Numerical Linear Algebra for Computational Science and Information Engineering

Krylov Subspace Methods for Linear Systems

Nicolas Venkovic
nicolas.venkovic@tum.de

Chair of Computational Mathematics
School of Computation, Information and Technology
Technical University of Munich

Outline I

Outline II

Projection methods for linear systems

General framework of projection methods for linear systems

- Let $\mathcal{K}_m$ be a proper m-dimensional subspace of $\mathbb{R}^n$, i.e., $\mathcal{K}_m \subset \mathbb{R}^n$, typically with $m \ll n$.
  We then seek for a $\tilde{x} \in \mathcal{K}_m$ which approximates the solution x of $Ax = b$.
  A typical way to form the approximation $\boxed{\tilde{x} \in \mathcal{K}_m}$ is to impose m **independent orthogonality conditions on the residual** $r := b - A\tilde{x}$ with respect to a m-dimensional **constraint subspace** $\mathcal{L}_m \subset \mathbb{R}^n$:

$$\boxed{r = b - A\tilde{x} \perp \mathcal{L}_m}. \tag{1}$$

  If $\mathcal{K}_m = \mathcal{L}_m$, then Eq. (1) is referred to as the **Galerkin condition**, and $\tilde{x}$ is formed by **orthogonal projection**.
  More generally, we have $\mathcal{L}_m \neq \mathcal{K}_m$, in which case Eq. (1) is referred to as the **Petrov-Galerkin condition**. Then, the process of forming $\tilde{x}$ is an **oblique projection**.
  A projection technique onto the approximation/search space $\mathcal{K}_m$ along the constraint subspace $\mathcal{L}_m$ is summarized as:

$$\boxed{\text{Find } \tilde{x} \in \mathcal{K}_m \text{ such that } b - A\tilde{x} \perp \mathcal{L}_m}.$$

General framework of projection methods for linear systems, cont'd

- The **projection techniques** presented in this lecture are **iterative**. That is, as a pair $(\mathcal{K}_m, \mathcal{L}_m) \subset \mathbb{R}^n \times \mathbb{R}^n$ of m-dimensional search and constraint subspaces is used to form an approximate solution $\tilde{x}$ of $Ax = b$, **the next iteration consists of expanding those subspaces**, leading to a pair $(\mathcal{K}_{m+1}, \mathcal{L}_{m+1})$ which is **then used to form a subsequent approximate solution**.
  A projection technique is deployed with an **initial iterate** $x_0 \in \mathbb{R}^n$. Subsequent iterates are then formed leveraging x_0 by searching in the **affine subspace** $x_0 + \mathcal{K}_m$. The projection technique is then summarized as

$$\boxed{\text{Find } \tilde{x} \in x_0 + \mathcal{K}_m \text{ such that } b - A\tilde{x} \perp \mathcal{L}_m}.$$

  If we write $\tilde{x} := x_0 + \hat{x}$ with $\hat{x} \in \mathcal{K}_m$, then the projection technique is reformulated as

$$\boxed{\text{Find } \hat{x} \in \mathcal{K}_m \text{ such that } r_0 - A\hat{x} \perp \mathcal{L}_m}$$

  where $r_0 := b - Ax_0$.

Matrix form of projection techniques for linear systems

- Let the columns of $V_m := [v_1, \ldots, v_m]$ and $W_m := [w_1, \ldots, w_m]$ form **bases of the search and constraints spaces**, respectively, i.e.,

$$\text{range}(V_m) = \mathcal{K}_m \text{ and } \text{range}(W_m) = \mathcal{L}_m.$$

Once equipped with such bases, one can recast the projection defined as finding $\tilde{x} \in x_0 + \mathcal{K}_m$ such that $b - A\tilde{x} \perp \mathcal{L}_m$ into

$$\text{Find } \tilde{y} \in \mathbb{R}^m \text{ such that } \tilde{x} := x_0 + V_m\tilde{y} \text{ and } b - A\tilde{x} \perp \text{range}(W_m).$$

Taking the dot product as inner product, this leads up to the following matrix form:

$$\text{Find } \tilde{y} \in \mathbb{R}^m \text{ such that } \tilde{x} := x_0 + V_m\tilde{y} \text{ and } W_m^T(r_0 - AV_m\tilde{y}) = 0.$$

If $W_m^T A V_m$ is not singular, we then have

$$\tilde{y} = (W_m^T A V_m)^{-1} W_m^T r_0$$

so that

$$\boxed{\tilde{x} = x_0 + V_m(W_m^T A V_m)^{-1} W_m^T r_0}.$$

Matrix form of projection techniques for linear systems, cont'd$_1$

- A proper projection technique to approximate the solution of a linear system in $x_0 + \text{range}(V_m)$ along $\text{range}(W_m)$ requires that $W_m^T A V_m$ is not singular.

  It can be shown that $W_m^T A V_m$ is **not singular** if and only if **no vector of the subspace** $A\mathcal{K}$ **is orthogonal to the constraints subspace** $\mathcal{L}_m$, i.e., $A\mathcal{K}_m \cap \mathcal{L}_m^\perp = \{0\}$.

  Saad (2003) states the following theorem:

Theorem (Non-singularity of $W_m^T A V_m$)

If A, $\mathcal{K}_m$ and $\mathcal{L}_m$ satisfy either of the two following conditions:

- *A is symmetric positive definite and $\mathcal{L}_m = \mathcal{K}_m$, or*
- *A is non-singular and $\mathcal{L}_m = A\mathcal{K}_m$.*

Then the $W_m^T A V_m$ matrix is non-singular for any full-rank V_m and W_m.

Saad, Y. (2003). Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics.

Matrix form of projection techniques for linear systems, cont'd$_2$

- In practical implementations of projection techniques to build approximate solutions to linear systems, we need to consider:
  - How to choose the search and constraints subspaces $\mathcal{K}_m$ and $\mathcal{L}_m$ at a given iteration m.
  - If an approximation is not good enough, how to expand those subspaces to $\mathcal{K}_{m+1}$ and $\mathcal{L}_{m+1}$.

  Of particular interest for the definition of projection techniques are the so-called **Krylov subspaces**:

$$\mathcal{K}_m(A, r_0) := \mathsf{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\} \subseteq \mathbb{F}^n$$

  which form a nested sequence:

$$\mathcal{K}_1(A, r_0) \subseteq \mathcal{K}_2(A, r_0) \subseteq \cdots \subseteq \mathcal{K}_m(A, r_0) \subseteq \ldots.$$

  A **Krylov subspace method** is a projection technique based on the subspace $\mathcal{K}_m(A, r_0)$. Different choices of a constraints subspace lead to different kinds of Krylov subspace methods.

Matrix form of projection techniques for linear systems, cont'd$_3$

- The choice of the constraint subspace $\mathcal{L}_m$ is often made so that the approximation in $\mathcal{K}_m$ possesses some **optimality properties**, such as **minimizing the residual norm** or the **norm of the forward error**.

  Some widely used Krylov subspace methods are proposed based on the choices

$$\mathcal{L}_m = \mathcal{K}_m(A, r_0),\ \mathcal{L}_m = A\mathcal{K}_m(A, r_0) \text{ and } \mathcal{L}_m = \mathcal{K}_m(A^T, r_0).$$

Methods for general linear systems

Full orthogonalization method (FOM)

- The full orthogonalization method (FOM), proposed by Saad (1981), is an **orthogonal projection in a Krylov subspace** $\mathcal{K}_m(A, r_0)$, with constraints subspace $\mathcal{L}_m = \mathcal{K}_m$, i.e., it reads

$$\boxed{\text{Find } x_m \in x_0 + \mathcal{K}_m(A, r_0) \text{ such that } b - Ax_m \perp \mathcal{K}_m(A, r_0)}.$$

Assuming that the columns of $V_m := [v_1, \dots, v_m]$ form a basis of the Krylov subspace $\mathcal{K}_m(A, r_0)$, the iterate formed by FOM is then given by

$$x_m := x_0 + V_m(V_m^T A V_m)^{-1} V_m^T r_0.$$

We saw in lecture 11 that, if the columns of V_m form an othonormal basis of $\mathcal{K}_m(A, r_0)$ as obtained by Arnoldi, we then have

$$V_m^T A V_m = H_m$$

where H_m is an upper-Hessenberg matrix.
Moreover, we have $v_1 := r_0/\beta$, where $\beta := \|r_0\|_2$, so that

$$V_m^T r_0 = [v_1, \dots, v_m]^T v_1 \beta = \beta e_1^{(m)} \text{ where } e_1^{(m)} := I_m[:, 1].$$

Saad, Y. (1981). Krylov subspace methods for solving large unsymmetric linear systems. Mathematics of computation, 37(155), 105-126.

Full orthogonalization method (FOM), cont'd$_1$

Consequently, we have

$$\boxed{x_m := x_0 + V_m\tilde{y} \;\text{ where } H_m\tilde{y} = \beta e_1^{(m)}}.$$

In most cases, the dimension m of the Krylov subspace $\mathcal{K}_m(A, r_0)$ is much smaller n, so that one can solve for $\tilde{y}$ such that $H_m\tilde{y} = \beta e_1^{(m)}$ using a direct method or, since H_m is Hessenberg, possibly also using a QR factorization.

- Let $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ be an iterate formed by FOM. Then, we have

$$\begin{aligned} r_m &:= b - Ax_m \\ &= b - A(x_0 + V_m\tilde{y}) \;\text{ where } H_m\tilde{y} = \beta e_1^{(m)} \\ &= r_0 - AV_m\tilde{y} \end{aligned}$$

where we recall the Arnoldi relation $AV_m = V_mH_m + h_{m+1,m}v_{m+1}e_m^{(m)T}$, so that

$$\begin{aligned} r_m &= r_0 - V_mH_my - h_{m+1,m}(e_m^{(m)T}\tilde{y})v_{m+1} \\ &= r_0 - \beta v_1 - h_{m+1,m}(e_m^{(m)T}\tilde{y})v_{m+1}. \end{aligned}$$

Full orthogonalization method (FOM), cont'd$_2$

But, remember that we have $r_0 = \beta v_1$, so that we obtain

$$\boxed{r_m = -h_{m+1,m}(e_m^{(m)T}\tilde{y})v_{m+1}}.$$

One can then promptly evaluate the residual norm $\|r_m\|_2$, without having to form the iterate x_m, nor to evaluate an additional matrix-vector product. Indeed, we have

$$\boxed{\|r_m\|_2 = |h_{m+1,m}||e_m^{(m)T}\tilde{y}|}.$$

- In practice, FOM is seldom used for the purpose of solving linear systems.

Full orthogonalization method (FOM), cont'd$_3$

- Implementations of the FOM method are defined by specifying a procedure to construct an orthonormal basis of the Krylov subspace $\mathcal{K}_m(A, r_0)$. This can be done using any variant of the Arnoldi algorithm, e.g.,

Algorithm 1 MGS-based FOM: $(x_0, \varepsilon) \mapsto x_j$

$r_0 := b - Ax_0$
$\beta := \|r_0\|_2$
$v_1 := r_0/\beta$
for $j = 1, 2 \ldots$ **do**
    $w := Av_j$
    for $i = 1, \ldots, j$ **do**
        $h_{ij} := w^T v_i$
        $w := w - h_{ij} v_i$
    $h_{j+1,j} := \|w\|_2$
    Solve for $\tilde{y}$ such that $H_j \tilde{y} = \beta e_1^{(j)}$
    if $h_{j+1,j} |e_j^{(j)T} \tilde{y}| < \varepsilon \|b\|_2$ **then**
        Stop   ▷ Stop if $\|r_j\|_2 < \varepsilon \|b\|_2$
    $v_{j+1} := w / h_{j+1,j}$
$x_j := x_0 + V_j \tilde{y}$

Generalized minimal residual (GMRES) method

- The generalized minimal residual (GMRES) method, proposed by Saad and Schultz (1986), is an **oblique projection in a Krylov subspace** $\mathcal{K}_m$, with constraints subspace $\mathcal{L}_m = A\mathcal{K}_m$, i.e., it reads

$$\boxed{\text{Find } x_m \in x_0 + \mathcal{K}_m(A, r_0) \text{ such that } b - Ax_m \perp A\mathcal{K}_m(A, r_0)}. \quad (2)$$

  Assuming that the columns of $V_m := [v_1, \dots, v_m]$ form a basis of the Krylov subspace $\mathcal{K}_m(A, r_0)$, the GMRES iterate is given by

$$x_m := x_0 + V_m((AV_m)^T AV_m)^{-1}(AV_m)^T r_0.$$

  However, it is more common and practical to derive the GMRES iterate based on its **optimality property**:

Theorem (Optimality of GMRES iterates)

The iterate x_m is the solution of Pb. 2 if and only it minimizes the residual norm $\|b - Ax\|_2$ over the affine subspace $x_0 + \mathcal{K}_m(A, r_0)$, i.e., if and only if

$$\|b - Ax_m\|_2 = \min_{x \in x_0 + \mathcal{K}_m(A, r_0)} \|b - Ax\|_2.$$

Saad, Y., & Schultz, M. H. (1986). GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on scientific and statistical computing, 7(3), 856-869.

