Chapter 9
Other Krylov Subspace Methods for Non-Symmetric Systems

The course gave only a survey of this chapter, presenting the main ideas of the methods and their main properties.

Remark 9.1. Motivation. The Krylov subspace methods GMRES and FOM for solving general linear systems of equations have the disadvantage that their costs (in memory and flops) increases with the number of iterations. A remedy is to use restarted versions, compare Remark 5.9. However, the restart might lead to a considerably slower rate of convergence. This section presents alternative approaches that are based on short recurrences but which fulfill the properties of minimizing the residual (like GMRES) or of the residual being orthogonal to the Krylov subspace (like FOM), respectively, only approximately.

Remark 9.2. Biorthogonal bases. The starting point of the alternative algorithms is the construction of a pair of biorthogonal bases

\[
\left\{ \underline{v}^{(1)}, \ldots, \underline{v}^{(k)} \right\} \text{ of } K_k \left( \underline{r}^{(0)}, A \right) = \text{span} \left\{ \underline{r}^{(0)}, A \underline{r}^{(0)}, \ldots, A^{k-1} \underline{r}^{(0)} \right\},
\]

\[
\left\{ \underline{w}^{(1)}, \ldots, \underline{w}^{(k)} \right\} \text{ of } K_k \left( \underline{r}^{(0)}, A^T \right) = \text{span} \left\{ \underline{r}^{(0)}, A^T \underline{r}^{(0)}, \ldots, (A^T)^{k-1} \underline{r}^{(0)} \right\}
\]

such that

\[
\left( \underline{w}^{(j)}, \underline{v}^{(i)} \right) = \delta_{ij}.
\]

These bases can be constructed with the Lanczos biorthogonalization procedure.

Algorithm 9.3. Lanczos biorthogonalization procedure. Given a matrix \( A \in \mathbb{R}^{n \times n} \) and \( \underline{r}^{(0)} \in \mathbb{R}^n \).

1. \( \underline{v}^{(1)} = \underline{r}^{(0)} / \| \underline{r}^{(0)} \|_2 \)
2. \( \underline{w}^{(1)} = \underline{v}^{(1)} \)
3. \( \beta_0 = 0, \gamma_0 = 0 \)
66 9 Other Krylov Subspace Methods for Non-Symmetric Systems

4. \( v^{(0)} := 0, \quad w^{(0)} = 0 \)
5. for \( j = 1 : k \)
6. \( s = A v^{(j)} \)
7. \( \tilde{s} = A^T w^{(j)} \)
8. \( \alpha_j = (w^{(j)}, s) \)
9. \( \tilde{v}^{(j+1)} = s - \alpha_j v^{(j)} - \beta_{j-1} v^{(j-1)} \)
10. \( \tilde{w}^{(j+1)} = \tilde{s} - \alpha_j \tilde{w}^{(j)} - \gamma_{j-1} w^{(j-1)} \)
11. \( \gamma_j = \| \tilde{v}^{(j+1)} \|_2 \)
12. \( \tilde{v}^{(j+1)} = \tilde{v}^{(j+1)}/\gamma_j \)
13. \( \beta_j = (\tilde{w}^{(j+1)}, v^{(j+1)}) \)
14. \( w^{(j+1)} = \tilde{w}^{(j+1)}/\beta_j \)
15. endfor

Remark 9.4. On the Lanczos biorthogonalization procedure.

- There is a short recurrence in Algorithm 9.3.
- Note that the basis of \( K_k (v^{(0)}, A) \) will be in general not orthogonal as well as the basis of \( K_k (w^{(0)}, A^T) \). For non-symmetric matrices, the computation of an orthogonal basis is not possible with a short recurrence.
- In the case \( A = A^T \), Algorithm 9.3 is exactly the Lanczos Algorithm 5.12.
- Algorithm 9.3 requires two matrix-vector products, lines 6 and 7.
- A critical point of Algorithm 9.3 is the product of the transposed of \( A \) with a vector, line 7. In some applications, \( A \) is not given explicitly or \( A \) is given only by some approximation. Then, \( A^T \) is usually not available. But much more important, the application of a number of preconditioners, see Chapter 8, becomes complicated if \( A^T \) appears in the algorithm.

Theorem 9.5. Computation of a pair of biorthogonal bases. Assume that \( (\tilde{w}^{(j)}, v^{(j)}) \neq 0 \) for all \( j = 1, \ldots, k \). Then, the Lanczos biorthogonalization procedure computes a pair of biorthogonal bases.

