Report no. 07/22

Approximation of the scattering amplitude

Gene H. Golub
Department of Computer Science, Stanford University,
Stanford, CA94305-9025, USA
golub@stanford.edu

Martin Stoll and Andy Wathen
Oxford University Computing Laboratory, Numerical Analysis Group,
Wolfson Building, Parks Road, Oxford OX1 3QD, U.K.,
{martin.stoll,andy.wathen}@comlab.ox.ac.uk

The simultaneous solution of $Ax = b$ and $A^T y = g$ 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 preconditioners.

Key words and phrases: Linear systems, Krylov subspaces, Gauss quadrature, Adjoint systems

Oxford University Computing Laboratory
Numerical Analysis Group
Wolfson Building
Parks Road
Oxford, England OX1 3QD

November, 2007

1 Introduction

Many applications require the solution of a linear system

$$Ax = b,$$

see [7]. 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 smaller problems. With increasing matrix dimensions the need for iterative methods arises, see [24,36] for more details. The most popular of these methods are the so-called Krylov subspace solvers which use the space

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

to find an appropriate approximation to the solution of the linear system. In the case of a symmetric matrix we would use CG [25] or MINRES [30] 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 amount of work per iteration step is increasing. There are alternative short-term recurrence approaches such as BICG [8], BICGSTAB [5] , QMR [10], ... 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

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

simultaneously. In [13] Giles and Süli provide an overview of the latest developments regarding adjoint methods with an excellent list of references. The applications given in [13] are widespread, ie. 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 the world of signal processing the scattering amplitude $g^T x$ connects the adjoint right hand side and the forward solution and can be viewed as the signal received by an antenna. In [38, 39] Smolarski and Saylor relate the scattering amplitude to Gaussian quadrature in the complex plane. We will discuss the methods introduced by Smolarski and Saylor in the course of this paper. Another paper concerned with the computation of the scattering amplitude is [21].

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 Equation 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 are of utmost importance. In [27] Darmofal and Lu introduce a QMR technique that simultaneously approximates the solutions to the forward and the adjoint system at the same time and also gives good estimates for the values of the primal and dual functional output described in (1.3) and (1.4).

The scattering amplitude also arises in nuclear physics [1], quantum mechanics [26] and CFD [12].

In the first part of this paper we will 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 will further introduce preconditioning for these methods and will discuss different preconditioners. The second part of this paper deals with the approximation of the outputs of the the primal and the dual functional as given in (1.3) and (1.4). We will conclude the paper by showing numerical experiments for the solution of the linear systems as well as for the approximation of the scattering amplitude in a direct way.

2 Solving the linear system

2.1 The QMR approach

In [27], 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 [40],

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

With the choice of $v_1 = r_0/\|r_0\|$ and $w_1 = s_0/\|s_0\|$ we can express the norm of forward and backward quasi-residual as

$$\|r_k\| = \|\|r_0\| e_1 - H_{k+1,k}y_k\| \text{ and } \|s_k\| = \left\|\|s_0\| e_1 - \hat{H}_{k+1,k}w_k\right\|.$$

It is also possible to introduce weights to improve the convergence behaviour, see [10].

2.2 The bidiagonalization or LSQR approach

Solving

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

simultaneously can be reformulated as

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

The system matrix of (2.1)

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

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. [14]) which is also the basis for the well-known LSQR method introduced by Paige and Saunders in [31].

In more detail, we assume that the bidiagonal factorization

$$A = UBV^T \tag{2.2}$$

is given, where U and V are orthogonal and B is bidiagonal. Hence, we can express (2.1) as

$$\begin{bmatrix} 0 & U \\ V & 0 \end{bmatrix} \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} 0 & V^T \\ U^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ g \end{bmatrix}. \tag{2.3}$$

From (2.3) we get for the solutions x and y that

$$UBV^T x = b$$

and

$$VB^TU^T y = g.$$

By using the orthogonality of U and V we can express the forward residual r and the adjoint residual s as

$$\|r\|_2 = \left\|UBV^T x - b\right\|_2 = \left\|U(BV^T x - U^T b)\right\|_2 = \left\|BV^T x - U^T b\right\|_2 \tag{2.4}$$

and

$$\|s\|_2 = \left\|VB^TU^T y - g\right\|_2 = \left\|V(B^TU^T y - V^T g)\right\|_2 = \left\|B^TU^T y - V^T g\right\|_2. \tag{2.5}$$

(2.4) and (2.5) show that for the solutions x and y we have to solve one system with B and one system with B^T as well as the application of orthogonal matrices.

So far we have assumed that an explicit bidiagonal factorization is given (Equation 2.2) 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. [14, 22, 31]). Therefore, we use the following matrix structures

$$\begin{array}{rcl} AV_k & = & U_{k+1}B_k \\ A^TU_{k+1} & = & V_kB_k^T + \alpha_{k+1}v_{k+1}e_{k+1}^T \end{array} \tag{2.6}$$

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 initial vectors of both sequences are linked by the relationship

$$A^T u_1 = \alpha_1 v_1. \tag{2.7}$$

We now use the iterative process described in (2.6) to obtain approximations to the solutions of the forward and the adjoint problem. The residuals at step k can be defined as

$$r_k = b - Ax_k \tag{2.8}$$

and

$$s_k = g - A^T y_k \tag{2.9}$$

with

$$x_k = x_0 + V_k z_k$$

and

$$y_k = y_0 + U_{k+1} w_k.$$

A typical choice for u_1 would be the normalized initial residual $u_1 = r_0 / \|r_0\|$ which then gives for the residual norms

$$\begin{aligned} \|r_k\|_2 &= \|b - Ax_k\|_2 \\ &= \|b - A(x_0 + V_k z_k)\|_2 \\ &= \|r_0 - AV_k z_k\|_2 \\ &= \|r_0 - U_{k+1} B_k z_k\|_2 \\ &= \|\|r_0\| e_1 - B_k z_k\|_2 \end{aligned} \tag{2.10}$$

using (2.6) and the orthogonality of U_{k+1}. The adjoint residual can now be expressed as

$$\begin{aligned} \|s_k\|_2 &= \left\|g - A^T y_k\right\|_2 \\ &= \left\|g - A^T(y_0 + U_{k+1} w_k)\right\|_2 \\ &= \left\|g - A^T y_0 - A^T U_{k+1} w_k\right\|_2 \\ &= \left\|s_0 - V_k B_k^T w_k - \alpha_{k+1} v_{k+1} e_{k+1}^T w_k\right\|_2. \end{aligned} \tag{2.11}$$

Notice, that (2.11) cannot be simplified to $\|s_0\| e_1 - B_k^T w_k$ since s_0 is not in the span of the current and all the following v_j-s. This represents the classical LSQR [31] approach where the focus is on obtaining an approximation that minimizes $\|r_k\|_2 = \|b - Ax_k\|_2$. The method is very successful and widely used in practice but is limited due to the restriction given by (2.7) in the case of simultaneous iteration for the adjoint problem. Figure 2.2 illustrates the behaviour we could observe for all our examples with the LSQR method. In particular the stagnation of the adjoint solution due to the coupling of the starting vectors. In the next section we will present a new approach that overcomes this drawback.