Generalized minimal residual (GMRES) method, cont'd$_1$

- Consequently, the GMRES iterate $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ is given by $\boxed{x_m := x_0 + V_m\tilde{y}}$, where

$$\begin{aligned}\tilde{y} &:= \arg\min_{y\in\mathbb{R}^m} \|b - A(x_0 + V_m y)\|_2 \\ &= \arg\min_{y\in\mathbb{R}^m} \|r_0 - AV_m y\|_2\end{aligned}$$

in which, we recall that $r_0 = \beta v_1$, where $\beta := \|r_0\|_2$, and, as the Arnoldi relation reads $AV_m = V_{m+1}\underline{H_m}$ in which $\underline{H_m} := V_{m+1}^T AV_m$, we obtain:

$$\begin{aligned}\tilde{y} &= \arg\min_{y\in\mathbb{R}^m} \|\beta v_1 - V_{m+1}\underline{H_m}y\|_2 \\ &= \arg\min_{y\in\mathbb{R}^m} \|V_{m+1}(\beta e_1^{(m+1)} - \underline{H_m}y)\|_2 \\ &= \arg\min_{y\in\mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{H_m}y\|_2 \text{ where } e_1^{(m+1)} := I_{m+1}[:, 1].\end{aligned}$$

Generalized minimal residual (GMRES) method, cont'd$_2$

- The least-squares problem $\min_{y\in\mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{H_m} y\|_2$ is solved using the QR decomposition of the Hessenberg matrix, which can be done efficiently provided that the dimension m of the approximation and constraints subspaces is not too large.

  Let $Q_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ be the orthogonal matrix s.t. $\underline{H_m} = Q_{m+1}^T \underline{R_m}$, where $\underline{R_m} \in \mathbb{R}^{(m+1)\times m}$ is an upper-triangular matrix.

  Then, the least-squares problem is recast into

$$\begin{aligned}\min_{y\in\mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{H_m} y\|_2 &= \min_{y\in\mathbb{R}^m} \|\beta e_1^{(m+1)} - Q_{m+1}^T \underline{R_m} y\|_2 \\ &= \min_{y\in\mathbb{R}^m} \|\beta Q_{m+1} e_1^{(m+1)} - \underline{R_m} y\|_2 \\ &= \min_{y\in\mathbb{R}^m} \left\|\beta q_1 - \begin{bmatrix} R_m \\ 0_{1\times m}\end{bmatrix} y\right\|_2\end{aligned}$$

  where $q_1 := Q_{m+1} e_1^{(m+1)} = Q_{m+1}[1:m+1, 1]$ and $R_m = \underline{R_m}[1\!:\!m, 1\!:\!m]$.

Generalized minimal residual (GMRES) method, cont'd$_3$

So that the least-squares problem is solved by solving the following triangular system:

$$\boxed{R_m\tilde{y} = \beta q_1[1:m]}.$$

- Then, the residual $r_m := b - Ax_m$ is s.t. $r_m = V_{m+1}(\beta e_1^{(m+1)} - \underline{H_m}\tilde{y})$ and

$$\begin{aligned}
\|r_m\|_2 &= \|\beta e_1^{(m+1)} - \underline{H_m}\tilde{y}\|_2 \\
&= \left\|\beta q_1 - \begin{bmatrix} R_m \\ 0_{1\times m}\end{bmatrix}\tilde{y}\right\|_2 \\
&= \left\|\beta q_1 - \begin{bmatrix} \beta q_1[1:m] \\ 0\end{bmatrix}\right\|_2 \\
&= \left\|\begin{bmatrix} 0_{m\times 1} \\ \beta q_1[m+1]\end{bmatrix}\right\|_2
\end{aligned}$$

so that $\boxed{\|r_m\|_2 = \beta|q_1[m+1]|}$.

Thus, one needs not to assemble the iterate x_m, nor to perform an additional matrix-vector product in order to monitor convergence.

Generalized minimal residual (GMRES) method, cont'd$_4$

- Just like with FOM, the workhorse of GMRES is the orthogonalization of Krylov basis vectors. In particular, this is most frequently implemented on the basis of the MGS procedure:

Algorithm 2 MGS-based GMRES: $(x_0, \varepsilon) \mapsto x_j$

$r_0 := b - Ax_0$
$\beta := \|r_0\|_2$
$v_1 := r_0/\beta$
for $j = 1, 2 \dots$ **do**
    $w := Av_j$
    for $i = 1, \dots, j$ **do**
        $h_{ij} := w^T v_i$
        $w := w - h_{ij} v_i$
    $h_{j+1,j} := \|w\|_2$
    Solve for $\tilde{y} = \arg\min_{y \in \mathbb{R}^j} \|\beta e_1^{(j+1)} - \underline{H}_j y\|$
    if $\|\beta e_1^{(j+1)} - \underline{H}_j \tilde{y}\| < \varepsilon \|b\|_2$ **then**
        Stop ▷ Stop if $\|r_j\|_2 < \varepsilon \|b\|_2$
    $v_{j+1} := w / h_{j+1,j}$
$x_j := x_0 + V_j \tilde{y}$

- Remember that the least-squares problem $\min_{y\in\mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{H_m} y\|_2$ is recast into the linear system $R_m\tilde{y} = \beta q_1[1\!:\!m]$ where $R_m := \underline{R_m}[1\!:\!m, 1\!:\!m]$ in which the QR decomposition $Q_{m+1}^T\underline{R_m} = \underline{H_m}$ is needed.

  Suppose that we have obtained the QR decomposition of the matrix $\underline{H_{j-1}}$, and we are interested in getting the decomposition of $\underline{H_j}$ with the least amount of work possible. Clearly, we have

$$\underline{H_j} = \begin{bmatrix} \underline{H_{j-1}} & h_{1:j,j} \\ 0_{1\times j-1} & h_{j+1,j} \end{bmatrix}.$$

  We saw in Lecture 07 that Givens rotations can be used to turn an upper Hessenberg matrix into triangular form. In particular, for $\underline{H_{j-1}}$, we have

$$\underline{R_{j-1}} = \begin{bmatrix} R_{j-1} \\ 0_{1\times(j-1)} \end{bmatrix} = G_{j-1}^{(j)} G_{j-2}^{(j)} \dots G_1^{(j)} \underline{H_{j-1}} = Q_j \underline{H_{j-1}}$$

Generalized minimal residual (GMRES) method, cont'd$_6$

where the Givens rotation matrices $G_1^{(j)}, \ldots, G_{j-1}^{(j)} \in \mathbb{R}^{j\times j}$ are given by

$$G_i^{(j)} = \begin{bmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & c_i & s_i & & & \\ & & & -s_i & c_i & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ i\text{-th row} \\ (i+1)\text{-th row} \\ \\ \\ \\ \end{matrix}$$

in which the scalars s_i and c_i are set so as to zero the $(i+1, i)$-entry of the Hessenberg matrix $G_i^{(j)}$ is applied to.

Clearly, we have

$$G_i^{(j+1)} = \begin{bmatrix} G_i^{(j)} & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix}$$

for $i = 1, \ldots, j-1$.

Generalized minimal residual (GMRES) method, cont'd$_7$

so that

$$\begin{aligned}
\underline{R_j} &= G_j^{(j+1)} \dots G_1^{(j+1)} \underline{H_j} \\
&= G_j^{(j+1)} \begin{bmatrix} G_{j-1}^{(j)} \dots G_1^{(j)} \underline{H_{j-1}} & G_{j-1}^{(j)} \dots G_1^{(j)} h_{1:j,j} \\ 0_{1\times(j-1)} & h_{j+1,j} \end{bmatrix} \\
&= G_j^{(j+1)} \begin{bmatrix} \underline{R_{j-1}} & G_{j-1}^{(j)} \dots G_1^{(j)} h_{1:j,j} \\ 0_{1\times(j-1)} & h_{j+1,j} \end{bmatrix} \\
&= \begin{bmatrix} \underline{R_{j-1}} & G_j^{(j+1)}[1:j, 1:j+1] \begin{bmatrix} G_{j-1}^{(j)} \dots G_1^{(j)} h_{1:j,j} \\ h_{j+1,j} \end{bmatrix} \\ 0_{1\times(j-1)} & 0 \end{bmatrix}.
\end{aligned}$$

Therefore, while performing the j-th iteration of GMRES, one is equipped with $\underline{R_{j-1}}$ and $\underline{H_j}$. In order to assemble $\underline{R_j}$, there only remains to apply the Givens rotations $G_1^{(j+1)}, \dots, G_j^{(j+1)}$ to the last column of $\underline{H_j}$, i.e.,

$$\boxed{\underline{R_j}[1:j+1, j] = G_j^{(j+1)} \dots G_1^{(j+1)} h_{1:j+1,j}}.$$

Generalized minimal residual (GMRES) method, cont'd$_8$

- We saw that the least-squares problem $\min_{x\in x_0+\mathcal{K}_j(A,r_0)}\|b-Ax\|_2$ can be recast in the linear system $R_j\tilde{y}=\underline{g_j}[1:j]$ where $\underline{g_j}:=\beta Q_{j+1}e_1^{(j+1)}$ so that

$$\underline{g_j}:=\beta G_j^{(j+1)}\dots G_1^{(j+1)}e_1^{(j+1)}=\begin{bmatrix}\gamma_1\\ \vdots\\ \gamma_{j-1}\\ c_j\gamma_j\\ -s_j\gamma_j\end{bmatrix}\quad\text{where}\quad\begin{bmatrix}\gamma_1\\ \vdots\\ \gamma_j\end{bmatrix}=\underline{g_{j-1}}$$

with $\underline{g_0}=\beta$, and in which the scalars s_i and c_i are given by

$$s_j=\frac{h_{j+1,j}}{\sqrt{\left(h_{jj}^{(j-1)}\right)^2+h_{j+1,j}^2}}\text{ and }c_j=\frac{h_{jj}^{(j-1)}}{\sqrt{\left(h_{jj}^{(j-1)}\right)^2+h_{j+1,j}^2}}.$$

where $\underline{H_j^{(j)}}:=\underline{R_j}$.

Generalized minimal residual (GMRES) method, cont'd$_9$

- In practice, the $\underline{R_1}, \ldots, \underline{R_m}$ and $\underline{g_1}, \ldots, \underline{g_m}$ are often computed in-place, stored in pre-allocated $\underline{H_m}$ and $\underline{g_m}$. This yields the following algorithm

Algorithm 3 Practical GMRES: $(x_0, m, \varepsilon) \mapsto x_j$

// Allocate $\underline{H} \in \mathbb{R}^{(m+1)\times m}$, $\underline{g} \in \mathbb{R}^{m+1}$ and $V \in \mathbb{R}^{n\times(m+1)}$
$r_0 := b - Ax_0$; $\beta := \|r_0\|_2$; $\underline{g} := [\beta, 0, \ldots, 0]^T$; $v_1 := r_0/\beta$
for $j = 1, 2 \ldots$ **do**
    Compute $h_{1:j+1,j}$ and v_{j+1}
    for $i = 1, \ldots, j-1$ **do**
        // Apply $G_i^{(j+1)}$ to $h_{1:j+1,j}$.
        $$\begin{bmatrix} h_{ij} \\ h_{i+1,j} \end{bmatrix} := \begin{bmatrix} c_i & s_i \\ -s_i & c_i \end{bmatrix} \begin{bmatrix} h_{ij} \\ h_{i+1,j} \end{bmatrix} \text{ where } \begin{cases} s_i := h_{i+1,i}/(h_{ii}^2 + h_{i+1,i}^2)^{1/2} \\ c_i := h_{ii}/(h_{ii}^2 + h_{i+1,i}^2)^{1/2} \end{cases}$$
    // Apply $G_j^{(j+1)}$ to $\underline{g}[1:j+1]$ and $h_{1:j+1,j}$
    $$\begin{bmatrix} \underline{g}[j] \\ \underline{g}[j+1] \end{bmatrix} := \begin{bmatrix} c_j & s_j \\ -s_j & c_j \end{bmatrix} \begin{bmatrix} \underline{g}[j] \\ 0 \end{bmatrix} \text{ where } \begin{cases} s_j := h_{j+1,j}/(h_{jj}^2 + h_{j+1,j}^2)^{1/2} \\ c_j := h_{jj}/(h_{jj}^2 + h_{j+1,j}^2)^{1/2} \end{cases}$$
    $h_{jj} := c_j h_{jj} + s_j h_{j+1,j}$; $h_{j+1,j} := 0$
    if $|\underline{g}[j+1]| < \varepsilon\|b\|_2$ **then**
        Stop ▷ Stop if $\|r_j\|_2 < \varepsilon\|b\|_2$
$x_j := x_0 + V_j\tilde{y}$ where $\tilde{y}$ is solution of triangular system $H[1:j, 1:j]\tilde{y} = \underline{g}[1:j]$