Proof. The theorem is proved by induction. The statement is true if \( k = 1 \) since \( w^{(1)} = v^{(1)} \), line 2 and \( \| v^{(1)} \|_2 = 1 \), line 1. For \( k = 2 \), it can be proved directly from the algorithm, in a similar way as for the general case.

Assume, the statement is proved for \( i = 1, \ldots, k - 1 \) with \( k - 1 \geq 2 \), and suppose \( (\tilde{w}^{(i)}, v^{(i)}) = (w^{(i)}, v^{(i)}) = 0 \) for \( i \neq j, 1 \leq i, j \leq k - 1 \) and \( (w^{(i)}, v^{(i)}) = 1, 1 \leq i \leq k - 1 \). The choice of \( \gamma_{k-1} \) implies \( \| v^{(k)} \|_2 = 1 \) and line 14 and the choice of \( \beta_{k-1} \) give

\[
(\tilde{w}^{(k)}, v^{(k)}) = \left( \frac{\tilde{w}^{(k)}}{(\tilde{w}^{(k)}, v^{(k)})}, v^{(k)} \right) = 1.
\]

Using lines 12 and 9, the assumption, and the definition of \( \alpha_{k-1} \) leads to
\[
\begin{align*}
&\left( w^{(k-1)}, v^{(k)} \right) = \left( w^{(k-1)}, \tilde{w}^{(k)} \right) \\
&= \frac{1}{\gamma_{k-1}} \left[ \left( w^{(k-1)}, \Lambda w^{(k-1)} \right) - \alpha_{k-1} \left( w^{(k-1)}, \underline{v}^{(k-1)} \right) \right] = 0.
\end{align*}
\]

One obtains analogously \( \left( w^{(k)}, \tilde{w}^{(k-1)} \right) = 0 \). Moreover, using the lines 12, 9, the assumption of the induction, line 10 (with \( \tilde{z} = A^T w^{(k-2)} \)), and the definition of \( \beta_{k-2} \) gives
\[
\begin{align*}
&\left( w^{(k-2)}, v^{(k)} \right) \\
&= \frac{1}{\gamma_{k-1}} \left[ \left( w^{(k-2)}, \Lambda w^{(k-1)} \right) - \alpha_{k-1} \left( w^{(k-2)}, \underline{v}^{(k-1)} \right) \right] = 0 \\
&= \frac{1}{\gamma_{k-1}} \left[ \left( w^{(k-2)}, \Lambda w^{(k-1)} \right) - \beta_{k-2} \right] \\
&= \frac{1}{\gamma_{k-1}} \left[ \left( w^{(k-2)}, \Lambda w^{(k-1)} \right) - \beta_{k-2} \right] \\
&= \frac{1}{\gamma_{k-1}} \left[ \left( \tilde{w}^{(k-1)}, \tilde{v}^{(k-1)} \right) + \alpha_{k-2} \left( \tilde{w}^{(k-2)}, w^{(k-1)} \right) \right] = 0.
\end{align*}
\]

Analogously, one checks that \( \left( w^{(k)}, \tilde{w}^{(k-2)} \right) = 0 \) and in a similar way, one obtains \( \left( w^{(k)}, \tilde{v}^{(j)} \right) = 0 \) and \( \left( w^{(j)}, \tilde{v}^{(k)} \right) = 0 \) for \( j < k - 2 \).

**Remark 9.6. Factorization of the system matrix.** For the matrix representation of the Lanczos biorthogonalization procedure, the matrices
\[
V_k = \left( \underline{v}^{(1)} \ldots \underline{v}^{(k)} \right), \quad W_k = \left( \tilde{w}^{(1)} \ldots \tilde{w}^{(k)} \right) \in \mathbb{R}^{n \times k}
\]
are introduced. From Algorithm 9.3, it follows that
\[
AV_k = V_{k+1} T_{k+1,k}, \quad A^TW_k = W_{k+1} \tilde{T}_{k+1,k} \tag{9.1}
\]
with
\[
T_{k+1,k} = \begin{pmatrix}
\alpha_1 & \beta_1 & \gamma_1 & \alpha_2 & \cdots & \beta_{k-1} & \gamma_{k-1} \\
\gamma_1 & \alpha_2 & \cdots & \beta_{k-1} & \gamma_{k-1} & \cdots & \beta_k \\
\vdots & \vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
\beta_{k-1} & \gamma_{k-1} & \cdots & \beta_k & \cdots & \cdots & \cdots \\
\end{pmatrix}, \quad \tilde{T}_{k+1,k} = \begin{pmatrix}
\alpha_1 & \gamma_1 & \beta_1 & \alpha_2 & \cdots & \beta_{k-1} & \gamma_{k-1} \\
\gamma_1 & \beta_1 & \alpha_2 & \cdots & \beta_{k-1} & \gamma_{k-1} & \cdots & \beta_k \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\beta_{k-1} & \gamma_{k-1} & \cdots & \beta_k & \cdots & \cdots & \cdots & \cdots \\
\end{pmatrix}.
\]

