & & -i & 1 & & \\ & & & i & 1 & & \\ & & & & & \ddots & \\ & & & & & -i & 1 \\ & & & & & i & 1 \end{pmatrix} =: \frac{1}{2}\kappa(\Lambda, q, \ell)F.$$

The idea is to compute the QR decomposition of $D\widetilde{A}$ from a QR decomposition of $\kappa(\Lambda, q, \ell)$ by solving an inverse eigenproblem similar to the approach of Reichel, Ammar, and Gragg in [17]. For this we need the following lemma which is a consequence of Theorem 2.12 and Theorem 3.2 in [7].

Lemma 1. *Given n distinct complex numbers $\{\lambda_k\}_{k=1}^n$ on the unit circle and associated positive weights $\{\nu_k^2\}_{k=1}^n$, there is a unique unreduced $n \times n$ Schur parameter pencil $G_o - \lambda G_e$ (with positive complementary Schur parameters) and unique unitary matrices Q and P such that*

$$\begin{aligned} Q^H e_1 &= \sigma_0^{-1}[\nu_1, ..., \nu_n]^T, \\ Q(\Lambda - \lambda I)P &= G_o - \lambda G_e, \\ \Lambda &= diag(\lambda_1, ..., \lambda_n), \end{aligned}$$

where $\sigma_0 = (\sum_{k=1}^n \nu_k^2)^{\frac{1}{2}}$.

The lemma shows that the reduction of Λ to an unreduced Schur parameter pencil $G_o - \lambda G_e = Q^H(\Lambda - \lambda I)P$ is unique if the first column of Q is given. Choosing

$$Qe_1 = \sigma_0^{-1}q, \qquad \sigma_0 = ||q||_2,$$

and using

$$\Lambda^k = (QG_oG_e^HQ^H)^k = Q(G_oG_e^H)^kQ^H,$$

we get

$$\Lambda^k q = \sigma_0 Q(G_oG_e^H)^k e_1 \qquad \text{and} \qquad (\Lambda^H)^k q = \sigma_0 Q(G_eG_o^H)^k e_1.$$

That is

$$\begin{aligned}\kappa(\Lambda, q, \ell) &= \sigma_0 Q[e_1, G_o G_e^H e_1, G_e G_o^H e_1, (G_o G_e^H)^2 e_1, ..., (G_o G_e^H)^\ell e_1, (G_e G_o^H)^\ell e_1] \\ &= \sigma_0 Q \kappa(G_o G_e^H, e_1, \ell) =: \sigma_0 QR.\end{aligned}$$

As can be seen, R is an $m \times n$ upper triangular matrix whose diagonal elements are products of the complementary Schur parameters. Therefore the upper $n \times n$ block of R is nonsingular with positive diagonal elements $R_{ii} = \sigma_1 \cdots \sigma_{i-1}$. Moreover

$$\left.\begin{array}{rcl} R_{1,2i} & = & \overline{R_{1,2i+1}} \\ R_{2i,2i+1} & = & -\sigma_1 \cdots \sigma_{2i-1}\overline{\gamma_{2i}} \\ R_{2i+1,2i+2} & = & -\sigma_1 \cdots \sigma_{2i}\gamma_{2i+1} \end{array}\right\}, \qquad i = 1, ..., \ell.$$

Thus we have $D\widetilde{A} = \frac{\sigma_0}{2} QRF$ with $RF =$

$$\begin{pmatrix}
2 & 2Im(R_{12}) & 2Re(R_{12}) & 2Im(R_{14}) & 2Re(R_{14}) & \cdots & 2Im(R_{1,2\ell}) & 2Re(R_{1,2\ell}) \\
0 & i(R_{23}-R_{22}) & R_{23}+R_{22} & i(R_{25}-R_{24}) & R_{25}+R_{24} & \cdots & i(R_{2,2\ell+1}-R_{2,2\ell}) & R_{2,2\ell+1}+R_{2,2\ell} \\
0 & iR_{33} & R_{33} & i(R_{35}-R_{34}) & R_{35}+R_{34} & \cdots & i(R_{3,2\ell+1}-R_{3,2\ell}) & R_{3,2\ell+1}+R_{3,2\ell} \\
0 & 0 & 0 & i(R_{45}-R_{44}) & R_{45}+R_{44} & \cdots & i(R_{4,2\ell+1}-R_{4,2\ell}) & R_{4,2\ell+1}+R_{4,2\ell} \\
0 & 0 & 0 & iR_{55} & R_{55} & \cdots & i(R_{5,2\ell+1}-R_{5,2\ell}) & R_{5,2\ell+1}+R_{5,2\ell} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & & i(R_{2\ell,2\ell+1}-R_{2\ell,2\ell}) & R_{2\ell,2\ell+1}+R_{2\ell,2\ell} \\
\vdots & \vdots & \vdots & \vdots & \vdots & & iR_{2\ell+1,2\ell+1} & R_{2\ell+1,2\ell+1} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0
\end{pmatrix}$$

In order to get a unique "skinny", real-valued QR decomposition of $D\widetilde{A}$ from the above, 2×2 blocks of the form

$$\sigma_1 \cdots \sigma_{2j-1} \begin{pmatrix} -i(1+\overline{\gamma_{2j}}) & (1-\overline{\gamma_{2j}}) \\ i\sigma_{2j} & \sigma_{2j} \end{pmatrix}$$

have to be transformed to upper triangular form with positive diagonal elements. Choosing $x_{2j} = \sigma_{2j}^2 + |1+\gamma_{2j}|^2 \in \mathbb{R}$, $s_{2j} = \sigma_{2j}/\sqrt{x_{2j}} \in \mathbb{R}$ and $c_{2j} = (1+\gamma_{2j})/\sqrt{x_{2j}} \in \mathbb{C}$ we get

$$\begin{aligned}&\sigma_1 \cdots \sigma_{2j-1} \begin{pmatrix} -i & \\ & 1 \end{pmatrix} \begin{pmatrix} -c_{2j} & s_{2j} \\ s_{2j} & \overline{c_{2j}} \end{pmatrix} \begin{pmatrix} -i(1+\overline{\gamma_{2j}}) & (1-\overline{\gamma_{2j}}) \\ i\sigma_{2j} & \sigma_{2j} \end{pmatrix} \\ &= 2(\sigma_{2j}^2 + |1+\gamma_{2j}|^2)^{-\frac{1}{2}} \sigma_1 \cdots \sigma_{2j-1} \begin{pmatrix} (1+Re(\gamma_{2j})) & Im(\gamma_{2j}) \\ 0 & \sigma_{2j} \end{pmatrix}\end{aligned}$$

and with

$$\begin{aligned}\mathcal{C}_{2k} &= diag(I_{2k-1}, \begin{bmatrix} ic_{2k} & -is_{2k} \\ s_{2k} & \overline{c_{2k}} \end{bmatrix}, I_{m-2k-1}), \\ \mathcal{C}_e &= \mathcal{C}_2 \mathcal{C}_4 \cdots \mathcal{C}_{2\ell} \in \mathbb{C}^{m \times m}\end{aligned}$$

we obtain $\mathcal{C}_e RF = \widetilde{R}$, such that $\widetilde{R}$ is an $m \times n$ upper triangular matrix with positive diagonal elements. Let $\widetilde{Q} = Q\mathcal{C}_e^H$. Then a QR decomposition of $D\widetilde{A}$ is given by

$$D\widetilde{A} = \frac{\sigma_0}{2} \widetilde{Q}\widetilde{R} = \frac{\sigma_0}{2} \widetilde{Q}_1 \widetilde{R}_1$$

where $\widetilde{Q}_1$ are the first n columns of $\widetilde{Q}$ and $\widetilde{R}_1$ is the upper $n \times n$ block of $\widetilde{R}$. Since $\widetilde{R}$ has positive diagonal elements, this "skinny" QR decomposition has to be unique. Therefore $\widetilde{Q}_1$ and $\widetilde{R}_1$ are real-valued matrices with $\widetilde{Q}_1 \in \mathbb{R}^{m \times n}$ and $\widetilde{R}_1 \in \mathbb{R}^{n \times n}$.