Methods for symmetric linear systems

Conjugate gradient (CG) method

- Here, we assume that the matrix A is **SPD**. Similarly to FOM, the CG method (Hestenes and Stiefel, 1952) is an **orthogonal projection** in the Krylov subspace $\mathcal{K}_m(A, r_0)$. That is, CG iterates are formed as follows:

$$\boxed{\text{Find } x_m \in x_0 + \mathcal{K}_m(A, r_0) \text{ such that } b - Ax_m \perp \mathcal{K}_m(A, r_0)}.$$

Once again, assuming that the columns of $V_m := [v_1, \ldots, v_m]$ form a basis of $\mathcal{K}_m(A, r_0)$, the CG iterate is given by

$$x_m := x_0 + V_m(V_m^T A V_m)^{-1} V_m^T r_0.$$

We saw in Lecture 11 that, if the columns of V_m form an orthonormal basis of $\mathcal{K}_m(A, r_0)$ as obtained by the Lanczos method, we then have

$$V_m^T A V_m = T_m$$

where T_m is a tridiagonal matrix.
Moreover, we have $v_1 := r_0/\beta$, where $\beta := \|r_0\|_2$, so that

$$V_m^T r_0 = [v_1, \ldots, v_m]^T v_1 \beta = \beta e_1^{(m)} \text{ where } e_1^{(m)} := I_m[:, 1].$$

Hestenes M. R. & Stiefel E. L. (1952). Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards, 49, 409–436.

Conjugate gradient (CG) method, cont'd$_1$

Consequently, we have

$$\boxed{x_m := x_0 + V_m\tilde{y} \text{ where } T_m\tilde{y} = \beta e_1^{(m)}}.$$

As formulated above, each CG iterate x_m requires to solve a linear system for $\tilde{y}$ with the tridiagonal matrix T_m.

As A is SPD, so is T_m. Thus, one can make use of the LU decomposition of T_m in order to solve $T_m\tilde{y} = \beta e_1^{(m)}$.

Let x_{m+1} denote the CG iterate in $x_0 + \mathcal{K}_{m+1}(A, r_0)$, i.e.,

$$x_{m+1} := x_0 + V_{m+1}\tilde{y} \;\text{ where }\; T_{m+1}\tilde{y} = \beta e_1^{(m+1)}.$$

In what follows, we present the steps enumerated by Bai and Pan (2021) in order to construct the CG iterate x_{m+1} given x_m.

Bai, Z. Z., & Pan, J. Y. (2021). Matrix analysis and computations. Society for Industrial and Applied Mathematics.

Conjugate gradient (CG) method, cont'd$_2$

Let the tridiagonal matrices T_m and T_{m+1} admit LU decompositions of the form L_mU_m and $L_{m+1}U_{m+1}$, respectively, in which we have

$$L_\ell = \begin{bmatrix} 1 & & & \\ \gamma_1 & 1 & & \\ & \ddots & \ddots & \\ & & \gamma_{\ell-1} & 1 \end{bmatrix} \text{ and } U_\ell = \begin{bmatrix} \eta_1 & \beta_1 & & \\ & \eta_2 & \ddots & \\ & & \ddots & \beta_{\ell-1} \\ & & & \eta_\ell \end{bmatrix} \text{ for } \ell = m, m+1.$$

That is, L_m and U_m are the m-th leading principal sub-matrices of L_{m+1} and U_{m+1}.

More precisely, we have

$$\begin{cases} \eta_1 := \alpha_1 & \\ \gamma_i := \beta_i / \eta_i & \text{for i=1,...,m} \\ \eta_{i+1} := \alpha_{i+1} - \gamma_i \beta_i & \text{for i=1,...,m} \end{cases}$$

where $\alpha_j := T_{jj} = v_j^T A v_j$ and $\beta_j := T_{j+1,j} := v_{j+1}^T A v_j = v_j^T A v_{j+1}$ denote the diagonal and off-diagonal components of T_m, respectively.

Conjugate gradient (CG) method, cont'd$_3$

Given those LU factorizations, the CG iterate $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ may be recast into

$$x_m := x_0 + P_m z^{(m)}$$

where $P_m := V_m U_m^{-1} \in \mathbb{R}^{n \times m}$ and $z^{(m)} := \beta L_m^{-1} e_1^{(m)} \in \mathbb{R}^m$. Then, we have

$$\begin{aligned} P_{m+1} := V_{m+1} U_{m+1}^{-1} = [V_m \, v_{m+1}] \begin{bmatrix} U_m^{-1} & *_{m \times 1} \\ 0_{1 \times m} & 1/\eta_{m+1} \end{bmatrix} &= [V_m U_m^{-1} \, p_{m+1}] \\ &= [P_m \, p_{m+1}]. \end{aligned}$$

And, from $V_{m+1} = P_{m+1} U_{m+1}$, we get

$$v_{m+1} = \beta_m p_m + \eta_{m+1} p_{m+1} \implies \boxed{p_{m+1} = (v_{m+1} - \beta_m p_m)/\eta_{m+1}}$$

for $m = 1, 2, \ldots$, while $\boxed{p_1 = v_1/\eta_1}$.

Conjugate gradient (CG) method, cont'd$_4$

Then, as we denote $z^{(m+1)} := [z^{(m)T} \, z_{m+1}]^T = [z_1, \dots, z_m, z_{m+1}]^T$, we see that

$$L_{m+1} z^{(m+1)} = \beta e_1^{(m+1)}$$

$$\begin{bmatrix} L_m z^{(m)} \\ \gamma_m z_m + z_{m+1} \end{bmatrix} = \begin{bmatrix} \beta e_1^{(m)} \\ 0 \end{bmatrix}$$

so that $\boxed{z_{m+1} = -\gamma_m z_m}$ for $m = 1, 2, \dots$ while $\boxed{z_1 = \beta}$. Therefore, we get

$$\begin{aligned} x_{m+1} &:= x_0 + P_{m+1} z^{(m+1)} \\ &= x_0 + [P_m \, p_{m+1}] \begin{bmatrix} z^{(m)} \\ z_{m+1} \end{bmatrix} \\ &= x_0 + P_m z^{(m)} + z_{m+1} p_{m+1} \end{aligned}$$

so that

$$\boxed{x_{m+1} := x_m + z_{m+1} p_{m+1} \text{ for } m = 0, 1, 2, \dots}.$$

Conjugate gradient (CG) method, cont'd$_5$

- Then, alongside an implementation of Lanczos procedure which generates a set of orthonormal basis vectors $v_1, v_2, \ldots, v_{m+1}$ spanning the subspace $\mathcal{K}_m(A, r_0)$ with the tridiagonal components $\alpha_1, \ldots, \alpha_{m+1}$ and $\beta_1, \ldots, \beta_m$, one can generate the sequence $x_1, x_2, \ldots, x_{m+1}$ of CG iterates as follows:

$r_0 := b - Ax_0$

$\beta := \|r_0\|_2;\; z_1 := \beta$

$v_1 := r_0/\beta;\; \alpha_1 := v_1^T A v_1;\; \eta_1 := \alpha_1;\; p_1 := v_1/\eta_1$

for $j = 1, \ldots, m$

  $x_j := x_{j-1} + z_j p_j$

  Compute α_{j+1}, β_j and v_{j+1} by Lanczos iteration

  $\gamma_j := \beta_j/\eta_j$

  $\eta_{j+1} := \alpha_{j+1} - \gamma_j \beta_j$

  $z_{j+1} := -\gamma_j z_j$

  $p_{j+1} := (v_{j+1} - \beta_j p_j)/\eta_{j+1}$

Conjugate gradient (CG) method, cont'd$_6$

- We will find it useful to consider **generic inner products** $(\cdot, \cdot)$ in place of the usual dot product. Two important results prove to be useful in deriving the CG algorithm. First, there is the **conjugacy** of the p vectors:

Theorem (A-orthogonality of p vectors)

*Assuming A is SPD, the vectors $p_1, \ldots, p_{m+1}$ built as described on the previous slides are A-**orthogonal** (or **conjugate**). That is,*

$$(p_i, p_j)_A := (Ap_i, p_j) = 0 \text{ if } i \neq j.$$

Second, there is the **orthogonality of residual vectors**:

Theorem (Orthogonality of residual vectors)

Let $r_j := b - Ax_j$ where x_j is the CG iterate in $x_0 + \mathcal{K}_m(A, r_0)$. Then,

$$r_j = \rho_j v_{j+1}, \text{ where } \rho_0 := \beta \text{ and } \rho_j := -\beta_j e_j^{(j)T} \tilde{y} \text{ s.t. } T_j \tilde{y} = \beta e_1^{(j)}$$

so that, by virtue of orthogonality of the Krylov basis vectors $v_1, \ldots, v_{m+1}$, the CG residual vectors $r_0, \ldots, r_m$ are orthogonal, i.e., $(r_i, r_j) = 0$ if $i \neq j$.

Conjugate gradient (CG) method, cont'd$_7$

- Now, let us define the **search direction** $\tilde{p}_{j+1} := \rho_j \eta_{j+1} p_{j+1}$ so that, using the fact that $r_j = \rho_j v_{j+1}$, we get

$$
\begin{aligned}
p_{j+1} &:= (v_{j+1} - \beta_j p_j)/\eta_{j+1} \\
\rho_j \eta_{j+1} p_{j+1} &:= \rho_j v_{j+1} - \rho_j \beta_j p_j \\
\tilde{p}_{j+1} &:= r_j - \rho_j \beta_j p_j \\
&\boxed{\tilde{p}_{j+1} := r_j + \tau_j \tilde{p}_j}
\end{aligned}
$$

where $\tau_j := -\rho_j \beta_j / (\rho_{j-1} \eta_j)$. These search directions are A-**orthogonal**. Then, from $x_j := x_{j-1} + z_j p_j$, we get

$$
\boxed{x_j := x_{j-1} + \xi_j \tilde{p}_j} \quad \text{where} \quad \xi_j := z_j / (\rho_{j-1} \eta_j).
$$

Also, the CG residual vector r_j can be reformulated as follows:

$$
r_j := b - Ax_j = b - A(x_{j-1} + \xi_j \tilde{p}_j) = b - Ax_{j-1} - \xi_j A \tilde{p}_j
$$

so that $\boxed{r_j := r_{j-1} - \xi_j A \tilde{p}_j}$.

Conjugate gradient (CG) method, cont'd$_8$

- Now, we are only left with finding alternative expressions for τ_j and ξ_j which do not explicitly depend on the tridiagonal form T_j and its LU decomposition.
  - First, using the stated **orthogonality of CG residuals**, we get

$$\begin{aligned}(r_j, r_{j-1}) &= 0\\ (r_{j-1} - \xi_j A\tilde{p}_j, r_{j-1}) &= 0\\ (r_{j-1}, r_{j-1}) - \xi_j (A\tilde{p}_j, r_{j-1}) &= 0\end{aligned}$$

    for which using the **conjugacy of search directions** as well as $\tilde{p}_{j+1} := r_j + \tau_j \tilde{p}_j$ leads to

$$\begin{aligned}(A\tilde{p}_j, r_{j-1}) &= (A\tilde{p}_j, \tilde{p}_j - \tau_{j-1}\tilde{p}_{j-1})\\ &= (A\tilde{p}_j, \tilde{p}_j) - \tau_{j-1}(A\tilde{p}_j, \tilde{p}_{j-1})\\ &= (A\tilde{p}_j, \tilde{p}_j)\end{aligned}$$

    so that $\boxed{\xi_j = (r_{j-1}, r_{j-1})/(A\tilde{p}_j, \tilde{p}_j)}$.

