On Computing the Scattering Amplitude and Linear Systems

Gene Golub[1] Martin Stoll[2] Andy Wathen[2]

[1]Department of Computer Science, Stanford

[2]Oxford University Computing Laboratory

Hong Kong Baptist University
October 2007

Workshop on Solution Methods for Saddle Point Systems

The linear systems

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

$$Ax = b \tag{1}$$

but also of the adjoint system

$$A^T y = g. \tag{2}$$

Our aim is to solve both systems simultaneously!

This is an easy task when working with direct solvers, ie. LU factorisation but for large-scale problems this is not always feasible.

The scattering amplitude

In signal processing the **scattering amplitude**

$$g^T x$$

has to be computed without looking for the approximation to x itself.
In optimization the scattering amplitude is sought for under the name of **primal linear output**

$$J^{pr}(x) = g^T x.$$

A reformulation

Solving

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

simultaneously can be reformulated as

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

using the bidiagonal factorization

$$A = UBV^T.$$

A reformulation

Solving

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

simultaneously can be reformulated as

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

using the bidiagonal factorization

$$A = UBV^T.$$

From above we get

$$UBV^T x = b$$

and

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

Residuals for forward and adjoint problem

We can now express the residuals of the adjoint and the forward problem as

$$\|r\|_2 = \left\|BV^T x - U^T b\right\|_2 \quad \text{and} \quad \|s\|_2 = \left\|B^T U^T y - V^T g\right\|_2. \qquad (3)$$

Residuals for forward and adjoint problem

We can now express the residuals of the adjoint and the forward problem as

$$\|r\|_2 = \left\|BV^T x - U^T b\right\|_2 \quad \text{and} \quad \|s\|_2 = \left\|B^T U^T y - V^T g\right\|_2. \qquad (3)$$

Decomposition

$$A = UBV^T$$

not feasible because the computational cost is high.

An iterative procedure

Therefore, we use the following iterative instance [G.&Kahan'65] or LSQR [Paige& Saunders'82]

$$\begin{aligned} AV_k &= U_{k+1}B_k \\ A^T U_{k+1} &= V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T \end{aligned} \tag{4}$$

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

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

The initial vectors of both sequences are linked by the relationship

$$A^T u_1 = \alpha_1 v_1. \tag{5}$$

The iterative residuals

Using the bidiagonalization we get

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \| \|r_0\| e_1 - B_k z_k\|_2$$

with $x_k = x_0 + V_k z_k$ and

$$\|s_k\|_2 = \left\|g - A^T y_k\right\|_2 = \left\|s_0 - V_k B_k^T w_k - \alpha_{k+1} v_{k+1} e_{k+1}^T w_k\right\|_2 .$$

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

The iterative residuals

Using the bidiagonalization we get

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \|\|r_0\| e_1 - B_k z_k\|_2$$

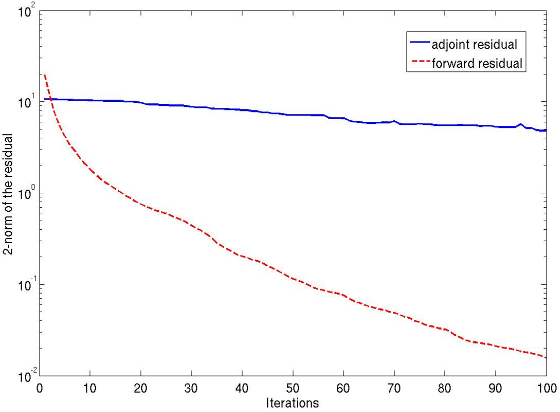
with $x_k = x_0 + V_k z_k$ and

$$\|s_k\|_2 = \left\|g - A^T y_k\right\|_2 = \left\|s_0 - V_k B_k^T w_k - \alpha_{k+1} v_{k+1} e_{k+1}^T w_k\right\|_2 .$$

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

s_k not in MINRES-like structure $\Longrightarrow$ LSQR approach *fails* for the adjoint solution. The reason is the link between the starting vectors for both sequences $A^T u_1 = \alpha_1 v_1$.

A numerical example using LSQR



Example for random matrix with typical stagnation for adjoint problem.