The minimum norm solution $\widetilde{t}$ of the least-squares problem (4) is obtained by

$$\widetilde{t} = 2\sigma_0^{-1} \widetilde{R}_1^{-1} \widetilde{Q}_1^H D\widehat{f}.$$

In order to solve the trigonometric approximation problem via the approach discussed here, we have to solve the inverse eigenvalue problem $Q^H(\Lambda - \lambda I)Q = G_o - \Lambda G_e$. In [11, 12] different methods for solving this problem are discussed: the Stieltjes-like procedure for orthogonal Laurent polynomials, the generalized Arnoldi procedure for unitary diagonal matrix pencils, and the algorithm for solving the inverse eigenvalue problem for Schur parameter pencils. Each of these methods requires $O(m^2)$ arithmetic operations to compute Q, the Schur parameters γ_1, ..., γ_m, and the complementary Schur parameters $\sigma_1, ..., \sigma_{m-1}$. As only the first n columns of $\widetilde{Q}$ (and Q) are required, these methods can be stopped after n steps without solving the entire inverse eigenvalue problem. Simple modifications of these algorithms yield $O(mn)$ algorithms to compute the first $n-1$ Schur parameters and to compute $\widetilde{Q}_1^H D\widehat{f} = \mathcal{C}_e^H Q_1^H D\widehat{f}$. The solution of (4)

$$||D(\widehat{f} - \widehat{t})||_2 = ||D\widehat{f} - D\widetilde{A}\widetilde{t}||_2 = ||\widetilde{Q}^H D\widehat{f} - \frac{\sigma_0}{2}\widetilde{R}\widehat{t}||_2$$

is now obtained by computing

$$2\sigma_0^{-1} \widetilde{R}^{-1} (\widetilde{Q}^H D\widehat{f}).$$

As $\widetilde{R} = \mathcal{C}_e R F$ and F and $\mathcal{C}_e$ are known and easily invertible, we have to invert the upper $n \times n$ block of $R = \kappa(G_o G_e^H, e_1, \ell)$. In [11] two $O(n^2)$ algorithms are developed to invert $R = \kappa(G_o G_e^H, e_1, \ell)$, that is to compute $S = [s_1, s_2, ..., s_n] \in \mathbb{R}^{n \times n}$ with $RS = I$. Numerical experiments show that the following of the two algorithms yields better results.

Algorithm 1
algorithm to invert $\kappa(G_o G_e^H, e_1, k)$

input : N = 2k+1, $\{\gamma_j\}_{j=1}^N, \{\sigma_j\}_{j=1}^N$
output : $S = [s_1 s_2 ... s_N]$ with $\kappa(G_o G_e^H, e_1, k) S = I$
$t_1 = e_1, s_1 = t_1$
for $j = 1, 2, ..., N-1$
$t_{j+1} = \sigma_j^{-1}(Jt_j + \gamma_j \widetilde{I}_j \overline{t_j})$
if j+1 even
then $s_{j+1} = \widehat{I}_{j+1}^{-1} t_{j+1}$
else $s_{j+1} = \widehat{I}_{j+1}^{-1} \overline{t_{j+1}}$
end if
end for

where

$$J = [e_2, e_3, ..., e_N, 0],$$
$$\widetilde{I}_j = [e_j, e_{j-1}, ..., e_2, e_1, e_{j+1}, ..., e_N],$$
$$\widehat{I}_{2j+1}^{-1} = [e_{2j}, e_{2j-2}, e_{2j-4}, ..., e_4, e_2, e_1, e_3, ..., e_{2j-1}, e_{2j+1}, e_{2j+2}, ..., e_N],$$
$$\widehat{I}_{2j}^{-1} = [e_{2j-1}, e_{2j-3}, e_{2j-5}, ..., e_3, e_1, e_2, e_4, ..., e_{2j-2}, e_{2j}, e_{2j+1}, ..., e_N].$$

This algorithm was obtained by the observation that s_{2k} is a permutation of the $2k$th column of the inverse of the Krylov matrix $K(H, e_1, 2\ell)$ and s_{2k+1} is a permutation of the $(2k+1)$st column of the inverse of the Krylov matrix $K(\overline{H}, e_1, 2\ell) = \overline{K(H, e_1, 2\ell)}$:

$$[e_1, He_1, H^2e_1, ..., H^{2\ell}e_1]\widehat{I}_{2k}s_{2k} = e_{2k},$$
$$[e_1, \overline{H}e_1, \overline{H}^2e_1, ..., \overline{H}^{2\ell}e_1]\widehat{I}_{2k+1}s_{2k+1} = e_{2k+1}.$$

Since $(K(H, e_1, 2\ell))^H K(H, e_1, 2\ell)$ is a Toeplitz matrix, the inverse $T = [t_1, ..., t_n]$ of $K(H, e_1, 2\ell)$ can be computed by a simple modification of the Levinson algorithm yielding $t_1 = e_1$, $t_{j+1} = \sigma_j^{-1}(Jt_j + \gamma_j \widetilde{I}_j \bar{t}_j)$.

3. Computation of $\widetilde{Q}_1$

In this section algorithms for computing $\widetilde{Q}_1$ are developed which use only real arithmetic and require merely $O(mn)$ arithmetic operations. Observing the effect of the transformation $\widetilde{Q}_1$ to the real and imaginary part of $\Lambda = C + iS, C = diag(\cos\theta_1, ..., \cos\theta_m), S = diag(\sin\theta_1, ..., \sin\theta_m)$ we obtain

$$\widetilde{Q}_1^T C \widetilde{Q}_1 = X,$$
$$\widetilde{Q}_1^T S \widetilde{Q}_1 = Y,$$
$$\widetilde{Q}_1 e_1 = \sigma_0^{-1}(\omega_1, ..., \omega_m)^T,$$

where $\widetilde{Q}_1 \in \mathbb{R}^{m\times n}$, X is a $(2\ell+1)\times(2\ell+1)$ symmetric pentadiagonal matrix of the form

$$\begin{pmatrix}
x & x & \oplus & & & & & & \\
x & x & x & \oplus & & & & & \\
\oplus & x & x & x & \oplus & & & & \\
 & \oplus & x & x & x & \oplus & & & \\
 & & \oplus & x & x & x & \oplus & & \\
 & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & & & \oplus & x & x & x & \oplus \\
 & & & & & \oplus & x & x & x \\
 & & & & & & \oplus & x & x
\end{pmatrix},$$

and Y is a $(2\ell+1)\times(2\ell+1)$ symmetric bordered block tridiagonal matrix of the form

(5)
$$\left(\begin{array}{c|cc|cc|cc|cc|cc}
x & \oplus & & & & & & & & & \\
\hline
\oplus & x & x & x & \ominus & & & & & & \\
 & x & x & x & x & & & & & & \\
\hline
 & x & x & x & x & x & \ominus & & & & \\
 & \ominus & x & x & x & x & x & & & & \\
\hline
 & & & x & x & x & x & x & \ominus & & \\
 & & & \ominus & x & x & x & x & x & & \\
\hline
 & & & & & \ddots & & \ddots & & \ddots & \\
\hline
 & & & & & & & x & x & x & x & x & \ominus \\
 & & & & & & & \ominus & x & x & x & x & x \\
\hline
 & & & & & & & & & x & x & x & x \\
 & & & & & & & & & \ominus & x & x & x
\end{array}\right).$$

Here x denotes any real-valued, $\oplus$ any positive and $\ominus$ any negative matrix element. The elements of the second subdiagonal of X are strictly positive, the elements of the third subdiagonal of Y are less than or equal to zero.

This observation motivates the following theorem.