\( T_{k+1,k}, \tilde{T}_{k+1,k} \in \mathbb{R}^{k+1 \times k} \). The biorthogonality property implies
\[
V_k^T W_k = I. \tag{9.2}
\]
9.1 The Bi-CG Method

Remark 9.7. Idea, costs. The Bi-CG method constructs the iterate

\[ x^{(k)} = x^{(0)} + V_k y_k, \quad y_k \in \mathbb{R}^k, \]

such that \( r^{(k)} = r^{(0)} - AV_k y_k \) is orthogonal to \( K_k (r^{(0)}, A^T) \), i.e.,

\[ 0 = W_k^T r^{(k)} = W_k^T r^{(0)} - W_k^T AV_k y_k. \]  

Using (9.1) and (9.2), one has

\[ W_k^T AV_k = W_k^T V_{k+1} T_{k+1,k} = W_k^T \left[ V_k e^{(k+1)} \right] T_{k+1,k} = [I_k \, 0] T_{k+1,k} =: T_k \]

with

\[ T_k = \begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 & \alpha_2 & \beta_2 & \gamma_2 & \ldots & \alpha_k & \beta_k & \gamma_k \\ \alpha_1 & \beta_1 & \gamma_1 & \alpha_2 & \beta_2 & \gamma_2 & \ldots & \alpha_k & \beta_k & \gamma_k \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \alpha_1 & \beta_1 & \gamma_1 & \alpha_2 & \beta_2 & \gamma_2 & \ldots & \alpha_k & \beta_k & \gamma_k \end{pmatrix} \in \mathbb{R}^{k \times k} \]

and

\[ W_k^T r^{(0)} = \| r^{(0)} \|_2 e_1, \quad e_1 \in \mathbb{R}^k, \]

since \( r^{(1)} = r^{(0)}/\| r^{(0)} \|_2 \) by line 1 of Algorithm 9.3 and \( w^{(j)} \perp r^{(1)} \) for \( j > 1 \).

Thus, the computation of \( y_k \) from (9.3) requires the solution of the tridiagonal system

\[ T_k y_k = \| r^{(0)} \|_2 e_1. \]

The necessity of solving a system with a tridiagonal matrix arose already in the CG method. The iterate of the CG method is given in (6.2), where \( \tilde{H}_k \) is a tridiagonal matrix since \( A \) is a symmetric matrix, see Remark 5.11. Now, an algorithm for implementing the Bi-CG method can be derived similarly to the derivation of the algorithm for the CG method, see Remark 6.7.

Algorithm 9.8. Biconjugate Gradient (Bi-CG). Given a non-singular matrix \( A \in \mathbb{R}^{n \times n} \), a right-hand side \( b \in \mathbb{R}^n \), an initial iterate \( x^{(0)} \in \mathbb{R}^n \) and a tolerance \( \varepsilon > 0 \).

1. \( \tilde{r}^{(0)} = b - Ax^{(0)} \), choose \( \tilde{r}^{(0)} \) such that \( (\tilde{r}^{(0)})^T \tilde{r}^{(0)} \neq 0 \)
2. \( \tilde{p}_1 = \tilde{r}^{(0)} \), \( \tilde{p}_1 = \tilde{r}^{(0)} \)
3. \( \tilde{k} = 0 \)
9.2 The QMR Method

