$$
\begin{array}{l}
v^1 = r^0/\|r^0\| \\
\textbf{do } j = 1, n \\
\quad v^{j+1} = Av^j \\
\quad \textbf{do } i = 1, j \\
\quad\quad h_{ij} = (v^i)^H A v^i \\
\quad\quad v^{j+1} = v^{j+1} - h_{i,j} v^i \\
\quad \textbf{end do} \\
\quad h_{j+1,j} = \|v^{j+1}\| \\
\quad v^{j+1} = v^{j+1}/h_{j+1,j} \\
\textbf{end do}
\end{array}
$$

Table 3.3: The Arnoldi method.

3.1.3 GMRES

The generalized minimal residual algorithm (GMRES) [37] solves the minimal residual problem (3.5) exactly in the l_2 norm. However, in order to compute the iterates, all the previous iterates are needed. Thus the amount of work and the need for storage grow linearly with the iteration number. In practice, GMRES is run for m steps after which the method is restarted. The restarted algorithm is called GMRES(m) or GMRES with memory length m.

In the GMRES algorithm, an orthonormal basis $\{v^i\}$ is built for the Krylov subspace $K_k(r^0, A)$ with the Arnoldi method. Here r^0 is the initial residual. The Arnoldi method is essentially the modified Gram-Schmidt algorithm for orthogonalizing a set of vectors and is given in Table 3.3.

The coefficients $h_{i,j}$ in the Arnoldi method form an upper hessenberg matrix H_k such that $H_k = V_k^H A V_k$, where V_k is the unitary matrix built form the vectors v^i, $i = 1, \ldots, k$. Now we build an augmented matrix $\tilde{H}_k$ of size $(k+1) \times k$, which is the same matrix as H_k except for an additional row whose only nonzero element is $h_{k+1,k}$. The vectors v^i and the matrix $\tilde{H}_k$ satisfy the relation

$$AV_k = V_{k+1}\tilde{H}_k. \tag{3.16}$$

Now the minimal residual equation (3.5) can be cast in the form

$$\min_{z \in K_k(r^0,A)} \|b - A(x^0 + z)\| = \min_{z \in K_k(r^0,A)} \|r^0 - Az\|. \tag{3.17}$$

$q^1 = b - Ax^0$
$z_1 = \|q^1\|$
$q^1 = q^1/z_1$
repeat
    do $k = 1, m$
        $q^{k+1} = Aq^k$
        do $i = 1, k$
            $h_{i,k} = (q^i)^H q^{k+1}$
            $q^{k+1} = q^{k+1} - h_{i,k} q^i$
        end do
        $h_{k+1,k} = \|q^{k+1}\|$
        $q^{k+1} = q^{k+1}/h_{k+1,k}$
    end do
    do $k = 1, m$
        $cc = \sqrt{|h_{k,k}|^2 + |h_{k+1,k}|^2}$
        $c = |h_{k,k}|/cc$
        $h_{k,k} = cc\, h_{k,k}/|h_{k,k}|$
        $s = h_{k+1,k}/h_{k,k}$
        do $i = k+1, m$
            $$\begin{pmatrix} h_{k,i} \\ h_{k+1,i} \end{pmatrix} = \begin{pmatrix} c & \overline{s} \\ -s & c \end{pmatrix} \begin{pmatrix} h_{k,i} \\ h_{k+1,i} \end{pmatrix}$$
        end do
        $$\begin{pmatrix} z_k \\ z_{k+1} \end{pmatrix} = \begin{pmatrix} c & \overline{s} \\ -s & c \end{pmatrix} \begin{pmatrix} z_k \\ 0 \end{pmatrix}$$
    end do
    $y_m = z_m/h_{m,m}$
    do $i = m, 1, -1$
        $$y_i = \left(z_i - \sum_{j=i+1}^{m} h_{i,j} y_j \right) / h_{i,i}$$
    end do
    $x = x + \sum_{i=1}^{m} y_i q^i$
    $r = b - Ax$
    $z_1 = \|r\|$
    $q^1 = r/z_1$
until $z_1 < \epsilon$

Table 3.4: The generalized minimal residual algorithm (GMRES).

Because $\{v^i\}$ is a basis for the Krylov subspace, z is of the form $z = V_k y$. If we denote $\beta = \|r^0\|$, the function to be minimized can be written as

$$J(y) = \|\beta v^1 - AV_k y\|. \tag{3.18}$$

When we use the orthonormality of V_k and Equation (3.16), we obtain

$$J(y) = \|\beta e^1 - \tilde{H}_k y\|, \tag{3.19}$$

where e^1 is the first unit vector in $\mathbb{R}^{k+1}$. In order to facilitate the minimization of $J(y)$ we use the QR factorization of $\tilde{H}_k$, namely

$$Q_k \tilde{H}_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix}, \tag{3.20}$$

where Q_k is a $(k+1) \times (k+1)$ unitary matrix and R_k is a $k \times k$ upper triangular matrix. With this decomposition, the function $J(y)$ finally reads

$$J(y) = \left\| \begin{pmatrix} g^k \\ g_{k,k} \end{pmatrix} - \begin{pmatrix} R_k \\ 0 \end{pmatrix} y \right\|, \tag{3.21}$$

where $\begin{pmatrix} g^k \\ g_{k,k} \end{pmatrix} = Q_k \beta e^1$. Now we can clearly see that $J(y)$ is minimized if y is the solution of the upper triangular system of equations $R_k y = g^k$ and that the minimal value of $J(y)$ is $|g_{k,k}|$. The solution in the original variables can be recovered from

$$x = x^0 + \sum_{i=1}^{k} y_i v^i. \tag{3.22}$$

The QR factorization of $\tilde{H}_k$ is updated after each step of the Arnoldi process. In the GMRES algorithm, the elements of $\tilde{H}_k$ are replaced by the elements of R_k. The new column of $\tilde{H}_k$ produced by the Arnoldi iteration is modified by a complex Givens rotation so that the last row of $\tilde{H}_k$ is zeroed. The same rotation is also applied to the vector g^k. The rotations are not needed in the rest of the algorithm, so that the matrix Q_k is never formed explicitly.