Theorem 1. *Let $n = 2\ell + 1 < m$. Let $C, S \in \mathbb{R}^{m\times m}$ be symmetric matrices with $C^2 + S^2 = I$ and $CS = SC$. Let $u = (\omega_1, ..., \omega_m)^T \in \mathbb{R}^m$ with $u^T u = 1$. Then there exists a unique $m \times n$ matrix $\hat{Q}$ with orthonormal columns such that*

$$\hat{Q}^T C \hat{Q} = X,$$
$$\hat{Q}^T S \hat{Q} = Y,$$
$$\hat{Q} e_1 = u,$$

where X is a symmetric pentadiagonal matrix with $x_{j+2,j} > 0$, and Y is a symmetric matrix of the desired form (5) with $y_{21} > 0, y_{2j+3,2j} < 0$ and $y_{2j+2,2j-1} = 0$.

Proof. See proofs of Theorem 3 and Theorem 4 in [11]. □

The existence proof in [11] constructs the matrices $\widetilde{Q}_1, X$, and Y columnwise in a Lanczos-like style from the equations

$$C\widetilde{q}_j = \widetilde{Q}_1 x_j,$$
$$S\widetilde{q}_j = \widetilde{Q}_1 y_j.$$

The first column of $\widetilde{Q}_1$ is given, the second column is found using the equation $S\widetilde{q}_1 = \widetilde{Q}_1 y_1$. The subsequent columns of $\widetilde{Q}_1$ can be computed using only the equation $C\widetilde{q}_j = \widetilde{Q}_1 x_j$. This construction leads to an $O(mn)$ algorithm for the computation of $\widetilde{Q}_1, X$, and Y. A problem (as for every Lanczos-like algorithm) is the loss of orthogonality between the columns of $\widetilde{Q}_1$.

In the following a different algorithm for computing $\widetilde{Q}, X$, and Y is developed, which builds up the matrices X and Y successively by adding two new triplets $(\cos\theta_{2k}, \sin\theta_{2k}, \omega_{2k})$ and $(\cos\theta_{2k+1}, \sin\theta_{2k+1}, \omega_{2k+1})$ at a time (similar to the idea of the IUQR-algorithm by Ammar, Gragg, and Reichel in [2]). That is, an orthogonal matrix $\widetilde{Q}$ is constructed such that $\widetilde{Q}^T e_1 = q = \sigma_0^{-1}(\omega_1, ..., \omega_m)^T$ and

$$\begin{pmatrix} 1 & \\ & \widetilde{Q} \end{pmatrix} \left(\begin{pmatrix} \delta & q^H \\ q & C \end{pmatrix} - \lambda \begin{pmatrix} \delta & 0^H \\ 0 & S \end{pmatrix} \right) \begin{pmatrix} 1 & \\ & \widetilde{Q}^H \end{pmatrix}$$
$$= \begin{pmatrix} \delta & e_1^H \\ e_1 & X \end{pmatrix} - \lambda \begin{pmatrix} \delta & 0^H \\ 0 & Y \end{pmatrix},$$

where X is a symmetric, pentadiagonal matrix with positive entries on the second subdiagonal and Y is a symmetric bordered block tridiagonal matrix of the form (5).

For our constructions we will use the following notation (as given by Bunse-Gerstner and Elsner in [7]). For $1 \le j < k \le n$ we denote by $Q(j, k, z)$ the Householder transformation defined below, which eliminates the entries $j + 1$ through k in the vector $z \in \mathbb{C}^n$, i.e.,

$$Q(j, k, z)z \in span\{e_1, ..., e_j, e_{k+1}, ..., e_n\}.$$

Here we have

$$Q(j, k, z) = I - \frac{1}{||v||_2^2} 2vv^H,$$

where $v^H = (0, ..., 0, \overline{z_j} + (sign\, \overline{z_j})\alpha, \overline{z_{j+1}}, ..., \overline{z_k}, 0, ..., 0)$ and $\alpha = (\sum_{l=j}^{k} |z_l|^2)^{\frac{1}{2}}$. If z is real, then $Q(j,k,z)$ is a real matrix. Note that

$$Q(j,k,z) = diag(I_{j-1}, \widehat{Q}, I_{n-k}).$$

For any $M \in \mathbb{C}^{n\times n}$ the vectors consisting of the columns and rows of the matrix are denoted by the corresponding small letter as $m_{*1}, ..., m_{*n}$ and $m_{1*}, ..., m_{n*}$, respectively. $\oslash$ denotes any matrix element not equal to zero, $\otimes$ denotes undesired matrix elements.

In the following $m = n$ is assumed for simplicity. For $n = 3$ the desired construction is trivial. Let $n > 3, n$ odd. We are given $u = (\omega_1, ..., \omega_n)^T, C = diag(c_1, ..., c_n)$ and $S = diag(s_1, ..., s_n)$ where $c_j^2 + s_j^2 = 1$. Assume that Q has been computed such that $Qu = (\omega_1, \omega_2, x, 0, ..., 0) = u'$, and

$$C' = QCQ^T = \begin{pmatrix} c_1 & & \\ & c_2 & \\ & & X' \end{pmatrix}, \qquad S' = QSQ^T = \begin{pmatrix} s_1 & & \\ & s_2 & \\ & & Y' \end{pmatrix},$$

where X' and Y' are $(n-2)\times(n-2)$ matrices of the desired form. Let $Q(1,n,u') = Q(1,3,u') = Q_1$, then we get $Q_1 u' = (x, 0, ..., 0)^T$ and

$$X_{(1)} = Q_1 C' Q_1^T = \begin{pmatrix} x & x & x & \otimes & \otimes & & & & & & \\ x & x & x & x & \otimes & & & & & & \\ x & x & x & x & x & & & & & & \\ \otimes & x & x & x & x & x & & & & & \\ \otimes & \otimes & x & x & x & x & x & & & & \\ & & & x & x & x & x & x & & & \\ & & & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & x & x & x & x & x & \\ & & & & & & x & x & x & x & \\ & & & & & & & x & x & x & \end{pmatrix},$$

$$Y_{(1)} = Q_1 S' Q_1^T = \begin{pmatrix} x & x & \otimes & \otimes & & & & & & & & \\ x & x & x & x & & & & & & & & \\ \otimes & x & x & x & & & & & & & & \\ \otimes & x & x & x & x & x & x & & & & & \\ & & & x & x & x & x & & & & & \\ & & & x & x & x & x & x & x & & & \\ & & & x & x & x & x & x & x & & & \\ & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & x & x & x & x & x & x & \\ & & & & & x & x & x & x & x & x & \\ & & & & & & & x & x & x & x & \\ & & & & & & & x & x & x & x & \end{pmatrix}.$$

Now a sequence of similarity transformation is performed to transform $X_{(1)}$ and $Y_{(1)}$ to matrices of the desired form. The first two steps are straightforward. Due to the desired form of the first columns/rows of X and Y, first we have to transform the first column/row of $Y_{(1)}$ to the desired form, then the first column/row of the X-matrix. Determine $Q(2, n, (Y_{(1)})_{*1}) = Q(2, 4, (Y_{(1)})_{*1}) = Q_2$ and transform $X_{(1)}$

and $Y_{(1)}$:

$$X_{(2)} = Q_2 X_{(1)} Q_2^T = \begin{pmatrix} x & x & x & \otimes & \otimes & & & & & & \\ x & x & x & x & \otimes & \otimes & & & & & \\ x & x & x & x & x & \otimes & & & & & \\ \otimes & x & x & x & x & x & & & & & \\ \otimes & \otimes & x & x & x & x & x & & & & \\ & \otimes & \otimes & x & x & x & x & x & & & \\ & & & & x & x & x & x & x & & \\ & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \end{pmatrix},$$

$$Y_{(2)} = Q_2 Y_{(1)} Q_2^T = \begin{pmatrix} x & \oslash & 0 & 0 & & & & & & & & \\ \oslash & x & x & x & x & \otimes & \otimes & & & & & \\ 0 & x & x & x & x & \otimes & \otimes & & & & & \\ 0 & x & x & x & x & x & x & & & & & \\ & x & x & x & x & x & x & & & & & \\ & \otimes & \otimes & x & x & x & x & x & x & & & \\ & \otimes & \otimes & x & x & x & x & x & x & & & \\ & & & & & x & x & x & x & x & x & \\ & & & & & x & x & x & x & x & x & \\ & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}.$$