4. while $\|z^{(k)}\|_2 > \varepsilon$
5. $k = k + 1$
6. $z = Ap_k$
7. $\tilde{z} = A^T\tilde{p}_k$
8. $\nu_k = \frac{\left(\tilde{z}^{(k-1)}\right)^T r^{(k-1)}}{\tilde{p}_k^T z}$
9. $\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \nu_k \tilde{p}_k$
10. $r^{(k)} = r^{(k-1)} - \nu_k \tilde{x}$
11. $\tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \nu_k \tilde{r}$
12. $\mu_{k+1} = \frac{\left(\tilde{r}^{(k)}\right)^T r^{(k)}}{\left(\tilde{r}^{(k-1)}\right)^T \tilde{r}^{(k-1)}}$
13. $\tilde{p}_{k+1} = \tilde{r}^{(k)} + \mu_{k+1} \tilde{p}_k$
14. $\tilde{p}_{k+1} = \tilde{r}^{(k)} + \mu_{k+1} \tilde{p}_k$
15. endwhile

Remark 9.9. On the Bi-CG method. If $A$ is symmetric and positive definite, the Bi-CG method reduces to the CG method, as can be easily seen by comparing Algorithm 6.8 and Algorithm 9.8. It is possible to implement the Bi-CG method with a short recurrence (not only the Lanczos part but also the projection part into $K_k (z^{(0)}, A^T)$. However, Bi-CG is not that often used. One reason might be the appearance of $A^T$ in the algorithm.

9.2 The QMR Method

Remark 9.10. Idea. The goal of the QMR method (quasi minimal residual) is the construction of

$$z^{(k)} = z^{(0)} + V_k U_k, \quad U_k \in \mathbb{R}^k,$$

such that the second factor of the residual

$$r^{(k)} = z^{(0)} - AV_k U_k = z^{(0)} - V_{k+1} T_{k+1, k} U_k = V_{k+1} \left(\|z^{(0)}\|_2 z_1 - T_{k+1, k} z_k\right)$$

becomes minimal. In this derivation, (9.1) and Line 1 of Algorithm 9.3 were used. Since the columns of $V_{k+1}$ are in general not mutually orthogonal, the left-hand side $r^{(k)}$ does not become minimal in the Euclidean norm. An analogously to the GMRES method, the solution of the least squares problem
\[
\min_{z \in \mathbb{R}^k} \left\| z^{(0)} \right\|_2 \leq T_{k+1,k} y \leq \epsilon_1 \leq \epsilon_1 - T_{k+1,k} y
\]

is defined to be \( y \). This problem possesses always a unique solution as long as the Lanczos biorthogonalization procedure does not terminate. □

**Lemma 9.11. Estimate of the residual of the QMR method.** Denote by \( \gamma^{(k)} \) the residual of QMR, then

\[
\left\| \gamma^{(k)} \right\|_2 \leq \sqrt{k + 1} \left\| \epsilon^{(0)} \right\|_2 \leq T_{k+1,k} y \leq \epsilon_1 - T_{k+1,k} y
\]

Proof. By the definition of the residual (9.4) it follows that

\[
\left\| \gamma^{(k)} \right\|_2 \leq \left\| \epsilon^{(0)} \right\|_2 \leq \epsilon_1 - T_{k+1,k} y
\]

One obtains for the first factor by using the definition of the spectral norm, the triangle inequality and the Cauchy–Schwarz inequality

\[
\left\| \epsilon^{(0)} \right\|_2 = \sup_{\| z \|_2 = 1} \left\| V_{k+1} z \right\|_2 \leq \sup_{\| z \|_2 = 1} \left\| \sum_{j=1}^{k+1} z_j z^{(j)} \right\|_2
\]

\[
\leq \sup_{\| z \|_2 = 1} \left\| \sum_{j=1}^{k+1} z_j \right\|_2 \left\| \sum_{j=1}^{k+1} z^{(j)} \right\|_2 \leq \left( \sum_{j=1}^{k+1} |z_j|^2 \right)^{1/2} \left( \sum_{j=1}^{k+1} \left\| z^{(j)} \right\|_2 \right)^{1/2} = (k + 1)^{1/2}.
\]

**Theorem 9.12. Relation between the residuals of the QMR method and of GMRES.** Denote by \( \gamma^{(k)} \) the residual of QMR and by \( \gamma^{(k)} \) the residual of GMRES in the \( k \)-th iteration, then it holds

\[
\left\| \gamma^{(k)} \right\|_2 \leq \frac{\lambda_{\max} (V_{k+1}^T V_{k+1})}{\lambda_{\min} (V_{k+1}^T V_{k+1})} \left\| \gamma^{(k)} \right\|_2.
\]

