Chapter 9
Other Krylov Subspace Methods for Non-Symmetric Systems

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, since there is no short recurrence. 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, not longer. □

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

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

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. $\underline{v}^{(1)} = \underline{r}^{(0)}/\left\|\underline{r}^{(0)}\right\|_2$
2. $\underline{w}^{(1)} = \underline{v}^{(1)}$
3. $\beta_0 = 0$, $\gamma_0 = 0$
4. $\underline{v}^{(0)} := \underline{0}$, $\underline{w}^{(0)} = \underline{0}$
5. for $j = 1 : k$
```

```
6.   $\underline{s} = A\underline{v}^{(j)}$
7.   $\underline{z} = A^T\underline{w}^{(j)}$
8.   $\alpha_j = (\underline{w}^{(j)}, \underline{s})$
9.   $\underline{\tilde{v}}^{(j+1)} = \underline{s} - \alpha_j\underline{v}^{(j)} - \beta_{j-1}\underline{v}^{(j-1)}$
10.  $\underline{\tilde{w}}^{(j+1)} = \underline{z} - \alpha_j\underline{w}^{(j)} - \gamma_{j-1}\underline{w}^{(j-1)}$
11.  $\gamma_j = \left\|\underline{\tilde{v}}^{(j+1)}\right\|_2$
12.  $\underline{v}^{(j+1)} = \underline{\tilde{v}}^{(j+1)}/\gamma_j$
13.  $\beta_j = \left(\underline{\tilde{w}}^{(j+1)}, \underline{v}^{(j+1)}\right)$
14.  $\underline{w}^{(j+1)} = \underline{\tilde{w}}^{(j+1)}/\beta_j$
15. endfor
```

□

Remark 9.4. On the Lanczos biorthogonalization procedure.

- There is a short recurrence in Algorithm 9.3.
- Note that the basis of $K_k\left(\underline{r}^{(0)}, A\right)$ will be in general not orthogonal as well as the basis of $K_k\left(\underline{r}^{(0)}, A^T\right)$. 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. 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 $\left(\underline{\tilde{w}}^{(j)}, \underline{v}^{(j)}\right) \neq 0$ for all $j = 1, \cdots, k$. 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 $\underline{w}^{(1)} = \underline{v}^{(1)}$, line 2 and $\left\|\underline{v}^{(1)}\right\|_2 = 1$, 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 $\left(\underline{\tilde{w}}^{(j)}, \underline{v}^{(i)}\right) = \left(\underline{w}^{(j)}, \underline{v}^{(i)}\right) = 0$ for $i \neq j$, $1 \leq i, j \leq k-1$ and $\left(\underline{w}^{(i)}, \underline{v}^{(i)}\right) = 1$, $1 \leq i \leq k-1$. The choice of γ_{k-1} implies $\left\|\underline{v}^{(k)}\right\|_2 = 1$ and line 14 and the choice of β_{k-1} give

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

Using lines 12 and 9, the assumption of the induction, and the definition of α_{k-1} leads to

$$
\begin{aligned}
&\left(\underline{w}^{(k-1)}, \underline{v}^{(k)}\right) = \left(\underline{w}^{(k-1)}, \frac{\underline{\tilde{v}}^{(k)}}{\gamma_{k-1}}\right) \\
&= \frac{1}{\gamma_{k-1}} \left[\left(\underline{w}^{(k-1)}, A\underline{v}^{(k-1)}\right) - \alpha_{k-1} \underbrace{\left(\underline{w}^{(k-1)}, \underline{v}^{(k-1)}\right)}_{=1} - \beta_{k-2} \underbrace{\left(\underline{w}^{(k-1)}, \underline{v}^{(k-2)}\right)}_{=0} \right] = 0.