Next choose $Q(3, n, (X_{(2)})_{*1}) = Q(3, 5, (X_{(2)})_{*1}) = Q_3$ to bring the first column/row of $X_{(2)}$ to the desired form

$$X_{(3)} = Q_3 X_{(2)} Q_3^T = \begin{pmatrix} x & x & \oslash & 0 & 0 & & & & & & \\ x & x & x & x & \otimes & \otimes & & & & & \\ \oslash & x & x & x & x & \otimes & \otimes & & & & \\ 0 & x & x & x & x & x & \otimes & & & & \\ 0 & \otimes & x & x & x & x & x & & & & \\ & \otimes & \otimes & x & x & x & x & x & & & \\ & & \otimes & \otimes & x & x & x & x & x & & \\ & & & & & x & x & x & x & x & \\ & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix},$$

$$Y_{(3)} = Q_3 Y_{(2)} Q_3^T = \begin{pmatrix} x & \oslash & & & & & & & & & \\ \oslash & x & x & x & x & \otimes & \otimes & & & & \\ & x & x & x & x & \otimes & \otimes & & & & \\ & x & x & x & x & x & x & & & & \\ & x & x & x & x & x & x & & & & \\ & \otimes & \otimes & x & x & x & x & x & x & & \\ & \otimes & \otimes & x & x & x & x & x & x & & \\ & & & & & x & x & x & x & x & x \\ & & & & & x & x & x & x & x & x \\ & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}.$$

Now different ways to further reduce $X_{(3)}$ and $Y_{(3)}$ to the desired form are possible. One possibility is (analogous to the Lanczos-like algorithm to compute $\widetilde{Q}$) to reduce $X_{(3)}$ columnwise to the desired form. If $Y_{(3)}$ is transformed in the same way, then Theorem 1 gives that $Y_{(3)}$ is transformed to the desired form as well. Numerical tests solving (4) showed that such a method for computing $\widetilde{Q}$ did not produce good results for all test examples. This method which works essentially on X produced very poor results if the values θ_k are chosen equidistant in the interval $[0, \pi)$.

A different possibility to further reduce $X_{(3)}$ and $Y_{(3)}$ is described below. We transform the second column of $X_{(3)}$ to the desired form by $Q_4 = Q(4, n, (X_{(3)})_{*2}) =$

$Q(4, 6, (X_{(3)})_{*2})$

$$X_{(4)} = Q_4 X_{(3)} Q_4^T = \begin{pmatrix}
x & x & \oslash & & & & & & & & & \\
x & x & x & \oslash & 0 & 0 & & & & & & \\
\oslash & x & x & x & x & \otimes & \otimes & & & & & \\
 & \oslash & x & x & x & x & \otimes & \otimes & & & & \\
 & 0 & x & x & x & x & x & \otimes & & & & \\
 & 0 & \otimes & x & x & x & x & x & & & & \\
 & & \otimes & \otimes & x & x & x & x & x & & & \\
 & & & \otimes & \otimes & x & x & x & x & x & & \\
 & & & & & & x & x & x & x & x & \\
 & & & & & & & x & x & x & x & x \\
 & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots
\end{pmatrix},$$

$$Y_{(4)} = Q_4 Y_{(3)} Q_4^T = \begin{pmatrix}
x & \oslash & & & & & & & & & & \\
\oslash & x & x & x & x & \otimes & \otimes & & & & & \\
 & x & x & x & x & \otimes & \otimes & & & & & \\
 & x & x & x & x & x & x & \otimes & \otimes & & & \\
 & x & x & x & x & x & x & \otimes & \otimes & & & \\
 & \otimes & \otimes & x & x & x & x & x & x & & & \\
 & \otimes & \otimes & x & x & x & x & x & x & & & \\
 & & & \otimes & \otimes & x & x & x & x & x & x & \\
 & & & \otimes & \otimes & x & x & x & x & x & x & \\
 & & & & & & & x & x & x & x & x & x \\
 & & & & & & & x & x & x & x & x & x \\
 & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{pmatrix}.$$

Subsequently the second column of $Y_{(4)}$ is transform to the desired form by $Q_5 = Q(5, n, (Y_{(4)})_{*2}) = Q(5, 7, (Y_{(4)})_{*2})$

$$X_{(5)} = Q_5 X_{(4)} Q_5^T = \begin{pmatrix}
x & x & \oslash & & & & & & & & & \\
x & x & x & \oslash & & & & & & & & \\
\oslash & x & x & x & x & \otimes & \otimes & & & & & \\
 & \oslash & x & x & x & x & \otimes & \otimes & & & & \\
 & & x & x & x & x & x & \otimes & \otimes & & & \\
 & & \otimes & x & x & x & x & x & \otimes & & & \\
 & & \otimes & \otimes & x & x & x & x & x & & & \\
 & & & \otimes & \otimes & x & x & x & x & x & & \\
 & & & & \otimes & \otimes & x & x & x & x & x & \\
 & & & & & & & x & x & x & x & x \\
 & & & & & & & & x & x & x & x & x \\
 & & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots
\end{pmatrix},$$

$$Y_{(5)} = Q_5 Y_{(4)} Q_5^T = \begin{pmatrix}
x & \oslash & & & & & & & & & & \\
\oslash & x & x & x & \oslash & 0 & 0 & & & & & \\
 & x & x & x & x & \otimes & \otimes & & & & & \\
 & x & x & x & x & x & x & \otimes & \otimes & & & \\
 & \oslash & x & x & x & x & x & \otimes & \otimes & & & \\
 & 0 & \otimes & x & x & x & x & x & x & & & \\
 & 0 & \otimes & x & x & x & x & x & x & & & \\
 & & & \otimes & \otimes & x & x & x & x & x & x & \\
 & & & \otimes & \otimes & x & x & x & x & x & x & \\
 & & & & & & & x & x & x & x & x & x \\
 & & & & & & & x & x & x & x & x & x \\
 & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{pmatrix}.$$

As $C+iS$ is unitary, $Q_5Q_4Q_3Q_2Q_1Q(C+iS)Q^TQ_1^TQ_2^TQ_3^TQ_4^TQ_5^T = X_{(5)}+iY_{(5)} = Z_{(5)}$ is unitary; that is $Z_{(5)}Z_{(5)}^H = I$ and especially $\sum_{k=1}^n (Z_{(5)})_{1k}(\overline{Z}_{(5)})_{6k} = (Z_{(5)})_{13}(\overline{Z}_{(5)})_{63} = 0$ as well as $\sum_{k=1}^n (Z_{(5)})_{1k}(\overline{Z}_{(5)})_{7k} = (Z_{(5)})_{13}(\overline{Z}_{(5)})_{73} = 0$. From $(Z_{(5)})_{13} = (X_{(5)})_{13} = (X_{(3)})_{13} \neq 0$, we get $(Z_{(5)})_{63} = (Z_{(5)})_{73} = 0$. Thus we obtain

$$X_{(5)} = \begin{pmatrix}
x & x & \oslash & & & & & & & & & \\
x & x & x & \oslash & & & & & & & & \\
\oslash & x & x & x & x & & & & & & & \\
 & \oslash & x & x & x & x & \otimes & \otimes & & & & \\
 & & x & x & x & x & x & \otimes & \otimes & & & \\
 & & & x & x & x & x & x & \otimes & & & \\
 & & & \otimes & x & x & x & x & x & & & \\
 & & & \otimes & \otimes & x & x & x & x & x & & \\
 & & & & \otimes & \otimes & x & x & x & x & x & \\
 & & & & & & & x & x & x & x & x \\
 & & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots
\end{pmatrix},$$

$$Y_{(5)} = \begin{pmatrix}
x & \oslash & & & & & & & & & & \\
\oslash & x & x & x & \oslash & & & & & & & \\
 & x & x & x & x & & & & & & & \\
 & x & x & x & x & x & x & \otimes & \otimes & & & \\
 & \oslash & x & x & x & x & x & \otimes & \otimes & & & \\
 & & & x & x & x & x & x & x & & & \\
 & & & x & x & x & x & x & x & & & \\
 & & & \otimes & \otimes & x & x & x & x & x & x & \\
 & & & \otimes & \otimes & x & x & x & x & x & x & \\
 & & & & & & & x & x & x & x & x & x \\
 & & & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots
\end{pmatrix};$$