Conjugate gradient (CG) method, cont'd₉

- Second, in order to find an alternative expression for τ_j, we start as follows from the statement of **conjugacy of search directions**:

$$
\begin{aligned}
(A\tilde{p}_j, \tilde{p}_{j+1}) &= 0 \\
(A\tilde{p}_j, r_j + \tau_j \tilde{p}_j) &= 0 \\
(A\tilde{p}_j, r_j) + \tau_j (A\tilde{p}_j, \tilde{p}_j) &= 0
\end{aligned}
$$

so that $\tau_j = -(A\tilde{p}_j, r_j)/(A\tilde{p}_j, \tilde{p}_j)$. Then, using $r_j := r_{j-1} - \xi_j A\tilde{p}_j$ as well as the **orthogonality of CG residuals**, we get

$$
\tau_j = -\frac{(A\tilde{p}_j, r_j)}{(A\tilde{p}_j, \tilde{p}_j)} = \frac{1}{\xi_j}\frac{(r_j - r_{j-1}, r_j)}{(A\tilde{p}_j, \tilde{p}_j)} = \frac{(A\tilde{p}_j, \tilde{p}_j)}{(r_{j-1}, r_{j-1})}\frac{(r_j, r_j)}{(A\tilde{p}_j, \tilde{p}_j)}
$$

so that $\boxed{\tau_j = (r_j, r_j)/(r_{j-1}, r_{j-1})}$.

Conjugate gradient (CG) method, cont'd$_{10}$

- Piecing together all the expressions for the update of ξ_j, x_j, r_j, τ_j and $\tilde{p}_{j+1}$, we get the following iteration for the CG method:

$$
\boxed{
\begin{array}{l}
r_0 := b - Ax_0 \\
\tilde{p}_1 := r_0 \\
\text{for } j = 1, \ldots, m \\
\quad \xi_j := (r_{j-1}, r_{j-1})/(A\tilde{p}_j, \tilde{p}_j) \\
\quad x_j := x_{j-1} + \xi_j \tilde{p}_j \\
\quad r_j := r_{j-1} - \xi_j A\tilde{p}_j \\
\quad \tau_j := (r_j, r_j)/(r_{j-1}, r_{j-1}) \\
\quad \tilde{p}_{j+1} := r_j + \tau_j \tilde{p}_j
\end{array}
}
$$

Conjugate gradient (CG) method, cont'd$_{11}$

- In order to reflect the most commonly encountered formulations of the CG method, the following changes of variables are operated

$$\xi_j \mapsto \alpha_j, \; \tau_j \mapsto \beta_j \; \text{ and } \tilde{p}_j \mapsto p_j$$

where α_j and β_j are not to be confused with the components of the tridiagonal form of A.
This leads to the following algorithm:

Algorithm 4 CG: $(x_0, \varepsilon) \mapsto x_j$

```
r_0 := b - A x_0
p_1 := r_0
for j = 1, 2 ... do
    α_j := (r_{j-1}, r_{j-1}) / (A p_j, p_j)
    x_j := x_{j-1} + α_j p_j
    r_j := r_{j-1} - α_j A p_j
    if ||r_j||_2 < ε ||b||_2 then
        Stop
    β_j := (r_j, r_j) / (r_{j-1}, r_{j-1})
    p_{j+1} := r_j + β_j p_j
```

Conjugate gradient (CG) method, cont'd$_{12}$

- Note that the CG method can be implemented allocating storage only for the iterate x, the search direction p, the matrix-vector product Ap and the residual r. Doing so leads to the following practical implementation:

Algorithm 5 Practical CG: $(x_0, \varepsilon) \mapsto x_j$

```
Allocate memory for x, p, w, r ∈ ℝ^n
r := b − A x_0
p := r
for j = 1, 2 ... do
    w := Ap
    α := (r, r)/(w, p)
    β := 1/(r, r)
    x := x + αp
    r := r − αw
    if ‖r‖_2 < ε‖b‖_2 then
        Stop
    β := β · (r, r)
    p := r + βp
```

Conjugate gradient (CG) method, cont'd$_{13}$

- An essential property of the CG method is that of **optimality**, namely

Theorem (Optimality of CG iterates)

Let A be SPD and $x_j \in x_0 + \mathcal{K}_j(A, r_0)$ denote the CG iterate approximating the solution of $Ax = b$. Then, x_j ***minimizes the A-norm of the error*** *over the search space, i.e.,*

$$\|x - x_j\|_A = \min_{y \in x_0 + \mathcal{K}_j(A, r_0)} \|x - y\|_A \text{ where } \|x\|_A := (Ax, x)^{1/2}.$$

Another important results on the CG method is about its convergence:

Theorem (Upper bound on the relative change of A-norm of the error)

Let A be SPD with smallest and largest eigenvalues given by λ_{min} and λ_{max}, respectively. Then, it holds that

$$\frac{\|x_j - x\|_A}{\|x_0 - x\|_A} \leq \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^j$$

where $\kappa_2(A) = \lambda_{max}/\lambda_{min}$ is the spectral condition number of A.

Conjugate gradient (CG) method, cont'd$_{14}$

- An alternative presentation of the CG method to that of **orthogonal projection in a Krylov subspace** is frequent in the field of **optimization**.
  - That is, considering an SPD matrix $A \in \mathbb{R}^{n\times n}$ and a vector $b \in \mathbb{R}^n$, the **quadratic function**

$$\begin{aligned} f : \mathbb{R}^n &\rightarrow \mathbb{R} \\ x &\mapsto x^T A x - x^T b \end{aligned}$$

    has $\nabla f(x) = Ax - b$ and $\nabla^2 f(x) = A$ for 1st and 2nd derivatives.
  - Since the Hessian $\nabla^2 f$ of f is SPD, the critical point x_* such that $\nabla f(x_*) = 0$ ($\implies Ax_* = b$), is a **minimizer** of the function $f(x)$.
  - An iterative procedure started with x_0 and aimed at finding x_* is devised upon setting a set of **search directions** $p_0, p_1, p_2, \ldots$, in the span of which subsequent approximations $x_1, x_2, \ldots$ of x_* are formed:

$$x_j := \sum_{i=0}^{j} \alpha_i p_i.$$

Conjugate gradient (CG) method, cont'd$_{15}$

- The search directions are chosen to be A-**orthogonal**, or **conjugate**, i.e., such that $(Ap_i, p_j) = 0$ for $i \neq j$.
- The initial **search direction** is chosen as the **opposite of the gradient of f at** x_0, i.e., $p_0 := -\nabla f(x_0) = b - Ax_0 =: r_0$.
- Subsequent search directions $p_1, p_2, \dots$ being A-orthogonal with respect to $p_0 \propto \nabla f(x_0)$, they are **conjugate to the gradient** $\nabla f(x_0)$, hence the name **conjugate gradient** given to the method.

Minimal residual (MINRES) method

- The **optimality property** of the CG method is reliant on the **assumption of positive definiteness** of A. Furthermore, in cases A is **not positive definite**, the CG method may **break down** (Paige et al., 1995).
  For cases where A is **symmetric but indefinite** (still non-singular), then, the minimal residual (MINRES) method (Paige and Saunders, 1975) is introduced as an **oblique projection in a Krylov subspace** $\mathcal{K}_m(A, r_0)$, with constraints subspace $\mathcal{L}_m := A\mathcal{K}_m$, i.e., similarly as GMRES, it reads

$$\boxed{\text{Find } x_m \in x_0 + \mathcal{K}_m(A, r_0) \text{ such that } b - Ax_m \perp A\mathcal{K}_m(A, r_0)}, \quad (3)$$

  the difference with GMRES being that A is symmetric.
  Assuming that the columns of $V_m := [v_1, \ldots, v_m]$ form a basis of the Krylov subspace $\mathcal{K}_m(A, r_0)$, the MINRES iterate is then given as follows from the **Petrov-Galerkin condition**:

$$x_m := x_0 + V_m((AV_m)^T AV_m)^{-1}(AV_m)^T r_0.$$

Paige, C. C., Parlett, B. N., & Van der Vorst, H. A. (1995). Approximate solutions and eigenvalue bounds from Krylov subspaces. Numerical linear algebra with applications, 2(2), 115-133.

Paige, C. C. & Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. SIAM Journal on Numerical Analysis, 12, 617–629.

Minimal residual (MINRES) method, cont'd$_1$

- However, similarly as for GMRES, it is more common and practical to derive the GMRES iterate based on the following **optimality property**:

Theorem (Optimality of MINRES iterates)

The iterate x_m is the solution of Pb. (3) *if and only if it minimizes the residual norm $\|b - Ax\|_2$ over the affine subspace $x_0 + \mathcal{K}_m(A, r_0)$, i.e., iff*

$$\|b - Ax_m\|_2 = \min_{x \in x_0 + \mathcal{K}_m(A, r_0)} \|b - Ax\|_2.$$

Consequently, the MINRES iterate $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ is given by $\boxed{x_m := x_0 + V_m \tilde{y}}$, where

$$\tilde{y} := \arg \min_{y \in \mathbb{R}^m} \|r_0 - AV_m y\|_2$$

in which, we recall that $r_0 = \beta v_1$, where $\beta := \|r_0\|_2$ and, as the Lanczos relation reads $AV_m = V_{m+1}\underline{T_m}$ in which $\underline{T_m} := V_{m+1}^T AV_m$, we obtain

$$\tilde{y} = \arg \min_{y \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{T_m} y\|.$$

Minimal residual (MINRES) method, cont'd$_2$

- Just as with GMRES, the least-squares problem $\min_{y \in \mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{T_m} y\|_2$ can be solved using the QR decomposition of the tridiagonal matrix.
  Let $Q_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ be the orthogonal matrix s.t. $\underline{T_m} = Q_{m+1}^T \underline{R_m}$, where $\underline{R_m} \in \mathbb{R}^{(m+1)\times m}$ is an upper-triangular matrix.
  Since $\underline{T_m}$ is tridiagonal, the upper-triangular matrix $\underline{R_m}$ is banded with a bandwidth of 3, i.e., we have

$$\underline{R_m} = \begin{bmatrix} \tau_1^{(1)} & \tau_1^{(2)} & \tau_1^{(3)} & & & \\ 0 & \tau_2^{(1)} & \tau_2^{(2)} & \tau_2^{(3)} & & \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \tau_{m-2}^{(1)} & \tau_{m-2}^{(2)} & \tau_{m-2}^{(3)} \\ \vdots & & & \ddots & \tau_{m-1}^{(1)} & \tau_{m-1}^{(2)} \\ \vdots & & & & \ddots & \tau_m^{(1)} \\ 0 & \dots & \dots & \dots & \dots & 0 \end{bmatrix} = \begin{bmatrix} R_m \\ 0_{1\times m} \end{bmatrix}$$

  where $R_m := \underline{R_m}[1\!:\!m, 1\!:\!m]$.

Minimal residual (MINRES) method, cont'd$_3$

The least-squares problem is recast into

$$\min_{y\in\mathbb{R}^m} \|\beta e_1^{(m+1)} - \underline{T_m} y\|_2 = \min_{y\in\mathbb{R}^m} \left\| \beta q_1 - \begin{bmatrix} R_m \\ 0_{1\times m} \end{bmatrix} y \right\|_2$$

where $q_1 := Q_{m+1} e_1^{(m+1)} = Q_{m+1}[1:m+1, 1]$.

Then, as we let $\underline{g_m} := \beta q_1 \in \mathbb{R}^{m+1}$ with $\underline{g_0} := \beta$, the least-squares problem is solved by solving the following triangular system:

$$\boxed{R_m \tilde{y} = \underline{g_m}[1:m]}\,.$$

Then, the residual $r_m := b - Ax_m$ is s.t. $r_m = V_{m+1}(\beta e_1^{(m+1)} - \underline{T_m}\tilde{y})$ and $\boxed{\|r_m\|_2 = \beta|q_1[m+1]| = |\underline{g_m}[m+1]|}$.

Thus, one needs not to assemble the iterate x_m, nor to perform an additional matrix-vector product in order to monitor convergence.