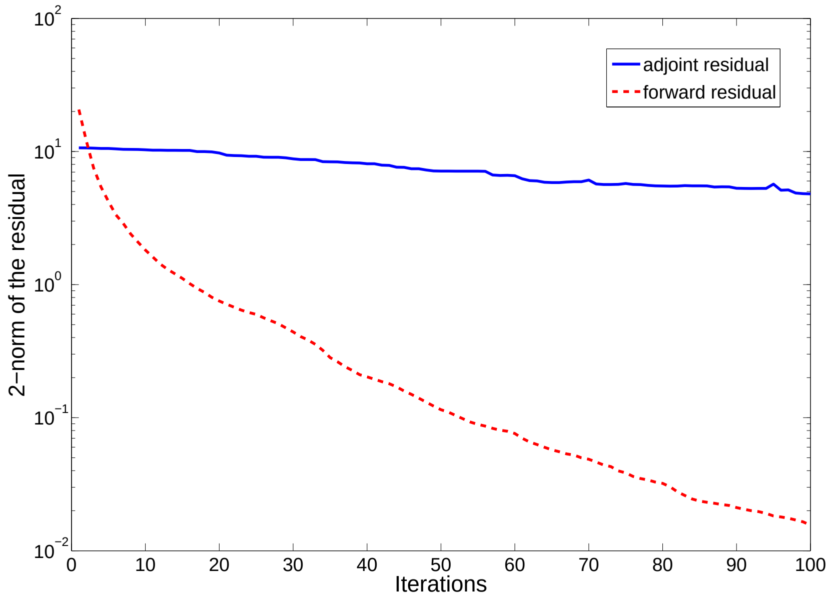

Figure 1: Solving a linear system with the LSQR approach

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 (2.7). In [37] Saunders et al. introduced a more general LSQR method that was also recently analyzed by Reichel and Ye, see [34]. Saunders and coauthors also hint 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 [34, 37]. The method of interest makes it possible to choose the starting vectors u_1 and v_1 independently, e.g. $u_1 = r_0/\left\|r_0\right\|$ and $v_1 = s_0/\left\|s_0\right\|$. The algorithm stated in [34, 37] is based on the following factorization

$$\begin{array}{rclcl} AV_k & = & U_{k+1}T_{k+1,k} & = & U_kT_{k,k} + \beta_{k+1}u_{k+1}e_k^T \\ A^TU_k & = & V_{k+1}S_{k+1,k} & = & V_kS_{k,k} + \eta_{k+1}v_{k+1}e_k^T \end{array} \tag{2.12}$$

where

$$V_k = [v_1, \ldots, v_k]$$

and

$$U_k = [u_1, \ldots, u_k]$$

are orthogonal matrices and

$$T_{k+1,k} = \begin{bmatrix} \alpha_1 & \gamma_1 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \gamma_{k-1} \\ & & \beta_k & \alpha_k \\ & & & \beta_{k+1} \end{bmatrix}$$

as well as

$$S_{k+1,k} = \begin{bmatrix} \delta_1 & \theta_1 & & \\ \eta_2 & \delta_2 & \ddots & \\ & \ddots & \ddots & \theta_{k-1} \\ & & \eta_k & \delta_k \\ & & & \eta_{k+1} \end{bmatrix}.$$

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

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

The matrix factorization given in (2.12) can be used to produce simple algorithmic statements of how to obtain new iterates for u_j and v_j:

$$\begin{array}{rcl} \beta_{k+1}u_{k+1} & = & Av_k - \alpha_k u_k - \gamma_{k-1}u_{k-1} \\ \eta_{k+1}v_{k+1} & = & A^T u_k - \delta_k v_k - \theta_{k-1}v_{k-1} \end{array}. \qquad (2.13)$$

The parameters $\alpha_j, \gamma_{j-1}, \delta_j, v_{j-1}$ could be determined via the Gram-Schmidt orthogonalization process in the classical or the modified version. However, since it is well understood that the classical Lanczos bidiagonalization process introduced in [14] 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 method given in [34, 37].

The generalized LSQR method given in Equation 2.13 looks very similar to the Lanczos process applied to the matrix

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

To see this we use the Lanczos iteration for this matrix and get

$$\nu_{k+1}\begin{bmatrix} u_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}\begin{bmatrix} u_k \\ v_k \end{bmatrix} - \xi_k \begin{bmatrix} u_k \\ v_k \end{bmatrix} - \varrho_{k-1}\begin{bmatrix} u_{k-1} \\ v_{k-1} \end{bmatrix} \qquad (2.14)$$

and the resulting recursions are then

$$\begin{array}{rcl} \nu_{k+1}u_{k+1} & = & Av_k - \xi_k u_k - \varrho_{k-1}u_{k-1} \\ \nu_{k+1}v_{k+1} & = & A^T u_k - \xi_k v_k - \varrho_{k-1}v_{k-1}. \end{array} \qquad (2.15)$$

[1]We will discuss breakdowns later in this section.

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

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

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

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

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 $v_{k+1}^T v_k = 0$. This shows that the the GLSQR method cannot be represented by just applying the symmetric Lanczos process.

In the following, we are going to study the similarity of the GLSQR and a special Block-Lanczos method. In [37] a connection to a Block-Lanczos for the matrix $A^T A$ was made. Here we will discuss a method based on

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

Hence, we assume the complete matrix decompositions

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

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

$$\begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix} \begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix} \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ g \end{bmatrix}. \tag{2.16}$$

We now introduce the perfect shuffle permutation

$$\Pi = [e_1, e_3, \ldots, e_2, e_4, \ldots]$$

and use Π to modify (2.16) which gives

$$\begin{bmatrix} U & \\ & V \end{bmatrix} \Pi^T \Pi \begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix} \Pi^T \Pi \begin{bmatrix} U^T & 0 \\ 0 & V^T \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ g \end{bmatrix}. \tag{2.17}$$

We now further analyze the matrices given in (2.17). The first two matrices can also be written as

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

Next, we are going to study the similarity transformation on

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

using Π which results in

$$\mathcal{T} = \Pi \begin{bmatrix} 0 & T \\ T^T & 0 \end{bmatrix} \Pi^T = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & B_2^T & & \\ & B_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \end{bmatrix}$$

with

$$M_i = \begin{bmatrix} 0 & \alpha_i \\ \alpha_i & 0 \end{bmatrix} \text{ and } B_i = \begin{bmatrix} 0 & \beta_{i+1} \\ \gamma_i & 0 \end{bmatrix}.$$

It is easy to see using the properties of the Reichel and Ye LSQR method 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

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

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

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

which can be further expressed as

$$\mathcal{U}_{k+1} \begin{bmatrix} 0 & \beta_{k+1} \\ \gamma_k & 0 \end{bmatrix} = \begin{bmatrix} 0 & Av_k - \alpha_k u_k - \gamma_{k-1} u_{k-1} \\ A^T u_k - \alpha_k v_k - \gamma_{k-1} v_{k-1} & 0 \end{bmatrix}.$$