the third columns/rows of $X_{(5)}$ and $Y_{(5)}$ are in the desired form. Choosing

$$Q_6 = Q(6,8,(X_{(5)})_{*4}), \quad X_{(6)} = Q_6X_{(5)}Q_6^T, \quad Y_{(6)} = Q_6Y_{(5)}Q_6^T,$$

and

$$Q_7 = diag(I_6,-1,I_{n-7})Q(7,9,(Y_{(6)})_{*4}), \quad X_{(7)} = Q_7X_{(6)}Q_7^T, \quad Y_{(7)} = Q_7Y_{(6)}Q_7^T,$$

the fourth columns/rows of $X_{(5)}$ and $Y_{(5)}$ can be transformed to the desired form. As above we can argue that the fifth columns/rows of $X_{(7)}$ and $Y_{(7)}$ are in the desired form. Now we have the same situation as after the construction of $X_{(5)}, Y_{(5)}$, solely the undesired elements are found 2 rows and columns further down. Therefore these undesired elements can be chased down along the diagonal analogous to the last two steps. This gives rise to the following sequence of similarity transformations to add two new triplets $(\cos\theta_{2k}, \sin\theta_{2k}, \omega_{2k})$, $(\cos\theta_{2k+1}, \sin\theta_{2k+1}, \omega_{2k+1})$ to X' and Y'.

Algorithm 2

IUQR-like algorithm to add two new triplets $(\cos\theta_{2k}, \sin\theta_{2k}, \omega_{2k})$, $(\cos\theta_{2k+1}, \sin\theta_{2k+1}, \omega_{2k+1})$ to X' and Y'

$Q_1 = Q(1,3,u')$, $X = Q_1 C' Q_1^T$, $Y = Q_1 S' Q_1^T$
$Q_2 = Q(2,4,Y_{*1})$, $X = Q_2 X Q_2^T$, $Y = Q_2 Y Q_2^T$
$Q_3 = Q(3,5,X_{*1})$, $X = Q_3 X Q_3^T$, $Y = Q_3 Y Q_3^T$
for $j = 4, ..., n-2$
  if j even
  then $Q_j = Q(j, j+2, X_{*,j-2})$
  else $Q_j = Q(j, j+2, Y_{*,j-3})$
  end if
  $X = Q_j X Q_j^T$, $Y = Q_j Y Q_j^T$
end for
$Q_{n-1} = Q(n-1, n, X_{*,n-3})$, $X = Q_{n-1} X Q_{n-1}^T$, $Y = Q_{n-1} Y Q_{n-1}^T$
if $y_{21} < 0$
then $Q_{n+1} = diag(1, -1, I_{n-2})$, $X = Q_{n+1} X Q_{n+1}^T$, $Y = Q_{n+1} Y Q_{n+1}^T$
end if
for $j = 3, ..., n$
  if $X_{j,j-2} < 0$
  then $Q_{n+j} = diag(I_{j-1}, -1, I_{n-j})$, $X = Q_{n+j} X Q_{n+j}^T$, $Y = Q_{n+j} Y Q_{n+j}^T$
  end if
end for

The last statements of the algorithm ensure that

$$y_{21} > 0$$
$$x_{k,k-2} > 0 \qquad \text{for } k \in \{3,4,6,8,...,n\}.$$

Theorem 1 gives

$$y_{k,k-3} > 0 \qquad \text{for } k \in \{5,7,9,...,n\}.$$

The given algorithm can easily be modified to an $O(mn)$ algorithm for computing $\widetilde{Q}$, X, and Y from $\{\theta_k\}_{k=1}^m$ and $\{\omega_k\}_{k=1}^m$. If $m > n = 2\ell + 1$, it should be observed that only the relevant $n \times n$ block in X and Y is required. For even n only one new pair of data has to be added in the last step; the transformation matrices Q_j reduce to Givens rotations. For more details and the modified algorithm see [11].

Numerical tests solving the trigonometric approximation problem showed that this method for computing $\widetilde{Q}$ did not produce good results for all test examples. Choosing the θ_k equidistant in $[0, \pi)$ we obtain good results. But for θ_k equidistant in $[0, 2\pi)$ this method does not work very well.

A detailed analysis of the method shows that in each step matrices of the form

$$X_k = \begin{pmatrix} x_{k+2,k} & x_{k+2,k+1} \\ x_{k+3,k} & x_{k+3,k+1} \\ x_{k+4,k} & x_{k+4,k+1} \\ 0 & x_{k+5,k+1} \end{pmatrix}, \qquad Y_k = \begin{pmatrix} y_{k+2,k} & y_{k+2,k+1} \\ y_{k+3,k} & y_{k+3,k+1} \\ y_{k+4,k} & y_{k+4,k+1} \\ y_{k+5,k} & y_{k+5,k+1} \end{pmatrix}$$

are transformed to

$$\begin{pmatrix} x & x \\ 0 & x \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \qquad \begin{pmatrix} y & y \\ y & y \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

In our method, the first columns of X_k and Y_k are transformed to the desired form. Theorem 1 shows that the second columns of X_k and Y_k also have the desired form. Implicitly the fact was used that the two second columns of X_k and Y_k are linearly dependent on the two first columns. Because of rounding errors the linear dependency is lost after only a few steps of the algorithm. The theoretically generated zeros in X_k and Y_k are affected with increasing rounding errors.

Numerical tests suggest that the dissimilar treatment of the four column vectors is the main reason for the increasing rounding errors. A method that uses all four vectors for the computation of the desired transformation could perhaps solve this problem (or at least diminish it). As the four vectors

$$\begin{pmatrix} x_{k+2,k} \\ x_{k+3,k} \\ x_{k+4,k} \\ 0 \end{pmatrix}, \quad \begin{pmatrix} x_{k+2,k+1} \\ x_{k+3,k+1} \\ x_{k+4,k+1} \\ x_{k+5,k+1} \end{pmatrix}, \quad \begin{pmatrix} y_{k+2,k} \\ y_{k+3,k} \\ y_{k+4,k} \\ y_{k+5,k} \end{pmatrix}, \quad \begin{pmatrix} y_{k+2,k+1} \\ y_{k+3,k+1} \\ y_{k+4,k+1} \\ y_{k+5,k+1} \end{pmatrix}$$

span a two dimensional subspace of $\mathbb{R}^4$, the matrix

$$M_k = \begin{pmatrix} x_{k+2,k} & x_{k+2,k+1} & y_{k+2,k} & y_{k+2,k+1} \\ x_{k+3,k} & x_{k+3,k+1} & y_{k+3,k} & y_{k+3,k+1} \\ x_{k+4,k} & x_{k+4,k+1} & y_{k+4,k} & y_{k+4,k+1} \\ 0 & x_{k+5,k+1} & y_{k+5,k} & y_{k+5,k+1} \end{pmatrix}$$

has rank 2. Thus M_k has only 2 (nonzero) singular values σ_1 and σ_2. The computation of an SVD of M_k requires information of all 4 column vectors. Therefore the idea is to compute the desired transformation by an SVD of M_k. From the SVD $M_k = U_k \Sigma_k V_k$ with $U_k, V_k \in \mathbb{R}^{4\times 4}$ unitary and $\Sigma_k = diag(\sigma_1, \sigma_2, 0, 0) \in \mathbb{R}^{4\times 4}$ we obtain

$$U_k^T M_k = \Sigma_k V_k = \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

A Givens rotation to eliminate the (2,1) element of $U_k^T M_k$ transforms this to the desired form

$$\begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Numerical tests (see Section 5) showed that this computational approach of the here developed method for the trigonometric approximation problem produces consistently accurate results similar to those of the method by Ammar, Gragg, und Reichel.