Proof. Using (2.4) and (9.4), one has for all \( y \in \mathbb{R}^k \)

\[
\left\| \gamma^{(k)} \right\|_2 = \left( \left\| \epsilon^{(0)} \right\|_2 \leq T_{k+1,k} y \leq \epsilon_1 - T_{k+1,k} y \right)^T \left( \left\| \epsilon^{(0)} \right\|_2 \leq T_{k+1,k} y \leq \epsilon_1 - T_{k+1,k} y \right)
\]

\[
\geq \lambda_{\min} (V_{k+1}^T V_{k+1}) \left\| \epsilon^{(0)} \right\|_2 \leq T_{k+1,k} y \leq \epsilon_1 - T_{k+1,k} y \geq \lambda_{\min} (V_{k+1}^T V_{k+1}) \left\| \epsilon^{(0)} \right\|_2 \leq T_{k+1,k} y \leq \epsilon_1 - T_{k+1,k} y
\]

---

1. Augustin Louis Cauchy (1789 – 1857)
2. Hermann Amandus Schwarz (1843 – 1921)
since \( y_k \) minimizes the second factor. Because GMRES minimizes the whole residual in (9.4) and this estimate holds for all \( y \in \mathbb{R}^k \), one obtains in particular that

\[
\|r^{(k)}_{\text{GMRES}}\|_2^2 \geq \lambda_{\text{min}} \left( V_{k+1}^T V_{k+1} \right) \left\| x^{(0)} \right\|_2^2 - T_{k+1,k} y_k \|_2^2.
\]  

(9.5)

On the other hand, it is, using the definition of the spectral norm,

\[
\|r^{(k)}_{\text{QMR}}\|_2^2 \leq \|V_{k+1}\|_2^2 \left\| x^{(0)} \right\|_2^2 - T_{k+1,k} y_k \|_2^2 = \lambda_{\text{max}} \left( V_{k+1}^T V_{k+1} \right) \left\| x^{(0)} \right\|_2^2 - T_{k+1,k} y_k \|_2^2
\]

such that

\[
\|x^{(0)}\|_2^2 - T_{k+1,k} y_k \|_2^2 \geq \frac{\|r^{(k)}_{\text{QMR}}\|_2^2}{\lambda_{\text{max}} \left( V_{k+1}^T V_{k+1} \right)}.
\]

Inserting this estimate into (9.5) finishes the proof.

\[\blacksquare\]

Remark 9.13. On the QMR method. The implementation of the QMR method is analogously to the implementation of MINRES. If \( A \) is symmetric, QMR reduces to MINRES.

\[\blacksquare\]

9.3 Transposed-Free Methods

Remark 9.14. Motivation. As already explained in Remark 9.4, the transposed matrix \( A^T \) is in a number of situations not available. Then, algorithms that require the multiplication with \( A^T \) cannot be used.

\[\blacksquare\]

Remark 9.15. Idea of the Conjugate Gradient Squared (CGS) method. The CGS method can be derived from the Bi-CG method, see Algorithm 9.8. Sucessive insertion shows that the residual vector can be expressed as

\[
r^{(k)} = \phi_k(A) x^{(0)} , \quad k \geq 0, \tag{9.6}
\]

where \( \phi_k(A) \) is a polynomial of degree \( k \) with \( \phi_k(0) = 1 \), compare (7.1). Similarly, it is

\[
p^{(k)} = \psi_{k+1}(A) x^{(0)} , \quad k \geq 1, \tag{9.7}
\]

where \( \psi_{k+1}(A) \) is a polynomial of degree \( (k - 1) \) with \( \psi_{k+1}(0) = 1 \). From Algorithm 9.8, it can be observed that the vectors \( \tilde{r}^{(k)} \) and \( \tilde{p}^{(k)} \) are defined in the same way with the same polynomials where \( A \) is replaced by \( A^T \)

\[
\tilde{r}^{(k)} = \phi_k(A^T) \tilde{r}^{(0)}; \quad \tilde{p}^{(k)} = \psi_{k+1}(A^T) \tilde{r}^{(0)}.
\]

(9.8)

Inserting (9.6) – (9.8) in the definition of \( \nu_k \) in line 8 of Algorithm 9.8 gives
\[ \nu_k = \frac{(\phi_{k-1}(A^T)\tilde{z}^{(0)})^T \phi_{k-1}(A)z^{(0)}}{(\psi_{k-1}(A^T)\tilde{z}^{(0)})^T A\psi_{k-1}(A)z^{(0)}} = \frac{(\tilde{z}^{(0)})^T \phi_{k-1}^2(A)z^{(0)}}{(\tilde{z}^{(0)})^T A\psi_{k-1}^2(A)z^{(0)}}. \] (9.9)