The GLSQR approach

[Saunders, Simon, Yip'88] and [Reichel&Ye'07] introduce a Generalized LSQR method where u_1 and v_1 can be chosen independently based on the factorization

$$\begin{array}{lllll} AV_k & = & U_{k+1}T_{k+1,k} & = & U_kT_{k,k} + \beta_{k+1}u_{k+1}e_k^T \\ A^TU_k & = & V_{k+1}S_{k+1,k} & = & V_kS_{k,k} + \eta_{k+1}v_{k+1}e_k^T \end{array} \qquad (6)$$

where

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

are orthogonal matrices and

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

Some remarks on GLSQR

- It represents a special Block-Lanczos method.
- Starting vectors can be chosen such that $u_1 = r_0 / \|r_0\|$ and $v_1 = s_0 / \|s_0\|$.
- Without breakdown (all breakdowns are lucky) we have $S_{k,k}^T = T_{k,k}$.

More on the Block-Lanczos

The block-tridiagonal matrix associated with GLSQR is now

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

where

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

with an orthogonal matrix $\mathcal{U} = [\mathcal{U}_1, \mathcal{U}_2, \cdots]$ where $\mathcal{U}_i^T \mathcal{U}_i = I_2$.

Thus, one particular instance at step k of the reformulated method reduces to

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

Solutions from GLSQR

With the choice of

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

we get for the residuals

$$\|r_k\|_2 = \|b - Ax_k\|_2 = \| \|r_0\| e_1 - T_{k+1,k} z_k\|_2$$

and

$$\|s_k\|_2 = \left\|g - A^T y_k\right\|_2 = \| \|s_0\| e_1 - S_{k+1,k} w_k\|_2$$

with

$$x_k = x_0 + V_k z_k \text{ and } y_k = y_0 + U_k w_k$$

Preconditioning in GLSQR

Introducing preconditioners we get

$$\widehat{A} = M_1^{-1} A M_2^{-1} \text{ and } \widehat{A}^T = M_2^{-T} A^T M_1^{-T}.$$

and can efficiently rewrite the GLSQR method, ie.

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

with the following updates

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

and

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

Possible preconditioners

- Incomplete LU factorization with $M_1 = L$ and $M_2 = U$.
- Since GLSQR is a Block-Lanczos for $A^T A$ an Incomplete Cholesky of $A^T A$ would be useful but numerically prohibitive.
- Instead, use Incomplete Orthogonal factorizations [Bai et al.'01, Papadopoulos et al.'05] where we get

$$A = QR + E \Longrightarrow \widehat{A}^T \widehat{A} = R^{-T} A^T Q Q^T A R^{-1} = R^{-T} A^T A R^{-1}$$

with $M_2 = R$ and $M_1 = Q$ and finally

$$M_2 = R \text{ and } M_1 = I.$$

The QMR method

The basis for QMR is the non-symmetric Lanczos

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

gives the quasi-residual

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

with the choice of $v_1 = r_0/\|r_0\|$ and $w_1 = s_0/\|s_0\|$.

Weights can be introduced see [Lu, Darmofal'01].

Formulation in terms of moments

Starting with the primal output $J^{pr}(x) = g^T x$, we use $x = A^{-1}b$ and get

$$J^{pr}(x) = g^T A^{-1} b$$

which can be written as

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

with $p = A^T b$. Equivalently,

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

A short course on Gauss quadrature

Using the eigendecomposition we see

$$A^TA = Q\Lambda Q^T \Longrightarrow f(A^TA) = Qf(\Lambda)Q^T$$

and therefore

$$u^Tf(A^TA)v = u^TQf(\Lambda)Q^Tv.$$

With $\alpha = Q^Tu$ and $\beta = Q^Tv$ we obtain

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

The last Equation can be viewed as a Riemann-Stieltes integral if $\alpha_i\beta_i \geq 0$

$$I[f] = u^Tf(A^TA)v = \int_a^b f(\lambda)d\alpha(\lambda)$$

where the measure α is defined as follows

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

A short course on Gauss quadrature ctd.

A numerical approximation via Gauss, Gauss-Radau or Gauss-Lobatto quadrature formulas gives

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

where the weights ω_j, v_k and the nodes t_j are unknowns and the nodes z_k are prescribed and

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

A short course on Gauss quadrature ctd.

Use the Gauss rule and Lanczos process for $A^T A$ which is simply the Golub-Kahan bidiagonalization process, ie.

$$A^T A V_N = V_N T_N + r_N e_N^T. \tag{7}$$

The eigenvalues of of T_N determine the nodes of

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

where

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

The weights for the Gauss rule are given by the squares of the first elements of the normalized eigenvectors of T_N.