The restarted algorithm GMRES(m) is given in Table 3.4.

3.1.4 QMR

The quasi-minimal residual (QMR) algorithm by Freund [15, 17] gives only approximate solutions to the minimal residual problem (3.5) so that it cannot converge in fewer iterations than the full GMRES algorithm. On the other hand, QMR iterates can be computed efficiently with short recurrences so that QMR competes favourably with GMRES. Furthermore, only a few previous iterates need to be saved in the algorithm. In this work we use a version of QMR tailored for complex symmetric matrices.

The following description of QMR is from [15]. QMR is based on the Lanczos process which can be used to calculate eigenvalues of matrices [25]. The complex symmetric Lanczos algorithm gives an orthogonal matrix V with respect to the indefinite inner product

$$(a, b) = b^T a \tag{3.23}$$

and a tridiagonal matrix T such that $V^T A V = T$. In the Lanczos algorithm the matrices V and T are built gradually. We set

$$V_k = \begin{pmatrix} v^1 & v^2 & \dots & v^k \end{pmatrix} \tag{3.24}$$

and

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \ddots & \vdots \\ 0 & \beta_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_k \\ 0 & \dots & 0 & \beta_k & \alpha_k \end{pmatrix}. \tag{3.25}$$

The Lanczos recursion is as follows:

1. Start:
   - Choose $r^0 \neq 0$;
   - Set $\tilde{v}^1 = r^0$ and $v^0 = 0$.
2. For $k = 1, 2, \cdots$ do:
   - Compute $\beta_k = \left((\tilde{v}^k)^T \tilde{v}^k\right)^{1/2}$;
   - If $\beta_k = 0$ then stop
   - Otherwise, set $v^k = \tilde{v}^k / \beta_k$;
   - Compute $\alpha_k = (v^k)^T A v^k$;
   - Set $\tilde{v}^{k+1} = A v^k - \alpha_k v^k - \beta_k v^{k-1}$.

QMR computes quasi-minimal solutions x^k to the minimal residual problem (3.5). It generates the matrices V_k and T_k by the Lanczos algorithm. QMR also forms the augmented matrix

$$\tilde{T}_k := \begin{pmatrix} T_k \\ \beta_{k+1}(e^k)^T \end{pmatrix} \text{ with } (e^k)^T = \begin{pmatrix} 0 & \cdots & 0 & 1 \end{pmatrix}, \tag{3.26}$$

so that $AV_k = V_{k+1}\tilde{T}_k$. The vectors v^i form a basis for the Krylov subspace $K_k(r^0, A)$ so that the k^{th} iterate of any Krylov subspace method is given by

$$x^k = x^0 + V_k z^k, \tag{3.27}$$

where z^k is a vector in $\mathbb{C}^k$. Because $r^0 = \beta_1 v^1$, it follows that the residual is given by

$$r^k = V_{k+1}(\beta_1 e^1 - \tilde{T}_k z^k). \tag{3.28}$$

Here e^1 is first unit vector $(1 \quad 0 \quad \cdots \quad 0)^T$.

The minimal residual problem (3.5) can be written as

$$\|b - Ax^k\| = \min_{z^k \in \mathbb{C}^k} \|V_{k+1}(\beta_1 e^1 - \tilde{T}_k z^k)\|, \tag{3.29}$$

Note that the matrix V_k is an orthonormal matrix but not a unitary matrix, i.e. the column vectors v^k are orthonormal with respect to the inner product $(x, y) = y^T x$ but not with respect to the inner product $(x, y) = y^H x$.

In the QMR approach we introduce a weight matrix

$$\Omega_{k+1} = \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_{k+1}) \text{ with } \omega_j > 0 \; \forall j. \tag{3.30}$$

Typically one takes $\omega_j = \|v_j\|$. Instead of solving (3.29) (like GMRES does), QMR solves a simpler problem

$$\min_{z \in \mathbb{C}^k} \|\omega_1 \beta_1 e^1 - \Omega_{k+1}\tilde{T}_k z\|. \tag{3.31}$$

whose solution can be computed easily with short recurrencies. If V were unitary, minimizing (3.31) would given the same answer as minimizing (3.29). In the practical implementation of QMR the QR factorization of $\Omega_{k+1}\tilde{T}_k$ is computed and updated during each iteration using Givens rotations.

The outline of the QMR algorithm is as follows:

1. Start:
   - Choose $x^0 \in C^n$.
   - Initialize all the other variables.
2. For $k = 1, 2, \ldots$ do:
   - Calculate one step of the Lanczos algorithm (Algorithm 1).
   - Update the QR factorization of $\Omega_{k+1}\tilde{T}_k$.
   - Solve (3.31) using the QR factorization giving z^k.

$\tilde{v}^1 = b - Ax^0$

$v^0 = p^0 = p^{-1} = 0$

$\omega_0 = 0$

$\beta_1 = ((\tilde{v}^1)^T \tilde{v}^1)^{1/2}$

$\omega_1 = \sqrt{(\mathrm{Re}\, \tilde{v}^1)^T (\mathrm{Re}\, \tilde{v}^1) + (\mathrm{Im}\, \tilde{v}^1)^T (\mathrm{Im}\, \tilde{v}^1)} / |\beta_1|$

$\tilde{\tau}_1 = \omega_1 \beta_1$

$c_0 = c_{-1} = 1$

$s_0 = s_{-1} = 0$

$k = 0$

repeat

 $k = k + 1$

 if $\beta_k = 0$ **stop** (x^{k-1} is the solution)

 $v^k = \tilde{v}^k / \beta_k$

 $\alpha_k = (v^k)^T A v^k$

 $\tilde{v}^{k+1} = Av^k - \alpha_k v^k - \beta_k v^{k-1}$

 $\beta_{k+1} = ((\tilde{v}^{k+1})^T \tilde{v}^{k+1})^{1/2}$

 $\omega_k = \sqrt{(\mathrm{Re}\, \tilde{v}^k)^T (\mathrm{Re}\, \tilde{v}^k) + (\mathrm{Im}\, \tilde{v}^k)^T (\mathrm{Im}\, \tilde{v}^k)} / |\beta_k|$