Similarly, one obtains for the parameter in line 12
\[ \mu_{k+1} = \frac{(\phi_k(A^T)\tilde{z}^{(0)})^T \phi_k(A)z^{(0)}}{(\phi_{k-1}(A^T)\tilde{z}^{(0)})^T \phi_{k-1}(A)z^{(0)}} = \frac{(\tilde{z}^{(0)})^T \phi_k^2(A)z^{(0)}}{(\tilde{z}^{(0)})^T A\psi_{k-1}^2(A)z^{(0)}}. \] (9.10)

Thus, if it would be possible to derive recursions for the vectors \( \phi_k^2(A)z^{(0)} \) and \( \psi_{k-1}^2(A)z^{(0)} \), then the computation of \( \nu_k \) and \( \mu_{k+1} \) would be possible without using the transposed of \( A \).

The goal of CGS consists now, besides the formulation of the recursions, to compute iterates whose residual satisfy
\[ r^{(k)} = \phi_k^2(A)z^{(0)} \] instead of (9.6).

To derive these recursions, one starts with a recursion for the polynomials. Inserting (9.6) and (9.7) in line 10 of Algorithm 9.8, one obtains
\[ \phi_k(A)z^{(0)} = \phi_{k-1}(A)z^{(0)} - \nu_k A\psi_{k-1}(A)z^{(0)}, \]

hence the recursion for the polynomial is
\[ \phi_k(t) = \phi_{k-1}(t) - \nu_k t\psi_{k-1}(t). \] (9.12)

Similarly, one obtains from line 13
\[ \psi_k(t) = \phi_k(t) + \mu_{k+1}\psi_{k-1}(t). \] (9.13)

Taking the square of the polynomials gives
\[ \phi_k^2(t) = \phi_{k-1}^2(t) - 2\nu_k t\phi_{k-1}(t)\psi_{k-1}(t) + \nu_k^2 t^2\psi_{k-1}^2(t), \]
\[ \psi_k^2(t) = \phi_k^2(t) + 2\mu_{k+1}\phi_k(t)\psi_{k-1}(t) + \mu_{k+1}^2\psi_{k-1}^2(t). \]

The difficulty for deriving a recursion comes from the cross terms. The idea to overcome this difficulty consists in constructing a recursion for one of the cross terms, too. It is with (9.13)
\[ \phi_{k-1}(t)\psi_{k-1}(t) = \phi_{k-1}(t)(\phi_{k-1}(t) + \mu_k\psi_{k-2}(t)) = \phi_{k-1}^2(t) + \mu_k\phi_{k-1}(t)\psi_{k-2}(t). \]

Collecting all equations and (9.12), one obtains the following relations (where for clarity of presentation the dependency on \( t \) will be neglected)
\[
\phi_k^2 = \phi_{k-1}^2 - \nu_k t \left(2\phi_{k-1}^2 + 2\mu_k \phi_{k-1} \psi_{k-1} - \nu_k t \psi_{k-1}^2\right), \tag{9.14}
\]
\[
\phi_k \psi_{k-1} = (\phi_{k-1} - \nu_k t \psi_{k-1}) \psi_{k-1} = \phi_{k-1} \psi_{k-1} - \nu_k t \psi_{k-1}^2,
\]
\[
\psi_k^2 = \phi_k^2 + 2\mu_{k+1} \phi_k \psi_{k-1} + \mu_k^2 \psi_{k-1}^2. \tag{9.16}
\]

Note that in (9.14) one can use the known quantities \(\phi_{k-1}^2\) and \(\psi_{k-1}^2\) for computing \(\nu_k\), see (9.9), and \(\phi_{k-1}^2\) and \(\phi_{k-1}^2\) for computing \(\mu_k\), see (9.10). Thus, the recursion (9.14) and (9.15) are well defined. After having computed \(\phi_k^2\), one can compute \(\mu_{k+1}\), which is needed in (9.16). Altogether, (9.14) – (9.16) is a recursion for the quantities \(\phi_k^2\), \(\psi_k^2\), and \(\phi_k \psi_{k-1}\).