The parameters β_{k+1} and γ_k as well as $\mathcal{U}_{k+1}$ have to be determined which can simply be done by normalizing the vectors on the right hand side and using the fact that the underlying process is a classical or modified Gram-Schmidt orthogonalization. Hence, we have shown that the GLSQR method can be viewed as a special Block-Lanczos method with stepsize 2, see [22, 28] 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 problem. We are now able to set u_1 and v_1 independently and choose for initial guesses x_0,y_0 and residuals $r_0 = b - Ax_0$, $s_0 = g - A^T y_0$

$$u_1 = \frac{r_0}{\|r_0\|}$$

and

$$v_1 = \frac{s_0}{\|s_0\|}.$$

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

$$x_k = x_0 + V_k z_k \tag{2.18}$$

for the forward problem and

$$y_k = y_0 + U_k w_k \tag{2.19}$$

for the linear system involving the adjoint. Using this and (2.12) we can express the residual at step k as follows; for the forward problem

$$\begin{aligned} \|r_k\|_2 &= \|b - Ax_k\|_2 \\ &= \|b - A(x_0 + V_k z_k)\|_2 \\ &= \|r_0 - AV_k z_k\|_2 \\ &= \|r_0 - U_{k+1} T_{k+1,k} z_k\|_2 \\ &= \left\|U_{k+1}^T r_0 - T_{k+1,k} z_k\right\|_2 \\ &= \left\|\|r_0\| e_1 - T_{k+1,k} z_k\right\|_2 \end{aligned} \tag{2.20}$$

and in complete analogy

$$\begin{aligned} \|s_k\|_2 &= \left\|g - A^T y_k\right\|_2 \\ &= \left\|V_{k+1}^T s_0 - S_{k+1,k} w_k\right\|_2 \\ &= \left\|\|s_0\| e_1 - S_{k+1,k} w_k\right\|_2 . \end{aligned} \tag{2.21}$$

The solutions z_k and w_k can be obtained by solving the least squares systems (2.20) and (2.21) 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

$$\begin{aligned}\begin{bmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{bmatrix} T_{k+1,k} &= \begin{bmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} T_{k,k-1} & \alpha_k e_k + \gamma_{k-1} e_{k-1} \\ 0 & \beta_{k+1} \end{bmatrix} \\ &= \begin{bmatrix} R_{k-1} & Q_{k-1}^T(\alpha_k e_k + \gamma_{k-1} e_{k-1}) \\ 0 & \beta_{k+1} \end{bmatrix}\end{aligned} \tag{2.22}$$

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 (2.20) and (2.21) efficiently. There is no need to store the whole basis V_k or U_k in order to update the solution as described in (2.18) and (2.19), see also [24]. 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 that $C_k \hat{R}_k = V_k$, ie.

$$c_{k-1} = (v_k - \hat{r}_{k-1,k} c_{k-2} - \hat{r}_{k-2,k} c_{k-3}) / \hat{r}_{k,k} \tag{2.23}$$

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

$$x_k = x_0 + \|r_0\| \, C_k \left(Q_k^T e_1\right)_{k\times 1} = x_{k-1} + a_{k-1} c_{k-1} \tag{2.24}$$

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 nonsymmetric Lanczos process as proposed by Lu and Darmofal in [27]. 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 $S_{k+1,k}$ and their corresponding upper triangular factors. Note, that these triangular matrices have only three nonzero diagonals and can be stored in $T_{k+1,k}$ and $S_{k+1,k}$ respectively. According to (2.23) the solutions x_k and y_k can be updated with only storing two vectors c_{k-2} and c_{k-3} for the forward problem and another two vectors for the adjoint solution. This shows that the solutions can be obtained by storing only a minimal amount of data.

In [34] 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. 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. We assume that the parameter $\beta_{k+1} = 0$ whereas $\eta_{k+1} \neq 0$ in which case Reichel and Ye proved in 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 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. A strategy implicitly proposed by Reichel and Ye is to compute a new u_k using

$$\beta_{k+1} u_{k+1} = 0 = A v_k - \alpha_k u_k - \gamma_{k-1} u_{k-1}$$

which gives

$$\alpha_k u_k = A v_k - \gamma_{k-1} u_{k-1}.$$

In terms of matrices this would result in a upper bidiagonal part of $T_{k+1,k}$ from the iteration where the breakdown occurred. 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

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

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 [34].

Theorem 2.1 *We assume that a first breakdown with $\beta_{k+1} = 0$ occurred and the procedure is continued with an update*

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

If the algorithm breaks down with $\alpha_k = 0$ at some step without the parameter $\eta_l = 0$ for some $l < k$ then the solution for the adjoint problem can be recovered using $y_k = y_0 + U_k w_k$.

Proof The solution w_k to the least squares problem

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

satisfies the following relation

$$S_{k+1,k}^T \left(\|r_0\| e_1 - S_{k+1,k} w_k \right) = 0. \tag{2.25}$$

We will show this by using the QR decomposition of $S_{k+1,k} = Q_{k+1} R_k$ with

$$R_k = \begin{bmatrix} \hat{R}_k \\ 0 \end{bmatrix}.$$

(2.25) can be reformulated now resulting in

$$(Q_{k+1} R_k)^T \left(\|r_0\| e_1 - Q_{k+1} R_k w_k \right) = 0 \tag{2.26}$$

which can be further rearranged using that $w_k = \|r_0\| \hat{R}_k^{-1} Q_{k+1}^T e_1$ as

$$\|r_0\| R_k^T Q_{k+1}^T e_1 - \|r_0\| R_k^T R_k \hat{R}_k^{-1} Q_{k+1}^T e_1 = 0. \tag{2.27}$$

Using (2.27) it is now easy to see that (2.25) holds. 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

$$A V_{k+1} = U_k T_{k,k+1}$$

and

$$A^T U_k = V_{k+1} S_{k+1,k}.$$

This results in,

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

using the fact that $S_{k+1,k}^T = T_{k,k+1}$, see Theorem 2.1 in [34]. Due to the assumption that A is nonsingular the solution for the adjoint problem is given by $y_k = y_0 + U_k w_k$.

$\square$

This shows that the GLSQR method is a well-suited process to find the solution of forward and adjoint problem at the same time. The breakdowns that occur in the process of the algorithm are all benign breakdowns which underlines the difference to methods based on the nonsymmetric Lanczos process. In order to give better reliability of the methods based on the nonsymmetric Lanczos process look-ahead strategies have to be implemented (cf. [9, 33]).

2.5 Preconditioned GLSQR

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

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

and its corresponding transpose 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

$$\begin{aligned} \beta_{j+1}u_{j+1} &= M_1^{-1}AM_2^{-1}v_j - \alpha_j u_j - \gamma_{j-1}u_{j-1} \\ \eta_{j+1}v_{j+1} &= M_2^{-T}A^TM_1^{-T}u_j - \delta_j v_j - \theta_{j-1}v_{j-1} \end{aligned} \tag{2.28}$$

to obtain an efficient implementation of the preconditioned procedure, ie.

$$\begin{aligned} \beta_{j+1}M_1u_{j+1} &= AM_2^{-1}v_j - \alpha_j M_1u_j - \gamma_{j-1}M_1u_{j-1} \\ \eta_{j+1}M_2^Tv_{j+1} &= A^TM_1^{-T}u_j - \delta_j M_2^Tv_j - \theta_{j-1}M_2^Tv_{j-1}. \end{aligned} \tag{2.29}$$

If we set $p_j = M_1u_j$, $M_2\hat{q}_j = v_j$, $q_j = M_2^Tv_j$ and $M_1^T\hat{p}_j = u_j$ we get

$$\begin{aligned} \beta_{j+1}p_{j+1} &= A\hat{q}_j - \alpha_j p_j - \gamma_{j-1}p_{j-1} \\ \eta_{j+1}q_{j+1} &= A^T\hat{p}_j - \delta_j q_j - \theta_{j-1}q_{j-1} \\ &. \end{aligned} \tag{2.30}$$

with the following updates

$$\hat{q}_j = M_2^{-1}v_j = M_2^{-1}M_2^{-T}q_j \tag{2.31}$$

and

$$\hat{p}_j = M_1^{-T}u_j = M_1^{-T}M_1^{-1}p_j. \tag{2.32}$$

We also want to compute the parameters α_j, γ_{j-1}, δ_j and θ_{j_1} The parameters α_j, γ_{j-1}, δ_j and θ_{j_1} can also 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 2.31 and 2.32. 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 Givens rotation 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 Equation 2.24. 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 rotatation 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 and then set $M_1 = L$ and $M_2 = U$. This is a very common approach when preconditioning for solving a linear system. The second method we present here will use the fact that the GLSQR method is a Block-Lanczos method for the normal equations, ie. the system matrix that has to be preconditioned is now A^TA. But rather then computing a Incomplete Cholesky factorization of this matrix we turn to a very interesting class of preconditioners; the Incomplete Orthogonal factorizations as described in [2, 32]. The main idea is to obtain a decomposition $A = QR + E$ with Q being orthogonal and E being the error term of this incomplete factorization. The matrix R now has different properties depending on the particular method that was chosen. In the most simple case, the so-called cIGO method, we restrict R to have the same sparsity pattern as the original matrix A. More methods and implementations can be found in [32]. 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 Moments, Matrices and Quadrature: An Introduction

In [16, 17] 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.