 $\theta_k = \overline{s_{k-2}} \omega_{k-1} \beta_k$

 $\eta_k = c_{k-1} c_{k-2} \omega_{k-1} \beta_k + \overline{s_{k-1}} \omega_k \alpha_k$

 $\tilde{\zeta}_k = c_{k-1} \omega_k \alpha_k - s_{k-1} c_{k-2} \omega_{k-1} \beta_k$

 $|\zeta_k| = (|\tilde{\zeta}_k|^2 + \omega_{k+1}^2 |\beta_{k+1}|^2)^{1/2}$

 if $\tilde{\zeta}_k \neq 0$ **then**

  $\zeta_k = |\zeta_k| \tilde{\zeta}_k / |\tilde{\zeta}_k|$

 else

  $\zeta_k = |\zeta_k|$

 end if

 $c_k = \tilde{\zeta}_k / \zeta_k$

 $s_k = \omega_{k+1} \beta_{k+1} / \zeta_k$

 $p^k = (v^k - \eta_k p^{k-1} - \theta_k p^{k-2}) / \zeta_k$

 $\tau_k = c_k \tilde{\tau}_k$

 $\tilde{\tau}_{k+1} = -s_k \tilde{\tau}_k$

 $x^k = x^{k-1} + \tau_k p^k$

until $\|b - Ax^k\| < \epsilon$

Table 3.5: The quasi-minimal residual (QMR) algorithm.

- Compute $x^k = x^0 + V_k z^k$.

The QMR algorithm is given in Table 3.5.

The QMR iterates can be computed with low storage requirements. In particular, the matrix V_k need not be saved, the calculation of x^k can be done with an easy recursion. Each QMR iteration consists of one matrix-vector product, a few inner products and vector updates plus some scalar operations. During the computation only a few vectors have to be saved.

3.1.5 Block QMR

When iterative methods are applied to problems with many right-hand sides, it may be advantageous to apply block iterative versions. O'Leary has constructed a block version of the conjugate gradient algorithm [31, 32]. Recently, Boyse and Seidl have developed a block version of QMR [5].

The block algorithms have been demonstrated to reduce the number of iterations in iterative methods. Theoretically, when m right-hand side vectors are used, the number of iterations could be reduced by a factor of m.

The condition number gives an upper limit to the convergence of the conjugate gradient algorithm. For Hermitian positive definite systems the condition number is given by the ratio of the largest to the smallest eigenvalue. The convergence of the block conjugate gradient is controlled by the ratio of the largest to the m^{th} smallest eigenvalue and may thus accelerate convergence [5].

Another way to look at the convergence acceleration of the block methods is to consider the corresponding Krylov subspaces. The ordinary (point) Krylov subspace algorithms generate iterates of the form

$$x^k \in x^0 + \operatorname{span}\{r^0, Ar^0, \ldots, A^{k-1}r^0\}. \tag{3.32}$$

The iterates of the block algorithms are characterized by

$$x^k \in x^0 + \operatorname{span}\{\operatorname{col}_i(r^0), A\operatorname{col}_i(r^0), \ldots, A^{k-1}\operatorname{col}_i(r^0)\}\ i = 1, \ldots, m. \tag{3.33}$$

Here x^k and r^0 are $n \times m$ matrices and $\operatorname{col}_i(r)$ is the i^{th} column of r.

The block algorithm works with an approximating subspace that is m times as big as in the point algorithm. This should produce better convergence properties. In practice the benefit from the block algorithm may vary. The Krylov subspaces for individual right-hand side vectors may or may not contain useful information for the solutions of other right-hand sides.

The block QMR algorithm is given in Table 3.6. The terms x^k, p^k, v^k, and $\tilde{v}^k$ are n by m matrices while a_k, b_k, c_k, d_k, α_k, β_k, θ_k, η_k, ζ_k, $\tilde{\zeta}_k$, τ_k, $\tilde{\tau}_k$, and ω_k are m by m matrices with ζ_k and β_k being upper triangular and ω_k being diagonal. The operations marked with $(*)$, e.g. $v^{k+1}\beta_{k+1} = \tilde{v}^{k+1}$, are so called quasi QR factorizations of the right-hand side [5]. This means that β_{k+1} is an upper triangular matrix and the columns of the matrix v^{k+1} are orthonormal with respect to the indefinite inner product

$$< x, y >_Q = x^T y. \tag{3.34}$$

This quasi QR factorization can be computed with the modified Gram-Schmidt process using the inner product (3.34). The operation marked with $(**)$ means that a $2m$ by $2m$ unitary matrix

$$Q(a_k, b_k, c_k, d_k) = \begin{pmatrix} a_k & b_k \\ c_k & d_k \end{pmatrix} \tag{3.35}$$

is computed such that

$$Q(a_k, b_k, c_k, d_k) \begin{pmatrix} \tilde{\zeta}_k \\ \omega_{k+1}\beta_{k+1} \end{pmatrix} = \begin{pmatrix} \zeta_k \\ 0 \end{pmatrix}, \tag{3.36}$$

where ζ_k is upper triangular. This can be done by computing the $2m$ by $2m$ QR factorization of the $2m$ by m matrix

$$\begin{pmatrix} \tilde{\zeta}_k \\ \omega_{k+1}\beta_{k+1} \end{pmatrix}. \tag{3.37}$$

The residual can be computed directly, using a special recursion or using the so called quasi residual as given by $\tilde{\tau}_k$ [5]. The recursion is derived as follows. The residual is given by

$$\begin{aligned} r^k &= V_{k+1}\Omega_{k+1}^{-1}(\omega_1 e^1 \beta_1 - \Omega_{k+1}\tilde{T}_k z^k) \\ &= V_{k+1}\Omega_{k+1}^{-1} Q_{k+1}^H \left(Q_{k+1}\omega_1 e^1 \beta_1 - \begin{pmatrix} R_k \\ 0 \end{pmatrix} z^k \right). \end{aligned} \tag{3.38}$$