\end{aligned}
$$

One obtains analogously $\left(\underline{w}^{(k)}, \underline{v}^{(k-1)}\right) = 0$. Moreover, using the lines 12, 9, the assumption of the induction, line 10 (with $\underline{z} = A^T \underline{w}^{(k-2)}$), and the definition of β_{k-2} gives

$$
\begin{aligned}
&\left(\underline{w}^{(k-2)}, \underline{v}^{(k)}\right) \\
&= \frac{1}{\gamma_{k-1}} \left[\left(\underline{w}^{(k-2)}, A\underline{v}^{(k-1)}\right) - \alpha_{k-1} \underbrace{\left(\underline{w}^{(k-2)}, \underline{v}^{(k-1)}\right)}_{=0} - \beta_{k-2} \underbrace{\left(\underline{w}^{(k-2)}, \underline{v}^{(k-2)}\right)}_{=1} \right] \\
&= \frac{1}{\gamma_{k-1}} \left[\left(\underline{w}^{(k-2)}, A\underline{v}^{(k-1)}\right) - \beta_{k-2} \right] \\
&= \frac{1}{\gamma_{k-1}} \left[\left(A^T\underline{w}^{(k-2)}, \underline{v}^{(k-1)}\right) - \beta_{k-2} \right] \\
&= \frac{1}{\gamma_{k-1}} \left[\left(\underline{\tilde{w}}^{(k-1)}, \underline{v}^{(k-1)}\right) + \alpha_{k-2} \underbrace{\left(\underline{\tilde{w}}^{(k-2)}, \underline{v}^{(k-1)}\right)}_{=0} + \gamma_{k-3} \underbrace{\left(\underline{\tilde{w}}^{(k-3)}, \underline{v}^{(k-1)}\right)}_{=0} - \beta_{k-2} \right] \\
&= 0.
\end{aligned}
$$

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

Remark 9.6. Matrix representation of the Lanczos biorthogonalization procedure. For the matrix representation of the Lanczos biorthogonalization procedure, the matrices

$$
V_k = \left(\underline{v}^{(1)} \ldots \underline{v}^{(k)}\right), \ W_k = \left(\underline{w}^{(1)} \ldots \underline{w}^{(k)}\right) \in \mathbb{R}^{n \times k}
$$

are introduced. From Algorithm 9.3, it follows that

$$
AV_k = V_{k+1} T_{k+1,k}, \quad A^T W_k = W_{k+1} \hat{T}_{k+1,k} \tag{9.1}
$$

with

$$
T_{k+1,k} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & & & \\ & \ddots & \ddots & \ddots & \\ & & & & \beta_{k-1} \\ & & & & \alpha_k \\ & & & & \gamma_k \end{pmatrix}, \ \hat{T}_{k+1,k} = \begin{pmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_1 & \alpha_2 & & & \\ & \ddots & \ddots & \ddots & \\ & & & & \gamma_{k-1} \\ & & & & \alpha_k \\ & & & & \beta_k \end{pmatrix},
$$

$T_{k+1,k}, \hat{T}_{k+1,k} \in \mathbb{R}^{(k+1)\times k}$. 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

$$\underline{x}^{(k)} = \underline{x}^{(0)} + V_k \underline{y}_k, \quad \underline{y}_k \in \mathbb{R}^k,$$

so that $\underline{r}^{(k)} = \underline{r}^{(0)} - AV_k\underline{y}_k$ is orthogonal to $K_k\left(\underline{r}^{(0)}, A^T\right)$, i.e.,

$$0 = W_k^T \underline{r}^{(k)} = W_k^T \underline{r}^{(0)} - W_k^T A V_k \underline{y}_k. \tag{9.3}$$

Using (9.1) and (9.2), one has

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

with

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & & & \\ & \ddots & \ddots & \ddots & \\ & & & \alpha_{k-1} & \beta_{k-1} \\ & & & \gamma_{k-1} & \alpha_k \end{pmatrix} \in \mathbb{R}^{k\times k}$$

and

$$W_k^T \underline{r}^{(0)} = \left\|\underline{r}^{(0)}\right\|_2 \underline{e}_1, \ \underline{e}_1 \in \mathbb{R}^k,$$

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

Thus, the computation of $\underline{y}_k$ from (9.3) requires the solution of the tridiagonal system

$$T_k \underline{y}_k = \left\|\underline{r}^{(0)}\right\|_2 \underline{e}_1.$$

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 $\underline{b} \in \mathbb{R}^n$, an initial iterate $\underline{x}^{(0)} \in \mathbb{R}^n$ and a tolerance $\varepsilon > 0$.