We will give a quick review of the process. Assume that the eigendecomposition of W is given by

$$W = Q\Lambda Q^T$$

with orthogonal Q. Let us further assume that the eigenvalues are ordered as follows

$$\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n.$$

As a result we get

$$f(W) = Qf(\Lambda)Q^T$$

which gives

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

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

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

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

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

where the measure α is defined as follows

$$\alpha(\lambda) = \begin{cases} 0 & \text{if } \lambda < a = \lambda_1 \\ \sum_{j=1}^{i} \alpha_j \beta_j & \text{if } \lambda_i < \lambda < \lambda_{i+1} \\ \sum_{j=1}^{n} \alpha_j \beta_j & \text{if } b = \lambda_n < \lambda \end{cases}$$

We can now express (3.3) as

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{j=1}^{N} \omega_j f(t_j) + \sum_{k=1}^{M} v_k f(z_k) + R[f], \tag{3.4}$$

where the weights ω_j, v_k and the nodes t_j are unknowns and the nodes z_k are prescribed. The residual can be expressed as

$$R[f] = \frac{f^{(2N+M)}(\eta)}{(2N+M)!} \int_a^b \prod_{k=1}^{M} (\lambda - z_k) \left[\prod_{j=1}^{N} (\lambda - t_j) \right]^2 d\alpha(\lambda), \quad a < \eta < b. \tag{3.5}$$

Following the analysis presented in [17] $R[f]$ can be rewritten for Gauss ($M = 0$), Gauss-Radau ($M = 1$, $z_1 = a$ or $z_1 = b$) or Gauss-Lobatto ($M = 2$, $z_1 = a$ and $z_2 = b$) quadrature formulas. A more detailed description can be found in [3, 4, 11, 18–20, 23].

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 [23]. 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

$$J^{pr}(x) = g^T (A^T A)^{-1} A^T b = g^T (A^T A)^{-1} p = g^T f(A^T A) p \tag{3.6}$$

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 (3.6)

$$J^{pr}(x) = \frac{1}{4} \left[(p+g)^T (A^T A)^{-1} (p+g) - (g-p)^T (A^T A)^{-1} (g-p) \right]. \tag{3.7}$$

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

$$A^TAV_N = V_NT_N + r_Ne_N^T \tag{3.8}$$

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 of T_N determine the nodes of

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{j=1}^N \omega_j f(t_j) + R_G[f], \tag{3.9}$$

with

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

which for the function $f(x)=\frac{1}{x}$ reduces to

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

due to the fact that $f^{(2N)}(\eta) = (2N)!\ \eta^{-(2N+1)}$. Notice, that since the matrix A^TA 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. In contrast, the residual for Gauss-Lobatto ($M=2$, $z_1=a$ and $z_2=b$)

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

will always be negative and therefore gives an overestimation whereas the Gauss-Radau($M=1$, $z_1=a$ or $z_1=b$) residual is either

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

or

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

and therefore negative and positive respectively which gives estimates on both sides.

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^TA we can simply use the Lanczos bidiagonalization procedure presented in Section 2.2. Trivially, the matrix T_N can be obtained from Equation 2.6 via $T_N = B_N^T B_N$. Since the matrix T_N is tridiagonal and similar to a symmetric matrix, it is relatively cheap to compute its eigenvalues and eigenvectors.

In [15] Golub and Meurant further show that the evaluation of the expression

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

can be simplified to

$$\sum_{j=1}^{N} \omega_j f(t_j) = e_1^T f(T_N) e_1 \tag{3.10}$$

which for $f(x) = 1/x$ reduces to $e_1^T T_N^{-1} e_1$. 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 [16, 17] give bounds on the elements of the inverse using Gauss, Gauss-Radau, Gauss-Lobatto rules depending on the Lanczos process. In more detail, the bound for the Gauss rule is

$$\frac{\sum_{k\neq i}\sum_{l\neq i} t_{k,i} t_{k,l} t_{l,i}}{t_{l,l}\sum_{k\neq i}\sum_{l\neq i} t_{k,i} t_{k,l} t_{l,i} - \left(\sum_{k\neq i} t_{k,i}^2\right)^2} \leq (T_N^{-1})_{1,1}$$

with $t_{i,j}$ elements of T_N, a lower and an upper bound for Gauss-Radau

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

with $s_1^2 = \sum_{j\neq 1} a_{j1}^2$ and for the Gauss-Lobatto

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

as an upper bound. These bounds are not sharp since they will improve with the number of Lanczos steps. It is also possible to obtain the given bounds using variational principles, see [35].

3.3 Approximation using GLSQR (the block case)

As shown in Section 2.3 the GLSQR represents a Block-Lanczos method. In [17] Golub and Meurant show how a block method can be used to generate quadrature formulas.

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

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{i=1}^{N} W_j f(T_j) W_j + R[f] \tag{3.11}$$

with T_j and W_j being symmetric 2×2 matrices. Equation 3.11 can be simplified using

$$T_j = Q_j \Lambda_j Q_j^T$$

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

$$\sum_{i=1}^{N} W_j Q_j^T f(\Lambda_j) Q_j W_j$$

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

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

we get for the quadrature rule

$$\sum_{i=1}^{2N} f(t_j) w_j w_j^T$$

where t_j is a scalar and w_j is a vector with two components. In [17] it is shown that there exist orthogonal matrix polynomials such that

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

with $p_0(\lambda) = I_2$ and $p_{-1}(\lambda) = 0$.

We can write the last Equation as

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

with

$$\mathcal{T}_N = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & B_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & B_{N-2} & M_{N-1} & B_{N-1}^T \\ & & & B_{N-1} & M_N \end{bmatrix}$$

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

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{i=1}^{2N} f(\theta_i) u_i u_i^T + R[f] \tag{3.12}$$

where $2N$ is the order of the matrix $\mathcal{T}_N$, θ_i eigenvalues of $\mathcal{T}_N$ and u_i is 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

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

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 behaviour.

This block method can now be used to estimate the scattering amplitude using GLSQR . The 2×2 matrix integral we are interested in is now

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

In order to use the approximation (3.12) we need a Block-Lanczos for the matrix

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

The GLSQR algorithm represents an implementation of a Block-Lanczos method for this matrix and can therefore be used to create the desired block-tridiagonal matrix $\mathcal{T}_N$. Thus, in every iteration step GLSQR gives an approximation to the scattering amplitude via

$$\sum_{i=1}^{2N} f(\lambda_i) u_i u_i^T = \begin{bmatrix} 0 & g^T x_N \\ g^T x_N & 0 \end{bmatrix}$$

without computing the approximate solution x_N directly.