Minimal residual (MINRES) method, cont'd$_4$

Suppose that we have obtained the QR decomposition of the matrix $\underline{T_{j-1}}$, and we are interested in getting the decomposition of $\underline{T_j}$ with the least amount of work possible. Clearly, we have

$$\underline{T_j} = \begin{bmatrix} \underline{T_{j-1}} & t_{1:j,j} \\ 0_{1\times j-1} & \beta_j \end{bmatrix} \text{ where } t_{1:j,j} = \begin{bmatrix} 0_{(j-2)\times 1} \\ \beta_{j-1} \\ \alpha_j \end{bmatrix}.$$

We saw in Lecture 07 that Givens rotations can be used to turn an upper Hessenberg matrix into triangular form. In particular, for $\underline{T_{j-1}}$, we have

$$\underline{R_{j-1}} = \begin{bmatrix} R_{j-1} \\ 0_{1\times(j-1)} \end{bmatrix} = G_{j-1}^{(j)} G_{j-2}^{(j)} \dots G_1^{(j)} \underline{T_{j-1}} = Q_j \underline{T_{j-1}}$$

where the Givens rotation matrix $G_i^{(j)} \in \mathbb{R}^{j\times j}$ zeroes the $(i+1, i)$-entry of the tridiagonal matrix it is applied to. Also, we have

$$G_i^{(j+1)} = \begin{bmatrix} G_i^{(j)} & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix} \text{ for } i = 1, \dots, j-1.$$

Minimal residual (MINRES) method, cont'd$_5$

As we had for GMRES, we have that $\underline{R_j}$ can be formed through minimal update of $\underline{R_{j-1}}$, i.e.,

$$\underline{R_j} = \begin{bmatrix} \underline{R_{j-1}} & G_j^{(j+1)}[1:j,1:j+1]\begin{bmatrix} G_{j-1}^{(j)} \dots G_1^{(j)} t_{1:j,j} \\ \beta_j \end{bmatrix} \\ 0_{1\times(j-1)} & 0 \end{bmatrix}.$$

Therefore, while performing the j-th iteration of MINRES, one is equipped with $\underline{R_{j-1}}$ and $\underline{T_j}$. In order to assemble $\underline{R_j}$, there only remains to apply the Givens rotations $G_1^{(j+1)}, \dots, G_j^{(j+1)}$ to the last column of $\underline{T_j}$, i.e.,

$$\underline{R_j}[1:j+1,j] = G_j^{(j+1)} \dots G_1^{(j+1)} t_{1:j+1,j}.$$

But, since $t_{1:j-2,j} = 0_{(j-2)\times 1}$, this simplifies to

$$\boxed{\underline{R_j}[1:j+1,j] = G_j^{(j+1)} G_{j-1}^{(j+1)} G_{j-2}^{(j+1)} t_{1:j+1,j} \text{ when } j>2}.$$

Minimal residual (MINRES) method, cont'd$_6$

- We recall that the MINRES iterate is given by $x_j := x_0 + V_j \tilde{y}$, where

$$R_j \tilde{y} = \underline{g_j}[1:j],$$

so that, for $j = 1, \dots, m$, we have $x_j = x_0 + P_j \underline{g_j}[1:j]$, in which $P_j = [p_1, \dots, p_j] := V_j R_j^{-1}$. But since R_j has a bandwidth of 3, we get

$$p_1 = v_1/\tau_1^{(1)}, \; p_2 = (v_2 - \tau_1^{(2)} p_1)/\tau_2^{(1)}$$
$$p_j = (v_j - \tau_{j-1}^{(2)} p_{j-1} - \tau_{j-2}^{(3)} p_{j-2})/\tau_j^{(1)} \quad \text{for} \quad j = 3, 4, \dots, m$$

so that the columns of P_j are an accessible by-product of the MINRES iteration. Finally, since $\underline{g_j}[1:j-1] = \underline{g_{j-1}}[1:j-1]$, we have

$$x_j = x_0 + P_j \underline{g_j}[1:j] = x_0 + [P_{j-1}\, p_j] \begin{bmatrix} \underline{g_j}[1:j-1] \\ \underline{g_j}[j] \end{bmatrix}$$
$$= x_0 + P_{j-1} \underline{g_{j-1}}[1:j-1] + \underline{g_j}[j] p_j$$

so that $\boxed{x_j = x_{j-1} + \underline{g_j}[j] p_j}$.

Minimal residual (MINRES) method, cont'd$_7$

- In practice, the $\underline{R}_1, \dots, \underline{R}_m$ and $\underline{g}_1, \dots, \underline{g}_m$ can be computed in-place, stored in pre-allocated $\underline{T}_m$ and $\underline{g}_m$. This yields the following algorithm

Algorithm 6 MINRES: $(x_0, m, \varepsilon) \mapsto x_j$

// Allocate $\underline{T} \in \mathbb{R}^{(m+1)\times m}$, $\underline{g} \in \mathbb{R}^{m+1}$

$r_0 := b - Ax_0$; $\beta := \|r_0\|_2$; $v_1 := r_0/\beta$; $\underline{g} := [\beta, 0, \dots, 0]^T$

for $j = 1, 2 \dots$ **do**

    // Perform Lanczos iteration

    $w_j := Av_j - \beta_{j-1}v_{j-1}$ where $\beta_0 := 0$ and $v_0 := 0$

    $\alpha_j := (w_j, v_j)$; $w_j := w_j - \alpha_j v_j$; $\beta_j := \|w_j\|_2$

    // Apply $G_{j-2}^{(j+1)}$ to $t_{1:j+1,j}$.

    if $j > 2$ **then**

$$\begin{bmatrix} t_{j-2,j} \\ t_{j-1,j} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} t_{j-2,j} \\ t_{j-1,j} \end{bmatrix} \text{ where } \begin{cases} s := t_{j-1,j-2}/(t_{j-2,j-2}^2 + t_{j-1,j-2}^2)^{1/2} \\ c := t_{j-2,j-2}/(t_{j-2,j-2}^2 + t_{j-1,j-2}^2)^{1/2} \end{cases}$$

    // Apply $G_{j-1}^{(j+1)}$ to $t_{1:j+1,j}$.

    if $j > 1$ **then**

$$\begin{bmatrix} t_{j-1,j} \\ t_{jj} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} t_{j-1,j} \\ t_{jj} \end{bmatrix} \text{ where } \begin{cases} s := t_{j,j-1}/(t_{j-1,j-1}^2 + t_{j,j-1}^2)^{1/2} \\ c := t_{j-1,j-1}/(t_{j-1,j-1}^2 + t_{j,j-1}^2)^{1/2} \end{cases}$$

Minimal residual (MINRES) method, cont'd$_8$

- In practice, the $\underline{R_1}, \ldots, \underline{R_m}$ and $\underline{g_1}, \ldots, \underline{g_m}$ can be computed in-place, stored in pre-allocated $\underline{T_m}$ and $\underline{g_m}$. This yields the following algorithm

Algorithm 6 cont'd MINRES: $(x_0, m, \varepsilon) \mapsto x_j$

// Apply $G_j^{(j+1)}$ to $\underline{g}[1:j+1]$ and $t_{1:j+1,j}$

$$\begin{bmatrix} \underline{g}[j] \\ \underline{g}[j+1] \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \underline{g}[j] \\ 0 \end{bmatrix} \text{ where } \begin{cases} s := t_{j+1,j}/(t_{jj}^2 + t_{j+1,j}^2)^{1/2} \\ c := t_{jj}/(t_{jj}^2 + t_{j+1,j}^2)^{1/2} \end{cases}$$

$t_{jj} := ct_{jj} + st_{j+1,j}$; $t_{j+1,j} := 0$

$p_j := (v_j - \tau_{j-1}^{(2)} p_{j-1} - \tau_{j-2}^{(3)} p_{j-2})/\tau_j^{(1)}$ where $p_0 := 0$ and $p_{-1} := 0$

$x_j := x_{j-1} + \underline{g}[j] p_j$

if $|\underline{g}[j+1]| < \varepsilon \|b\|_2$ **then** Stop ▷ Stop if $\|r_j\|_2 < \varepsilon \|b\|_2$

$v_{j+1} := w_j / \beta_j$

SYMMLQ method

- The SYMMLQ method (Paige and Saunders, 1975) is an **orthogonal projection** in a Krylov subspace $\mathcal{K}_m(A, r_0)$ where A is **symmetric**, possibly **indefinite**. Thus, equivalently to the CG method, it sums up to

$$\boxed{\text{Find } x_m \in x_0 + \mathcal{K}_m(A, r_0) \text{ such that } b - Ax_m \perp \mathcal{K}_m(A, r_0)}.$$

Assuming that the columns of $V_m := [v_1, \ldots, v_m]$ form a basis of the Krylov subspace $\mathcal{K}_m(A, r_0)$, the SYMMLQ iterate is given by

$$x_m := x_0 + V_m T_m^{-1} V_m^T r_0$$

where $T_m := V_m^T A V_m$ is the tridiagonal matrix of a Lanczos procedure. The main difference with CG stems from the assumed factorization of T_m. While CG assumes that T_m admits an **LU** factorization **without pivoting** (not guaranteed to exist for an indefinite A), the SYMMLQ method relies on a **LQ** decomposition of T_m (guaranteed to exist for all non-singular A). That is, we search for the lower-triangular $\tilde{L}_m \in \mathbb{R}^{m \times m}$ and an orthogonal $Q_m \in \mathbb{R}^{m \times m}$ such that $T_m = \tilde{L}_m Q_m$.

Paige C. C. & Saunders M. A. (1975). Solution of sparse indefinite systems of linear equations. SIAM Journal on Numerical Analysis, 12, 617–629.

SYMMLQ method, cont'd$_1$

- Given an LQ decomposition of the tridiagonal matrix T_j, the SYMMLQ iterate can be recast into

$$x_j = x_0 + \tilde{P}_j \tilde{z}^{(j)} \text{ where } \tilde{L}_j \tilde{z}^{(j)} = \beta e_1^{(j)} \text{ and } \tilde{P}_j := V_j Q_j^T.$$

Since T_j is tridiagonal, it is also Hessenberg, and its LQ decomposition can be constructed through the application of **Givens rotations**:

$$\tilde{L}_j = T_j G_1^{(j)} \dots G_{j-1}^{(j)} \text{ so that } Q_j = \left(G_1^{(j)} \dots G_{j-1}^{(j)} \right)^T.$$

Since T_j is tridiagonal, $\tilde{L}_j$ is banded with a bandwidth of 3.

Let $\tilde{Q}_{j+1} := \begin{bmatrix} Q_j & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix}$. Then, we have

$$G_j^{(j+1)T} \tilde{Q}_{j+1} = G_j^{(j+1)T} \begin{bmatrix} G_{j-1}^{(j)}{}^T \dots G_1^{(j)T} & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix}$$

SYMMLQ method, cont'd$_2$

$$
\begin{aligned}
G_j^{(j+1)T}\tilde{Q}_{j+1} &= G_j^{(j+1)T}\begin{bmatrix} G_{j-1}^{(j)}{}^T & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix} \dots \begin{bmatrix} G_1^{(j)T} & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix} \\
&= G_j^{(j+1)T} G_{j-1}^{(j+1)T} \dots G_1^{(j+1)T}
\end{aligned}
$$

so that $G_j^{(j+1)T}\tilde{Q}_{j+1} = Q_{j+1}$. Then, we have

$$
\begin{aligned}
T_{j+1}Q_{j+1}^T &= T_{j+1}\tilde{Q}_{j+1}^T G_j^{(j+1)} \\
&= \begin{bmatrix} T_j & t_{1:j,j+1} \\ t_{j+1,1:j} & \alpha_{j+1} \end{bmatrix} \begin{bmatrix} Q_j^T & 0_{j\times 1} \\ 0_{1\times j} & 1 \end{bmatrix} G_j^{(j+1)} \\
&= \begin{bmatrix} T_j Q_j^T & t_{1:j,j+1} \\ t_{j+1,1:j}Q_j^T & \alpha_{j+1} \end{bmatrix} G_j^{(j+1)}
\end{aligned}
$$

where

$$
\begin{aligned}
t_{j+1,1:j}Q_j^T &= [0_{1\times(j-1)}\ \beta_j] G_1^{(j+1)} \dots G_{j-1}^{(j+1)} \\
&= [0_{1\times(j-1)}\ \beta_j] G_{j-1}^{(j+1)} \\
&= [0_{1\times(j-2)}\ \ -s_{j-1}\beta_j\ \ c_j\beta_j].
\end{aligned}
$$

SYMMLQ method, cont'd$_3$

We can see that the application of $G_j^{(j+1)}$ to the right of $T_{j+1}\tilde{Q}_{j+1}$:

- zeroes the only non-zero component over the diagonal in the last column of $T_{j+1}\tilde{Q}_{j+1}$;
- modifies the $(j+1, j)$-entry of $T_{j+1}\tilde{Q}_{j+1}$;
- modifies the (j, j)-entry of $(T_{j+1}Q_{j+1}^T)[1:j, 1:j] = T_jQ_j^T = \tilde{L}_j$.

Consequently, the components of $\tilde{L}_j$ can be denoted as follows:

$$\tilde{L}_j = \begin{bmatrix} \ell_1^{(1)} & 0 & \dots & \dots & \dots & 0 \\ \ell_2^{(2)} & \ell_2^{(1)} & \ddots & & & \vdots \\ \ell_3^{(3)} & \ell_3^{(2)} & \ell_3^{(1)} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ell_{j-1}^{(3)} & \ell_{j-1}^{(2)} & \ell_{j-1}^{(1)} & 0 \\ 0 & \dots & 0 & \ell_j^{(3)} & \ell_j^{(2)} & \tilde{\ell}_j^{(1)} \end{bmatrix}$$

where the $\tilde{}$ over $\tilde{L}_j$ marks the difference with $L_j := \tilde{L}_{j+1}[1:j, 1:j]$. That is, only the (j, j)-entry differ between $\tilde{L}_j$ and L_j.

