Electronic Transactions on Numerical Analysis.
Volume 31, pp. 178-203, 2008.

ISSN 1068-9613.

ETNA
Kent State University
http://etna.math.kent.edu

APPROXIMATION OF THE SCATTERING AMPLITUDE AND LINEAR SYSTEMS*

GENE H. GOLUB†, MARTIN STOLL‡, AND ANDY WATHEN‡

Abstract. The simultaneous solution of $Ax = b$ and $A^T y = g$, where A is a non-singular matrix, is required in a number of situations. Darmofal and Lu have proposed a method based on the Quasi-Minimal Residual algorithm (QMR). We will introduce a technique for the same purpose based on the LSQR method and show how its performance can be improved when using the generalized LSQR method. We further show how preconditioners can be introduced to enhance the speed of convergence and discuss different preconditioners that can be used. The scattering amplitude $g^T x$, a widely used quantity in signal processing for example, has a close connection to the above problem since x represents the solution of the forward problem and g is the right-hand side of the adjoint system. We show how this quantity can be efficiently approximated using Gauss quadrature and introduce a block-Lanczos process that approximates the scattering amplitude, and which can also be used with preconditioning.

Key words. Linear systems, Krylov subspaces, Gauss quadrature, adjoint systems.

AMS subject classifications. 65F10, 65N22, 65F50, 76D07.

1. Introduction. Many applications require the solution of a linear system

$$Ax = b,$$

with $A \in \mathbb{R}^{n\times n}$; see [8]. This can be done using different solvers, depending on the properties of the underlying matrix. A direct method based on the LU factorization is typically the method of choice for small problems. With increasing matrix dimensions, the need for iterative methods arises; see [25, 39] for more details. The most popular of these methods are the so-called Krylov subspace solvers, which use the space

$$\mathcal{K}_k(A, r_0) = \mathrm{span}(r_0, Ar_0, A^2 r_0, \ldots, A^{k-1} r_0)$$

to find an appropriate approximation to the solution of the linear system. In the case of a symmetric matrix we would use CG [26] or MINRES [32], which also guarantee some optimality conditions for the current iterate in the existing Krylov subspace. For a nonsymmetric matrix A it is much harder to choose the best-suited method. GMRES is the most stable Krylov subspace solver for this problem, but has the drawback of being very expensive, due to large storage requirements and the fact that the amount of work per iteration step is increasing. There are alternative short-term recurrence approaches, such as BICG [9], BICGSTAB [46], QMR [11], . . . , mostly based on the nonsymmetric Lanczos process. These methods are less reliable than the ones used for symmetric systems, but can nevertheless give very good results.

In many cases we are not only interested in the solution of the forward linear system

$$Ax = b, \tag{1.1}$$

but also of the adjoint system

$$A^T y = g \tag{1.2}$$

*Received November 11, 2007. Accepted October 1, 2008. Published online on February 24, 2009. Recommended by Zdeněk Strakoš.

†Department of Computer Science, Stanford University, Stanford, CA94305-9025, USA.

‡Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom ({martin.stoll,andy.wathen}@comlab.ox.ac.uk).

ETNA
Kent State University
http://etna.math.kent.edu

simultaneously. In [14] Giles and Süli provide an overview of the latest developments regarding adjoint methods with an excellent list of references. The applications given in [14] are widespread: optimal control and design optimization in the context of fluid dynamics, aeronautical applications, weather prediction and data assimilation, and many more. They also mention a more theoretical use of adjoint equations, regarding a posteriori error estimation for partial differential equations.

In signal processing, the scattering amplitude $g^T x$ connects the adjoint right-hand side and the forward solution. For a given vector g this means that $Ax = b$ determines the field x from the signal b. This signal is then received on an antenna characterised by the vector g which is the right-hand side of the adjoint system $A^T y = g$, and can be expressed as $g^T x$. This is of use when one is interested in what is reflected when a radar wave is impinging on a certain object; one typical application is the design of stealth planes. The scattering amplitude also arises in nuclear physics [2], quantum mechanics [28] and CFD [13].

The scattering amplitude is also known in the context of optimization as the primal linear output of a functional

$$J^{pr}(x) = g^T x, \tag{1.3}$$

where x is the solution of (1.1). The equivalent formulation of the dual problem results in the output

$$J^{du}(y) = y^T b, \tag{1.4}$$

with y being the solution of the adjoint equation (1.2). In some applications the solution to the linear systems (1.1) and (1.2) is not required explicitly, but a good approximation to the primal and dual output is important. In [29] Darmofal and Lu introduce a QMR technique that simultaneously approximates the solutions to the forward and the adjoint system, and also gives good estimates for the values of the primal and dual functional output described in (1.3) and (1.4).

In the first part of this paper we describe the QMR algorithm followed by alternative approaches to compute the solutions to the linear systems (1.1) and (1.2) simultaneously, based on the LSQR and GLSQR methods. We further introduce preconditioning for these methods and discuss different preconditioners.

In the second part of the paper we discuss how to approximate the scattering amplitude without computing a solution to the linear system. The principal reason for this approach, rather than computing x_k and then the inner product of g with x_k, relates to numerical stability: the analysis in Section 10 of [43] for Hermitian systems, and the related explanation in [45] for non-Hermitian systems, shows that approach to be sensitive in finite precision arithmetic, whereas our approach based on Gauss quadrature is more reliable. We briefly discuss a technique recently proposed by Strakoš and Tichý in [45] and methods based on BICG (cf. [9]) introduced by Smolarski and Saylor [41, 42], who indicate that there may be additional benefits in using Gauss quadrature for the calculation of the scattering amplitude in the context of high performance computing. Another paper concerned with the computation of the scattering amplitude is [21].

We conclude the paper by showing numerical experiments for the solution of the linear systems as well as for the approximation of the scattering amplitude by Gauss quadrature.

ETNA
Kent State University
http://etna.math.kent.edu

2. Solving the linear systems.

2.1. The QMR approach. In [29], Lu and Darmofal presented a technique using the standard QMR method to obtain an algorithm that would approximate the solution of the forward and the adjoint problem at the same time. The basis of QMR is the nonsymmetric Lanczos process (see [11, 47])

$$\begin{aligned} AV_k &= V_{k+1}T_{k+1,k}, \\ A^T W_k &= W_{k+1}\hat{T}_{k+1,k}. \end{aligned}$$

The nonsymmetric Lanczos algorithm generates two sequences V_k and W_k which are biorthogonal, i.e., $V_k^T W_k = I$. The matrices $T_{k+1,k}$ and $\hat{T}_{k+1,k}$ are of tridiagonal structure where the blocks $T_{k,k}$ and $\hat{T}_{k,k}$ are not necessarily symmetric. With the choice $v_1 = r_0/\left\|r_0\right\|$, where $r_0 = b - Ax_0$ and $x_k = x_0 + V_k c_k$, we can express the residual as

$$\|r_k\| = \|b - Ax_0 - AV_k c_k\| = \|r_0 - V_{k+1}T_{k+1,k}c_k\| = \|V_{k+1}(\|r_0\|\, e_1 - T_{k+1,k}c_k)\|\,.$$

This gives rise to the *quasi-residual* $r_k^Q = \|r_0\|\, e_1 - T_{k+1,k}c_k$, and we know that

$$\|r_k\| \leq \|V_{k+1}\|\, \|r_k^Q\|;$$

see [11, 25] for more details. The idea presented by Lu and Darmofal was to choose $w_1 = s_0/\left\|s_0\right\|$, where $s_0 = g - A^T y_0$ and $y_k = y_0 + W_k d_k$, to obtain the adjoint quasi-residual

$$\|s_k^Q\| = \left\|\|s_0\|\, e_1 - \hat{T}_{k+1,k}d_k\right\|$$

in a similar fashion to the forward quasi-residual. The two least-squares solutions $c_k, d_k \in \mathbb{R}^k$ can be obtained via an updated QR factorization; see [32, 11] for details. It is also theoretically possible to introduce weights to improve the convergence behaviour; see [11].

2.2. The bidiagonalization or LSQR approach. Solving

$$Ax = b, \qquad A^T y = g$$

simultaneously can be reformulated as solving

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ g \end{bmatrix}. \tag{2.1}$$

The coefficient matrix of system (2.1)

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \tag{2.2}$$

is symmetric and indefinite. Furthermore, it is heavily used when computing singular values of the matrix A and is also very important in the context of linear least squares problems. The main tool used for either purpose is the Golub-Kahan bidiagonalization (cf. [15]), which is also the basis for the well-known LSQR method introduced by Paige and Saunders in [34].

In more detail, we assume that the bidiagonal factorization

$$A = UBV^T \tag{2.3}$$

ETNA
Kent State University
http://etna.math.kent.edu

is given, where U and V are orthogonal and B is bidiagonal. Hence, we can express forward and adjoint systems as

$$UBV^Tx = b \qquad \text{and} \qquad VB^TU^Ty = g.$$

So far we have assumed that an explicit bidiagonal factorization (2.3) is given, which is a rather unrealistic assumption for large sparse matrices. In practice we need an iterative procedure that represents instances of the bidiagonalization process; cf. [15, 23, 34]. To achieve this, we use the following matrix structures

$$\begin{aligned} AV_k &= U_{k+1}B_k, \\ A^TU_{k+1} &= V_kB_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T, \end{aligned} \tag{2.4}$$

where $V_k = [v_1, \ldots, v_k]$ and $U_k = [u_1, \ldots, u_k]$ are orthogonal matrices and

$$B_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}.$$

The Golub-Kahan bidiagonalization is nothing else than the Lanczos process applied to the matrix A^TA, i.e., we multiply the first equation of (2.4) by A^T on the left, and then use the second to get the Lanczos relation for A^TA,

$$A^TAV_k = A^TU_{k+1}B_k = \left(V_kB_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T\right)B_k = V_kB_k^TB_k + \hat{\alpha}_{k+1}v_{k+1}e_{k+1}^T,$$

with $\hat{\alpha}_{k+1} = \alpha_{k+1}\beta_{k+1}$; see [4, 27] for details. The initial vectors of both sequences are linked by the relationship

$$A^Tu_1 = \alpha_1 v_1. \tag{2.5}$$

We now use the iterative process described in (2.4) to obtain approximations to the solutions of the forward and the adjoint problem. The residuals at step k can be defined as

$$r_k = b - Ax_k \tag{2.6}$$

and

$$s_k = g - A^Ty_k, \tag{2.7}$$

with

$$x_k = x_0 + V_kz_k \qquad \text{and} \qquad y_k = y_0 + U_{k+1}w_k.$$

A typical choice for u_1 would be the normalized initial residual $u_1 = r_0/\left\|r_0\right\|$. Hence, we get for the residual norms that

$$\begin{aligned} \|r_k\| &= \|b - Ax_k\| = \|b - A(x_0 + V_kz_k)\| = \|r_0 - AV_kz_k\| \\ &= \|r_0 - U_{k+1}B_kz_k\| = \left\| \|r_0\| e_1 - B_kz_k \right\|, \end{aligned}$$