3.4 Preconditioned GLSQR

In Section 2.5 the preconditioned GLSQR method was introduced and we will 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 scattering amplitude using that

$$\begin{bmatrix} 0 & g^T A^{-1} b \\ b^T A^{-T} g & 0 \end{bmatrix}.$$

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 $\widehat{A} = M_1^{-1}AM_2^{-1}$, $\hat{x} = M_2x$ and $\hat{b} = M_1^{-1}b$ and with initial residuals $r_0 = M_1^{-1}b$ and $\tilde{r}_0 = M_2^{-T}g$. The scattering amplitude can therefore be approximated via

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

3.5 Complex Gaussian quadrature and BICG

The methods we presented so far are based on Lanczos methods for A^TA. The algorithm introduced in this Section connects BICG a method based on the nonsymmetric Lanczos process and Gaussian quadrature in the complex plane. We follow the derivation of Saylor and Smolarski in [38, 39] in which the scattering amplitude is approximated employing this connection.

Saylor and Smolarski use *formally orthogonal polynomials*, ie. a set of polynomials $\{\phi_i\}_{i=0,\ldots}$ such that $(\phi_i, \phi_j)_w = 0$ if $i \neq j$ with $(\cdot,\cdot)_w$ a bilinear form. They further show that the BICG polynomials are formally orthogonal and link them to Gaussian quadrature. More details can be found in [39]. Again the nodes and weights can be determined from a certain version of the Jacobi matrix which can be generated from the parameters of the BICG iteration. We want to emphasize again that the nonsymmetric Lanczos process and therefore BICG can break down and look-ahead strategies have to be implemented.

We allow that all quantities are complex and use A^* for matrices and $\bar{\lambda}$ for the conjugate transpose and complex conjugate respectively. In more detail, if $x_0 = \tilde{x}_0 = 0$ where x_0 is the initial guess for the forward problem and $\tilde{x}_0$ the initial guess for the adjoint problem we can expand the residuals

$$r_0 = \sum_{i=1}^{N} \theta_i \lambda_i g_i \text{ and } \tilde{r}_0 = \sum_{i=1}^{N} \tilde{\theta}_i \bar{\lambda}_i \tilde{g}_i$$

where λ_i,$\bar{\lambda}_i$ and g_i, $\tilde{g}_i$ are the eigenvalues and eigenvectors of A and A^* respectively. For any two vectors v and $\tilde{v}$ with

$$v \in V_k = span\left\{r_0, Ar_0, \ldots, A^{k-1}r_0\right\} \text{ and } \tilde{v} \in \tilde{V}_k = span\left\{\tilde{r}_0, A^*\tilde{r}_0 \ldots, (A^*)^{k-1}\tilde{r}_0\right\}$$

we have polynomials p_k and $\tilde{p}_k$ such that

$$v = p_k(A)r_0 = p_k(A)\sum_{i=1}^{N} \theta_i \lambda_i g_i = \sum_{i=1}^{N} \theta_i \lambda_i p_k(\lambda_i) g_i$$

and

$$\tilde{v} = \tilde{p}_k(A^*)r_0 = \tilde{p}_k(A^*)\sum_{i=1}^{N} \tilde{\theta}_i \bar{\lambda}_i \tilde{g}_i = \sum_{i=1}^{N} \tilde{\theta}_i \bar{\lambda}_i \tilde{p}_k(\bar{\lambda}_i) \tilde{g}_i.$$

Now proceeding formally we insist that $\tilde{g}_i^* g_i$ which results in

$$\tilde{v}^* v = \sum_{i=1}^{N} \bar{\tilde{\theta}}_i \theta_i \lambda_i^2 \overline{\tilde{p}_k(\bar{\lambda}_i)} p_k(\lambda_i).$$

Therefore, by introducing the discrete measure

$$w(\lambda) = \begin{cases} \theta_i \bar{\tilde{\theta}}_i \lambda_i^2 & \text{for } \lambda = \lambda_i, i = 1, \ldots, N \\ 0 & \text{for } \lambda \neq \lambda_i, i = 1, \ldots, N \end{cases}$$

we obtain

$$\tilde{v}^* v = \sum_{i=1}^{N} \bar{\tilde{\theta}}_i \theta_i \lambda_i^2 \overline{\tilde{p}_k(\bar{\lambda}_i)} p_k(\lambda_i) = \int_\gamma \overline{\tilde{p}_k(\bar{\lambda}_i)} p_k(\lambda_i) w(\lambda) \, |d\lambda|$$

where γ is any arc connecting the eigenvalues of A. For p_0 and $\tilde{p}_0$ as so-called residual polynomials, ie. $p_0(0) = \tilde{p}_0(0) = 1$, we get

$$r_0 = b \text{ and } \tilde{r}_0 = g.$$

and thus

$$g^* b = \int w(\lambda) \, |d\lambda| \, .$$

More usefully, for every function f, Gaussian quadrature gives

$$\int_\gamma f(\lambda) w(\lambda) \, |d\lambda| \approx \sum_{i=1}^{k} \omega_i f(\zeta_i),$$

where ω_i and ζ_i can be determined from the eigensystem of the tridiagonal matrix associated with the appropriate form of BICG (cf. [39]). On the other hand, we know

$$\int_\gamma f(\lambda) \cdot 1 \cdot w(\lambda) \, |d\lambda| = \sum \bar{\tilde{\theta}}_i \theta_i \lambda_i^2 f(\lambda_i) \cdot 1 = (f(A)b)^* g.$$

For $f(A) = A^{-1}$, the result is

$$\int_\gamma f(\lambda) w(\lambda) \, |d\lambda| = \int_\gamma \frac{1}{\lambda} w(\lambda) \, |d\lambda| = (A^{-1}b)^* g = g^* A^{-1} b$$

and

$$g^* A^{-1} b \approx \sum_{i=1}^{k} \frac{\omega_i}{\zeta_i}$$

which gives an approximation to the desired scattering amplitude. Note, that no bounds on the quality of the approximation are given in the case of the BICG based approximation in contrast to the bounds based on the symmetric Lanczos process, see Section 3.2.

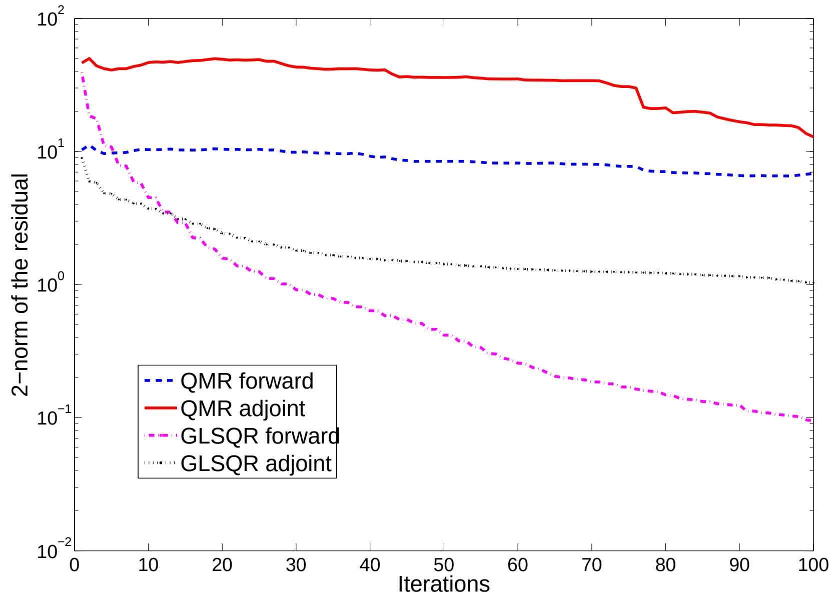

Figure 2: QMR and GLSQR for a matrix of dimension 100

We also mention that preconditioning is possible when working with the above method based on the following observation

$$\begin{aligned}
\hat{g}^T\hat{x} &= \hat{g}^T\hat{A}^{-1}\hat{b}\\
&= (M_2^{-T}g)^T(M_1^{-1}AM_2^{-1})^{-1}(M_1^{-1}b)\\
&= g^TM_2^{-1}M_2A^{-1}M_1M_1^{-1}b\\
&= g^TA^{-1}b\\
&= g^Tx
\end{aligned}$$

with initial residuals $r_0 = M_1^{-1}b$ and $\tilde{r}_0 = M_2^{-T}g$ for BICG .