A different way of computing the desired transformation using all four column vectors is the use of the rank-revealing QR decomposition of M_k [8]. With this approach the here developed method produces slightly poorer results than with the SVD approach. As the operation count for an SVD of a 4×4 matrix is not much higher than for the rank-revealing QR decomposition, all tests in Section 5 were done using the SVD approach.

4. Computation of $\widetilde{R}_1^{-1}$

In this section an algorithm for inverting the upper $n \times n$ block of $\widetilde{R}$ is developed which uses only real arithmetic and requires merely $O(n^2)$ arithmetic operations. From Section 2 we have

$$\begin{aligned}
\widetilde{R} &= \mathcal{C}_e R F \in \mathbb{R}^{m\times n} \qquad \text{with } n = 2\ell + 1,\\
R &= \kappa(G_o G_e^H, e_1, \ell) \in \mathbb{C}^{m\times n},\\
F &= diag(2, K, K, ..., K) \in \mathbb{C}^{n\times n},\\
K &= \begin{pmatrix} -i & 1 \\ i & 1 \end{pmatrix},\\
\mathcal{C}_e &= \mathcal{C}_2\mathcal{C}_4\cdots\mathcal{C}_{2l} \in \mathbb{C}^{m\times m},\\
\mathcal{C}_{2k} &= diag(I_{2k-1}, \begin{bmatrix} ic_{2k} & -is_{2k} \\ s_{2k} & \overline{c_{2k}} \end{bmatrix}, I_{m-2k-1}).
\end{aligned}$$

From $\mathcal{C}_e(G_o G_e^H)^k = (\mathcal{C}_e G_o \mathcal{C}_e^T)^k \mathcal{C}_e$ and $\mathcal{C}_e(G_e G_o^H)^k = \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)}^k \mathcal{C}_e$ for $1 \le k \le \ell$ follows

$$\begin{aligned}
\mathcal{C}_e R &= \mathcal{C}_e[e_1, G_o G_e^H e_1, G_e G_o^H e_1, (G_o G_e^H)^2 e_1, (G_e G_o^H)^2 e_1, ..., (G_o G_e^H)^\ell, (G_e G_o^H)^\ell e_1]\\
&= [e_1, \mathcal{C}_e G_o \mathcal{C}_e^T e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)} e_1, ..., (\mathcal{C}_e G_o \mathcal{C}_e^T)^\ell e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)^\ell} e_1]\\
&= \kappa(\mathcal{C}_e G_o \mathcal{C}_e^T, e_1, \ell),
\end{aligned}$$

where $\kappa(\mathcal{C}_e G_o \mathcal{C}_e^T, e_1, \ell)$ is an upper block triangular matrix of the form

$$\begin{pmatrix} \Upsilon \\ 0 \end{pmatrix} = \begin{pmatrix}
x & x & x & x & x & x & x & \cdots & x & x \\
 & x & x & x & x & x & x & \cdots & x & x \\
 & x & x & x & x & x & x & \cdots & x & x \\
 & & & x & x & x & x & \cdots & x & x \\
 & & & x & x & x & x & \cdots & x & x \\
 & & & & & x & x & \cdots & x & x \\
 & & & & & x & x & \cdots & x & x \\
 & & & & & & & \ddots & \vdots & \vdots \\
 & & & & & & & & x & x \\
 & & & & & & & & x & x \\
 & & & & & & & & 0 & 0 \\
 & & & & & & & & \vdots & \vdots \\
 & & & & & & & & 0 & 0
\end{pmatrix}$$

with $\Upsilon = \kappa(\mathcal{C}_{e,2\ell} G_{o,2\ell+1} \mathcal{C}_{e,2\ell}^T, e_1, \ell)$.

Let $S = [s_1, s_2, ..., s_{2l+1}] \in \mathbb{R}^{n\times n}$ be the inverse of the upper triangular matrix ΥF, that is of the upper $n \times n$ block of $\widetilde{R}$. Then the vectors s_k are the solutions of the equations $\Upsilon F s_k = e_k$ for $k = 1, ..., n$. Noting that the last $n - k$ columns of ΥF have no influence on the solution of these equations since the last $n - k$ entries in s_k are zero, we have to solve

$$\begin{aligned}
[e_1, (\mathcal{C}_e G_o \mathcal{C}_e^T) e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)} e_1, ..., (\mathcal{C}_e G_o \mathcal{C}_e^T)^k e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)^k} e_1, *] F s_{2k} &= e_{2k},\\
[e_1, (\mathcal{C}_e G_o \mathcal{C}_e^T) e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)} e_1, ..., (\mathcal{C}_e G_o \mathcal{C}_e^T)^k e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)^k} e_1, *] F s_{2k+1} &= e_{2k+1}.
\end{aligned}$$

Tedious calculation yields (for a detailed derivation see [11]) for $k = 1, 2, ..., \ell - 1$

$$s_{2k+2} = x_{2k+2,2k}^{-1}[(\frac{1}{2}P_k - x_{2k,2k}I)s_{2k} - x_{2k-2,2k}s_{2k-2} - x_{2k-1,2k}s_{2k-1} - x_{2k+1,2k}s_{2k+1}],$$

$$s_{2k+3} = x_{2k+3,2k+1}^{-1}[(\frac{1}{2}P_k - x_{2k+1,2k+1}I)s_{2k+1} - x_{2k-1,2k+1}s_{2k-1} - x_{2k,2k+1}s_{2k} - x_{2k+2,2k+1}s_{2k+2}]$$

where the relevant first $2k + 1$ columns of P_k are given by:

$$\begin{aligned} P_k e_1 &= 2e_3, \\ P_k e_2 &= e_4, \\ P_k e_j &= e_{j-2} + e_{j+2}, \qquad j = 3, 4, ..., 2k + 1. \end{aligned}$$

If s_1, s_2, s_3 are known, then $s_4, s_5, ..., s_n$ can be computed from the above formulae. For s_1, s_2, s_3 we have

$$[e_1, (\mathcal{C}_e G_o \mathcal{C}_e^T)e_1, \overline{(\mathcal{C}_e G_o \mathcal{C}_e^T)}e_1, *]F[s_1, s_2, s_3] = [e_1, e_2, e_3]$$

or

$$[2e_1, 2Ye_1, 2Xe_1, *][s_1, s_2, s_3] = [e_1, e_2, e_3].$$

This is equivalent to

$$2\begin{pmatrix} 1 & y_{11} & x_{11} \\ 0 & y_{21} & x_{21} \\ 0 & 0 & x_{31} \end{pmatrix}\begin{pmatrix} s_{11} & s_{21} & s_{31} \\ 0 & s_{22} & s_{32} \\ 0 & 0 & s_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Thus s_1, s_2, s_3 can be computed directly from the above equation. We obtain the following $O(n^2)$ algorithm for computing the inverse of the upper $n \times n$ block of $\widetilde{R}$.

Algorithm 3

algorithm for inverting the upper $n \times n$ block of $\widetilde{R}$

input : X, y_{11}, y_{21}

output : $S = [s_1, s_2, ..., s_{2l+1}]$ with $\widetilde{R}_1 S = I$

$s_1 = \frac{1}{2}e_1$

$s_2 = (2y_{21})^{-1}(-y_{11}e_1 + e_2)$

$s_3 = (2x_{31})^{-1}(-y_{21}^{-1}x_{21}e_2 + e_3 - (y_{21}^{-1}y_{11}x_{21} + x_{11})e_1)$

$s_4 = x_{42}^{-1}[(\frac{1}{2}P_1 - x_{22}I)s_2 - x_{12}s_1 - x_{32}s_3]$

for $k = 5, 6, ..., 2l + 1$

if k even

then $j := (k - 2)/2$

else $j := (k - 3)/2$

end if

$s_k = x_{k,k-2}^{-1}[(\frac{1}{2}P_j - x_{k-2,k-2}I)s_{k-2} - x_{k-4,k-2}s_{k-4} - x_{k-3,k-2}s_{k-3} - x_{k-1,k-2}s_{k-1}]$

end for

5. Numerical results

We present some numerical examples that compare the accuracy of the following methods for solving the trigonometric approximation problem (2):

- AGR : the algorithm proposed in [17] as sketched in the introduction. The least-squares problem (3) $||DAc - Dg||_2 = min$ is solved via QR decomposition of DA, where the desired Q-factor of the QR decomposition is computed by an inverse unitary Hessenberg eigenvalue problem and the inverse of the upper square subblock of R is computed by an algorithm closely related to the Levinson algorithm (with complex arithmetic).