SYMMLQ method, cont'd$_4$

- Let us introduce $z^{(j)} \in \mathbb{R}^j$ such that

$$L_j z^{(j)} = \beta e_1^{(j)},$$

which differs only in its last entry from $\tilde{z}^{(j)}$, which we previously introduced as the solution of $\tilde{L}_j \tilde{z}^{(j)} = \beta e_1^{(j)}$.
That is, we have

$$z^{(j)} = \begin{bmatrix} z^{(j-1)} \\ z_j \end{bmatrix} \text{ and } \tilde{z}^{(j)} = \begin{bmatrix} z^{(j-1)} \\ \tilde{z}_j \end{bmatrix}$$

where $z^{(j-1)}$ is the solution of $L_{j-1} z^{(j-1)} = \beta e_1^{(j-1)}$.
Given that L_j and $\tilde{L}_j$ are both lower-triangular and differ from each other only in their (j, j)-entry, we have

$$\tilde{z}_j = \ell_j^{(1)} z_j / \tilde{\ell}_j^{(1)}$$

where $\ell_j^{(1)} = L_j[j, j]$ and $\tilde{\ell}_j^{(1)} = \tilde{L}_j[j, j]$.

SYMMLQ method, cont'd$_5$

- It follows from $L_j z^{(j)} = \beta e_1^{(j)}$ that

$$\begin{cases} z_1 = \beta / \ell_1^{(1)}, \\ z_2 = -\ell_2^{(2)} z_1 / \ell_2^{(1)}, \\ z_j = -\left(\ell_j^{(3)} z_{j-2} + \ell_j^{(2)} z_{j-1}\right) / \ell_j^{(1)} \text{ for } j = 3, 4, \ldots, m. \end{cases}$$

Given $\tilde{P}_j = V_j Q_j^T$ and $\tilde{P}_{j+1} = V_{j+1} Q_{j+1}^T$, we introduce

$$P_{j-1} := \tilde{P}_j[1:n, 1:j-1] \text{ and } P_j := \tilde{P}_{j+1}[1:n, 1:j],$$

and we write $\tilde{P}_j = [P_{j-1}\ \tilde{p}_j]$ and $\tilde{P}_{j+1} = [P_j\ \tilde{p}_{j+1}]$. Then, we have

$$\tilde{P}_{j+1} = V_{j+1} Q_{j+1}^T = [V_j\ v_{j+1}] \begin{bmatrix} Q_j^T & 0_{j \times 1} \\ 0_{1 \times j} & 1 \end{bmatrix} G_j^{(j+1)} = [V_j Q_j^T\ v_{j+1}] G_j^{(j+1)}$$

so that

$$\tilde{P}_{j+1} = [\tilde{P}_j\ v_{j+1}] G_j^{(j+1)} = [P_{j-1}\ \tilde{p}_j\ v_{j+1}] G_j^{(j+1)}.$$

SYMMLQ method, cont'd$_6$

Therefore, we have

$$\tilde{P}_{j+1}[1:n, 1:j] = [P_{j-1} \; (c_j \tilde{p}_j - s_j v_{j+1})]$$

so that $P_j = [P_{j+1} \; p_j]$, where

$$\begin{cases} \tilde{p}_1 = v_1 \\ p_j = c_j \tilde{p}_j - s_j v_{j+1} \\ \tilde{p}_{j+1} = s_j \tilde{p}_j + c_j v_{j+1} \; \text{ for } \; j = 1, 2, \ldots, m. \end{cases}$$

- Consider the iterate given by $\tilde{x}_j := x_0 + P_j z^{(j)}$, then we have

$$\tilde{x}_j = x_0 + [P_{j-1} \; p_j] \begin{bmatrix} z^{(j-1)} \\ z_j \end{bmatrix} = x_0 + P_{j-1} z^{(j-1)} + z_j p_j = \tilde{x}_{j-1} + z_j p_j.$$

The new iterate $x_{j+1} := x_0 + \tilde{P}_{j+1} \tilde{z}^{(j+1)}$ can then be recast as follows:

$$x_j = x_0 + [P_j \; \tilde{p}_{j+1}] \begin{bmatrix} z^{(j)} \\ \tilde{z}_{j+1} \end{bmatrix} = x_0 + P_j z^{(j)} + \tilde{z}_{j+1} \tilde{p}_{j+1} = \tilde{x}_j + \tilde{z}_{j+1} \tilde{p}_{j+1}$$

so that x_{j+1} can be formed effciently from $\tilde{x}_j$.

SYMMLQ method, cont'd$_7$

- We recall that, as an orthogonal projection in the Krylov subspace range(V_j), the SYMMLQ iterate is equivalently given by

$$x_j = x_0 + V_j \tilde{y} \;\text{ where }\; T_j \tilde{y} = \beta e_1^{(j)}.$$

But since A, and thus T_j are symmetric, we have

$$\begin{aligned} T_j^T \tilde{y} &= \beta e_1^{(j)} \\ (\tilde{L}_j Q_j)^T \tilde{y} &= \beta e_1^{(j)} \\ Q_j^T \tilde{L}_j^T \tilde{y} &= \beta e_1^{(j)} \\ \tilde{L}_j^T \tilde{y} &= \beta Q_j e_1^{(j)}. \end{aligned}$$

By comparing the last entries on both sides of $\tilde{L}_j^T \tilde{y} = \beta Q_j e_1^{(j)}$, we have

$$\begin{aligned} e_j^{(j)T} \tilde{L}_j^T \tilde{y} &= \beta e_j^{(j)T} Q_j e_1^{(j)} \\ \tilde{\ell}_j^{(1)} (e_j^{(j)} \tilde{y}) &= \beta e_j^{(j)T} (G_1^{(j)} \dots G_{j-1}^{(j)})^T e_1^{(j)} \\ &= \beta (G_1^{(j)} \dots G_{j-1}^{(j)} e_j^{(j)})^T e_1^{(j)} \end{aligned}$$

SYMMLQ method, cont'd$_8$

so that

$$\tilde{\ell}_j^{(1)}(e_j^{(j)}\tilde{y}) = \beta s_1 s_2 \dots s_{j-1}. \quad (4)$$

Also, by construction of $G_j^{(j+1)}$, it can be shown that $s_j\tilde{\ell}_j^{(1)} + c_j\beta_j = 0$.

Then, recalling the Lanczos relation, i.e., $AV_j = V_jT_j + \beta_j v_{j+1} e_j^{(j)T}$, the SYMMLQ residual $r_j := b - Ax_j$ is recast as follows:

$$r_j = r_0 - AV_j\tilde{y} = r_0 - (V_jT_j + \beta_j v_{j+1} e_j^{(j)T})\tilde{y} = \beta v_1 - V_jT_j\tilde{y} - \beta_j(e_j^{(j)T}\tilde{y})v_{j+1}$$

where $T_j\tilde{y} = \beta e_1^{(j)}$, so that

$$r_j = \beta v_1 - \beta V_j e_1^{(j)} - \beta_j(e_j^{(j)T}\tilde{y})v_{j+1} = -\beta_j(e_j^{(j)T}\tilde{y})v_{j+1}$$

in which we use Eq. (4) to obtain

$$r_j = -\left(\beta s_1 \dots s_{j-1}/\tilde{\ell}_j^{(1)}\right) v_{j+1} = \left(\beta s_1 \dots s_j/c_j\right) v_{j+1}.$$

SYMMLQ method, cont'd₉

Then, as we have

$$\|r_{j-1}\|_2 = |\beta s_1 \dots s_{j-1}/c_{j-1}| \text{ and } \|r_j\|_2 = |\beta s_1 \dots s_j/c_j|$$

so that

$$\|r_j\|_2 = \left|\frac{c_{j-1}s_j}{c_j}\right| \|r_{j-1}\|_2.$$

Thus, the convergence of SYMMLQ can be monitored without forming the iterate x_j, or even solve the tridiagonal system for $\tilde{y}$, neither forming r_j nor computing its vector norm.

SYMMLQ method, cont'd$_{10}$

- Now we are equipped to put the SYMMLQ algorithm together:

Algorithm 7 SYMMLQ: $(x_0, m, \varepsilon) \mapsto x_j$

// Allocate $\underline{T} \in \mathbb{R}^{(m+1)\times m}$, $\underline{g} \in \mathbb{R}^{m+1}$
$r_0 := b - Ax_0$; $\beta := \|r_0\|_2$; $v_1 := r_0/\beta$; $\underline{g} := [\beta, 0, \dots, 0]^T$; $\tilde{x}_0 := x_0$
for $j = 1, 2 \dots$ **do**
    // Perform Lanczos iteration
    $w_j := Av_j - \beta_{j-1}v_{j-1}$ where $\beta_0 := 0$ and $v_0 := 0$
    $\alpha_j := (w_j, v_j)$; $w_j := w_j - \alpha_j v_j$; $\beta_j := \|w_j\|_2$
    if $j = 1$ **then** $\tilde{\ell}_j^{(1)} := \alpha_j$
    // Apply $G_{j-2}^{(j)}$ to the last row of T_j
    if $j > 2$ **then** $\begin{bmatrix} \ell_j^{(3)} & \beta_{j-1} \end{bmatrix} := \begin{bmatrix} 0 & \beta_{j-1} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ where $\begin{cases} s := s_{j-2} \\ c := c_{j-2} \end{cases}$
    // Apply $G_{j-1}^{(j)}$ to the last 2 columns of $T_j\tilde{Q}_j$
    if $j > 1$ **then**
        $\ell_{j-1}^{(1)} := \sqrt{\left(\tilde{\ell}_{j-1}^{(1)}\right)^2 + \beta_{j-1}^2}$
        $\begin{bmatrix} \ell_j^{(2)} & \tilde{\ell}_j^{(1)} \end{bmatrix} := \begin{bmatrix} \beta_{j-1} & \alpha_j \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$ where $\begin{cases} s := s_{j-1} \\ c := c_{j-1} \end{cases}$

SYMMLQ method, cont'd$_{11}$

- Now we are equipped to put the SYMMLQ algorithm together:

Algorithm 7 cont'd SYMMLQ: $(x_0, m, \varepsilon) \mapsto x_j$

// Compute z_{j-1}
if $j = 2$ **then** $z_1 := \beta / \ell_1^{(1)}$
if $j = 3$ **then** $z_2 := -\ell_2^{(2)} z_1 / \ell_2^{(1)}$
if $j > 3$ **then** $z_{j-1} := -\left(\ell_{j-1}^{(3)} z_{j-3} + \ell_{j-1}^{(2)} z_{j-2}\right) / \ell_{j-1}^{(1)}$
if $j = 1$ **then** $\tilde{p}_1 := v_1$
if $j > 1$ **then**
  $p_{j-1} := c_{j-1}\tilde{p}_{j-1} - s_{j-1} v_j$
  $\tilde{p}_j := s_{j-1}\tilde{p}_{j-1} + c_{j-1} v_j$
  $\underline{g}[j] := \tilde{x}_{j-2} + z_{j-1} p_{j-1}$
  $\underline{g}[j] := (c_{j-2} s_{j-1} / c_{j-1}) \underline{g}[j-1]$ where $c_0 := 1$
  if $|\underline{g}[j]| > \varepsilon \|b\|_2$ **then**
    $x_{j-1} := \tilde{x}_{j-2} + \left(\ell_{j-1}^{(1)} / \tilde{\ell}_{j-1}^{(1)}\right) \tilde{p}_{j-1}$
    Stop

More methods for non-symmetric linear systems

Bi-orthogonalization process

- The **bi-orthogonalization** process is an extension of the Lanczos procedure to **non-symmetric matrices**.

  It is sometimes called the **two-sided Lanczos** procedure.

- This procedure generates a pair of **bi-orthogonal** bases in the columns of $V_j = [v_1, \ldots, v_j] \in \mathbb{R}^{n \times j}$ and $W_j = [w_1, \ldots, w_j] \in \mathbb{R}^{n \times j}$ for Krylov subspaces of A and A^T, respectively, i.e., that is, we have

$$\text{range}(V_j) = \mathcal{K}_j(A, r_0) \;\text{ and }\; \text{range}(W_j) = \mathcal{K}_j(A^T, \tilde{r}_0)$$

  such that $V_j^T W_j = W_j^T V_j = I_j$ where $\tilde{r}_0$ is an **auxiliary vector** used to generate the **left Krylov subspace** $\mathcal{K}_j(A^T, \tilde{r}_0)$ with $(r_0, \tilde{r}_0) \neq 0$.