ETNA
Kent State University
http://etna.math.kent.edu

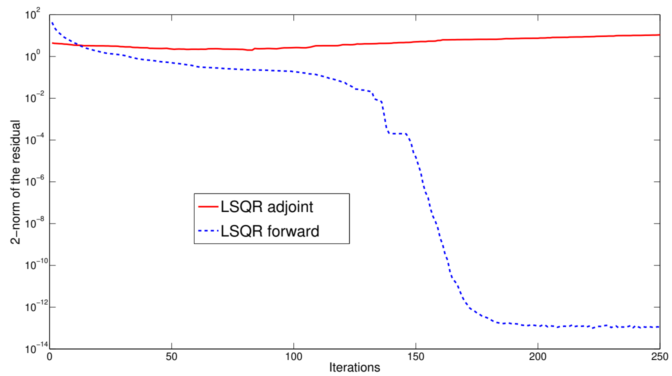

FIGURE 2.1. *Solving a linear system of dimension* 100×100 *with the* LSQR *approach.*

using (2.4) and the orthogonality of U_{k+1}. The adjoint residual can now be expressed as

$$\begin{aligned} \|s_k\| &= \left\|g - A^T y_k\right\| = \left\|g - A^T(y_0 + U_{k+1}w_k)\right\| \\ &= \left\|g - A^T y_0 - A^T U_{k+1} w_k\right\| \\ &= \left\|s_0 - V_k B_k^T w_k - \alpha_{k+1} v_{k+1} e_{k+1}^T w_k\right\|. \end{aligned} \tag{2.8}$$

Notice that (2.8) cannot be simplified to the desired structure $\left\|\|s_0\| e_1 - B_k^T w_k\right\|$, since the initial adjoint residual s_0 is not in the span of the current and all the following v_j vectors. This represents the classical approach LSQR [33, 34], where the focus is on obtaining an approximation that minimizes $\|r_k\| = \|b - Ax_k\|$. The method is very successful and widely used in practice, but is limited due to the restriction given by (2.5) in the case of simultaneous iteration for the adjoint problem. In more detail, we are not able to choose the second starting vector independently, and therefore cannot obtain the desired least squares structure obtained for the forward residual. Figure 2.1 illustrates the behaviour observed for all our examples with the LSQR method. Here, we are working with a random matrix of dimension 100×100. Convergence for the forward solution could be observed when a large number of iteration steps was executed, whereas the convergence for the adjoint residual could not be achieved at any point, which is illustrated by the stagnation of the adjoint solution. As already mentioned, this is due to the coupling of the starting vectors. In the next section we present a new approach that overcomes this drawback.

2.3. Generalized LSQR (GLSQR). The simultaneous computation of forward and adjoint solutions based on the classical LSQR method is not very successful, since the starting vectors u_1 and v_1 depend on each other through (2.5). In [40] Saunders et al. introduced a more general LSQR method which was also recently analyzed by Reichel and Ye [37]. Saunders and coauthors also mention in their paper that the method presented can be used to solve forward and adjoint problem at the same time. We will discuss this here in more detail and will also present a further analysis of the method described in [37, 40]. The method of interest makes it possible to choose the starting vectors u_1 and v_1 independently, namely, $u_1 = r_0/\|r_0\|$ and $v_1 = s_0/\|s_0\|$. The algorithm stated in [37, 40] is based on the following factorization

$$\begin{aligned} AV_k &= U_{k+1}T_{k+1,k} = U_k T_{k,k} + \beta_{k+1} u_{k+1} e_k^T, \\ A^T U_k &= V_{k+1} S_{k+1,k} = V_k S_{k,k} + \eta_{k+1} v_{k+1} e_k^T, \end{aligned} \tag{2.9}$$

ETNA
Kent State University
http://etna.math.kent.edu

where

$$V_k = [v_1, \ldots, v_k] \qquad \text{and} \qquad U_k = [u_1, \ldots, u_k]$$

are orthogonal matrices and

$$T_{k+1,k} = \begin{bmatrix} \alpha_1 & \gamma_1 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \gamma_{k-1} \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}, \qquad S_{k+1,k} = \begin{bmatrix} \delta_1 & \theta_1 & & \\ \eta_2 & \delta_2 & \ddots & \\ & \ddots & \ddots & \theta_{k-1} \\ & & \eta_k & \delta_k \\ & & & \eta_{k+1} \end{bmatrix}.$$

In the case of no *breakdown*[1], the following relation holds

$$S_{k,k}^T = T_{k,k}.$$

The matrix factorization given in (2.9) can be used to produce simple algorithmic statements of how to obtain new iterates for u_j and v_j:

$$\begin{aligned} \beta_{k+1} u_{k+1} &= A v_k - \alpha_k u_k - \gamma_{k-1} u_{k-1}, \\ \eta_{k+1} v_{k+1} &= A^T u_k - \delta_k v_k - \theta_{k-1} v_{k-1}. \end{aligned} \tag{2.10}$$

The parameters $\alpha_j, \gamma_j, \delta_j, \theta_j$ can be determined via the Gram-Schmidt orthogonalization process in the classical or the modified version. Furthermore, β_j and η_j are determined from the normalization of the vectors in (2.10).

Since it is well understood that the classical Golub-Kahan bidiagonalization process introduced in [15] can be viewed as the Lanczos algorithm applied to the matrix A^TA, we want to analyze whether a similar connection can be made for the GLSQR method given in [37, 40]. Note that if the Lanczos process is applied to the matrix (2.2) with starting vector $[u_1, 0]^T$, we get equivalence to the Golub-Kahan bidiagonalization; see [4, 27] for details.

The generalized LSQR method (GLSQR) given in [37, 40] looks very similar to the Lanczos process applied to the matrix (2.2) and we will now show that in general GLSQR can not be seen as a Lanczos process applied to this matrix. The Lanczos iteration then gives

$$\nu_{k+1} \begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \xi_k \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \varrho_{k-1} \begin{bmatrix} u_{k-1} \\ v_{k-1} \end{bmatrix}, \tag{2.11}$$

and the resulting recursions are

$$\begin{aligned} \nu_{k+1} u_{k+1} &= A v_k - \xi_k u_k - \varrho_{k-1} u_{k-1}, \\ \nu_{k+1} v_{k+1} &= A^T u_k - \xi_k v_k - \varrho_{k-1} v_{k-1}. \end{aligned}$$

The parameters ϱ_{k-1}, ξ_k and ν_{k+1} are related to the parameters from the GLSQR process via

$$\begin{aligned} \xi_k &= u_k^T A v_k + v_k^T A^T u_k = \alpha_k + \delta_k, \\ \varrho_{k-1} &= u_{k-1}^T A v_k + v_{k-1}^T A^T u_k = \gamma_{k-1} + \eta_{k-1}, \end{aligned}$$

and since the Lanczos process generates a symmetric tridiagonal matrix, we also get

$$\nu_{k+1} = \varrho_k = \gamma_k + \eta_k.$$

[1] We discuss breakdowns later in this section.

The orthogonality condition imposed by the symmetric Lanczos process ensures that

$$\begin{bmatrix} u_{k+1}^T & v_{k+1}^T \end{bmatrix} \begin{bmatrix} u_k \\ v_k \end{bmatrix} = 0,$$

which reduces to $u_{k+1}^T u_k + v_{k+1}^T v_k = 0$. This criteria would be fulfilled by the vectors coming from the GLSQR method, because it creates two sequences of orthonormal vectors. In general, the vectors coming from the symmetric Lanczos process do not satisfy $u_{k+1}^T u_k = 0$ and $v_{k+1}^T v_k = 0$.

In the following, we study the similarity of GLSQR and a special block-Lanczos method. In [40] a connection to a block-Lanczos for the matrix $A^T A$ was made. Here we will discuss a method based on the augmented matrix (2.2).

Hence, we assume the complete matrix decompositions

$$AV = UT \qquad \text{and} \qquad A^T U = VT^T,$$

with $S = T^T$. Using this relations, we can rewrite the linear system (2.1) as

$$\begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix} \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ g \end{bmatrix}. \tag{2.12}$$

We now introduce the perfect shuffle permutation

$$\Pi = [e_1, e_3, \ldots, e_2, e_4, \ldots] \tag{2.13}$$

and use Π to modify (2.12), obtaining

$$\begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \Pi^T \Pi \begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix} \Pi^T \Pi \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ g \end{bmatrix}. \tag{2.14}$$

We now further analyze the matrices given in (2.14). The first two matrices can also be written as

$$\mathcal{U} = \left[\begin{array}{ccc|ccc} | & | & | & | & | & | \\ u_1 & u_2 & \vdots & 0 & 0 & 0 \\ | & | & | & | & | & | \\ \hline | & | & | & | & | & | \\ 0 & 0 & 0 & v_1 & v_2 & \vdots \\ | & | & | & | & | & | \end{array}\right] \Pi^T = \left[\begin{array}{cccccc} | & | & | & | & | & | \\ u_1 & 0 & u_2 & 0 & \vdots & \vdots \\ | & | & | & | & | & | \\ \hline | & | & | & | & | & | \\ 0 & v_1 & 0 & v_2 & \vdots & \vdots \\ | & | & | & | & | & | \end{array}\right].$$

Next, we study the similarity transformation on

$$\begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix}$$

using Π, which results in

$$\mathcal{T} = \Pi \begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix} \Pi^T = \begin{bmatrix} \Theta_1 & \Psi_1^T & & & \\ \Psi_1 & \Theta_2 & \Psi_2^T & & \\ & \Psi_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \end{bmatrix}, \tag{2.15}$$

ETNA
Kent State University
http://etna.math.kent.edu

with

$$\Theta_i = \begin{bmatrix} 0 & \alpha_i \\ \alpha_i & 0 \end{bmatrix} \qquad \text{and} \qquad \Psi_i = \begin{bmatrix} 0 & \beta_{i+1} \\ \gamma_i & 0 \end{bmatrix}.$$

Using the properties of the LSQR method by Reichel and Ye [37], we see that the matrix $\mathcal{U}$ is an orthogonal matrix and furthermore that if we write $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}_2, \cdots]$, where

$$\mathcal{U}_i = \left[\begin{array}{cc} | & | \\ u_i & 0 \\ | & | \\ \hline | & | \\ 0 & v_i \\ | & | \end{array}\right],$$

then $\mathcal{U}_i^T \mathcal{U}_i = I$ for all i. Thus, one particular instance at step k of the reformulated method reduces to

$$\mathcal{U}_{k+1}\Psi_{k+1} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \mathcal{U}_k - \mathcal{U}_k\Theta_k - \mathcal{U}_{k-1}\Psi_{k-1}^T.$$

Hence, we have shown that the GLSQR method can be viewed as a special block-Lanczos method with stepsize 2; see [22, 23, 30] for more details on the block-Lanczos method.