Now define

$$\tilde{w}^{k+1} = V_{k+1}\Omega_{k+1}^{-1} Q_{k+1}^H \begin{pmatrix} 1 \\ \vdots \\ 0 \\ I_m \end{pmatrix}, \tag{3.39}$$

after which the residual is given by

$$r^k = \tilde{w}^{k+1}\tilde{\tau}_{k+1}. \tag{3.40}$$

$$
\begin{array}{l}
v^0 = p^0 = p^{-1} = 0 \\
c_0 = c_{-1} = b_0 = b_{-1} = 0 \\
a_0 = a_{-1} = d_0 = d_{-1} = I_m \\
\tilde{v}^1 = b - Ax^0 \\
\omega_0 = 0 \\
v^1 \beta_1 = \tilde{v}^1 \ (*) \\
\omega_1 = \mathrm{Diag}\{\|\mathrm{col}_i(v^1)\|\} \\
\tilde{\tau}_1 = \omega_1 \beta_1 \\
k = 0 \\
\textbf{repeat} \\
\qquad k = k + 1 \\
\qquad \tilde{v}^{k+1} = Av^k - v^{k-1}\beta_k^T \\
\qquad \alpha_k = (v^k)^T \tilde{v}^{k+1} \\
\qquad \tilde{v}^{k+1} = \tilde{v}^{k+1} - v^k \alpha_k \\
\qquad v^{k+1}\beta_{k+1} = \tilde{v}^{k+1} \ (*) \\
\qquad \omega_{k+1} = \mathrm{Diag}\{\|\mathrm{col}_i(v^{k+1})\|\} \\
\qquad \theta_k = b_{k-2}\omega_{k-1}\beta_k^T \\
\qquad \eta_k = a_{k-1}d_{k-2}\omega_{k-1}\beta_k^T + b_{k-1}\omega_k \alpha_k \\
\qquad \tilde{\zeta}_k = c_{k-1}d_{k-2}\omega_{k-1}\beta_k^T + d_{k-1}\omega_k \alpha_k \\
\qquad \text{Find } Q(a_k, b_k, c_k, d_k) \text{ and } \zeta_k \ (**) \\
\qquad p^k = (v^k - p^{k-1}\eta_k - p^{k-2}\theta_k)\zeta_k^{-1} \\
\qquad \tau_k = a_k \tilde{\tau}_k \\
\qquad x^k = x^{k-1} + p^k \tau_k \\
\qquad \tilde{\tau}_{k+1} = c_k \tilde{\tau}_k \\
\textbf{until } \max\limits_{1 \le i \le m} \|\mathrm{col}_i(b - Ax^k)\| < \epsilon
\end{array}
$$

Table 3.6: The block QMR algorithm. The operations marked with $(*)$ and $(**)$ are explained in the text.

We obtain a recursion for $\tilde{w}^{k+1}$ by writing

$$\tilde{w}^{k+1} = \begin{pmatrix} V_k & v^{k+1} \end{pmatrix} \begin{pmatrix} \Omega_k^{-1} & 0 \\ 0 & \omega_{k+1}^{-1} \end{pmatrix} \begin{pmatrix} Q_k^H & 0 \\ 0 & I_m \end{pmatrix} \begin{pmatrix} I_{(k-1)m} & 0 \\ 0 & \begin{matrix} a^H & c^H \\ b^H & d^H \end{matrix} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 0 \\ I_m \end{pmatrix}. \tag{3.41}$$

This simplifies to

$$\tilde{w}^{k+1} = \tilde{w}^k c_k^H + v^{k+1} \omega_{k+1}^{-1} d_k^H. \tag{3.42}$$

In the algorithm we use the residual norm given by

$$\max_i \|\mathrm{col}_i(r^k)\|. \tag{3.43}$$

3.1.6 BCG

The biconjugate gradient method (BCG) [26] is related to the Lanczos algorithm. BCG generates two sequences of residuals r^i and $\tilde{r}^i$ and search directions p^i and $\tilde{p}^i$ that are generated from A and A^H, respectively, by the same recursion as in the conjugate gradient. The coefficients α and β in the conjugate gradient algorithm are the same for both sequences; they force the sequences to be bi-orthogonal:

$$(\tilde{r}^i)^H r^j = (\tilde{p}^i)^H A p^j = 0, \ i \neq j. \tag{3.44}$$

We used the complex symmetric BCG algorithm, given in Table 3.7, which is based on the complex symmetric Lanczos algorithm [15],

3.1.7 CGS

In the general biconjugate gradient algorithm, the residual is the product of the initial residual and a polynomial in A:

$$r^i = P_i(A) r^0. \tag{3.45}$$

Similarly,

$$\tilde{r}^i = P_i(A) \tilde{r}^0. \tag{3.46}$$

so that

$$(\tilde{r}^i, r^i) = (\tilde{r}^0, P_i^2(A) r^0). \tag{3.47}$$

This suggests that we could obtain a "transpose-free" algorithm (i.e. A^H is not needed) that would apply the BCG polynomial twice. This algorithm is called the conjugate gradient squared (CGS), given in table 3.8 [39].

$$
\begin{array}{l}
r^0 = b - Ax^0 \\
p^0 = r^0 \\
k = 0 \\
\textbf{repeat} \\
\quad k = k + 1 \\
\quad \alpha_k = \dfrac{(r^{k-1})^T r^{k-1}}{(p^{k-1})^T A p^{k-1}} \\
\quad x^k = x^{k-1} + \alpha_k p^{k-1} \\
\quad r^k = r^{k-1} - \alpha_k A p^{k-1} \\
\quad \beta_k = \dfrac{(r^k)^T r^k}{(r^{k-1})^T r^{k-1}} \\
\quad p^k = r^k + \beta_k p^{k-1} \\
\textbf{until } \|r^k\| < \epsilon
\end{array}
$$

Table 3.7: The biconjugate gradient method (BCG).