A short course on Gauss quadrature ctd.

BUT we don't have to compute eigenvalues of T_N since

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

which in our case reduces to

$$e_1^T T_N^{-1} e_1$$

and **even better** we can compute bounds on the elements of the inverse using Gauss, Gauss-Radau, Gauss-Lobatto rules, e.g. from the Gauss-Radau rule we get

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

with $t_{i,j}$ elements of T_N and $s_1^2 = \sum_{j \neq 1} a_{j1}^2$.

Gauss quadrature and the Block-Lanczos

$\int_a^b f(\lambda)d\alpha(\lambda)$ is a 2×2 symmetric matrix and a quadrature formula is of the form

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{i=1}^{N} W_j f(T_j) W_j + \mathit{error} \tag{8}$$

with T_j and W_j being symmetric 2×2 matrices. Equation 8 can be simplified using

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

is an eigendecomposition of T_j and

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

Gauss quadrature and the Block-Lanczos

$\int_a^b f(\lambda)d\alpha(\lambda)$ is a 2×2 symmetric matrix and a quadrature formula is of the form

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{i=1}^{N} W_j f(T_j) W_j + error \tag{8}$$

with T_j and W_j being symmetric 2×2 matrices. Equation 8 can be simplified using

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

is an eigendecomposition of T_j and

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

In terms of orthogonal matrix polynomials we get

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

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

Gauss quadrature and the Block-Lanczos

Therefore,

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

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-triangular matrix. Therefore, we can define the quadrature rule as

$$\int_a^b f(\lambda)d\alpha(\lambda) = \sum_{i=1}^{2N} f(\lambda_i)u_i u_i^T + error \tag{9}$$

where $2N$ is the order of the matrix $\mathcal{T}_N$, λ_i an eigenvalue of $\mathcal{T}_N$ and u_i is the vector consisting of the first two elements of the corresponding normalized eigenvector.

Block-Lanczos, GLSQR and the scattering amplitude

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

$$\int_a^b f(\lambda) d\alpha(\lambda) =$$

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

This can be approximated if a Block-Lanczos for

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

is given.

Block-Lanczos, GLSQR and the scattering amplitude

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

$$\int_a^b f(\lambda) d\alpha(\lambda) =$$

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

This can be approximated if a Block-Lanczos for

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

is given. Which is what GLSQR does!

Preconditioned Gauss quadrature

GLSQR gives approximation to

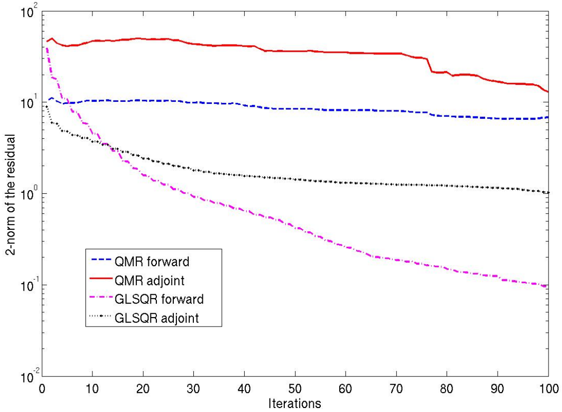
$$\begin{bmatrix} 0 & g^T A^{-1} b \\ b^T A^{-T} g & 0 \end{bmatrix}.$$

Reformulating this in terms of the preconditioned method gives,

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

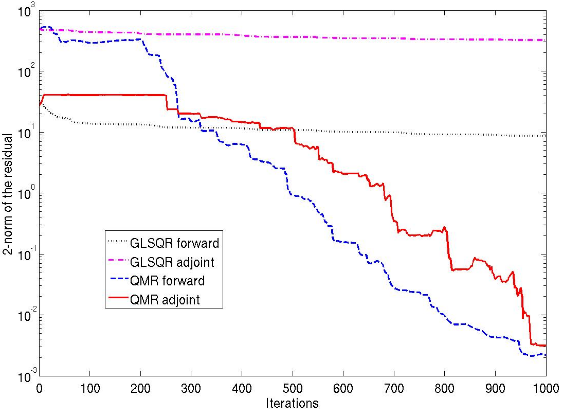
which shows that the natural choice of the preconditioned residuals in the preconditioned GLSQR gives an approximation for the scattering amplitude.