2.4. GLSQR and linear systems. The GLSQR process analyzed above can be used to obtain approximate solutions to the linear system and the adjoint system. We are now able to set u_1 and v_1 independently and choose, for initial guesses x_0, y_0 and residuals $r_0 = b - Ax_0$, $s_0 = g - A^T y_0$,

$$u_1 = \frac{r_0}{\|r_0\|} \qquad \text{and} \qquad v_1 = \frac{s_0}{\|s_0\|}.$$

Hence, our approximations for the solution at each step are given by

$$x_k = x_0 + V_k z_k \tag{2.16}$$

for the forward problem and

$$y_k = y_0 + U_k w_k \tag{2.17}$$

for the linear system involving the adjoint. Using this and (2.9) we can express the residual at step k as follows: for the forward problem

$$\begin{aligned} \|r_k\| &= \|b - Ax_k\| = \|b - A(x_0 + V_k z_k)\| = \|r_0 - AV_k z_k\| \\ &= \|r_0 - U_{k+1}T_{k+1,k} z_k\| = \left\|U_{k+1}^T r_0 - T_{k+1,k} z_k\right\| \\ &= \left\|\|r_0\| e_1 - T_{k+1,k} z_k\right\| \end{aligned} \tag{2.18}$$

and, in complete analogy,

$$\begin{aligned} \|s_k\| &= \left\|g - A^T y_k\right\| = \left\|V_{k+1}^T s_0 - S_{k+1,k} w_k\right\| \\ &= \left\|\|s_0\| e_1 - S_{k+1,k} w_k\right\|. \end{aligned} \tag{2.19}$$

The solutions z_k and w_k can be obtained by solving the least squares systems (2.18) and (2.19), respectively. The QR factorization is a well known tool to solve least squares systems of the above form. We therefore have to compute the QR factorization of $T_{k+1,k}$ and $S_{k+1,k}$. The factorization can be updated at each step using just one Givens rotation. In more detail, we assume that the QR factorization of $T_{k,k-1} = Q_{k-1}R_{k-1}$ is given, with

$$R_{k-1} = \begin{bmatrix} \hat{R}_{k-1} \\ 0 \end{bmatrix}$$

and $\hat{R}_{k-1}$ an upper triangular matrix. To obtain the QR factorization of $T_{k+1,k}$ we eliminate the element β_{k+1} from

$$\begin{aligned} \begin{bmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{bmatrix} T_{k+1,k} &= \begin{bmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} T_{k,k-1} & \alpha_k e_k + \gamma_{k-1} e_{k-1} \\ 0 & \beta_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} R_{k-1} & Q_{k-1}^T(\alpha_k e_k + \gamma_{k-1} e_{k-1}) \\ 0 & \beta_{k+1} \end{bmatrix} \end{aligned} \tag{2.20}$$

by using one Givens rotation. The same argument holds for the QR decomposition of the matrix $S_{k+1,k}$. Thus, we have to compute two Givens rotations at every step to solve the systems (2.18) and (2.19) efficiently. There is no need to store the whole basis V_k or U_k in order to update the solution as described in (2.16) and (2.17); see also [25]. The matrix R_k of the QR decomposition of the tridiagonal matrix $T_{k+1,k}$ has only three non-zero diagonals. Let us define $C_k = [c_0, c_1, \ldots, c_{k-1}] = V_k \hat{R}_k^{-1}$. Note that c_0 is a multiple of v_1 and we can compute successive columns using $C_k \hat{R}_k = V_k$, i.e.,

$$c_{k-1} = (v_k - \hat{r}_{k-1,k} c_{k-2} - \hat{r}_{k-2,k} c_{k-3}) / \hat{r}_{k,k}, \tag{2.21}$$

where the $\hat{r}_{i,j}$ are elements of $\hat{R}_k$. Therefore, we can update the solution

$$x_k = x_0 + \|r_0\| \, C_k \left(Q_k^T e_1\right)_{k \times 1} = x_{k-1} + a_{k-1} c_{k-1}, \tag{2.22}$$

where a_{k-1} is the kth entry of $\|r_0\| \, Q_k^T e_1$.

The storage requirements for the GLSQR method are similar to the storage requirements for a method based on the non-symmetric Lanczos process, as proposed by Lu and Darmofal [29]. We need to store the vectors u_j, v_j, u_{j-1}, and v_{j-1}, to generate the basis vectors for the next Krylov space. Furthermore, we need to store the sparse matrices $T_{k+1,k}$ and $S_{k+1,k}$. This can be done in a parameterized fashion (remember that they are tridiagonal matrices) and since $T_{k,k} = S_{k,k}^T$, until the first breakdown occurs, the storage requirement can be reduced even further. The triangular factors of $T_{k+1,k}$ and $S_{k+1,k}$ can also be stored very efficiently, since they only have three nonzero diagonals. According to (2.21) the solutions x_k and y_k can be updated storing only two vectors c_{k-2} and c_{k-3} for the forward problem, and another two vectors for the adjoint solution. Thus the solutions can be obtained by storing only a minimal amount of data in addition to the original problem.

In [37], Reichel and Ye solve the forward problem and introduce the term *breakdown* in the case that the matrix $S_{k+1,k}$ associated with the adjoint problem has a zero entry on the subdiagonal. Note that until a breakdown occurs it is not necessary to distinguish between the parameters of the forward and adjoint sequence, since $T_{k,k} = S_{k,k}^T$. We will discuss these breakdowns and show that they are indeed *lucky breakdowns*, which means that the solution can be found in the current space. When the breakdown occurs, we assume that the parameter $\beta_{k+1} = 0$ whereas $\eta_{k+1} \neq 0$, in which case Reichel and Ye proved in [29, Theorem 2.2] that the solution x_k for the forward problem can be obtained via $x_k = x_0 + \|r_0\| \, V_k T_{k,k}^{-1} e_1$. The same holds if $\beta_{k+1} \neq 0$ whereas $\eta_{k+1} = 0$, in which case the solution y_k can be obtained via $y_k = y_0 + \|s_0\| U_k S_{k,k}^{-1} e_1$. In the case when $\beta_{k+1} = 0$ and $\eta_{k+1} = 0$ at the same time, both problems are solved and we can stop the algorithm. Note that this is in contrast to the breakdowns that can occur in the non-symmetric Lanczos process.

In both cases, we have to continue the algorithm since only the solution to one of the two problems is found. Without loss of generality, we assume that $\beta_{k+1} = 0$ whereas $\eta_{k+1} \neq 0$, which means that the forward problem has already been solved. Considering that now we have

$$\beta_{k+1} u_{k+1} = 0 = Av_k - \alpha_k u_k - \gamma_{k-1} u_{k-1},$$

we can use

$$\alpha_{k+1} u_{k+1} = Av_{k+1} - \gamma_k u_k$$

to compute u_{k+1}, a strategy implicitly proposed by Reichel and Ye in [37].

From the point where the breakdown occurs, the band structure of the matrix $T_{k+1,k}$ would not be tridiagonal anymore, but rather upper bidiagonal since we are computing the vector u_{k+1} based on $\alpha_{k+1} u_{k+1} = Av_{k+1} - \gamma_k u_k$. There is no need to update the solution x_k in further steps of the method. The vectors u_{k+1} generated by this two-term recurrence are used to update the solution for the adjoint problem in a way we will now describe. First, we obtain a new basis vector v_{j+1}

$$\eta_{j+1} v_{j+1} = A^T u_j - \delta_j v_j - \theta_{j-1} v_{j-1}$$

and then update the QR factorization of $S_{k+1,k}$ to get a new iterate y_k. If the parameter $\eta_{j+1} = 0$, the solution for the adjoint problem is found and the method can be terminated. In the case of the parameter α_{k+1} becoming zero, the solution for the adjoint problem can be obtained using the following theorem, which stands in complete analogy to Theorem 2.3 in [37].