$$
\begin{array}{l}
r^0 = b - Ax^0 \\
\textbf{choose } \tilde{r} \text{ so that } (\tilde{r}, r^0) \neq 0 \\
d^1 = u^0 = r^0 \\
k = 0 \\
\textbf{repeat} \\
\quad k = k + 1 \\
\quad \alpha_k = \dfrac{(\tilde{r}, r^{k-1})}{(\tilde{r}, Ad^k)} \\
\quad q^k = u^{k-1} - \alpha_k A d^k \\
\quad w^k = u^{k-1} + q^k \\
\quad x^k = x^{k-1} + \alpha_k w^k \\
\quad r^k = r^{k-1} - \alpha_k A w^k \\
\quad \beta_k = \dfrac{(\tilde{r}, r^k)}{(\tilde{r}, r^{k-1})} \\
\quad u^k = r^k + \beta_k q^k \\
\quad d^{k+1} = u^k + \beta_k (q^k + \beta_k d^k) \\
\textbf{until } \|r^k\| < \epsilon
\end{array}
$$

Table 3.8: The conjugate gradient squared algorithm (CGS).

$$
\begin{array}{l}
r^0 = b - Ax^0 \\
\textbf{choose } \tilde{r} \text{ so that } (\tilde{r}, r^0) \neq 0 \\
d^1 = r^0 \\
k = 0 \\
\textbf{repeat} \\
\quad k = k + 1 \\
\quad \alpha_k = \dfrac{(\tilde{r}, r^{k-1})}{(\tilde{r}, Ad^k)} \\
\quad u^k = r^{k-1} - \alpha_k Ad^k \\
\quad \omega_k = (Au^k, u^k)/(Au^k, Au^k) \\
\quad x^k = x^{k-1} + \alpha_k d^k + \omega_k u^k \\
\quad r^k = u^k - \omega_k Au^k \\
\quad \beta_k = \dfrac{(\tilde{r}, r^k)}{(\tilde{r}, r^{k-1})} \alpha_k \omega_k \\
\quad d^{k+1} = r^k + \beta_k (d^k - \omega_k Ad^k) \\
\textbf{until } \|r^k\| < \epsilon
\end{array}
$$

Table 3.9: The biconjugate gradients stabilized (BiCGStab).

3.1.8 BiCGStab

Both BCG and CGS often exhibit wildly erratic convergence behaviour with a highly oscillating residual norm. The biconjugate gradients stabilized (BiCGStab) [41] produces smoother residuals by constructing the residual as $Q_i(A)P_i(A)r^0$, where $Q_i(A) = \prod_{j=1}^{i}(I - \omega_j A)$. The coefficients ω_j are chosen so that the residual norm $\|r^i\|$ is minimized.

The BiCGStab algorithm is given in Table 3.9.

3.2 Spectrum of the coefficient matrix

The convergence of iterative methods is governed by the eigenvalues or spectrum of the coefficient matrix. For four pseudospherical clusters of dipoles, the corresponding eigenvalue spectra are shown in Figures 3.1–3.4. These matrices arise from four different discretizations of the same particle; while the size of the cluster is increased, the size parameter x_0 of the computational unit is decreased. Here the index of

N	x_0	N_L	p_L
96	0.3	80	83%
408	0.188	393	96%
912	0.144	900	98.7%
3192	0.0947	3180	99.6%

Table 3.10: The total number of eigenvalues N, the number of eigenvalues on the line segment N_L, and the percentage of eigenvalues on the line segment p_L.

refraction is $m = 1.6 + 0.05i$. Figure 3.5 show the eigenvalues of the linear system arising from a discretization of a diffusion limited aggregation (DLA) particle of 500 dipoles.

In the pseudosphere spectra, most of the eigenvalues lie on a line segment and the eigenvalues get sparser on the line as one goes towards the end of the line segment. This is demonstrated in Figure 3.6, where the distances of the eigenvalues from the leftmost eigenvalue on the line are plotted against the eigenvalue number. The percentage of eigenvalues on and off the line is shown in table 3.10. When the number of dipoles is increased, the spectrum converges so that almost every eigenvalue is on the line, only a constant number of eigenvalues are outside the line. In the DLA case, there are many more eigenvalues that do not lie on the line.

The structure of the spectrum probably accounts for the good performance of QMR. In a further study the spectrum of the continuous operator $\mathcal{K}$, defined in Equation (2.15), will be computed and compared to the eigenvalues of the linear equation. It is most probable that the spectrum of the continuous operator explains the linear part of the spectrum. In that case, the eigenvalues off the line should converge towards the line as discretization is refined. However, this was not the case in our simulations. Some eigenvalues seem to converge to points that are far away from the line. This is probably some artifact that is due to the discretization and that should disappear when better discretization of the integral equation is used. Better discretization means that the integrals needed in forming the linear equations should be computed using more accurate integration formulas that described in chapter 2. The DDA formalism is equivalent to using one-point integration formulas on each computational box.

The orientation and length of the line segment in the spectrum is determined for a fixed geometry by the coefficient $k^3\alpha$ in equation (2.4). The angle of the line with respect to the x axis is the same as the argument of $k^3\alpha$.

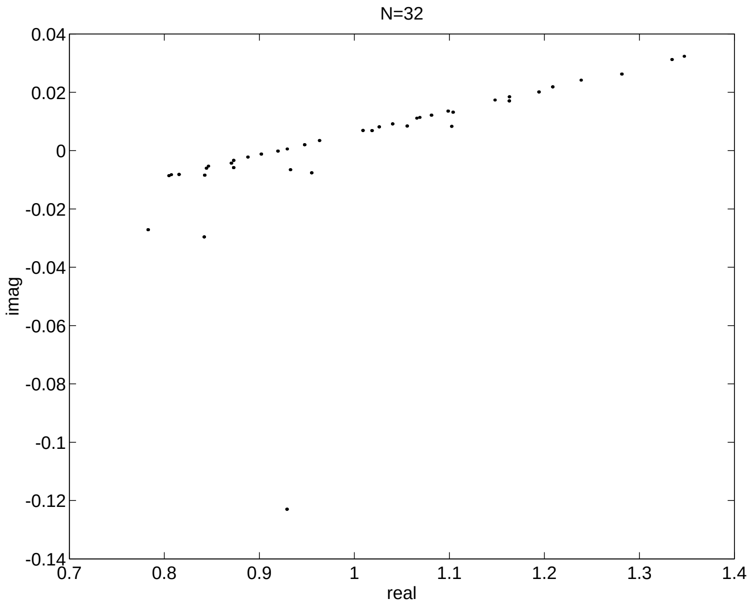

Figure 3.1: Spectrum of the linear system arising from the DDA calculations. The number of dipoles is 32 and the number of unknowns is 96.