A random example



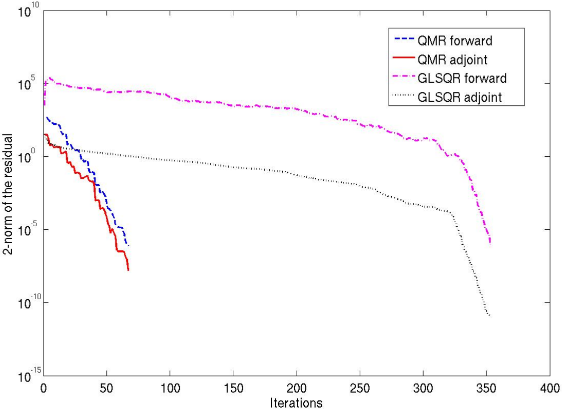

```
A=sprandn(n,n,0.2)+speye(n);
```

A matrix market example



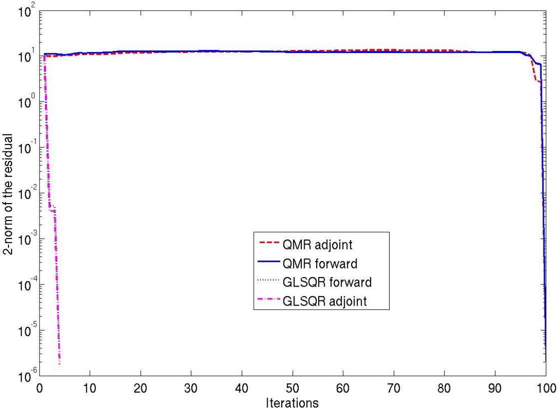
Here the matrix *orsirr_1.mtx* was used without preconditioning.

A matrix market example-preconditioned



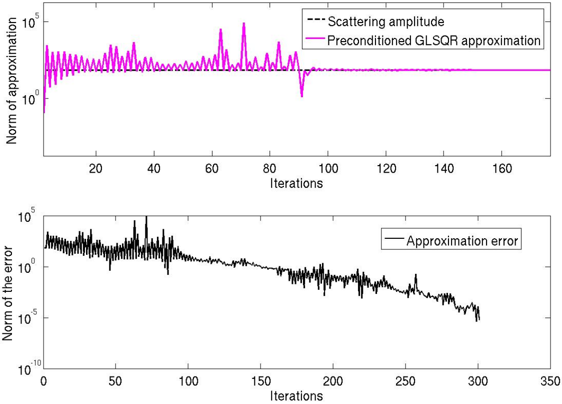
Matrix *orsirr_1.mtx* was used with ILU preconditioning.

An almost orthogonal matrix

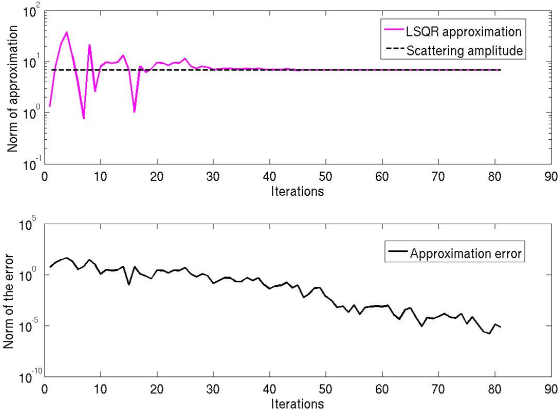


Approximating the scattering amplitude

Preconditioned LSQR for Matrix *orsirr_1.mtx*



Approximating the scattering amplitude
LSQR for 187×187 Navier-Stokes



Conclusions

- We illustrated how GLSQR can be used to compute the solution to the forward and the backward system simultaneously.
- We introduced preconditioning for GLSQR
- We approximated the scattering amplitude directly using Gauss quadrature and the connection to the Golub-Kahan bidiagonalization
- We illustrated how the interpretation of GLSQR as a Block-Lanczos method can help to approximate the scattering amplitude using the block version of Gauss quadrature.
- We introduced preconditioning for the Block-Gauss quadrature and GLSQR

We would like to thank James Lu, Michael Saunders and Gerard Meurant for many helpful comments.