```
 1. $\underline{r}^{(0)} = \underline{b} - A\underline{x}^{(0)}$, choose $\underline{\tilde{r}}^{(0)}$ such that $\left(\underline{\tilde{r}}^{(0)}\right)^T \underline{r}^{(0)} \neq 0$
 2. $\underline{p}_1 = \underline{r}^{(0)}$, $\underline{\tilde{p}}_1 = \underline{\tilde{r}}^{(0)}$
 3. $k = 0$
 4. while $\left\|\underline{r}^{(k)}\right\|_2 > \varepsilon$
 5.     $k = k + 1$
 6.     $\underline{s} = A\underline{p}_k$
 7.     $\underline{z} = A^T\underline{\tilde{p}}_k$
 8.     $\nu_k = \dfrac{\left(\underline{\tilde{r}}^{(k-1)}\right)^T \underline{r}^{(k-1)}}{\underline{\tilde{p}}_k^T \underline{s}}$
 9.     $\underline{x}^{(k)} = \underline{x}^{(k-1)} + \nu_k \underline{p}_k$
10.     $\underline{r}^{(k)} = \underline{r}^{(k-1)} - \nu_k \underline{s}$
11.     $\underline{\tilde{r}}^{(k)} = \underline{\tilde{r}}^{(k-1)} - \nu_k \underline{z}$
12.     $\mu_{k+1} = \dfrac{\left(\underline{\tilde{r}}^{(k)}\right)^T \underline{r}^{(k)}}{\left(\underline{\tilde{r}}^{(k-1)}\right)^T \underline{r}^{(k-1)}}$
13.     $\underline{p}_{k+1} = \underline{r}^{(k)} + \mu_{k+1}\underline{p}_k$
14.     $\underline{\tilde{p}}_{k+1} = \underline{\tilde{r}}^{(k)} + \mu_{k+1}\underline{\tilde{p}}_k$
15. endwhile
```

□

Remark 9.9. On the Bi-CG method. If A is symmetric and positive definit and $\underline{r}^{(0)} = \underline{\tilde{r}}^{(0)}$, the Bi-CG method reduces to the CG method, as can be easily seen by comparing Algorithm 6.8 and Algorithm 9.8. As can be seen in Algorithm 9.8, the Bi-CG method can be implemented with a short recurrence (not only the Lanczos part but also the projection part into $K_k(\underline{r}^{(0)}, A^T)$).

If not only $A\underline{x} = \underline{b}$ should be solved but also $A^T\underline{\tilde{x}} = \underline{\tilde{b}}$ for given $\underline{\tilde{b}} \in \mathbb{R}^n$, then one should choose $\underline{\tilde{r}}^{(0)} = \underline{\tilde{b}} - A^T\underline{\tilde{x}}^{(0)}$ for some initial iterate $\underline{\tilde{x}}^{(0)}$ and an appropriate update for the solution of the system with the transposed matrix has to be inserted in the algorithm, see Saad (2003). Otherwise, $\underline{\tilde{r}}^{(0)} = \underline{r}^{(0)}$ is a common choice.

Altogether, 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

$$\underline{x}^{(k)} = \underline{x}^{(0)} + V_k \underline{y}_k, \quad \underline{y}_k \in \mathbb{R}^k,$$

such that the second factor of the residual

$$\underline{r}^{(k)} = \underline{r}^{(0)} - AV_k\underline{y}_k = \underline{r}^{(0)} - V_{k+1}T_{k+1,k}\underline{y}_k = V_{k+1}\left(\left\|\underline{r}^{(0)}\right\|_2 \underline{e}_1 - T_{k+1,k}\underline{y}_k\right) \tag{9.4}$$

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 $\underline{r}^{(k)}$ does not become minimal in the Euclidean norm. Analogously to the GMRES method, the solution of the least squares problem

$$\min_{\underline{y}\in\mathbb{R}^k} \left\| \left\|\underline{r}^{(0)}\right\|_2 \underline{e}_1 - T_{k+1,k}\underline{y}\right\|_2$$

is defined to be $\underline{y}_k$. This problem possesses always a unique solution as long as the Lanczos biorthogonalization procedure does not terminate. The solution can be performed with a short recurrence since $T_{k+1,k}$ is tridiagonal, compare the discussion for MINRES in Remark 5.15. □

Lemma 9.11. Upper bound for the residual of the QMR method. *Denote by $\underline{r}^{(k)}_{\mathrm{QMR}}$ the residual of QMR, then*