An efficient implementation of the recursion leads to the CGS method. This method was developed in Sonneveld (1989). One step of this method requires two matrix-vector products with the matrix \(A\) but no product with \(A^T\). CGS works well in many cases (Saad, 2003, Section 7.4.1), however rounding errors might become important since the residual polynomial is squared. An irregular convergence might occur that can even lead to an overflow.

Remark 9.16. Idea of the Bi-Conjugate Gradient Stabilized (BiCGStab) method. The BiCGStab method is a generalization of the CGS method that was developed in van der Vorst (1992) to stabilize the CGS method. Instead of computing a residual vector of the form (9.11), the residual vector computed with BiCGStab should fulfill
\[
\xi^{(k)} = \pi_k(A) \phi_k(A) \xi^{(0)},
\]
where \(\phi_k(t)\) is the polynomial associated with the Bi-CG algorithm and \(\pi_k(t)\) is a new polynomial of degree \(k\). The choice of \(\pi_k(A)\) should contribute to the stabilization of the algorithm.

In the BiCGStab method, the polynomial \(\pi_k(t)\) is defined by the recurrence
\[
\pi_{k+1}(t) = (1 - \omega_k t) \pi_k(t),
\]
where \(\omega_k \in \mathbb{R}\) has to be yet determined. The choice of this polynomial leads in fact to a more stable algorithm than CGS. The parameter \(\omega_k\) is chosen in such a way that the norm of a certain residual that occurs in the algorithm is minimized, for more details see van der Vorst (1992) and (Saad, 2003, Section 7.4.2).

Algorithm 9.17. Bi-Conjugate Gradient Stabilized (BiCGStab).
Given a non-singular matrix \(A \in \mathbb{R}^{n \times n}\), a right-hand side \(b \in \mathbb{R}^n\), an initial iterate \(\xi^{(0)} \in \mathbb{R}^n\) and a tolerance \(\varepsilon > 0\).

1. \(\xi^{(0)} = b - A \xi^{(0)}\), choose \(\xi^{(0)}\) arbitrarily such that \(\xi^{(0)} \neq 0\)
2. \(\xi_{k+1} = \xi^{(0)}\)
3. \(k = 0\)
4. while $\|z^{(k)}\|_2 > \varepsilon$
5. $k = k + 1$
6. $z = Ap_k$
7. $\nu_k = \left( \tilde{r}^{(0)} \right)^T z^{(k-1)}$
8. $w = z^{(k-1)} - \nu_k z$
9. $\tilde{z} = Aw$
10. $\omega_k = \tilde{z}^T w$
11. $p^{(k)} = z^{(k-1)} + \nu_k p_k + \omega_k w$
12. $r^{(k)} = w - \omega_k \tilde{z}$
13. $\mu_{k+1} = \frac{\left( \tilde{r}^{(0)} \right)^T r^{(k)} \nu_k}{\left( \tilde{r}^{(0)} \right)^T r^{(k-1)} \omega_k}$
14. $p_{k+1} = r^{(k)} + \mu_{k+1} \left( p_k - \omega_k \tilde{z} \right)$
15. endwhile

- In van der Vorst (1992) it is proposed to use $\tilde{r}^{(0)} = r^{(0)}$.
- In exact arithmetics, BiCGStab terminates in at most $n$ iterations.
- There is the possibility that BiCGStab terminates in finite precision without having computed the solution, e.g., if $\left( \tilde{r}^{(0)} \right)^T r^{(k-1)}$ is (close to) zero. Then, one can try to use a different vector $\tilde{r}^{(0)}$ or one can switch to another method like GMRES.
- There are other variants of BiCGStab than given in Algorithm 9.17 which are equivalent in exact arithmetics but not in finite precision computation, see van der Vorst (1992). For instance, in MATLAB there is different variant implemented. Different variants might behave differently.
- In Algorithm 9.17, one should compute and store $\left( \tilde{r}^{(0)} \right)^T r^{(k-1)}$ before line 7 to be used again in line 13. Likewise, one should compute and store $\left( \tilde{r}^{(0)} \right)^T r^{(k)}$ before line 13 to be used in the next iteration at line 7.
- BiCGStab is besides GMRES the most popular iterative scheme for solving large linear systems of equations with non-symmetric matrices.

Remark 9.19. Transposed-free QMR. There is also a transposed-free version of the QMR algorithm, see (Saad, 2003, Section 7.4.3). However, this algorithm is not popular.