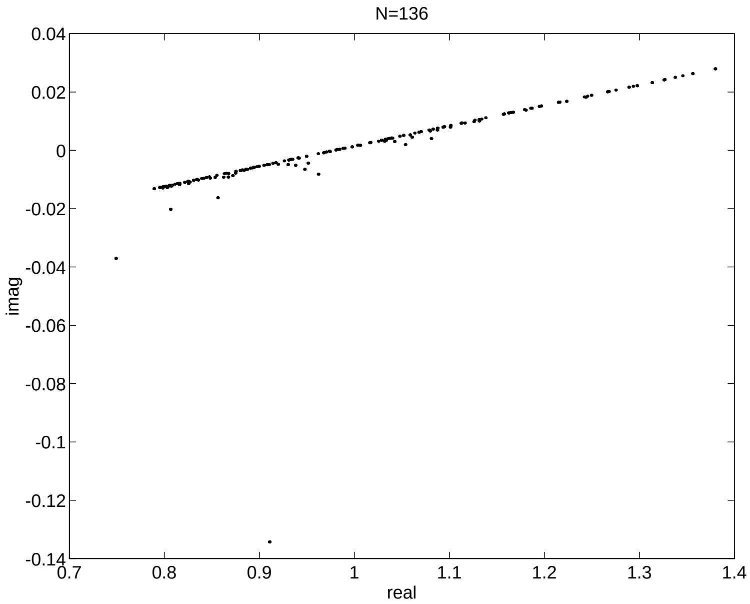

Figure 3.2: Spectrum of the linear system for 136 dipoles.

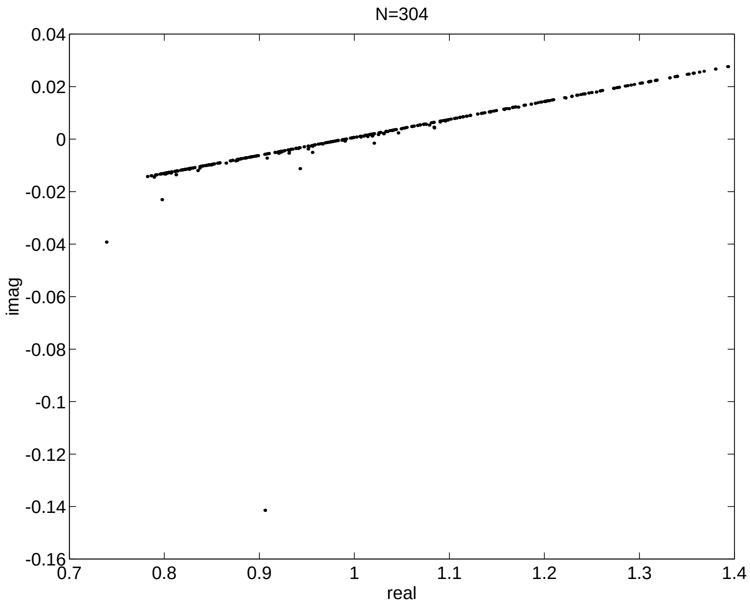

Figure 3.3: Spectrum of the linear system for 304 dipoles.

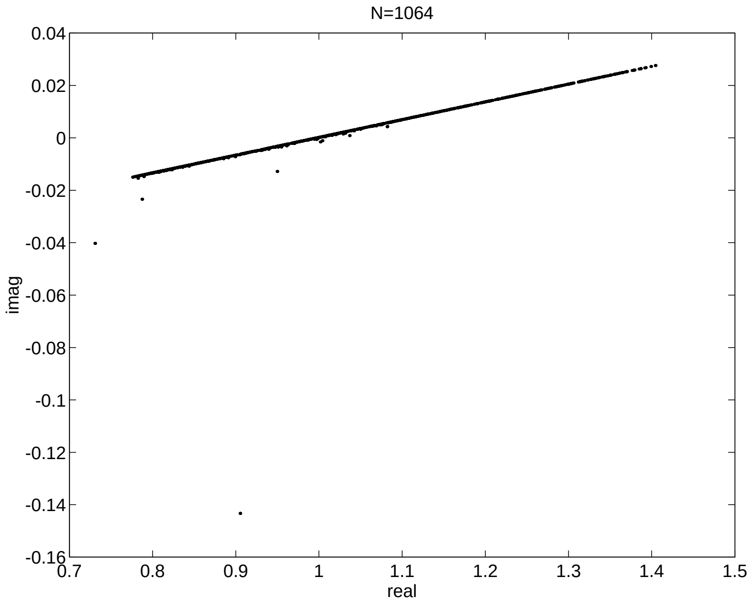

Figure 3.4: Spectrum of the linear system for 1064 dipoles.

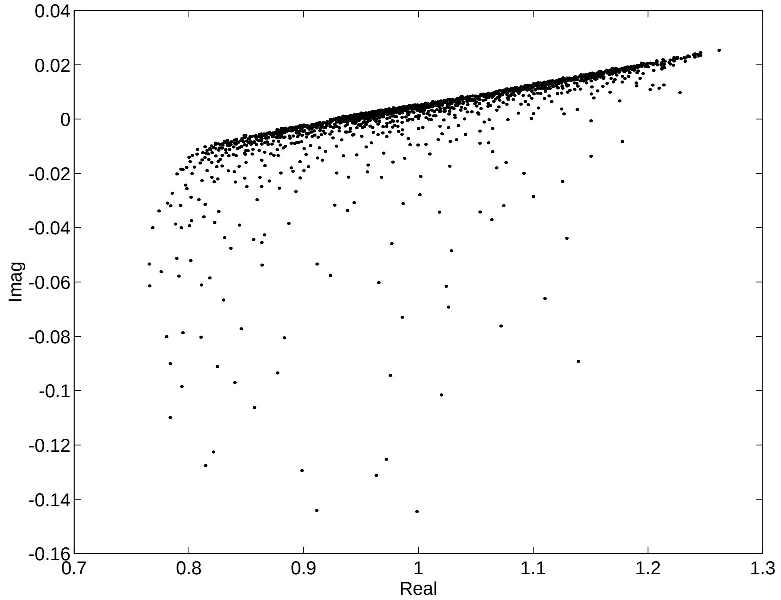

Figure 3.5: Spectrum of the linear system for the DLA particle of 500 dipoles.