$$\left\|\underline{r}^{(k)}_{\mathrm{QMR}}\right\|_2 \le \sqrt{k+1}\left\| \left\|\underline{r}^{(0)}\right\|_2 \underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2.$$

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

$$\left\|\underline{r}^{(k)}_{\mathrm{QMR}}\right\|_2 \le \left\|V_{k+1}\right\|_2 \left\| \left\|\underline{r}^{(0)}\right\|_2 \underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2.$$

One obtains for the first factor by using the definition of the spectral norm, the triangle inequality, the Cauchy[1]–Schwarz[2] inequality, and line 12 of Algorithm 9.3

$$\begin{aligned}\left\|V_{k+1}\right\|_2 &= \sup_{\underline{z}\in\mathbb{R}^{k+1},\|\underline{z}\|_2=1} \left\|V_{k+1}\underline{z}\right\|_2 = \sup_{\underline{z}\in\mathbb{R}^{k+1},\|\underline{z}\|_2=1} \left\|\sum_{j=1}^{k+1} z_j \underline{v}^{(j)}\right\|_2 \\ &\le \sup_{\underline{z}\in\mathbb{R}^{k+1},\|\underline{z}\|_2=1} \sum_{j=1}^{k+1} |z_j| \left\|\underline{v}^{(j)}\right\|_2\end{aligned}$$

[1] Augustin Louis Cauchy (1789 – 1857)

[2] Hermann Amandus Schwarz (1843 – 1921)

$$\leq \sup_{\underline{z}\in\mathbb{R}^{k+1},\|\underline{z}\|_2=1} \underbrace{\left(\sum_{j=1}^{k+1}|z_j|^2\right)^{1/2}}_{=1} \left(\sum_{j=1}^{k+1}\underbrace{\left\|\underline{v}^{(j)}\right\|_2^2}_{=1}\right)^{1/2} = (k+1)^{1/2}.$$

■

Theorem 9.12. Relation between the residual of the QMR method and the minimal residual. *Denote by* $\underline{r}_{\text{QMR}}^{(k)}$ *the residual of QMR in the k-th iteration and by* $\underline{r}_{\min}^{(k)}$ *the residual that is obtained by minimizing the whole residual, then it holds*

$$\left\|\underline{r}_{\text{QMR}}^{(k)}\right\|_2 \leq \frac{\lambda_{\max}\left(V_{k+1}^T V_{k+1}\right)}{\lambda_{\min}\left(V_{k+1}^T V_{k+1}\right)}\left\|\underline{r}_{\min}^{(k)}\right\|_2.$$

Proof. Using (2.4), after having multiplied the expression appropriately, and (9.4), one has for all $\underline{y}\in\mathbb{R}^k$

$$\begin{aligned}\left\|\underline{r}^{(k)}\right\|_2^2 &= \left(\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}\right)^T \underbrace{V_{k+1}^T V_{k+1}}_{s.p.d.}\left(\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}\right)\\ &\geq \lambda_{\min}\left(V_{k+1}^T V_{k+1}\right)\left\|\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}\right\|_2^2\\ &\geq \lambda_{\min}\left(V_{k+1}^T V_{k+1}\right)\left\|\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2^2,\end{aligned}$$

since $\underline{y}_k$ minimizes the second factor. Because this estimate holds for all $\underline{y}\in\mathbb{R}^k$, one obtains in particular for the vector which leads to the minimal residual that

$$\left\|\underline{r}_{\min}^{(k)}\right\|_2^2 \geq \lambda_{\min}\left(V_{k+1}^T V_{k+1}\right)\left\|\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2^2. \tag{9.5}$$

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

$$\begin{aligned}\left\|\underline{r}_{\text{QMR}}^{(k)}\right\|_2^2 &\leq \|V_{k+1}\|_2^2\left\|\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2^2\\ &= \lambda_{\max}\left(V_{k+1}^T V_{k+1}\right)\left\|\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2^2\end{aligned}$$

so that

$$\left\|\left\|\underline{r}^{(0)}\right\|_2\underline{e}_1 - T_{k+1,k}\underline{y}_k\right\|_2^2 \geq \frac{\left\|\underline{r}_{\text{QMR}}^{(k)}\right\|_2^2}{\lambda_{\max}\left(V_{k+1}^T V_{k+1}\right)}.$$