THEOREM 2.1. *We assume that* GLSQR *does not break down until step m of the algorithm. At step m we get $\beta_{m+1} = 0$ and $\eta_{m+1} \neq 0$, which corresponds to the forward problem being solved. The process is continued for $k \geq m$ with the updates*

$$\alpha_{k+1} u_{k+1} = Av_{k+1} - \gamma_k u_k$$

and

$$\eta_{k+1} v_{k+1} = A^T u_k - \delta_k v_k - \theta_{k-1} v_{k-1}.$$

If the breakdown occurs at step k, the solution of the adjoint problem can now be obtained from one of the following two cases:

1. *if the parameter $\eta_{k+1} = 0$, then the adjoint solution is given by*

$$y_k = y_0 + \|s_0\| U_k S_{k,k}^{-1} e_1;$$

2. *if the parameter $\alpha_{k+1} = 0$, then the adjoint problem can be recovered using*

$$y_k = y_0 + U_k w_k.$$

Proof. The proof of the first point is trivial since, for $\eta_{k+1} = 0$, the least squares error in

$$\min_{w \in \mathbb{R}^k} \left\| \|r_0\| e_1 - S_{k+1,k} w_k \right\|$$

is equal to zero. For the second point, we note that the solution w_k to the least squares problem

$$\min_{w\in\mathbb{R}^k} \left\| \|r_0\| \, e_1 - S_{k+1,k} w_k \right\|$$

satisfies the following relation

$$S_{k+1,k}^T \left(\|s_0\| \, e_1 - S_{k+1,k} w_k \right) = 0.$$

The breakdown with $\alpha_{k+1} = 0$ results in

$$\alpha_{k+1} u_{k+1} = 0 = A v_{k+1} - \gamma_k u_k,$$

which means that no new u_{k+1} is generated in this step. In matrix terms we get

$$\begin{aligned} AV_{k+1} &= U_k T_{k,k+1}, \\ A^T U_k &= V_{k+1} S_{k+1,k}. \end{aligned}$$

This results in,

$$\begin{aligned} A(g - A^T y) &= A(s_0 - A^T U_k w_k) = A(s_0 - V_{k+1} S_{k+1,k} w_k) \\ &= As_0 - AV_{k+1} S_{k+1,k} w_k = \|s_0\| \, AV_{k+1} e_1 - AV_{k+1} S_{k+1,k} w_k \\ &= \|s_0\| \, U_k T_{k,k+1} e_1 - U_k T_{k,k+1} S_{k+1,k} w_k \\ &= U_k T_{k,k+1} \left(\|s_0\| \, e_1 - S_{k+1,k} w_k \right) \\ &= U_k S_{k+1,k}^T \left(\|s_0\| \, e_1 - S_{k+1,k} w_k \right) = 0, \end{aligned}$$

using the fact that $S_{k+1,k}^T = T_{k,k+1}$; see Theorem 2.1 in [37]. Due to the assumption that A is nonsingular the solution for the adjoint problem is given by $y_k = y_0 + U_k w_k$. □

This theorem shows that the GLSQR method is a well-suited process to find the solution of the forward and adjoint problems at the same time. The breakdowns that may occur in the algorithm are all benign, which underlines the difference to methods based on the nonsymmetric Lanczos process. In order to give better reliability of the method based on the nonsymmetric Lanczos process, look-ahead strategies have to be implemented; cf. [10, 36].

2.5. Preconditioned GLSQR. In practice the GLSQR method can show slow convergence, and therefore has to be enhanced using preconditioning techniques. We assume the preconditioner $M = M_1 M_2$ is given. Note that in general $M_1 \neq M_2$. The preconditioned matrix is now

$$\widehat{A} = M_1^{-1} A M_2^{-1},$$

and its transpose is given by

$$\widehat{A}^T = M_2^{-T} A^T M_1^{-T}.$$

Since we do not want to compute the matrix $\widehat{A}$, we have to rewrite the GLSQR method

$$\begin{aligned} \beta_{j+1} u_{j+1} &= M_1^{-1} A M_2^{-1} v_j - \alpha_j u_j - \gamma_{j-1} u_{j-1}, \\ \eta_{j+1} v_{j+1} &= M_2^{-T} A^T M_1^{-T} u_j - \delta_j v_j - \theta_{j-1} v_{j-1}, \end{aligned} \tag{2.23}$$

to obtain an efficient implementation of the preconditioned procedure, i.e.,

$$\begin{aligned} \beta_{j+1} M_1 u_{j+1} &= A M_2^{-1} v_j - \alpha_j M_1 u_j - \gamma_{j-1} M_1 u_{j-1}, \\ \eta_{j+1} M_2^T v_{j+1} &= A^T M_1^{-T} u_j - \delta_j M_2^T v_j - \theta_{j-1} M_2^T v_{j-1}. \end{aligned} \tag{2.24}$$

If we set $p_j = M_1 u_j$, $M_2 \hat{q}_j = v_j$, $q_j = M_2^T v_j$, and $M_1^T \hat{p}_j = u_j$, we get

$$\begin{aligned} \beta_{j+1} p_{j+1} &= A\hat{q}_j - \alpha_j p_j - \gamma_{j-1} p_{j-1}, \\ \eta_{j+1} q_{j+1} &= A^T \hat{p}_j - \delta_j q_j - \theta_{j-1} q_{j-1}, \end{aligned} \tag{2.25}$$

with the following updates

$$\hat{q}_j = M_2^{-1} v_j = M_2^{-1} M_2^{-T} q_j, \tag{2.26}$$
$$\hat{p}_j = M_1^{-T} u_j = M_1^{-T} M_1^{-1} p_j. \tag{2.27}$$

We also want to compute the parameters α_j, γ_{j-1}, δ_j, and θ_{j-1}, which can be expressed in terms of the vectors $\hat{p}_j$, $\hat{q}_j$, p_j, and q_j. Namely, we get

$$\begin{aligned} \alpha_j &= (\widehat{A} v_j, u_j) = (A\hat{q}_j, \hat{p}_j), \\ \gamma_{j-1} &= (\widehat{A} v_j, u_{j-1}) = (A\hat{q}_j, \hat{p}_{j-1}), \\ \delta_j &= (\widehat{A}^T u_j, v_j) = (A^T \hat{p}_j, \hat{q}_j), \\ \theta_{j-1} &= (\widehat{A}^T u_j, v_{j-1}) = (A^T \hat{p}_j, \hat{q}_{j-1}), \end{aligned}$$

which can be computed cheaply. Note, that we need to evaluate $A^T \hat{p}_j$ and $A\hat{q}_j$ once in every iteration step. The parameters β_{j+1} and η_{j+1} can be computed using equations (2.26) and (2.27); see Algorithm 1 for a summary of this method.

ALGORITHM 1 (Preconditioned GLSQR).

for $k = 0, 1, \dots$ **do**
  Solve $(M_2^T M_2)\hat{q}_j = q_j$
  Solve $(M_1 M_1^T)\hat{p}_j = p_j$
  Compute $A\hat{q}_j$.
  Compute $\alpha_j = (A\hat{q}_j, \hat{p}_j)$ and $\gamma_{j-1} = (A\hat{q}_j, \hat{p}_{j-1})$.
  Compute β_{j+1} and p_{j+1} via $\beta_{j+1} p_{j+1} = A\hat{q}_j - \alpha_j p_j - \gamma_{j-1} p_{j-1}$
  Compute $A^T \hat{p}_j$
  Compute $\delta_j = (A^T \hat{p}_j, \hat{q}_j)$ and $\theta_{j-1} = (A^T \hat{p}_j, \hat{q}_{j-1})$.
  Compute η_{j+1} and q_{j+1} via $\eta_{j+1} q_{j+1} = A^T \hat{p}_j - \delta_j q_j - \theta_{j-1} q_{j-1}$
end for

This enables us to compute the matrices $T_{k+1,k}$ and $S_{k+1,k}$ efficiently. Hence, we can update the QR factorizations in every step using one Givens rotation for the forward problem and one for the adjoint problem. The solutions x_k and y_k can then be updated without storing the whole Krylov space, but with a recursion similar to (2.22). The norm of the preconditioned residual can be computed via the well known recursion

$$\|r_k\| = |\sin(\theta_k)| \, \|r_{k-1}\|,$$

where $\sin(\theta_k)$ is associated with the Givens rotation at step k. There are different preconditioning strategies for enhancing the spectral properties of A to make the GLSQR method converge faster. One possibility would be to use an incomplete LU factorization of A and then set $M_1 = L$ and $M_2 = U$; see [39] for more details.

Another technique is to use the fact that the GLSQR method is also a block-Lanczos method for the normal equations, i.e., the system matrix that has to be preconditioned is now

A^TA. We therefore consider preconditioning techniques that are well-suited for the normal equations.

One possibility would be to compute an incomplete Cholesky factorization of A^TA, but, since the matrix A^TA is typically less sparse than A and we never want to form the matrix A^TA explicitly, we consider preconditioners coming from an LQ decomposition of A. In [39] incomplete LQ preconditioners are discussed and used as a preconditioner to solve the system with AA^T. This strategy can be adopted when trying to find a solution to a system with A^TA.

Another approach is based on incomplete orthogonal factorizations, where a decomposition $A = QR + E$, with Q orthogonal and E the error term, is computed. There are different variants of this decomposition [3, 35] which result in a different structure of the matrix R. In the simple case of the so-called cIGO (column-Incomplete Givens Orthogonalization) method, where entries are only dropped based upon their position, we restrict R to have the same sparsity pattern as the original matrix A. We now use Q and R from the incomplete factorization and set $M_1 = Q$ and $M_2 = R$, which gives $\widehat{A} = Q^TAR^{-1}$ for the normal equations $\widehat{A}^T\widehat{A} = R^{-T}A^TQQ^TAR^{-1} = R^{-T}A^TAR^{-1}$. Hence, we can use R as a preconditioner for the normal equations and therefore for the GLSQR method.

3. Approximating the scattering amplitude.

In Section 2 we gave a detailed overview of how to compute the solution to the forward and adjoint linear system simultaneously. In the following, we present methods that allow the approximation of the scattering amplitude or primal output functional directly, without computing approximate solutions to the linear systems.

3.1. Matrices, moments and quadrature: an introduction.

In [18, 19] Golub and Meurant show how Gauss quadrature can be used to approximate

$$u^Tf(W)v,$$

where W is a symmetric matrix and f is some function, not necessarily a polynomial.