- ver2.1 : the algorithm proposed in [11]. The least-squares problem (4) $||D\widehat{f} - D\widetilde{A}\widetilde{t}||_2 = min$ is solved via QR decomposition of $D\widetilde{A}$, where the desired Q-factor of the QR decomposition is computed by an inverse eigenvalue problem for Schur parameter pencils [11, 12] and the inverse of the upper square subblock of R is computed by Algorithm 1 (with complex arithmetic).

- ver4.1 : the algorithm proposed in [11]. The least-squares problem (4) $||D\widehat{f} - D\widetilde{A}\widetilde{t}||_2 = min$ is solved via QR decomposition of $D\widetilde{A}$, where the desired Q-factor of the QR decomposition is computed by simultaneous reduction of the real and imaginary part of Λ to a compact form (to X and Y) as discussed in Section 3 (using the SVD approach) and the inverse of the upper square subblock of R is computed by Algorithm 3 (with real arithmetic).

- linpack : The least-squares problem (4) $||D\widehat{f} - D\widetilde{A}\widetilde{t}||_2 = min$ is solved via the explicit formation of the matrix $D\tilde{A}$ and the use of the LINPACK [10] routines sqrdc and sqrsl (with real arithmetic)

For comparison of accuracy we compute the solution $\tilde{t}_d$ of the system $min||D\tilde{A}\tilde{t} - D\tilde{f}||_2$ in double precision using the NAG routine F04AMF. The figures display the relative error $||\tilde{t} - \tilde{t}_d||_2/||\tilde{t}_d||_2$ where $\tilde{t}$ is the coefficient vector computed in single precision by the method under consideration. Each graph displays the errors for $m = 50$ and increasing values of n. The arguments of the nodes are either equispaced in the interval $[0, \pi)$, $[0, 3/2\pi)$ or $[0, 2\pi)$ or the arguments are randomly generated uniformly distributed numbers in $[0, 2\pi)$. The weights are all equal to one, the elements of the real vector $\tilde{f}$ are randomly generated uniformly distributed numbers in $[-5, 5]$.

A comparison of the methods AGR, linpack and ver2.1 is given in Figure 1, a comparison of the methods AGR, linpack and ver4.1 is given in Figure 2. The graphs at the top of Figure 1 and Figure 2 display the relative errors in the coefficient vectors for equispaced nodes in intervals smaller than 2π. As n increases, and the problem becomes more ill conditioned, the LINPACK routines are the first to produce inaccurate results. ver2.1 produces errors that are somewhat smaller than AGR, while ver4.1 produces errors that are about the same as AGR. The graphs at the bottom of Figure 1 and Figure 2 display the relative error when the arguments are equispaced in $[0, 2\pi)$ and when the arguments are randomly generated uniformly distributed numbers in $[0, 2\pi)$. In the first case the LINPACK routines and ver2.1 produce smaller errors than AGR, while ver4.1 produces slightly larger errors. Note that in this case we are computing the Fourier transform and thus the FFT is a better method for solving this problem. When the arguments are randomly generated uniformly distributed points in $[0, 2\pi)$ the least-squares problem is relatively well conditioned and the algorithms AGR, ver2.1 and ver4.1 yield roughly the same accuracy as n gets close to m.

We obtained similar results to those in Figure 1 and Figure 2 with other choices for the nodes and the weights. For more numerical examples and a more detailed discussion see [11].

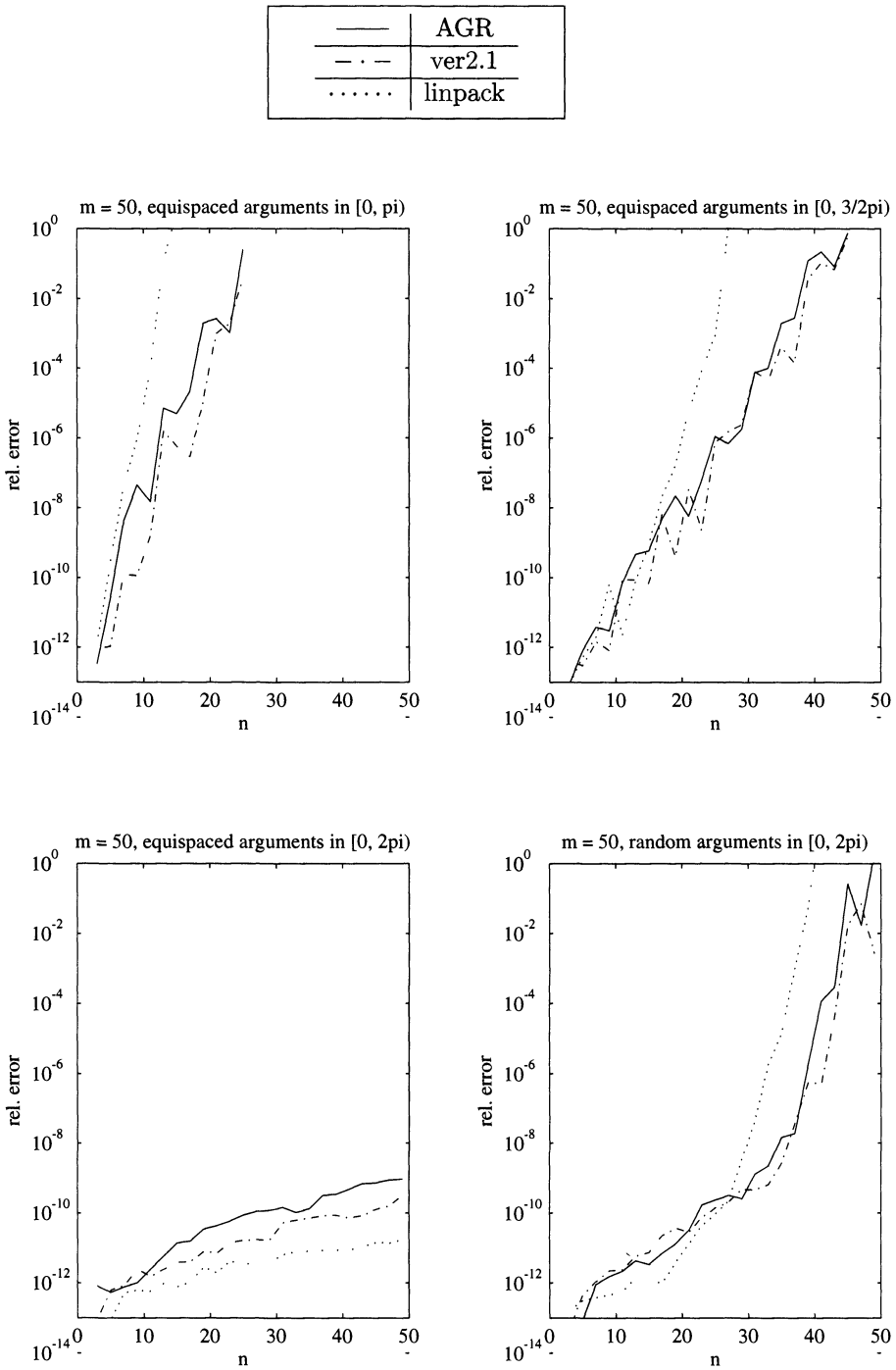
AGR
ver2.1
linpack
m = 50, equispaced arguments in [0, pi)
m = 50, equispaced arguments in [0, 3/2pi)
m = 50, equispaced arguments in [0, 2pi)
m = 50, random arguments in [0, 2pi)
rel. error
n
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
0
10
20
30
40
50