Inserting this estimate in (9.5) finishes the proof. ■

Remark 9.13. On the QMR method. The implementation of the QMR method is analogous to the implementation of MINRES. If A is symmetric, QMR reduces to MINRES. □

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. $\square$

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

$$\underline{r}^{(k)} = \phi_k(A)\underline{r}^{(0)}, \quad k \geq 0, \tag{9.6}$$

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

$$\underline{p}_k = \psi_{k-1}(A)\underline{r}^{(0)}, \quad k \geq 1, \tag{9.7}$$

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 $\tilde{\underline{r}}^{(k)}$ and $\tilde{\underline{p}}_k$ are defined in the same way with the same polynomials where A is replaced by A^T

$$\tilde{\underline{r}}^{(k)} = \phi_k\left(A^T\right)\tilde{\underline{r}}^{(0)}, \quad \tilde{\underline{p}}_k = \psi_{k-1}\left(A^T\right)\tilde{\underline{r}}^{(0)}. \tag{9.8}$$

Inserting (9.6) – (9.8) in the definition of ν_k in line 8 of Algorithm 9.8 gives

$$\nu_k = \frac{\left(\phi_{k-1}\left(A^T\right)\tilde{\underline{r}}^{(0)}\right)^T \phi_{k-1}(A)\underline{r}^{(0)}}{\left(\psi_{k-1}\left(A^T\right)\tilde{\underline{r}}^{(0)}\right)^T A\psi_{k-1}(A)\underline{r}^{(0)}} = \frac{\left(\tilde{\underline{r}}^{(0)}\right)^T \phi_{k-1}^2(A)\underline{r}^{(0)}}{\left(\tilde{\underline{r}}^{(0)}\right)^T A\psi_{k-1}^2(A)\underline{r}^{(0)}}. \tag{9.9}$$

Similarly, one obtains for the parameter in line 12

$$\mu_{k+1} = \frac{\left(\phi_k\left(A^T\right)\tilde{\underline{r}}^{(0)}\right)^T \phi_k(A)\underline{r}^{(0)}}{\left(\phi_{k-1}\left(A^T\right)\tilde{\underline{r}}^{(0)}\right)^T \phi_{k-1}(A)\underline{r}^{(0)}} = \frac{\left(\tilde{\underline{r}}^{(0)}\right)^T \phi_k^2(A)\underline{r}^{(0)}}{\left(\tilde{\underline{r}}^{(0)}\right)^T \phi_{k-1}^2(A)\underline{r}^{(0)}}. \tag{9.10}$$

Thus, if it would be possible to derive recursions for the vectors $\phi_k^2(A)\underline{r}^{(0)}$ and $\psi_{k-1}^2(A)\underline{r}^{(0)}$, then the computation of ν_k and μ_{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

$$\underline{r}^{(k)} = \phi_k^2(A)\underline{r}^{(0)}, \tag{9.11}$$

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

$$\phi_k(A)\underline{r}^{(0)} = \phi_{k-1}(A)\underline{r}^{(0)} - \nu_k A\psi_{k-1}(A)\underline{r}^{(0)},$$

hence the recursion for the polynomial is

$$\phi_k(t) = \phi_{k-1}(t) - \nu_k t\psi_{k-1}(t). \tag{9.12}$$

Similarly, one obtains from line 13

$$\psi_k(t) = \phi_k(t) + \mu_{k+1}\psi_{k-1}(t). \tag{9.13}$$

Taking the square of the polynomials gives

$$\begin{aligned}\phi_k^2(t) &= \phi_{k-1}^2(t) - 2\nu_k t\phi_{k-1}(t)\psi_{k-1}(t) + \nu_k^2 t^2\psi_{k-1}^2(t),\\ \psi_k^2(t) &= \phi_k^2(t) + 2\mu_{k+1}\phi_k(t)\psi_{k-1}(t) + \mu_{k+1}^2\psi_{k-1}^2(t).\end{aligned}$$

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)

$$\begin{aligned}\phi_{k-1}(t)\psi_{k-1}(t) &= \phi_{k-1}(t)\left(\phi_{k-1}(t) + \mu_k\psi_{k-2}(t)\right)\\ &= \phi_{k-1}^2(t) + \mu_k\phi_{k-1}(t)\psi_{k-2}(t).\end{aligned}$$

Collecting all equations, (9.12), and (9.13), one obtains the following relations (where for clarity of presentation the dependency on t will be neglected)

$$\phi_k^2 = \phi_{k-1}^2 - \nu_k t\left(2\phi_{k-1}^2 + 2\mu_k\phi_{k-1}\psi_{k-2} - \nu_k t\psi_{k-1}^2\right), \tag{9.14}$$