This can be done using the eigendecomposition $W = Q\Lambda Q^T$, with orthogonal Q, and we assume $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. As a result we get

$$u^Tf(W)v = u^TQf(\Lambda)Q^Tv. \tag{3.1}$$

By introducing $\alpha = Q^Tu$ and $\beta = Q^Tv$, we can rewrite (3.1) as

$$u^Tf(W)v = \alpha^Tf(\Lambda)\beta = \sum_{i=1}^{n} f(\lambda_i)\alpha_i\beta_i. \tag{3.2}$$

Formula (3.2) can be viewed as a Riemann-Stieltes integral

$$I[f] = u^Tf(W)v = \int_a^b f(\lambda)\, d\alpha(\lambda); \tag{3.3}$$

see [18] for more details. We can now express (3.3) as the quadrature formula

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{j=1}^{N} \omega_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f],$$

where the weights ω_j, v_k and the nodes t_j are unknowns and the nodes z_k are prescribed. Expressions for the remainder $R[f]$ can be found in [18], and for more details we recommend [5, 6, 12, 16, 17, 24]. We will see in the next section that, in the case of $u = v$, we can compute the weights and nodes of the quadrature rule by simply applying the Lanczos process to the symmetric matrix W; see [24]. Then, the eigenvalues of the tridiagonal matrix will represent the nodes of the quadrature rule, and the first component of the corresponding eigenvector can be used to compute the weights.

3.2. The Golub-Kahan bidiagonalization. The scattering amplitude or primal output $J^{pr}(x) = g^T x$ can now be approximated using the connection between Gauss quadrature and the Lanczos process. To be able to apply the theory of Golub and Meurant, we need the system matrix to be symmetric, which can be achieved by

$$J^{pr}(x) = g^T (A^T A)^{-1} A^T b = g^T (A^T A)^{-1} p = g^T f(A^T A) p, \tag{3.4}$$

using the fact that $x = A^{-1} b$ and $p = A^T b$. In order to use the Lanczos process to obtain nodes and weights of the quadrature formula, we need a symmetrized version of (3.4)

$$J^{pr}(x) = \frac{1}{4}\left[(p+g)^T (A^T A)^{-1} (p+g) - (g-p)^T (A^T A)^{-1} (g-p)\right].$$

Good approximations to $(p+g)^T (A^T A)^{-1} (p+g)$ and $(p-g)^T (A^T A)^{-1} (p-g)$ will result in a good approximation to the scattering amplitude. Here, we present the analysis for the Gauss rule (i.e., $M = 0$) where we apply the Lanczos process to $A^T A$ and get

$$A^T A V_N = V_N T_N + r_N e_N^T, \tag{3.5}$$

with orthogonal V_N and

$$T_N = \begin{bmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_N \\ & & \beta_N & \alpha_N \end{bmatrix}.$$

The eigenvalues of T_N determine the nodes of

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{j=1}^{N} \omega_j f(t_j) + R_G[f],$$

where $R_G[f]$ for the function $f(x) = \frac{1}{x}$ is given by

$$R_G[f] = \frac{1}{\eta^{2N+1}} \int_a^b \left[\prod_{j=1}^{N} (\lambda - t_j)\right]^2 d\alpha(\lambda).$$

Notice that, since the matrix $A^T A$ has only positive eigenvalues, the residual $R_G[f]$ will always be positive, and therefore the Gauss rule will always give an underestimation of the scattering amplitude.

The weights for the Gauss rule are given by the squares of the first elements of the normalized eigenvectors of T_N. Instead of applying the Lanczos process to $A^T A$, we can simply use the Golub-Kahan bidiagonalization procedure presented in Section 2.2. The matrix T_N can be trivially obtained from (2.4), via $T_N = B_N^T B_N$. Since T_N is tridiagonal and similar to a symmetric matrix, it is relatively cheap to compute its eigenvalues and eigenvectors.

ETNA
Kent State University
http://etna.math.kent.edu

In [20] Golub and Meurant further show that the evaluation of the expression

$$\sum_{j=1}^{N} \omega_j f(t_j)$$

can be simplified to

$$\sum_{j=1}^{N} \omega_j f(t_j) = e_1^T f(T_N) e_1,$$

which reduces to $e_1^T T_N^{-1} e_1$ for $f(x) = 1/x$. The last expression simply states that we have to find a good approximation for the $(1,1)$ element of the inverse of T_N. If we can find such a good approximation for $(T_N^{-1})_{(1,1)}$, the computation becomes much more efficient, since no eigenvalues or eigenvectors have to be computed to determine the Gauss quadrature rule. Another possibility is to solve the system $T_N z = e_1$, which is relatively cheap for the tridiagonal matrix T_N.

Golub and Meurant [18, 19] give bounds on the elements of the inverse using Gauss, Gauss-Radau, Gauss-Lobatto rules, depending on the Lanczos process. These bounds can then be used to give a good approximation to the scattering amplitude without solving a linear system with T_N or using its eigenvalues and eigenvectors. We will only give the bounds connected to the Gauss-Radau rule, i.e.,

$$\frac{t_{1,1} - b + \frac{s_1^2}{b}}{t_{1,1}^2 - t_{1,1} b + s_1^2} \leq (T_N^{-1})_{1,1} \leq \frac{t_{1,1} - a + \frac{s_1^2}{a}}{t_{1,1}^2 - t_{1,1} a + s_1^2},$$

with $s_1^2 = \sum_{j \neq 1} a_{j1}^2$, and $t_{i,j}$ the elements of T_N. These bounds are not sharp since they will improve with the number of Lanczos steps, and the approximation to the scattering amplitude will improve as the algorithm progresses. It is also possible to obtain the given bounds using variational principles; see [38]. In the case of CG applied to a positive definite matrix A, the $(1,1)$-element of T_N^{-1} can be easily approximated using

$$(T_N^{-1})_{(1,1)} = \frac{1}{\|r_0\|^2} \sum_{j=0}^{N-1} \alpha_j \|r_j\|^2,$$

where α_j and $\|r_j\|$ are given at every CG step. This formula is discussed in [1, 43, 44], where it is shown that it is numerically stable. From [43] we get that the remainder $R_G[f]$ in the Gauss quadrature where f is the reciprocal function, is equal to the error at step k of CG for the normal equations, i.e.,

$$\|x - x_k\|_{A^T A} / \|r_0\| = R_G[f].$$

Hence, the Golub-Kahan bidiagonalization can be used to approximate the error for CG for the normal equations [45].

3.3. Approximation using GLSQR (the block case). We now want to use a block method to estimate the scattering amplitude using GLSQR. The 2×2 matrix integral we are interested in is now

$$\begin{aligned} \int_a^b f(\lambda)\, d\alpha(\lambda) &= \begin{bmatrix} b^T & 0 \\ 0 & g^T \end{bmatrix} \begin{bmatrix} 0 & A^{-T} \\ A^{-1} & 0 \end{bmatrix} \begin{bmatrix} b & 0 \\ 0 & g \end{bmatrix} \\ &= \begin{bmatrix} 0 & b^T A^{-T} g \\ g^T A^{-1} b & 0 \end{bmatrix}. \end{aligned} \tag{3.6}$$

ETNA
Kent State University
http://etna.math.kent.edu

In [18], Golub and Meurant show how a block method can be used to generate quadrature formulae. In more detail, the integral $\int_a^b f(\lambda)\, d\alpha(\lambda)$ is now a 2×2 symmetric matrix, and the most general quadrature formula is of the form

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{i=1}^{k} C_j f(H_j) C_j + R[f], \tag{3.7}$$

with H_j and C_j being symmetric 2×2 matrices. Expression (3.7) can be simplified using

$$H_j = Q_j \Lambda_j Q_j^T,$$

where Q_j is the eigenvector matrix and Λ_j the 2×2 diagonal matrix containing the eigenvalues. Hence,

$$\sum_{i=1}^{k} C_j f(H_j) C_j = \sum_{i=1}^{k} C_j Q_j^T f(\Lambda_j) Q_j C_j,$$

and if we write $C_j Q_j^T f(\Lambda_j) Q_j C_j$ as

$$f(\lambda_1) z_1 z_1^T + f(\lambda_2) z_2 z_2^T,$$

where z_j is the j-th column of the matrix $C_j Q_j^T$. Consequently, we obtain for the quadrature rule

$$\sum_{i=1}^{2k} f(\lambda_j) z_j z_j^T,$$

where λ_j is a scalar and $z_j = \left[z_j^{(1)}, z_j^{(2)}\right]^T \in \mathbb{R}^2$. In [18], it is shown that there exist orthogonal matrix polynomials such that

$$\lambda p_{j-1}(\lambda) = p_j(\lambda) B_j + p_{j-1}(\lambda) D_j + p_{j-2}(\lambda) B_{j-1}^T,$$

with $p_0(\lambda) = I_2$ and $p_{-1}(\lambda) = 0$. We can write the last equation as

$$\lambda \left[p_0(\lambda), \ldots, p_{N-1}(\lambda)\right] = \left[p_0(\lambda), \ldots, p_{k-1}(\lambda)\right] \mathcal{T}_k + \left[0, \ldots, 0, p_N(\lambda) B_k\right]^T,$$

with

$$\mathcal{T}_k = \begin{bmatrix} D_1 & B_1^T & & & \\ B_1 & D_2 & B_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & B_{k-2} & D_{k-1} & B_{k-1}^T \\ & & & B_{k-1} & D_k \end{bmatrix},$$

which is a block-tridiagonal matrix. Therefore, we can define the quadrature rule as

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \sum_{i=1}^{2k} f(\theta_i) u_i u_i^T + R[f], \tag{3.8}$$

where $2k$ is the order of the matrix $\mathcal{T}_k$, θ_i the eigenvalues of $\mathcal{T}_k$, and u_i the vector consisting of the first two elements of the corresponding normalized eigenvector. The remainder $R[f]$ can be approximated using a Lagrange polynomial and we get

$$R[f] = \frac{f^{(2k)}(\eta)}{(2k)!} \int_a^b s(\lambda)\, d\alpha(\lambda),$$

ETNA
Kent State University
http://etna.math.kent.edu

where $s(x) = (x-\theta_1)(x-\theta_2)\dots(x-\theta_{2N})$. The sign of the function s is not constant over the interval $[a, b]$. Therefore, we cannot expect that the block-Gauss rule always underestimates the scattering amplitude. This might result in a rather oscillatory behavior. In [18], it is also shown that

$$\sum_{i=1}^{2k} f(\theta_i) u_i u_i^T = e^T f(\mathcal{T}_k) e,$$

with $e = (I_2, 0, \dots, 0)$. In order to use the approximation (3.8), we need a block-Lanczos algorithm for the matrix

$$\left[\begin{array}{cc} 0 & A \\ A^T & 0 \end{array}\right].$$

The GLSQR algorithm represents an implementation of a block-Lanczos method for this matrix and can therefore be used to create a block-tridiagonal matrix $\mathcal{T}_k$ as introduced in Section 2.3. Using this, we show in the second part of this section that we can compute an approximation to the integral given in (3.6). Hence, the scattering amplitude is approximated via

$$\sum_{i=1}^{2k} f(\lambda_i) u_i u_i^T \approx \left[\begin{array}{cc} 0 & g^T x \\ g^T x & 0 \end{array}\right]$$

without computing an approximation to x directly.