4 Numerical Experiments

4.1 Solving the linear system

In this Section we want to show numerical experiments for the methods introduce in Section 2.

Example 4.1 *In the first example, we apply the* QMR *method and the* GLSQR *to a random sparse matrix of dimension* 100*, eg.* `A=sprandn(n,n,0.2)+speye(n);`*. Figure 2 indicates that* GLSQR *outperforms* QMR *for this example.*

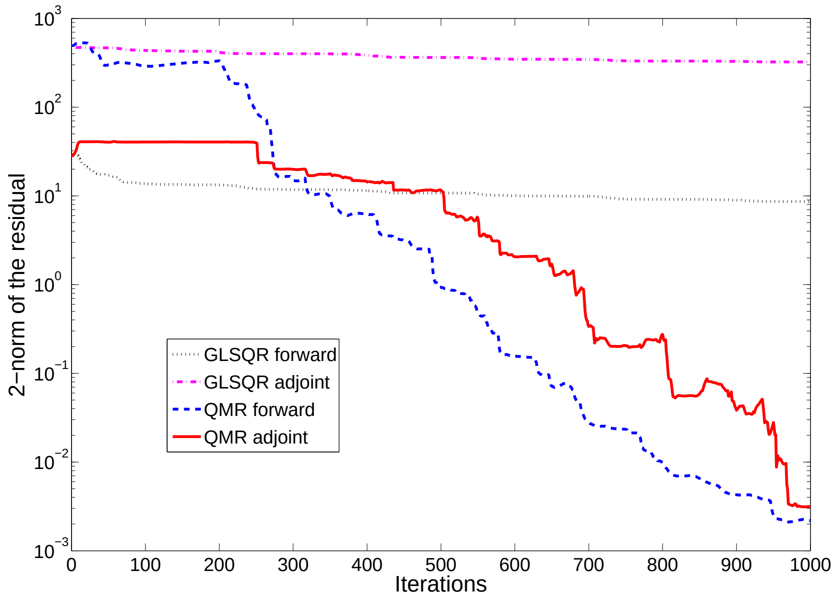

Figure 3: GLSQR and QMR for the matrix: orsirr_1.mtx

Example 4.2 *The second example is a matrix from the Matrix Market*[2] *collection, in particular the matrix ORSIRR 1 which represents a linear system used in oil reservoir modelling. The matrix size is* 1030. *The results without preconditioning are shown in Figure 3. Results using the Incomplete LU (ILU) factorization as a preconditioner for* GLSQR *and* QMR *are given Figure 4. Clearly,* QMR *outperforms* GLSQR *in both.*

Example 4.3 *Figure 5 shows the comparison of ILU preconditioner and cIGO preconditioner for an academic example of a* 50×50 *matrix. It shows that the cIGO preconditioner seems to be of rather theoretical use for large square matrices and the results are not necessarily better than the ones coming from the application of an Incomplete LU decomposition.*

Example 4.4 *The next example is motivated by [29] where Nachtigal et al. introduce examples that show that different solvers for nonsymmetric systems can outperform others by a large factor. The original example in [29] is given by the matrix*

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

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

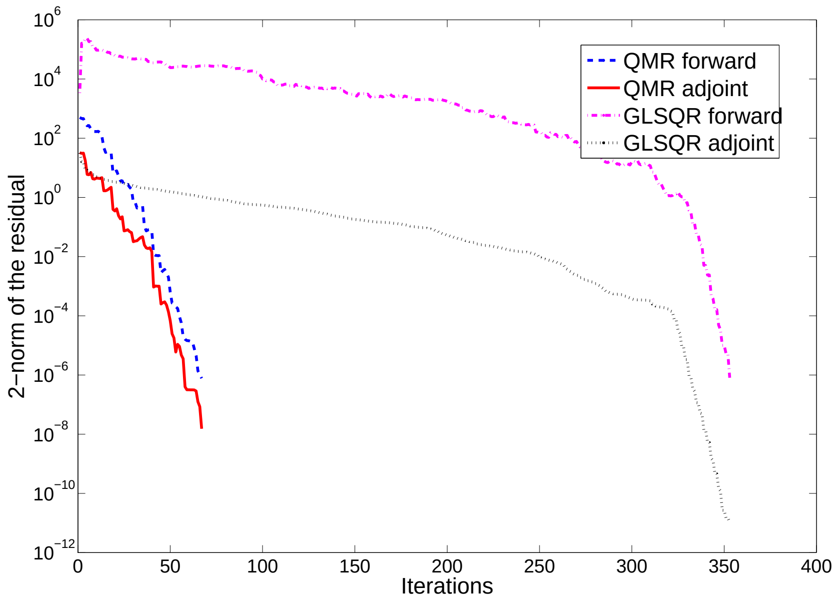

Figure 4: ILU preconditioned GLSQR and QMR for the matrix: orsirr_1.mtx

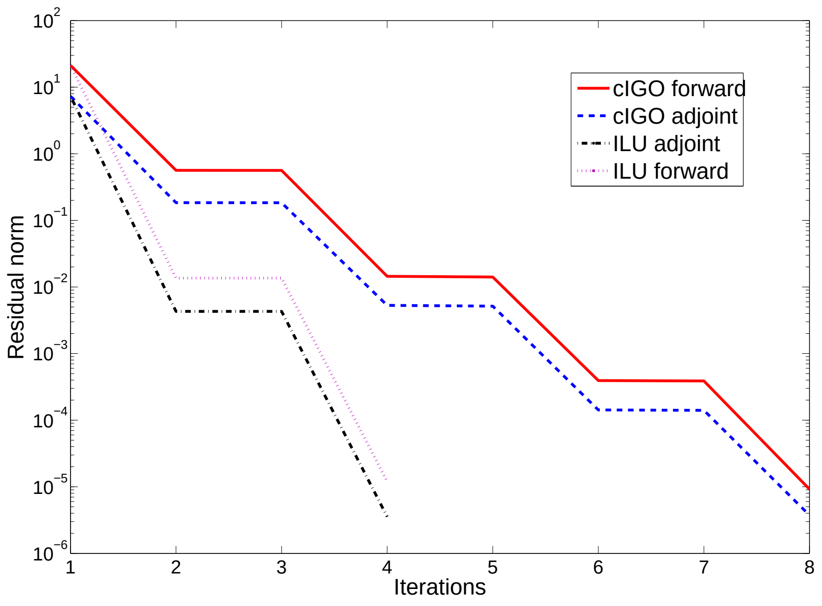

Figure 5: cIGO and ILU results for a 50×50 example.