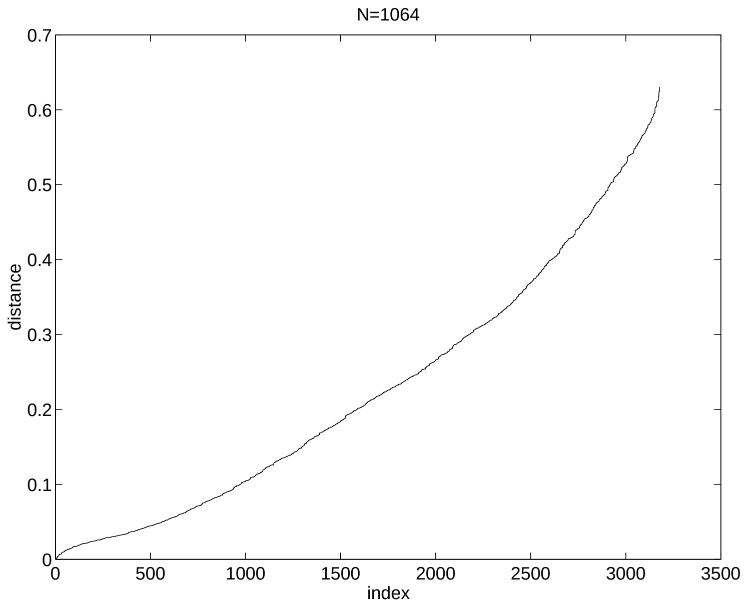

Figure 3.6: The distances of the eigenvalues on the linear part of the spectrum from the leftmost eigenvalue on the line.

3.2.1 Optimal reduction factor

The spectrum of the coefficient matrix together with the initial residual vector determine the properties of the iterative methods. We would like to obtain an estimate on the convergence speed of iterative methods from the spectrum. This can be done using the machinery described in a book by Olavi Nevanlinna [30].

Let us consider iterative methods for a fixed point problem

$$x = \mathcal{K}x + g, \tag{3.48}$$

where $\mathcal{K}$ is a bounded linear operator in a Banach space and g is a given vector. We shall be working with polynomial acceleration and define P_n to be the set of polynomials p_n of degree not exceeding n together with normalization $p_n(1) = 1$. In this formalism, Krylov subspace methods generate iterates of the form

$$x^k = p_k(\mathcal{K})x^0 + q_{k-1}(\mathcal{K})g, \tag{3.49}$$

where the polynomial p_k belongs to P_k and $p_k(\lambda) = 1-(1-\lambda)q_{k-1}(\lambda)$. The residuals are defined by $d^k = g - (1-\mathcal{K})x^k$. In terms of the polynomial p_k, the residual is given by

$$d^k = p_k(\mathcal{K})d^0. \tag{3.50}$$

Thus a good Krylov subspace method would generate polynomials for which $\|p_k(\mathcal{K})\|$ is small.

We now ask how fast can $\|p_k(\mathcal{K})\|$ decrease as k grows. The answer is given by the *optimal reduction factor* defined by

$$\eta(\mathcal{K}) = \inf \|p_k(\mathcal{K})\|^{1/k}, \tag{3.51}$$

where the infimum is over all positive k and all $p_k \in P_k$. The residual can then decrease with a rate

$$\|d^k\| = (\eta(\mathcal{K}))^k \|d^0\|. \tag{3.52}$$

The optimal reduction factor depends only on the spectrum $\sigma(\mathcal{K})$ of the operator $\mathcal{K}$ by

$$\eta(\mathcal{K}) = \eta_{\sigma(\mathcal{K})}, \tag{3.53}$$

where

$$\eta_E = \inf \|p_k\|_E^{1/k} \tag{3.54}$$

and

$$\|f\|_E = \max_{\lambda \in E} |f(\lambda)|. \tag{3.55}$$

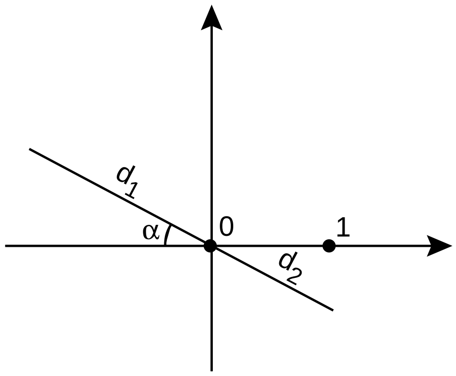

Figure 3.7: The linear part of the spectrum for the operator $\mathcal{K}$.

Here E is a compact subset of the complex plane. Finally, η_E is given by

$$\eta_E = \frac{1}{|\phi(1)|}, \tag{3.56}$$

where ϕ is the conformal map that maps G_∞ to the outside of the closed unit disk and G_∞ is the unbounded component of the complement of E. Here we assume that G_∞ is simply connected.

Thus in order to find the optimal reduction factor for a given operator, we need to find a conformal map that maps the complement of the spectrum to the outside of the closed unit disk and evaluate this at unity. For the scattering problem, the line part of the spectrum can be modeled as a line segment that goes through unity, makes an angle α with the positive x-axis and extends a distance d_1 in the upper half plane and a distance d_2 in the lower. However, the coefficient matrix corresponds to the operator $I - \mathcal{K}$ and thus the spectrum of $\mathcal{K}$ is a line that goes through the origin, makeas an angle α with the negative x-axis, and has the lengths d_1 and d_2 as described above. This is visualized in Figure 3.7.