Further simplification of the above can be achieved following a result in [45]: since from (2.15)

$$\mathcal{T}_k = \Pi_{2k} \left[\begin{array}{cc} 0 & T_k \\ T_k^T & 0 \end{array}\right] \Pi_{2k}^T,$$

where Π_{2k} is the permutation (2.13) of dimension $2k$, in the case of the reciprocal function we have

$$\begin{aligned} e^T \mathcal{T}_k^{-1} e &= e^T \Pi_{2k} \left[\begin{array}{cc} 0 & T_k^{-T} \\ T_k^{-1} & 0 \end{array}\right] \Pi_{2k}^T e \\ &= \left[\begin{array}{cc} 0 & e_1^T T_k^{-T} e_1 \\ e_1^T T_k^{-1} e_1 & 0 \end{array}\right]. \end{aligned}$$

Note that with the settings $r_0 = b - Ax_0$ and $s_0 = g - A^T y_0$, the scattering amplitude can be written as

$$g^T A^{-1} b = s_0^T A^{-1} r_0 + s_0^T x_0 + y_0^T b.$$

With our choice of $x_0 = y_0 = 0$, we get that the scattering amplitude is approximated by $s_0^T A^{-1} r_0$. Starting the GLSQR block-Lanczos process with

$$\left[\begin{array}{cc} u_1 & 0 \\ 0 & v_1 \end{array}\right],$$

where $u_1 = r_0 / \|r_0\|$ and $v_1 = s_0 / \|s_0\|$, results in $v_1^T A^{-1} u_1 = e_1^T T_N^{-1} e_1$. An approximation to the scattering amplitude $g^T A^{-1} b$ is thus obtained via

$$s_0^T A^{-1} r_0 = \|r_0\| \, \|s_0\| \, e_1^T T_N^{-1} e_1.$$

ETNA
Kent State University
http://etna.math.kent.edu

3.4. Preconditioned GLSQR. The preconditioned GLSQR method was introduced in Section 2.5, and we now show that we can use this method to approximate the scattering amplitude directly. In the above we showed that GLSQR gives an approximation to the scattering amplitude using that

$$\int_a^b f(\lambda)\, d\alpha(\lambda) = \begin{bmatrix} 0 & \hat{x}^T g \\ g^T \hat{x} & 0 \end{bmatrix}.$$

Reformulating this in terms of the preconditioned method gives,

$$\begin{aligned} \hat{g}^T \hat{x} &= \hat{g}^T \widehat{A}^{-1} \hat{b} = (M_2^{-T} g)^T (M_1^{-1} A M_2^{-1})^{-1} (M_1^{-1} b) \\ &= g^T M_2^{-1} M_2 A^{-1} M_1 M_1^{-1} b = g^T A^{-1} b = g^T x, \end{aligned}$$

which shows that the scattering amplitude for the preconditioned system

$$\widehat{A}\hat{x} = \hat{b},$$

with $\widehat{A} = M_1^{-1} A M_2^{-1}$, $\hat{x} = M_2 x$ and $\hat{b} = M_1^{-1} b$, is equivalent to the scattering amplitude of the original system. The scattering amplitude can therefore be approximated via

$$\begin{bmatrix} 0 & g^T \hat{x} \\ \hat{x}^T g & 0 \end{bmatrix}.$$

3.5. BICG and the scattering amplitude. The methods we presented so far are based on Lanczos methods for $A^T A$. The algorithm introduced in this section connects BICG (see Algorithm 2) and [9], a method based on the nonsymmetric Lanczos process and the scattering amplitude.

ALGORITHM 2 (Biconjugate Gradient Method (BICG)).

for $k = 0, 1, \ldots$ **do**

$\alpha_k = \frac{s_k^T r_k}{q_k^T A p_k}$

$x_{k+1} = x_k + \alpha_k p_k$

$y_{k+1} = y_k + \alpha_k q_k$

$r_{k+1} = r_k - \alpha_k A p_k$

$s_{k+1} = s_k - \alpha_k A^T q_k$

$\beta_{k+1} = \frac{s_{k+1}^T r_{k+1}}{s_k^T r_k}$

$p_{k+1} = r_{k+1} + \beta_{k+1} p_k$

$q_{k+1} = s_{k+1} + \beta_{k+1} q_k$

end for

Using $r_j = b - Ax_j$ and $s_j = g - A^T y_j$, the scattering amplitude can be expressed as

$$g^T A^{-1} b = \sum_{j=0}^{N-1} \alpha_j s_j^T r_j + s_N^T A^{-1} r_N, \tag{3.9}$$

where N is the dimension of A; cf. [45]. To show this, we use $r_0 = b$, $s_0 = g$, and

$$\begin{aligned} s_j^T A^{-1} r_j - s_{j+1}^T A^{-1} r_{j+1} &= (g - A^T y_j)^T A^{-1} (b - A x_j) - s_{j+1}^T A^{-1} r_{j+1} \\ &= (g - A^T y_j + A^T y_{j+1} - A^T y_{j+1})^T A^{-1} (b - A x_j + A^T x_{j+1} - A^T x_{j+1}) \\ &\quad - s_{j+1}^T A^{-1} r_{j+1} \\ &= (s_{j+1} + A^T (y_{j+1} - y_j))^T A^{-1} (r_{j+1} + A(x_{j+1} - x_j)) - s_{j+1}^T A^{-1} r_{j+1} \\ &= \alpha_j (q_j^T r_{j+1} + s_{j+1}^T p_j + \alpha_j q_j^T A p_j) = \alpha_j s_j^T r_j, \end{aligned}$$

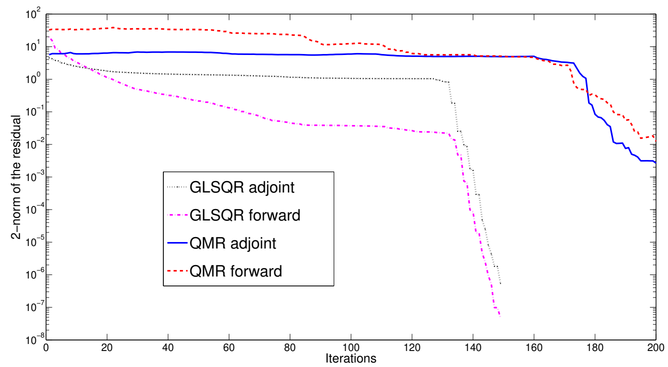

FIGURE 4.1. QMR *and* GLSQR *for a matrix of dimension* 100 *(Example 4.1).*

where we use $\alpha_j = \frac{\langle s_j, r_j \rangle}{\langle q_j, Ap_j \rangle}$. An approximation to the scattering amplitude at step k is then given by

$$g^T A^{-1} b \approx \sum_{j=0}^{k} \alpha_j s_j^T r_j. \tag{3.10}$$

It can be shown that (3.9) also holds for the preconditioned version of BICG with system matrix $\widehat{A} = M_1^{-1} A M_2^{-1}$ and preconditioned initial residuals $r_0 = M_1^{-1} b$ and $s_0 = M_2^{-T} g$.

Another way to approximate the scattering amplitude via BICG was given by Saylor and Smolarski [42, 41], in which the scattering amplitude is connected to Gaussian quadrature in the complex plane. The scattering amplitude is then given by

$$g^T A^{-1} b \approx \sum_{i=1}^{k} \frac{\omega_i}{\zeta_i}, \tag{3.11}$$

where ω_i and ζ_i are the eigenvector components and the eigenvalues, respectively, of the tridiagonal matrix associated with the appropriate formulation of BICG; see [41] for details. In [45] it is shown that (3.10) and (3.11) are mathematically equivalent. Note that, in a similar way to Section 3.4, it can be shown that the scattering amplitude of the preconditioned system is equivalent to the scattering amplitude of the preconditioned version of BICG.

4. Numerical experiments.

4.1. Solving the linear system. In this Section we want to show numerical experiments for the methods introduced in Section 2.

EXAMPLE 4.1. In the first example, we apply the QMR and the GLSQR methods to a random sparse matrix of dimension 100; e.g., `A=sprandn(n,n,0.2)+speye(n)` in Matlab notation. The maximal iteration number for both methods is 200, and it can be observed in Figure 4.1 that GLSQR outperforms QMR for this example.