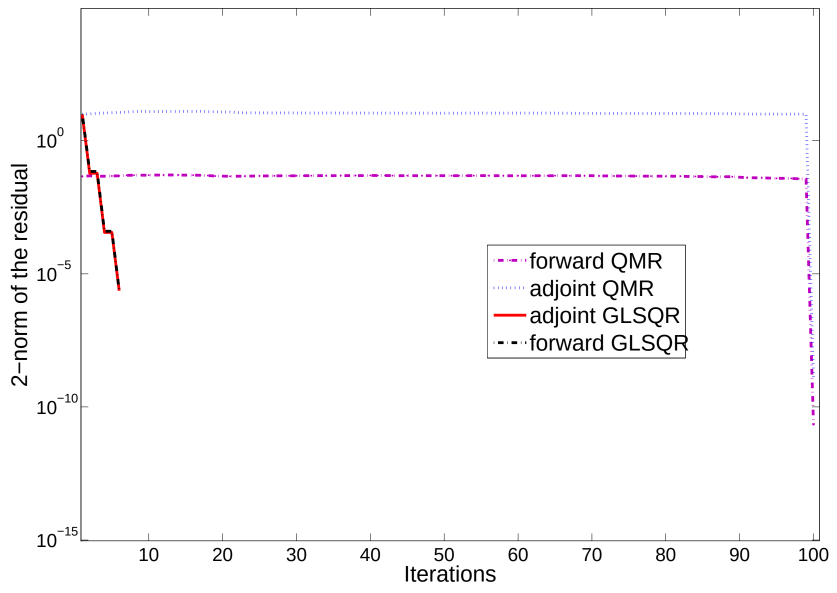

Figure 6: An almost orthogonal matrix

The results shown in Figure 6 are for a sparse permutation of the matrix J, ie. in Matlab notation `A=1e-3*sprandn(n,n,0.2)+J;`*. The matrix J is orthogonal which explains the rapid convergence of* GLSQR *as a method for $A^T A$.*

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 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 7 and 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
.

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 8 and the bottom part shows the norm of the error against the number of iterations.*

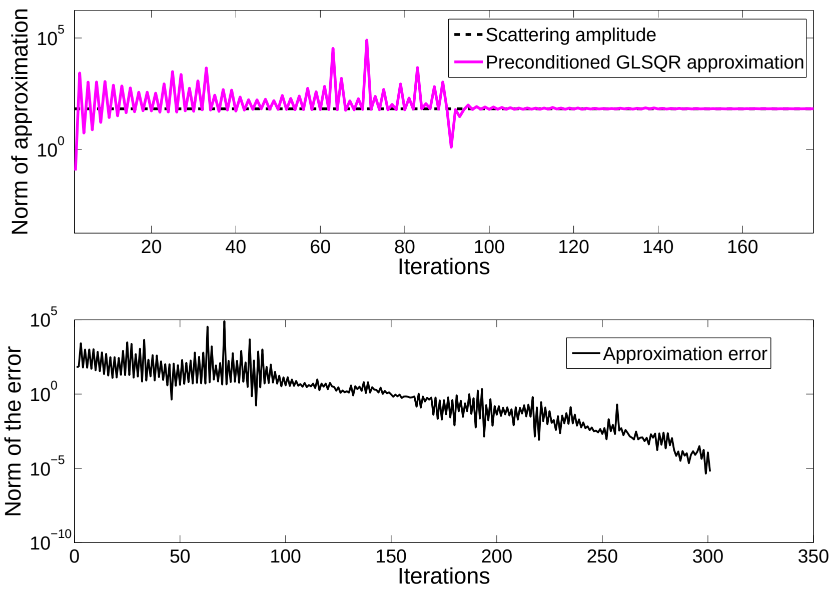

Figure 7: Approximations to the scattering amplitude and error

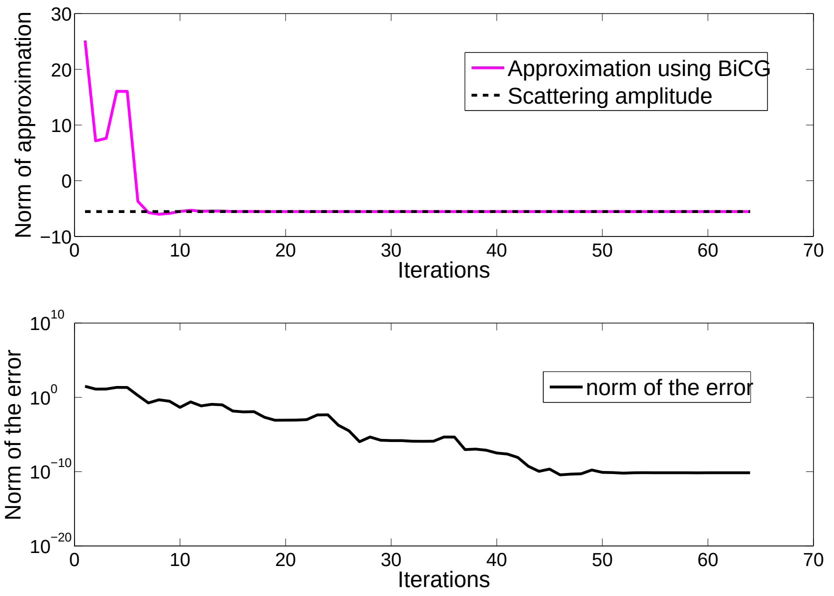

Figure 8: Approximations to the scattering amplitude and error

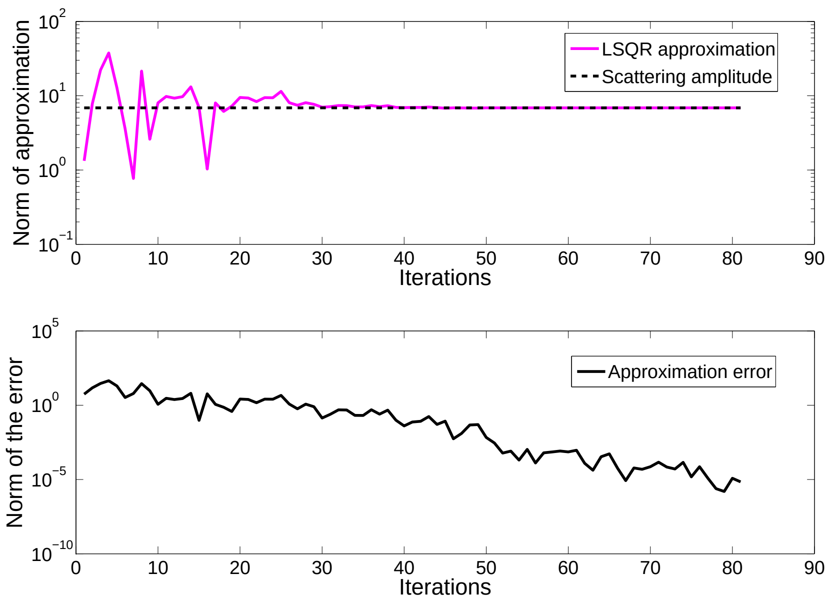

Figure 9: Approximations to the scattering amplitude and error

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 [6]. The result is shown in Figure 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 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 forward and adjoint system at the same time. Furthermore, we introduced preconditioning for the GLSQR method and proposed different preconditioners.

The approximation of the scattering amplitude without first computing solutions to the linear systems was introduced based on the Golub-Kahan bidiagonalisation 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 directly by using the connection to Block-Gauss quadrature. Furthermore, we introduced preconditioning for the Block-Gauss quadrature. We implemented a nonsymmetric Lanczos method proposed by Saylor and Smolarski and showed how in that situation preconditioners can also be incorporated.