- During the bi-orthogonalization process, instead of forming v_{j+1} by orthonormalizing Av_j against v_j and v_{j-1}, it is done by orthonormalizing against w_j and w_{j-1}.

  Simultaneously, w_{j+1} is obtained by orthonormalizing $A^T w_j$ against v_j and v_{j-1}.

Bi-orthogonalization process, cont'd$_1$

- The resulting procedure is given by the following algorithm:

Algorithm 8 Bi-Orthogonalization: $(r_0, \tilde{r}_0, m) \mapsto (V_m, W_m)$

```
// r_0 and r̃_0 must be such that (r,r̃_0) ≠ 0
β := ‖r_0‖_2; v_1 := r_0/β; w_1 := β r̃_0/(r̃_0, r_0); β_0 := 0; γ_0 := 0
for j = 1, 2, ..., m do
    v_{j+1} := A v_j − β_{j−1} v_{j−1} where v_0 := 0
    w_{j+1} := A^T w_j − γ_{j−1} w_{j−1} where w_0 := 0
    α_j := (v_j, w_{j+1})
    v_{j+1} := v_{j+1} − α_j v_j
    w_{j+1} := w_{j+1} − α_j w_j
    γ_j := sqrt(|(v_{j+1}, w_{j+1})|)
    β_j := (v_{j+1}, w_{j+1})/γ_j
    v_{j+1} := v_{j+1}/γ_j
    w_{j+1} := w_{j+1}/β_j
```

Bi-orthogonalization process, cont'd$_2$

- We obtain the following **three-term recurrences** from the last algorithm:

$$\boxed{\begin{cases} \gamma_j v_{j+1} = Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1}, \\ \beta_j w_{j+1} = A^T w_j - \alpha_j w_j - \gamma_{j-1} w_{j-1} \quad \text{for} \quad j = 2, \dots, m \end{cases}}.$$

- We can show that the bases stored in the columns of V_m and W_m are orthonormal.
  - For that, we first note that $(v_1, w_1) = (r_0/\beta, \beta\tilde{r}_0/(\tilde{r}_0, r_0)) = 1$.
  - Then, for $j = 1$, we have

$$\begin{aligned}
(v_{j+1}, w_{j+1}) &= (Av_1 - \alpha_1 v_1, A^T w_1 - \alpha_1 w_1)/(\beta_1\gamma_1) \\
&= \left((Av_1, A^T w_1) - \alpha_1(v_1, A^T w_1)\right)/(\beta_1\gamma_1) \\
&\quad - \left(\alpha_1(Av_1, w_1) - \alpha_1^2(v_1, w_1)\right)/(\beta_1\gamma_1) \\
&= \left((Av_1, A^T w_1) - \alpha_1^2 - \alpha_1(Av_1, w_1) + \alpha_1^2\right)/(\beta_1\gamma_1) \\
&= (Av_1, A^T w_1 - \alpha_1 w_1)/(\beta_1\gamma_1)
\end{aligned}$$

where $\beta_1 = (Av_1 - \alpha_1 v_1, A^T w_1 - \alpha_1 w_1)/\gamma_1 = (Av_1, A^T w_1 - \alpha_1 w_1)/\gamma_1$ so that $(v_{j+1}, w_{j+1}) = 1$.

Bi-orthogonalization process, cont'd$_3$

- For $j = 2, \dots, m$, we have

$$\begin{aligned}(v_{j+1}, w_{j+1}) &= (\gamma_j v_{j+1}, \beta_j w_{j+1})/(\gamma_j \beta_j)\\ &= (Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1}, A^T w_j - \alpha_j w_j - \gamma_{j-1} w_{j-1})/(\gamma_j \beta_j)\end{aligned}$$

  where $\beta_j = (Av_j - \alpha_j v_j - \beta_{j-1} v_{j-1}, A^T w_j - \alpha_j w_j - \gamma_{j-1} w_{j-1})/\gamma_j$ so that $(v_1, w_1) = \cdots = (v_{m+1}, w_{m+1}) = 1$.

- There remains to show $(v_i, w_j) = 0$ if $i \neq j$. Let us proceed by induction and show that, for an integer j with $2 \leq j \leq m+1$, we have

$$(v_i, w_j) = (v_j, w_i) = 0 \;\text{ for }\; i = 1, \dots, j-1 \tag{5}$$

- For $j = 2$, we have

$$\begin{aligned}(v_1, w_2) = (v_1, A^T w_1 - \alpha_1 w_1)/\beta_1 &= \left((v_1, A^T w_1) - \alpha_1 (v_1, w_1)\right)/\beta_1\\ &= (\alpha_1 - \alpha_1)/\beta_1 = 0\end{aligned}$$

  and

$$\begin{aligned}(v_2, w_1) = (Av_1 - \alpha_1 v_1, w_1)/\gamma_1 &= \left((Av_1, w_1) - \alpha_1 (v_1, w_1)\right)/\gamma_1\\ &= \left((v_1, A^T w_1) - \alpha_1\right)/\gamma_1 = (\alpha_1 - \alpha_1)/\gamma_1 = 0.\end{aligned}$$

Bi-orthogonalization process, cont'd$_4$

- Suppose that Eq: (5) holds for j, then we need to show that

$$(v_i, w_{j+1}) = (v_{j+1}, w_i) = 0 \;\text{ for }\; i = 1, \dots, j.$$

First, we have

$$\alpha_j = (v_j, A^T w_j - \gamma_{j-1} w_{j-1}) = (v_j, A^T w_j) - \gamma_{j-1}(v_j, w_{j-1}) = (v_j, A^T w_j).$$

We also have

$$\begin{aligned}
(v_j, w_{j+1}) &= (v_j, A^T w_j - \alpha_j w_j - \gamma_{j-1} w_{j-1})/\beta_j \\
&= \left((v_j, A^T w_j) - \alpha_j (v_j, w_j)\right)/\beta_j \\
&= (\alpha_j - \alpha_j)/\beta_j = 0.
\end{aligned}$$

as well as

$$\begin{aligned}
(v_{j-1}, w_{j+1}) &= (v_{j-1}, A^T w_j - \alpha_j w_j - \gamma_{j-1} w_{j-1})/\beta_j \\
&= \left((v_{j-1}, A^T w_j) - \gamma_{j-1}(v_{j-1}, w_{j-1})\right)/\beta_j \\
&= \left((A v_{j-1}, w_j) - \gamma_{j-1}\right)/\beta_j \\
&= \left((\gamma_{j-1} v_j + \alpha_{j-1} v_{j-1} + \beta_{j-2} v_{j-2}, w_j) - \gamma_{j-1}\right)/\beta_j \\
&= (\gamma_{j-1} - \gamma_{j-1})/\beta_j = 0.
\end{aligned}$$

Bi-orthogonalization process, cont'd$_5$

- and, for $i = 1, \dots, j-2$, we get

$$\begin{aligned}(v_i, w_{j+1}) &= (v_i, A^T w_j - \alpha_j w_j - \gamma_{j-1} w_{j-1})/\beta_j \\ &= (v_i, A^T w_j)/\beta_j \\ &= (Av_i, w_j)/\beta_j \\ &= (\gamma_i v_{i+1} + \alpha_i v_i + \beta_{i-1} v_{i-1}, w_j)/\beta_j = 0.\end{aligned}$$

- We have shown that $(v_i, w_{j+1}) = 0$ for $i = 1, \dots, j$.

  Similarly, we can show that $(v_{j+1}, w_i) = 0$ for $i = 1, \dots, j$, after what the bi-orthonormality of the bases is proven.

- In the case of the dot product, the stated orthonormality implies

$$\boxed{V_m^T W_m = W_m^T V_m = I_m}.$$

Bi-orthogonalization process, cont'd$_6$

- The three-term recurrence formulae can be cast into matrix form as follows:

$$\begin{aligned} AV_m &= V_{m+1}\underline{T_m} \\ &= V_m T_m + \gamma_m v_{m+1} e_m^{(m)T} \end{aligned}$$

$$\begin{aligned} A^T W_m &= W_{m+1}\underline{\tilde{T}_m^T} \\ &= W_m T_m^T + \beta_m w_{m+1} e_m^{(m)T} \end{aligned}$$

where the tridiagonal matrices $\underline{T_m} \in \mathbb{R}^{(m+1)\times m}$ and $\underline{\tilde{T}_m^T} \in \mathbb{R}^{(m+1)\times m}$ are given by

$$\underline{T_m} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \gamma_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \gamma_{m-1} & \alpha_m \\ & & & \gamma_m \end{bmatrix} \text{ and } \underline{\tilde{T}_m^T} = \begin{bmatrix} \alpha_1 & \gamma_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_{m-1} \\ & & \beta_{m-1} & \alpha_m \\ & & & \beta_m \end{bmatrix}$$

with $T_m := \underline{T_m}[1:m, 1:m] = \underline{\tilde{T}_m}[1:m, 1:m]$.

Bi-orthogonalization process, cont'd$_7$

- Combining the matrix form of the first three-term recurrence formula with the statement of bi-orthonormality, we obtain:

$$
\begin{aligned}
AV_m &= V_m T_m + \gamma_m v_{m+1} e_m^{(m)T} \\
W_m^T A V_m &= W_m^T V_m T_m + \gamma_m W_m^T v_{m+1} e_m^{(m)T} \\
W_m^T A V_m &= T_m
\end{aligned}
$$

  where, as for a regular Lanczos procedure, T_m is tridiagonal, although this time not symmetric.

- In general, neither $\{v_1, \ldots, v_m\}$ nor $\{w_1, \ldots, w_m\}$ are orthogonal by themselves, i.e., $V_m^T V_m \neq I_m$ and $W_m^T W_m \neq I_m$.
- The bi-orthogonalization procedure is similar to Arnoldi in that they both apply to non-symmetric matrices.

  The advantage of the bi-orthogonalization method is that relies on short recurrences, unlike Arnoldi, which requires full orthogonalization against all previously formed vectors.

Bi-conjugate gradient (BiCG) method

- The BiCG method (Lanczos, 1952; Fletcher, 1976) is an **oblique projection** method in a Krylov subspace $\mathcal{K}_m(A, r_0)$, with a left Krylov constraints subspace $\mathcal{L}_m := \mathcal{K}_m(A^T, \tilde{r}_0)$ and iterates given by

$$\text{Find } x_m \in x_0 + \mathcal{K}_m(A, r_0) \text{ such that } b - Ax_m \perp \mathcal{K}_m(A^T, \tilde{r}_0).$$

- From a two-sided Lanczos procedure, we get $V_m, W_m \in \mathbb{R}^{n \times m}$ such that

$$\text{range}(V_m) = \mathcal{K}_m(A, r_0) \;\text{ and }\; \text{range}(W_m) = \mathcal{K}_m(A^T, \tilde{r}_0)$$

so that $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ implies that there exists $\tilde{y} \in \mathbb{R}^m$ such that $x_m = x_0 + V_m\tilde{y}$. Along with the Petrov-Galerkin condition, this yields

$$W_m^T(b - A(x_0 + V_m\tilde{y})) = 0$$
$$W_m^T r_0 - W_m^T A V_m \tilde{y} = 0$$
$$\beta W_m^T v_1 - T_m\tilde{y} = 0$$

so that the bi-orthonormality of the bases implies $T_m\tilde{y} = \beta e_1^{(m)}$.

Lanczos, C. (1952). Solution of systems of linear equations by minimized iterations, Journal of Research of the National Bureau of Standards, 49, 33–53.

Fletcher, R. (1976). Conjugate gradient methods for indefinite systems, in "Proceeding of the Dundee Conference on Numerical Analysis 1975", G. A. Watson (Editor), Lecture Notes in Mathematics, Springer-Verlag, Berlin, 506, pp. 73–89.

Bi-conjugate gradient (BiCG) method, cont'd$_1$

- Analogously to the CG method, we can introduce an LU decomposition with no pivoting of the tridiagonal T_m to derive the BiCG iteration. This leads to the following algorithm:

Algorithm 9 BiCG: $(x_0, \varepsilon) \mapsto x_j$

```
r_0 := b − A x_0
Pick r̃_0 such that (r_0, r̃_0) ≠ 0                    ▷ E.g., r̃_0 := r_0
p_1 := r_0; p̃_1 := r̃_0
for j = 1, 2 ... do
    α_j := (r_{j−1}, r̃_{j−1}) / (A p_j, p̃_j)
    x_j := x_{j−1} + α_j p_j
    r_j := r_{j−1} − α_j A p_j
    if ‖r_j‖_2 < ε‖b‖_2 then Stop
    r̃_j := r̃_{j−1} − α_j A^T p̃_j
    β_j := (r_j, r̃_j) / (r_{j−1}, r̃_{j−1})
    p_{j+1} := r_j + β_j p_j
    p̃_{j+1} := r̃_j + β_j p̃_j
```

Clearly, if A is SPD and $\tilde{r}_0 = r_0$, then the BICG iterates are the same as those from CG.