EXAMPLE 4.2. The second example is the matrix *ORSIRR_1*, from the Matrix Market[2] collection, which represents a linear system used in oil reservoir modelling. The matrix size is 1030. The results without preconditioning are shown in Figure 4.2. Results using the Incomplete LU (ILU) factorization with zero fill-in as a preconditioner for GLSQR and QMR are

[2] http://math.nist.gov/MatrixMarket/

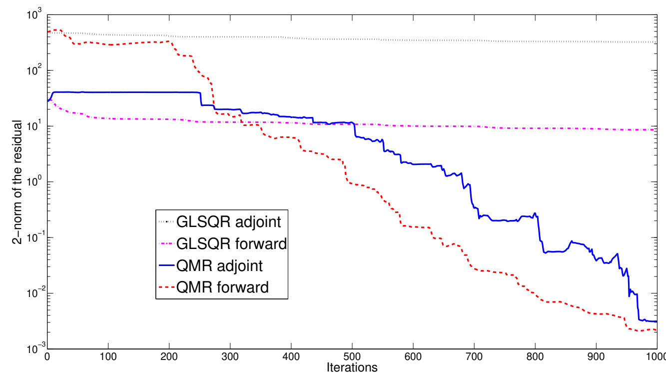

FIGURE 4.2. GLSQR *and* QMR *for the matrix: ORSIRR_1 (Example 4.2).*

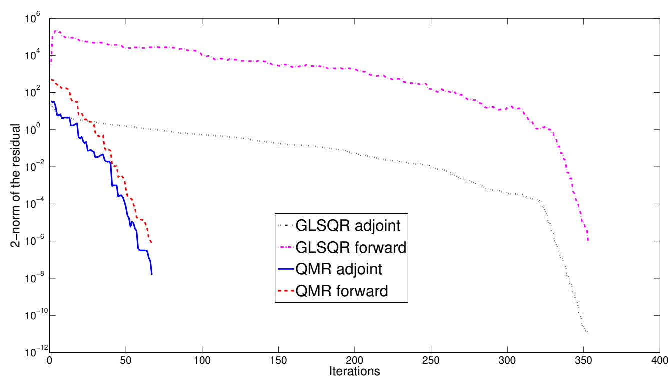

FIGURE 4.3. *ILU preconditioned* GLSQR *and* QMR *for the matrix: ORSIRR_1 (Example 4.2).*

given in Figure 4.3. Clearly, QMR outperforms GLSQR in both cases. The choice of using ILU as a preconditioner is mainly motivated by the fact that we are not aware of existing more sophisticated implementations of incomplete orthogonal factorizations or incomplete modified Gram-Schmidt decompositions that can be used in Matlab. Our tests with the basic implementations of cIGO and IMGS did not yield better numerical results than the ILU preconditioner, and we have therefore omitted these results in the paper. Nevertheless, we feel that further research in the possible use of incomplete orthogonal factorizations might result in useful preconditioners for GLSQR.

EXAMPLE 4.3. The next example is motivated by [31], where Nachtigal et al. introduce examples that show how different solvers for nonsymmetric systems can outperform others by a large factor. The original example in [31] is given by the matrix

$$J = \begin{bmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ 1 & & & 0 \end{bmatrix}.$$

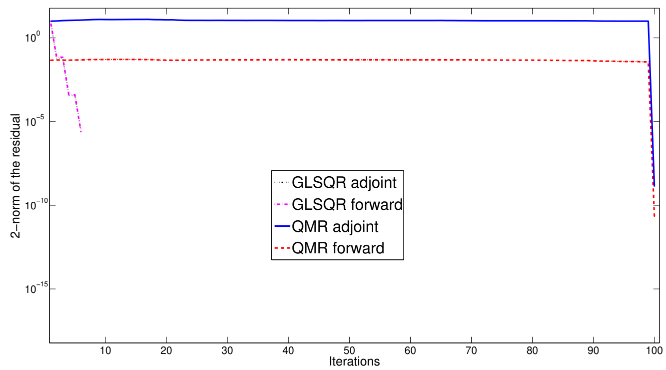

FIGURE 4.4. *Perturbed circulant shift matrix (Example 4.3).*

The results, shown in Figure 4.4, are for a sparse perturbation of the matrix J, i.e., in Matlab notation, `A=1e-3*sprandn(n,n,0.2)+J`. It is seen that QMR convergence for both forward and adjoint systems is slow, whereas GLSQR convergence is essentially identical for the forward and adjoint systems, and is rapid.

The convergence of GLSQR has not yet been analyzed, but we feel that using the connection to the block-Lanczos process for $A^T A$ we can try to look for similarities to the convergence of CG for the normal equations (CGNE). It is well known [31] that the convergence of CGNE is governed by the singular values of the matrix A. We therefore illustrate in the next example how the convergence of GLSQR is influenced by the distribution of the singular values of A. This should not be seen as a concise description of the convergence behaviour, but rather as a starting point for further research.

EXAMPLE 4.4. In this example we create a diagonal matrix $\Sigma = \operatorname{diag}(D_1, D_2)$ with

$$D_1 = \begin{bmatrix} 1000 & & \\ & \ddots & \\ & & 1000 \end{bmatrix} \in \mathbb{R}^{p,p} \quad \text{and} \quad D_2 = \begin{bmatrix} 1 & & & \\ & 2 & & \\ & & \ddots & \\ & & & q \end{bmatrix} \in \mathbb{R}^{q,q},$$

with $p + q = n$. We then create $A = U\Sigma V^T$, where U and V are orthogonal matrices. For $n = 100$ the results of GLSQR, for $D_1 \in \mathbb{R}^{90,90}$, $D_1 \in \mathbb{R}^{10,10}$, and $D_1 \in \mathbb{R}^{50,50}$, are given in Figure 4.5. It is seen that there is a better convergence when there are fewer distinct singular values. Figure 4.6 shows the comparison of QMR and GLSQR without preconditioning on an example with $n = 1000$ and D_1 of dimension 600; clearly GLSQR is superior in this example.

4.2. Approximating the functional. In this section we want to present results for the methods that approximate the scattering amplitude directly, avoiding the computation of approximate solutions for the linear systems with A and A^T.

EXAMPLE 4.5. In this example we compute the scattering amplitude using the preconditioned GLSQR approach for the oil reservoir example *ORSIRR_1*. The matrix size is 1030. We use the Incomplete LU (ILU) factorization as a preconditioner. The absolute values of the approximation from GLSQR are shown in the top part of Figure 4.7, while the bottom part shows the norm of the error against the number of iterations. Note that the non-monotonicity of the remainder term can be observed for the application of GLSQR .

ETNA
Kent State University
http://etna.math.kent.edu

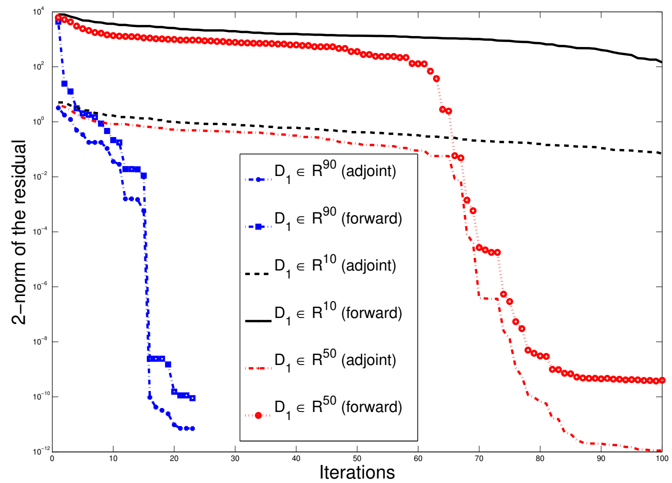

FIGURE 4.5. GLSQR *for different* D_1 *(Example 4.4).*

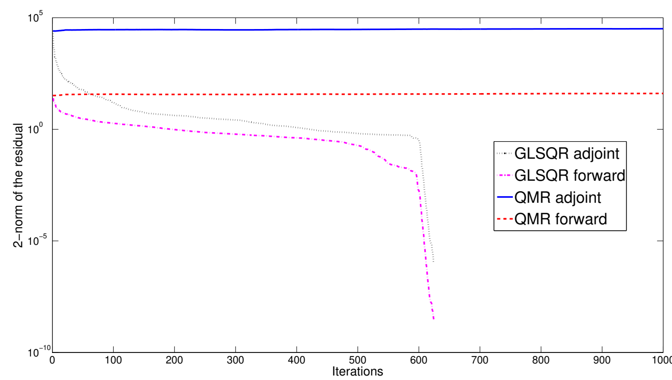

FIGURE 4.6. GLSQR *and* QMR *for matrix of dimension* 1000 *(Example 4.4).*

EXAMPLE 4.6. In this example we compute the scattering using the preconditioned BICG approach for the oil reservoir example *ORSIRR_1*. The matrix size is 1030. We use the Incomplete LU (ILU) factorization as a preconditioner. The absolute values of the approximation from BICG are shown in the top part of Figure 4.8, and the bottom part shows the norm of the error against the number of iterations.

EXAMPLE 4.7. In this example we compute the scattering amplitude by using the LSQR approach presented in Section 2.2. The test matrix is of size 187×187 and represents a Navier-Stokes problem generated by the IFISS package [7]. The result is shown in Figure 4.9, again with approximations in the top part and the error in the bottom part.

5. Conclusions. We studied the possibility of using LSQR for the simultaneous solution of forward and adjoint problems. Due to the link between the starting vectors of the two sequences, this method did not show much potential for a practical solver. As a remedy, we proposed to use the GLSQR method, which we carefully analyzed showing its relation to a

ETNA
Kent State University
http://etna.math.kent.edu

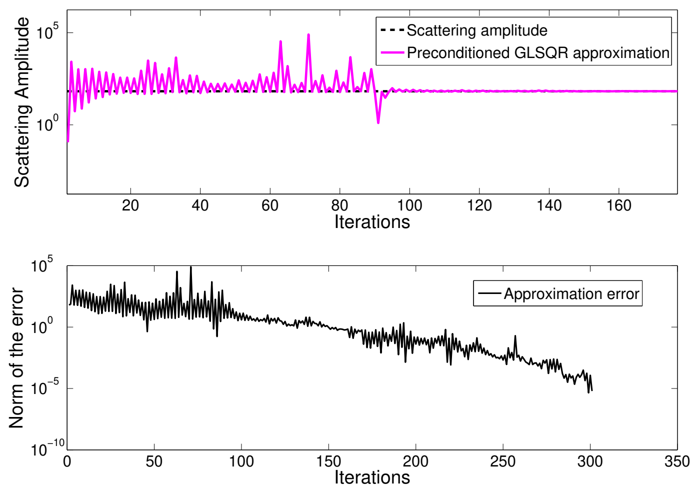

FIGURE 4.7. *Approximations to the scattering amplitude and error (Example 4.5).*

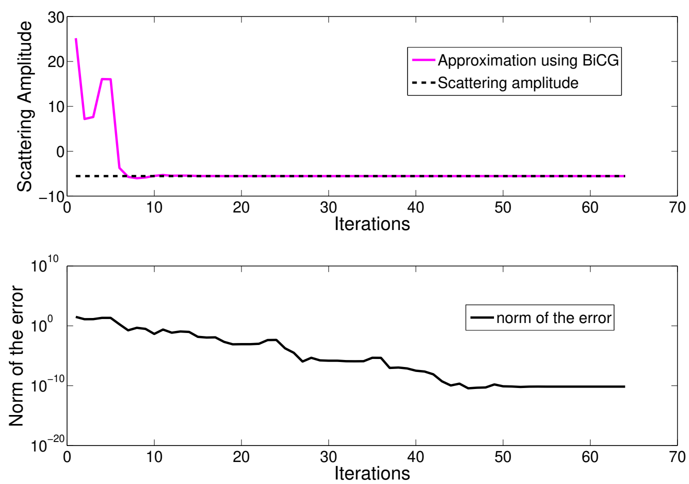

FIGURE 4.8. *Approximations to the scattering amplitude and error (Example 4.6).*

block-Lanczos method. Due to its special structure, we are able to choose the two starting vectors independently, and can therefore approximate the solutions for the forward and adjoint systems at the same time. Furthermore, we introduced preconditioning for the GLSQR method and proposed different preconditioners. We feel that more research has to be done to fully understand which preconditioners are well-suited for GLSQR, especially with regard to the experiments where different singular value distributions were used.