$$\begin{aligned}\phi_k\psi_{k-1} &= \left(\phi_{k-1} - \nu_k t\psi_{k-1}\right)\psi_{k-1} = \phi_{k-1}\psi_{k-1} - \nu_k t\psi_{k-1}^2\\ &= \phi_{k-1}^2 + \mu_k\phi_{k-1}\psi_{k-2} - \nu_k t\psi_{k-1}^2,\end{aligned} \tag{9.15}$$

$$\psi_k^2 = \phi_k^2 + 2\mu_{k+1}\phi_k\psi_{k-1} + \mu_{k+1}^2\psi_{k-1}^2. \tag{9.16}$$

Notice that in (9.14) one can use the known quantities ϕ_{k-1}^2 and ψ_{k-1}^2 for computing ν_k, see (9.9), and ϕ_{k-1}^2 and ϕ_{k-2}^2 for computing μ_k, see (9.10). Thus, the recursion (9.14) and (9.15) are well defined. After having computed ϕ_k^2, one can compute μ_{k+1}, which is needed in (9.16). Altogether, (9.14) – (9.16) is a recursion for the quantities ϕ_k^2, ψ_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

$$\underline{r}^{(k)} = \pi_k(A)\phi_k(A)\underline{r}^{(0)},$$

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

$$\pi_{k+1}(t) = (1 - \omega_k t)\,\pi_k(t), \quad \pi_0(t) = 1,$$

where $\omega_k \in \mathbb{R}$ has yet to be determined. The choice of this polynomial leads in fact to a more stable algorithm than CGS. The parameter ω_k is chosen in such a way that the norm of a certain residual vector 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 $\underline{b} \in \mathbb{R}^n$, an initial iterate $\underline{x}^{(0)} \in \mathbb{R}^n$ and a tolerance $\varepsilon > 0$.

```
1.  $\underline{r}^{(0)} = \underline{b} - A\underline{x}^{(0)}$, choose $\tilde{\underline{r}}^{(0)}$ arbitrarily such that
    $\left(\tilde{\underline{r}}^{(0)}\right)^T \underline{r}^{(0)} \neq 0$
2.  $\underline{p}_1 = \underline{r}^{(0)}$
3.  $k = 0$
4.  while $\left\|\underline{r}^{(k)}\right\|_2 > \varepsilon$
5.      $k = k + 1$
6.      $\underline{s} = A\underline{p}_k$
7.      $\nu_k = \dfrac{\left(\tilde{\underline{r}}^{(0)}\right)^T \underline{r}^{(k-1)}}{\left(\tilde{\underline{r}}^{(0)}\right)^T \underline{s}}$
8.      $\underline{w} = \underline{r}^{(k-1)} - \nu_k \underline{s}$
9.      $\underline{z} = A\underline{w}$
10.     $\omega_k = \dfrac{\underline{z}^T \underline{w}}{\underline{z}^T \underline{z}}$
11.     $\underline{x}^{(k)} = \underline{x}^{(k-1)} + \nu_k \underline{p}_k + \omega_k \underline{w}$
12.     $\underline{r}^{(k)} = \underline{w} - \omega_k \underline{z}$
13.     $\mu_{k+1} = \dfrac{\left(\tilde{\underline{r}}^{(0)}\right)^T \underline{r}^{(k)}}{\left(\tilde{\underline{r}}^{(0)}\right)^T \underline{r}^{(k-1)}} \dfrac{\nu_k}{\omega_k}$
14.     $\underline{p}_{k+1} = \underline{r}^{(k)} + \mu_{k+1}\left(\underline{p}_k - \omega_k \underline{s}\right)$
15. endwhile
```

□

Remark 9.18. BiCGStab.

- Each iteration requires two matrix-vector products with A.
- In van der Vorst (1992), it is proposed to use $\underline{\tilde{r}}^{(0)} = \underline{r}^{(0)}$.
- 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 $\left(\underline{\tilde{r}}^{(0)}\right)^T \underline{r}^{(k-1)}$ is (close to) zero. Then, one can try to use a different vector $\underline{\tilde{r}}^{(0)}$ or one can switch to another method like GMRES.
- There are other variants of BiCGStab than given in Algorithm 9.17 that 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.
- One should compute and store $\left(\underline{\tilde{r}}^{(0)}\right)^T \underline{r}^{(k)}$ before line 13 to be used in the next iteration at line 7 and at line 13.
- 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. □