We know that the outside of interval $[-1, 1]$ can be mapped to the outside of the unit disk with the conformal map [30]

$$\phi_3(\lambda) = \lambda + \sqrt{\lambda^2 - 1}. \tag{3.57}$$

In the scattering case, the conformal map can be obtained as a combination of three maps. First rotate the spectrum so that it aligns with the x-axis. This can be done with the map

$$\phi_1(\lambda) = e^{i\alpha}\lambda. \tag{3.58}$$

Then this line segment is mapped to the interval $[-1, 1]$ with the map

$$\phi_2(\lambda) = \frac{2\lambda + d_1 - d_2}{d_1 + d_2}. \tag{3.59}$$

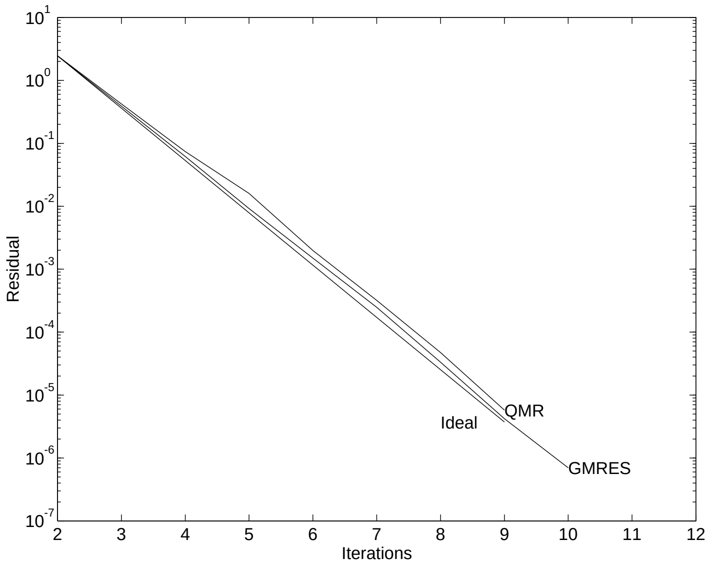

Figure 3.8: The convergence of QMR and GMRES together with the optimal reduction rate. The geometry is a pseudosphere of 1064 dipoles.

After this we can use the map ϕ_3. Thus the optimal reduction factor of $\mathcal{K}$ is given by

$$\eta = \frac{1}{|\phi_3(\lambda_1)|}, \tag{3.60}$$

where

$$\lambda_1 = \frac{2e^{i\alpha} + d_1 - d_2}{d_1 + d_2}. \tag{3.61}$$

Now we shall compute how well the optimal reduction factor characterizes the practical convergence. We shall consider the linear system whose spectrum was shown in Figure 3.4. From the spectrum we can estimate $\alpha = 0.067$, $d_1 = 0.4055$, $d_2 = 0.224$. This gives an optimal reduction factor of 0.147. When we solve the corresponding linear system with GMRES, the linear convergence rate is 0.152. The agreement is excellent as can be seen in Figure 3.8.

The spectrum of the linear system arising from the discretization of a DLA particle was given in Figure 3.5. In this case, the spectrum is more spread out, but we can still try to predict the convergence of iterative solvers from the line part of the spectrum. The estimated line parameters for the DLA case are $\alpha = 0.081$, $d_1 = 0.262$, and $d_2 = 0.202$. This gives $\eta = 0.114$ for the optimal reduction factor. In practical calculations, GMRES converges with rate 0.155, as can be seen from Figure 3.9. As expected, the optimal reduction factor gives a little too optimistic convergence speed because the eigenvalues off the line are neglected.

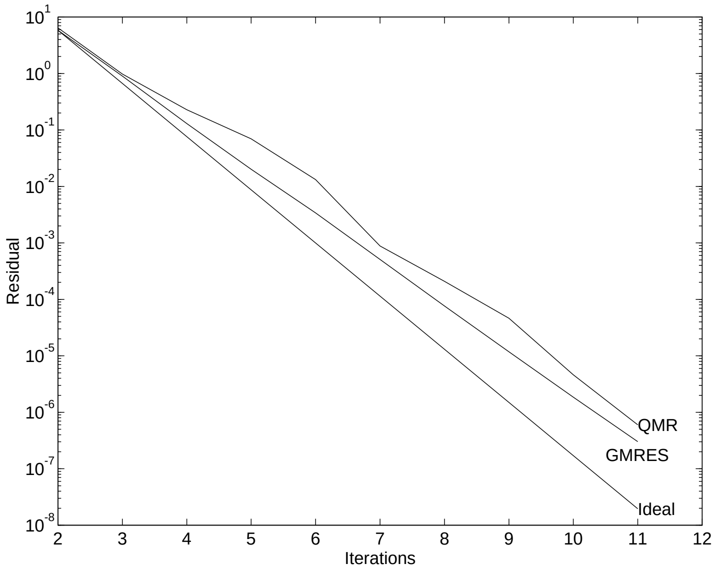

Figure 3.9: The convergence of QMR and GMRES together with the optimal reduction rate. The geometry is a DLA particle of 500 dipoles.

3.3 Comparison of iterative methods

In this work, the iterative methods described in section 3.1 were implemented and tested. The convergence of these methods depends on the geometry of the dipoles, the index of refraction m, and the size parameter x_0. All calculations were done on a Convex C3840 at the Center for Scientific Computing using double precision arithmetic. We typically indicate the size of the system by giving the number of dipoles N_{dip}. The number of unknowns in the linear system is $3N_{dip}$.

Figure 3.10 shows the convergence of CGNR, GMRES, QMR, BCG, CGS, and BiCGStab in a typical small problem. The horizontal axis shows the number of matrix-vector products so it is roughly proportional to the CPU time. The number of matrix-vector products per iteration for these methods is show in Table 3.11. In this simulation the index of refraction was $m = 3 + 4i$, the size parameter was $x_0 = 0.3$, and the number of dipoles was $N_{dip} = 304$. In the figure, the memory length of GMRES was 20. In bigger systems, CGS had trouble to converge, and the residuals of BCG started to behave nonmonotonically. This is illustrated in Figure 3.11, which is computed with similar parameters as Figure 3.10 except for the number of dipoles which is 1064. Figure 3.12 shows the convergence of the four best solvers for a linear system with $N_{dip} = 5232$ dipoles.

In the optics and astrophysics communities, the method of choice for indefinite linear equations has been CG applied to the normal equations [9]. In [28] QMR and CGNR were compared for various geometries and for many values of the index of refraction