FIGURE 1

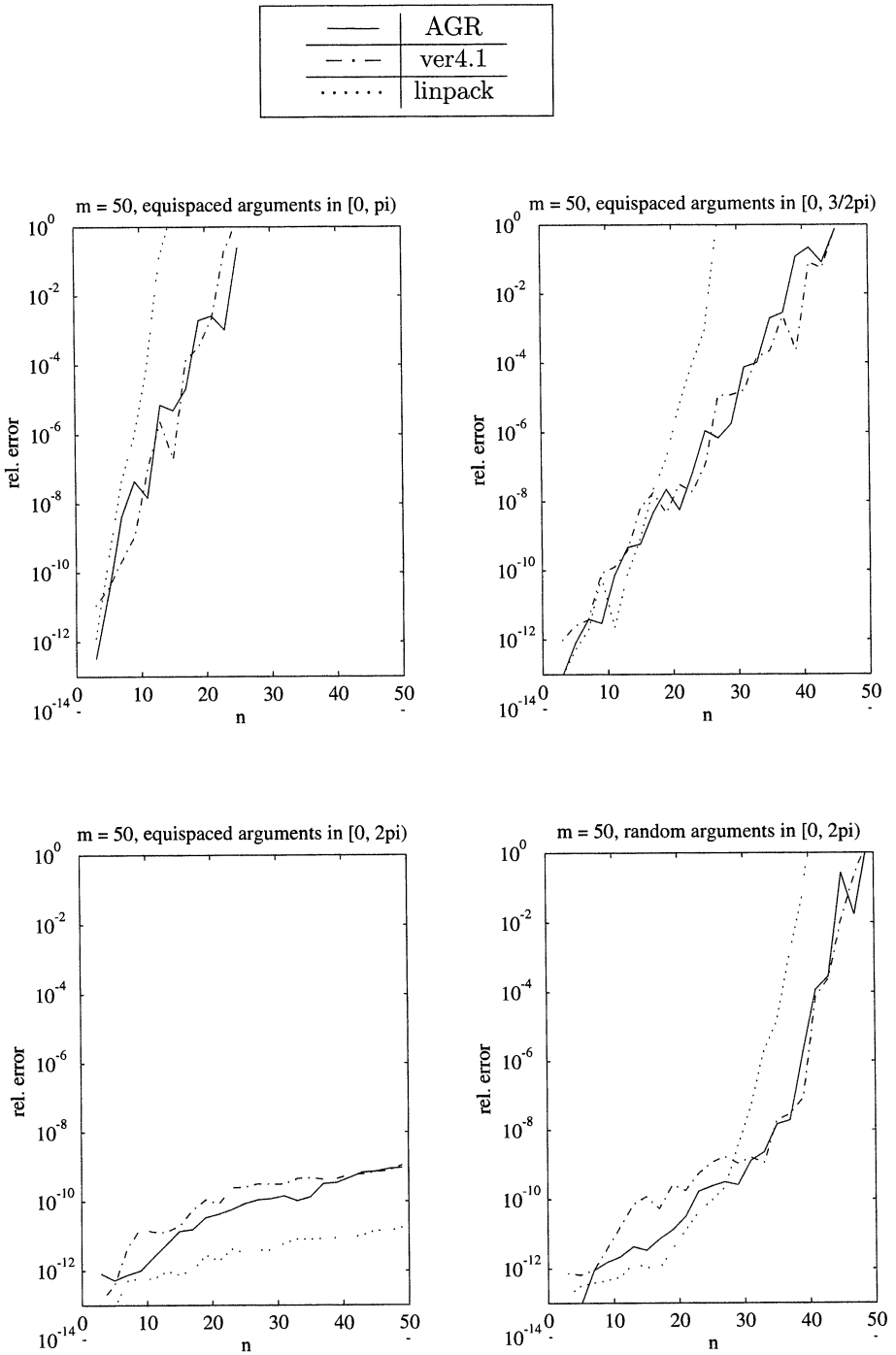
AGR
ver4.1
linpack
m = 50, equispaced arguments in [0, pi)
m = 50, equispaced arguments in [0, 3/2pi)
m = 50, equispaced arguments in [0, 2pi)
m = 50, random arguments in [0, 2pi)
rel. error
n
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
0
10
20
30
40
50

FIGURE 2

The numerical experiments have shown that generally the method ver2.1 produces more accurate results than the method AGR. On the other hand, method ver2.1 requires about 3 times as much time to solve the problem than method AGR. For the discussed examples the method linpack requires more time than the method AGR, and is the method that produces inaccurate results first. The method ver4.1 uses only real arithmetic (as opposed to the methods AGR and ver2.1 which use complex arithmetic). The relative errors in the coefficient vector produced by ver4.1 are about the same as those produced by AGR. AGR, ver2.1 and ver4.1 are algorithms to solve the trigonometric approximation problem in $O(mn)$ arithmetic operations, while the method linpack requires $O(mn^2)$ operations.

6. Final remarks

This note is a partial summary of [11].

References

1. G. S. Ammar, W. B. Gragg, and L. Reichel, *On the Eigenproblem for Orthogonal Matrices*, Proc. 25th IEEE Conference on Decision and Control, 1986, pp. 1963 – 1966.
2. ———, *Constructing a Unitary Hessenberg Matrix from Spectral Data*, Numerical Linear Algebra, Digital Signal Processing, and Parallel Algorithms (G.H. Golub and P. Van Dooren, eds.), Springer-Verlag, Berlin, 1991, pp. 385–396. MR **92j:**15003
3. M. Van Barel and A. Bultheel, *Discrete linearized least squares rational approximation on the unit circle*, Report TW179, Department of Computer Science, K.U. Leuven, Belgium, 1992.
4. J.-P. Berrut, *Baryzentrische Formeln zur trigonometrischen Interpolation I + II*, Appl. Math. Phys. (ZAMP) **35** (1984), 91 – 105, 193 – 205. MR **86e:**65007; MR **86e:**65015
5. Å. Björck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math. Comp. **24** (1970), 893 – 903. MR **44:**7721
6. A. Bultheel and M. Van Barel, *Vector orthogonal polynomials and least squares approximation*, Report TW184, Department of Computer Science, K.U. Leuven, Belgium, 1992.
7. A. Bunse-Gerstner and L. Elsner, *Schur Parameter Pencils for the Solution of the Unitary Eigenproblem*, Lin. Alg. and its Appl. **154 - 156** (1991), 741 – 778. MR **92c:**65048
8. T. F. Chan, *Rank-Revealing QR Factorizations*, Lin. Alg. and its Applic. **88/89** (1987), 67 – 82. MR **88c:**15011
9. C. J. Demeure, *Fast QR factorization of Vandermonde matrices*, Lin. Alg. and its Appl. **122 - 124** (1989), 165 – 194. MR **91a:**65068
10. J. Dongarra, J.R. Bunch, C. Moler, and G.W. Stewart, *Linpack user's guide*, SIAM, Philadelphia, PA, 1979.
11. H. Faßbender, *Numerische Verfahren zur diskreten trigonometrischen Polynomapproximation*, Dissertation Universität Bremen, 1993.
12. ———, *Inverse unitary eigenproblems and related orthogonal functions*, to appear in Numerische Mathematik.
13. ———, *A note on Newbery's algorithm for computing discrete least-squares approximation by trigonometric polynomials*, Electron. Trans. Numer. Anal. **4** (1996), 64–74. CMP 96:16
14. G. H. Golub and C. F. Van Loan, *Matrix Computation*, second ed., The John Hopkins University Press, 1989. MR **90d:**65055
15. P. Henrici, *Applied and Computational Analysis*, vol. 3, Wiley, 1986. MR **87h:**30002
16. A. C. R. Newbery, *Trigonometric Interpolation and Curve-Fitting*, Math. Comp. **24** (1970), 869 – 876. MR **43:**5687
17. L. Reichel, G. S. Ammar, and W. B. Gragg, *Discrete Least Squares Approximation by Trigonometric Polynomials*, Math. Comp. **57** (1991), 273 – 289. MR **91j:**65027

Universität Bremen, Fachbereich 3 Mathematik und Informatik, 28334 Bremen, Germany

E-mail address: heike@mathematik.uni-bremen.de