The approximation of the scattering amplitude, without first computing solutions to the linear systems, was introduced based on the Golub-Kahan bidiagonalization and its connection to Gauss quadrature. In addition, we showed how the interpretation of GLSQR as a block-Lanczos procedure can be used to allow approximations of the scattering amplitude

ETNA
Kent State University
http://etna.math.kent.edu

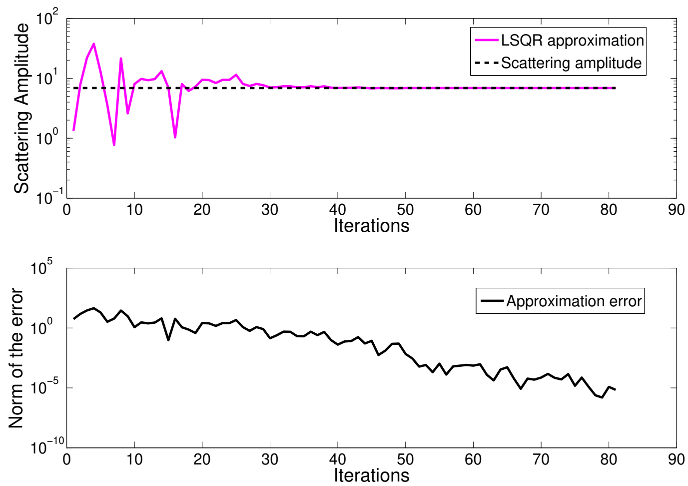

FIGURE 4.9. *Approximations to the scattering amplitude and error (Example 4.7).*

directly, by using the connection to block-Gauss quadrature.

We showed that for some examples the linear systems approach using GLSQR can outperform QMR, which is based on the nonsymmetric Lanczos process, and others where QMR performed better. We also showed how LSQR and GLSQR can be used to approximate the scattering amplitude on real world examples.

Acknowledgment. The authors would like to thank James Lu, Gerard Meurant, Michael Saunders, Zdeněk Strakoš, Petr Tichý, and an anonymous referee for their helpful comments.

Martin Stoll and Andy Wathen would like to mention what a wonderful time they had working on this paper with Gene Golub. He always provided new insight and a bigger picture. They hope wherever he is now he has access to ETNA to see the final result.

REFERENCES

[1] M. ARIOLI, *A stopping criterion for the conjugate gradient algorithms in a finite element method framework*, Numer. Math., 97 (2004), pp. 1–24.

[2] D. ARNETT, *Supernovae and Nucleosynthesis: An Investigation of the History of Matter, from the Big Bang to the Present*, Princeton University Press, 1996.

[3] Z.-Z. BAI, I. S. DUFF, AND A. J. WATHEN, *A class of incomplete orthogonal factorization methods. I. Methods and theories*, BIT, 41 (2001), pp. 53–70.

[4] Å. BJÖRCK, *A bidiagonalization algorithm for solving ill-posed system of linear equations*, BIT, 28 (1988), pp. 659–670.

[5] G. DAHLQUIST, S. C. EISENSTAT, AND G. H. GOLUB, *Bounds for the error of linear systems of equations using the theory of moments*, J. Math. Anal. Appl., 37 (1972), pp. 151–166.

[6] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, Computer Science and Applied Mathematics, Academic Press Inc, Orlando, second ed., 1984.

[7] H. C. ELMAN, A. RAMAGE, AND D. J. SILVESTER, *Algorithm 886: IFISS, a Matlab toolbox for modelling incompressible flow*, ACM Trans. Math. Software, 33 (2007), 14 (18 pages).

[8] H. C. ELMAN, D. J. SILVESTER, AND A. J. WATHEN, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005.

[9] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis (Proc 6th Biennial Dundee Conf., Univ. Dundee, Dundee, 1975), G. Watson, ed., vol. 506 of Lecture Notes in Math., Springer, Berlin, 1976, pp. 73–89.

[10] R. W. FREUND, M. H. GUTKNECHT, AND N. M. NACHTIGAL, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.
[11] R. W. FREUND AND N. M. NACHTIGAL, *QMR: a quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.
[12] W. GAUTSCHI, *Construction of Gauss-Christoffel quadrature formulas*, Math. Comp., 22 (1968), pp. 251–270.
[13] M. B. GILES AND N. A. PIERCE, *An introduction to the adjoint approach to design*, Flow, Turbulence and Combustion, 65 (2000), pp. 393–415.
[14] M. B. GILES AND E. SÜLI, *Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality*, Acta Numer., 11 (2002), pp. 145–236.
[15] G. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser. B Numer. Anal, 2 (1965), pp. 205–224.
[16] G. H. GOLUB, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.
[17] ———, *Bounds for matrix moments*, Rocky Mountain J. Math., 4 (1974), pp. 207–211.
[18] G. H. GOLUB AND G. MEURANT, *Matrices, moments and quadrature*, in Numerical Analysis 1993 (Dundee, 1993), D. Griffiths and G. Watson, eds., vol. 303 of Pitman Res. Notes Math. Ser., Longman Sci. Tech, Harlow, 1994, pp. 105–156.
[19] G. H. GOLUB AND G. MEURANT, *Matrices, moments and quadrature. II. How to compute the norm of the error in iterative methods*, BIT, 37 (1997), pp. 687–705.
[20] ———, *Matrices, Moments and Quadrature with Applications*. Draft, 2007.
[21] G. H. GOLUB, G. N. MINERBO, AND P. E. SAYLOR, *Nine ways to compute the scattering cross section (i): Estimating $c^T x$ iteratively*. Draft, 2007.
[22] G. H. GOLUB AND R. UNDERWOOD, *The block Lanczos method for computing eigenvalues*, in Mathematical software, III (Proc. Sympos., Math. Res. Center, Univ. Wisconsin, 1977), J. R. Rice, ed., Academic Press, New York, 1977, pp. 361–377.
[23] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, third ed., 1996.
[24] G. H. GOLUB AND J. H. WELSCH, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.
[25] A. GREENBAUM, *Iterative Methods for Solving Linear Systems*, vol. 17 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1997.
[26] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Stand, 49 (1952), pp. 409–436.
[27] I. HNĚTYNKOVÁ AND Z. STRAKOŠ, *Lanczos tridiagonalization and core problems*, Linear Algebra Appl., 421 (2007), pp. 243–251.
[28] L. D. LANDAU AND E. LIFSHITZ, *Quantum Mechanics*, Pergamon Press, Oxford, 1965.
[29] J. LU AND D. L. DARMOFAL, *A quasi-minimal residual method for simultaneous primal-dual solutions and superconvergent functional estimates*, SIAM J. Sci. Comput., 24 (2003), pp. 1693–1709.
[30] G. MEURANT, *Computer Solution of Large Linear Systems*, vol. 28 of Studies in Mathematics and its Applications, North-Holland, Amsterdam, 1999.
[31] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
[32] C. C. PAIGE AND M. A. SAUNDERS, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal, 12 (1975), pp. 617–629.
[33] C. C. PAIGE AND M. A. SAUNDERS, *Algorithm 583; LSQR: sparse linear equations and least-squares problems*, ACM Trans. Math. Software, 8 (1982), pp. 195–209.
[34] ———, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.
[35] A. T. PAPADOPOULOS, I. S. DUFF, AND A. J. WATHEN, *A class of incomplete orthogonal factorization methods. II. Implementation and results*, BIT, 45 (2005), pp. 159–179.
[36] B. N. PARLETT, D. R. TAYLOR, AND Z. A. LIU, *A look-ahead Lánczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.
[37] L. REICHEL AND Q. YE, *A generalized LSQR algorithm*, Numer. Linear Algebra Appl., 15 (2008), pp. 643–660.
[38] P. D. ROBINSON AND A. J. WATHEN, *Variational bounds on the entries of the inverse of a matrix*, IMA J. Numer. Anal., 12 (1992), pp. 463–486.
[39] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, 2003. Second edition.
[40] M. A. SAUNDERS, H. D. SIMON, AND E. L. YIP, *Two conjugate-gradient-type methods for unsymmetric linear equations*, SIAM J. Numer. Anal, 25 (1988), pp. 927–940.
[41] P. E. SAYLOR AND D. C. SMOLARSKI, *Why Gaussian quadrature in the complex plane?*, Numer. Algorithms, 26 (2001), pp. 251–280.

ETNA
Kent State University
http://etna.math.kent.edu

[42] ———, *Addendum to: "Why Gaussian quadrature in the complex plane?" [Numer. Algorithms* **26** *(2001), pp. 251–280]*, Numer. Algorithms, 27 (2001), pp. 215–217.
[43] Z. STRAKOŠ AND P. TICHÝ, *On error estimation in the conjugate gradient method and why it works in finite precision computations*, Electron. Trans. Numer. Anal., 13 (2002), pp. 56–80. http://etna.math.kent.edu/vol.13.2002/pp56-80.dir/.
[44] ———, *Error estimation in preconditioned conjugate gradients*, BIT, 45 (2005), pp. 789–817.
[45] ———, *Estimation of $c^* A^{-1} b$ via matching moments*. Submitted, 2008.
[46] H. A. VAN DER VORST, *BiCGSTAB: A fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.
[47] H. A. VAN DER VORST, *Iterative Krylov Methods for Large Linear Systems*, vol. 13 of Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2003.