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 were 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 and Michael Saunders for their helpful comments.

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

References

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

[2] Zhong-Zhi Bai, Iain S. Duff and Andrew J. Wathen, 2001. A class of incomplete orthogonal factorization methods. I. Methods and theories. *BIT*, **41**(1): 53–70.

[3] Germund Dahlquist, Stanley C. Eisenstat and Gene H. Golub, 1972. Bounds for the error of linear systems of equations using the theory of moments. *J. Math. Anal. Appl.*, **37**: 151–166.

[4] Philip J. Davis and Philip Rabinowitz, 1984. *Methods of numerical integration.* Computer Science and Applied Mathematics. Academic Press Inc., Orlando, FL, second edition.

[5] Van der Vorst, 1992. BiCGSTAB: A fast and smoothly converging variant of biCG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput*, **13**: 631–644.

[6] Howard C. Elman, Alison Ramage and David J. Silvester, June 2007. Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow. *ACM Transactions on Mathematical Software*, **33**(2).

[7] Howard C. Elman, David J. Silvester and Andrew J. Wathen, 2005. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics.* Numerical Mathematics and Scientific Computation. Oxford University Press, New York.

[8] R. Fletcher, 1976. Conjugate gradient methods for indefinite systems. In *Numerical analysis (Proc 6th Biennial Dundee Conf., Univ. Dundee, Dundee, 1975)*, pages 73–89. Lecture Notes in Math., Vol. 506. Springer, Berlin.

[9] Roland W. Freund, Martin H. Gutknecht and Noël M. Nachtigal, 1993. An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. *SIAM J. Sci. Comput.*, **14**(1): 137–158.

[10] Roland W. Freund and Noël M. Nachtigal, 1991. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, **60**(3): 315–339.

[11] Walter Gautschi, 1968. Construction of Gauss-Christoffel quadrature formulas. *Math. Comp.*, **22**: 251–270.

[12] M.B. Giles and N.A. Pierce, 2000. An Introduction to the Adjoint Approach to Design. *Flow, Turbulence and Combustion*, **65**(3): 393–415.

[13] Michael B. Giles and Endre Süli, 2002. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. *Acta Numer.*, **11**: 145–236.

[14] G. Golub and W. Kahan, 1965. Calculating the singular values and pseudo-inverse of a matrix. *J. Soc. Indust. Appl. Math. Ser. B Numer. Anal.*, **2**: 205–224.

[15] G. H. Golub and G. Meurant. Matrices, moments and quadrature with applications. Draft.

[16] G. H. Golub and G. Meurant, 1997. Matrices, moments and quadrature. II. How to compute the norm of the error in iterative methods. *BIT*, **37**(3): 687–705. Direct methods, linear algebra in optimization, iterative methods (Toulouse, 1995/1996).

[17] G. H. Golub and Gérard Meurant, 1994. Matrices, moments and quadrature. In *Numerical analysis 1993 (Dundee, 1993)*, Volume 303 of *Pitman Res. Notes Math. Ser.*, pages 105–156. Longman Sci. Tech., Harlow.

[18] Gene H. Golub, 1973. Some modified matrix eigenvalue problems. *SIAM Rev.*, **15**: 318–334.

[19] Gene H. Golub, 1974. Bounds for matrix moments. In *Proceedings of the International Conference on Padé Approximants, Continued Fractions and Related Topics (Univ. Colorado, Boulder, Colo., 1972; dedicated to the memory of H. S. Wall)*, Volume 4, pages 207–211.

[20] Gene H. Golub, 1974. Bounds for matrix moments. In *Proceedings of the International Conference on Padé Approximants, Continued Fractions and Related Topics (Univ. Colorado, Boulder, Colo., 1972; dedicated to the memory of H. S. Wall)*, Volume 4, pages 207–211.

[21] Gene H. Golub, Gerald N. Minerbo and Paul E. Saylor. Nine ways to compute the scattering cross section (i): Estimating $c^T x$ iteratively.

[22] Gene H. Golub and Charles F. Van Loan, 1996. *Matrix computations.* Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition.

[23] Gene H. Golub and John H. Welsch, 1969. Calculation of Gauss quadrature rules. *Math. Comp. 23 (1969), 221-230; addendum, ibid.*, **23**(106, loose microfiche suppl): A1–A10.

[24] Anne Greenbaum, 1997. *Iterative methods for solving linear systems*, Volume 17 of *Frontiers in Applied Mathematics.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.

[25] Magnus R. Hestenes and Eduard Stiefel, 1952. Methods of conjugate gradients for solving linear systems. *J. Research Nat. Bur. Standards*, **49**: 409–436 (1953).

[26] L.D. Landau and EM Lifshitz, 1965. Quantum mechanics. *Course of theoretical physics.*

[27] James Lu and David L. Darmofal, 2003. A quasi-minimal residual method for simultaneous primal-dual solutions and superconvergent functional estimates. *SIAM J. Sci. Comput.*, **24**(5): 1693–1709 (electronic).

[28] G. Meurant, 1999. *Computer solution of large linear systems*, Volume 28 of *Studies in Mathematics and its Applications.* North-Holland Publishing Co., Amsterdam.

[29] Noël M. Nachtigal, Satish C. Reddy and Lloyd N. Trefethen, 1992. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix Anal. Appl.*, **13**(3): 778–795. Iterative methods in numerical linear algebra (Copper Mountain, CO, 1990).

[30] C. C. Paige and M. A. Saunders, 1975. Solutions of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, **12**(4): 617–629.

[31] Christopher C. Paige and Michael A. Saunders, 1982. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, **8**(1): 43–71.

[32] A. T. Papadopoulos, I. S. Duff and A. J. Wathen, 2005. A class of incomplete orthogonal factorization methods. II. Implementation and results. *BIT*, **45**(1): 159–179.

[33] B. N. Parlett, D. R. Taylor and Z. S. Liu, 1984. The look ahead Lánczos algorithm for large unsymmetric eigenproblems. In *Computing methods in applied sciences and engineering, VI (Versailles, 1983)*, pages 87–96. North-Holland, Amsterdam.

[34] Lothar Reichel and Qiang Ye, 2007. A generalized LSQR algorithm. *Numer. Linear Algebra Appl.*

[35] P. D. Robinson and A. J. Wathen, 1992. Variational bounds on the entries of the inverse of a matrix. *IMA J. Numer. Anal.*, **12**(4): 463–486.

[36] Y. Saad, 2003. *Iterative methods for sparse linear systems.* Society for Industrial and Applied Mathematics, Philadelphia, PA.

[37] M. A. Saunders, H. D. Simon and E. L. Yip, 1988. Two conjugate-gradient-type methods for unsymmetric linear equations. *SIAM J. Numer. Anal.*, **25**(4): 927–940.

[38] Paul E. Saylor and Dennis C. Smolarski, 2001. Addendum to: "Why Gaussian quadrature in the complex plane?" [Numer. Algorithms **26** (2001), no. 3, 251–280; MR1832543 (2002a:65050)]. *Numer. Algorithms*, **27**(2): 215–217.

[39] Paul E. Saylor and Dennis C. Smolarski, 2001. Why Gaussian quadrature in the complex plane? *Numer. Algorithms*, **26**(3): 251–280.

[40] Henk A. van der Vorst, 2003. *Iterative Krylov methods for large linear systems*, Volume 13 of *Cambridge Monographs on Applied and Computational Mathematics.* Cambridge University Press, Cambridge.