Bi-conjugate gradient (BiCG) method, cont'd$_2$

- Six vectors need be allocated for a practical implementation:

Algorithm 10 Practical BiCG: $(x_0, \varepsilon) \mapsto x_j$

```
Allocate memory for x, p, p̃, w, r, r̃ ∈ ℝ^n
r := b − A x_0
Pick r̃ such that (r, r̃) ≠ 0                ▷ E.g., r̃ := r
p := r; r̃ := p̃
for j = 1, 2 ... do
    w := A p
    α := (r, r̃)/(w, p̃)
    β := 1/(r, r̃)
    x := x + α p
    r := r − α w
    if ‖r‖_2 < ε‖b‖_2 then Stop
    w := A^T p̃
    r̃ := r̃ − α w
    β := β · (r, r̃)
    p := r + β p
    p̃ := r̃ + β p̃
```

Bi-conjugate gradient (BiCG) method, cont'd$_3$

- In addition to $Ax = b$, a dual system

$$A^T\tilde{x} = \tilde{b}$$

  can be solved by BiCG iteration upon setting $\tilde{r}_0 := \tilde{b} - A^T\tilde{x}_0$ for some initial iterate $\tilde{x}_0$, in which the dual iterate, given by

$$\tilde{x}_j := \tilde{x}_{j-1} + \alpha_j \tilde{p}_j$$

  is such that

$$\tilde{x}_j \in \tilde{x}_0 + \mathcal{K}_j(A^T, \tilde{r}_0) \text{ with } \tilde{r}_j := \tilde{b} - A^T\tilde{x}_j \perp \mathcal{K}_j(A, r_0).$$

- Similarly as for CG, we assumed that T_j admits an LU decomposition *without pivoting*. However, for a general matrix A, this may not be true. We have also assumed that T_j is *not singular* which also is not guaranteed.
- Analogously to what we did for the CG method, one can show that the residuals and their duals are orthogonal, while the search directions and their duals are A-orthogonal. That is

$$(r_i, \tilde{r}_j) = 0 \text{ and } (Ap_i, \tilde{p}_j) = 0 \text{ for } i \neq j.$$

Quasi-minimal residual (QMR) method

- The BiCG method is notoriously unstable (Gutknecht & Strakoš, 2000) and it often displays irregular convergence behaviors, i.e., no monotone decrease of residual norm, unlike GMRES.
- The QMR method (Freund & Nachtigal, 1991) can be viewed as an extension of the GMRES method in the sense that it builds iterates as

$$\begin{aligned}&\text{Find } \ x_m \in x_0 + \mathcal{K}_m(A, r_0)\\ &\text{such that } \ \|r_m\|_2 := \|b - Ax_m\|_2 = \min_{x \in x_0 + \mathcal{K}_m(A, r_0)} \|b - Ax\|_2\end{aligned}$$

  with the important difference that the basis of $\mathcal{K}_m(A, r_0)$ is produced by bi-orthogonalization.
  For a given V_{m+1} such that range$(V_m) = \mathcal{K}_m(A, r_0)$, similarly as with GMRES, we have

$$r_m := b - Ax_m = r_0 - AV_m\tilde{y} = \beta v_1 - V_{m+1}\underline{T_m}\tilde{y} = V_{m+1}(\beta e_1^{m+1} - \underline{T_m}\tilde{y}).$$

Gutknecht, M. H. & Strakoš, Z. (2000). Accuracy of two three-term and three two-term recurrences for Krylov space solvers, SIAM Journal on Matrix Analysis and Applications, 22, 213–229.

Freund, R. W. & Nachtigal, N. M. (1991). QMR: A quasi-minimal residual method for non-Hermitian linear systems, SIAM Journal: Numer. Math. 60, pp. 315–339.

Quasi-minimal residual (QMR) method, cont'd$_1$

The main difference with a basis produced by Arnoldi is that V_{m+1} is *not orthogonal*. Thus, we are left with

$$\|r_m\|_2 = \|V_{m+1}(\beta e_1^{(m+1)} - \underline{T_m}\tilde{y})\|_2$$

Although we have

$$\|r_m\|_2 \leq \|V_{m+1}\|_2 \cdot \|\beta e_1^{(m+1)} - \underline{T_m}\tilde{y}\|_2$$

Like in GMRES, we still form the iterate by minimizing $\|\beta e_1^{(m+1)} - \underline{T_m}y\|_2$, which here, is referred to as the *quasi-residual norm*, hence the name of *quasi-minimal residual* method.

- Because of the tridiagonal structure of $\underline{T_m}$, minimizing the quasi-residual norm is a bit simpler than minimizing the residual norm in GMRES.

  In particular, updating the QR factorization of the tridiagonal requires only up to three applications of Givens rotations.

Quasi-minimal residual (QMR) method, cont'd$_2$

- The least-squares problem $\min_{y\in\mathbb{R}^j} \|\beta e_1^{(j+1)} - \underline{T_j} y\|_2$ is, once again, solved by making use of a QR decomposition of $\underline{T_j}$. We have

$$\underline{R_j} = \begin{bmatrix} R_j \\ 0_{1\times j}\end{bmatrix} = G_j^{(j+1)} \dots G_1^{(j+1)} \underline{T_j} = G_j^{(j+1)} G_{j-1}^{(j+1)} G_{j-2}^{(j+1)} \underline{T_j} = Q_{j+1}\underline{T_j}$$

and $\underline{g_j} := \beta Q_{j+1} e_1^{(j+1)}$, so that the least-squares problem is recast in a banded triangular linear system:

$$R_j \tilde{y} = \underline{g_j}[1:j]$$

where R_j has a bandwidth of three. R_j and $\underline{g_j}$ are updated as follows, with minimal effort, given R_{j-1} and $\underline{g_{j-1}}$:

$$\underline{R_j} = \begin{bmatrix} \underline{R_{j-1}} & G_j^{(j+1)}[1:j, 1:j+1] \begin{bmatrix} G_{j-1}^{(j)} G_{j-2}^{(j)} t_{1:j,j} \\ \beta_j \end{bmatrix} \\ 0_{1\times j-1} & 0 \end{bmatrix}$$

so that updating $\underline{R_j}$ boils down to computing

$$\underline{R_j}[1:j+1, j] = G_j^{(j+1)} G_{j-1}^{(j+1)} G_{j-2}^{(j+1)} t_{1:j+1,j}.$$

Quasi-minimal residual (QMR) method, cont'd$_3$

and $\underline{g_j}$ is updated as follows:

$$\underline{g_j} = \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_{j-1} \\ c_j\gamma_j \\ -s_j\gamma_j \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_j \end{bmatrix} := \underline{g_{j-1}}$$

with

$$s_j := \frac{t_{j+1,j}}{\sqrt{\left(t_{jj}^{(j-1)}\right)^2 + t_{j+1,j}^2}} \quad \text{and} \quad c_j := \frac{t_{j+1,j}^{(j-1)}}{\sqrt{\left(t_{jj}^{(j-1)}\right)^2 + t_{j+1,j}^2}}$$

in which $\underline{T_j}^{(j)} := \underline{R_j}$.

- Finally, given $R_j\tilde{y} = \underline{g_j}[1:j]$, we obtain

$$r_j = V_{j+1}(\beta e_1^{(j+1)} - \underline{T_j}\tilde{y}) = V_{j+1}\begin{bmatrix} 0_{j\times 1} \\ \underline{g_j}[j+1] \end{bmatrix} \quad \text{so that} \quad \|r_j\|_2 = |\underline{g_j}[j+1]|.$$

Quasi-minimal residual (QMR) method, cont'd$_4$

- Finally, the QMR iteration is given as follows:

Algorithm 11 QMR: $(x_0, \varepsilon) \mapsto x_j$

// Allocate $\underline{T} \in \mathbb{R}^{(m+1)\times m}$ and $\underline{g} \in \mathbb{R}^{m+1}$
$r_0 := b - Ax_0$; $\beta := \|r_0\|_2$; $\underline{g} := [\beta, 0, \dots, 0]^T$; $v_1 := r_0/\beta$
Pick $\tilde{r}_0$ such that $(r_0, \tilde{r}_0) \neq 0$ ▷ E.g., $\tilde{r}_0 := r_0$
$w_1 := \beta\tilde{r}_0/(r_0, \tilde{r}_0)$
for $j = 1, 2 \dots$ **do**
    Get v_{j+1} and $t_{1:j+1,j}$ from iteration of two-sided Lanczos
    // Apply $G_{j-2}^{(j+1)}$ to $t_{1:j+1,j}$.
    if $j > 2$ **then**

$$\begin{bmatrix} t_{j-2,j} \\ t_{j-1,j} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} t_{j-2,j} \\ t_{j-1,j} \end{bmatrix} \text{ where } \begin{cases} s := t_{j-1,j-2}/(t_{j-2,j-2}^2 + t_{j-1,j-2}^2)^{1/2} \\ c := t_{j-2,j-2}/(t_{j-2,j-2}^2 + t_{j-1,j-2}^2)^{1/2} \end{cases}$$

    // Apply $G_{j-1}^{(j+1)}$ to $t_{1:j+1,j}$.
    if $j > 1$ **then**

$$\begin{bmatrix} t_{j-1,j} \\ t_{jj} \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} t_{j-1,j} \\ t_{jj} \end{bmatrix} \text{ where } \begin{cases} s := t_{j,j-1}/(t_{j-1,j-1}^2 + t_{j,j-1}^2)^{1/2} \\ c := t_{j-1,j-1}/(t_{j-1,j-1}^2 + t_{j,j-1}^2)^{1/2} \end{cases}$$

Quasi-minimal residual (QMR) method, cont'd$_5$

- Finally, the QMR iteration is given as follows:

Algorithm 11 QMR: $(x_0, \varepsilon) \mapsto x_j$

// Apply $G_j^{(j+1)}$ to $\underline{g}[1:j+1]$ and $t_{1:j+1,j}$

$\begin{bmatrix} \underline{g}[j] \\ \underline{g}[j+1] \end{bmatrix} := \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} \underline{g}[j] \\ 0 \end{bmatrix}$ where $\begin{cases} s := t_{j+1,j}/(t_{jj}^2 + t_{j+1,j}^2)^{1/2} \\ c := t_{jj}/(t_{jj}^2 + t_{j+1,j}^2)^{1/2} \end{cases}$

$t_{jj} := c \cdot t_{jj} + s \cdot t_{j+1,j}$; $t_{j+1,j} := 0$

$p_j := (v_j - \tau_{j-1}^{(2)} p_{j-1} - \tau_{j-2}^{(3)} p_{j-2})/\tau_j^{(1)}$ where $p_0 := 0$ and $p_{-1} := 0$

$x_j := x_{j-1} + \underline{g}[j] p_j$

if $|\underline{g}[j+1]| < \varepsilon \|b\|_2$ **then** Stop ▷ Stop if $\|r_j\|_2 < \varepsilon \|b\|_2$

The QMR usually exhibits a much smoother convergence behavior than BiCG.

Transpose-free methods

Matrix polynomials

- Let $A \in \mathbb{R}^{n\times n}$, and consider the scalar **polynomial** of *degree* m given by

$$\begin{aligned} p_m : \mathbb{C} &\to \mathbb{C} \\ t &\mapsto a_0 + a_1 t + a_2 t^2 + \cdots + a_m t^m. \end{aligned}$$

That is, $a_m \neq 0$. An associated **matrix polynomial** is then given by

$$\begin{aligned} p_m : \mathbb{R}^{n\times n} &\to \mathbb{R}^{n\times n} \\ A &\mapsto a_0 I_n + a_1 A + a_2 A^2 + \cdots + a_m A^m. \end{aligned}$$

Theorem (Eigenvalues of matrix polynomials)

Let $p : \mathbb{C} \to \mathbb{C}$ be a scalar polynomial, and $\theta \in \mathbb{C}$ be an eigenvalue of $A \in \mathbb{R}^{n\times n}$ with an associated eigenvector $y \in \mathbb{C}^n$. Then, $p(\theta)$ is an eigenvalue of $p(A)$, and y is an associated eigenvector, i.e., $p(A)y = p(\theta)y$.

Theorem (Cayley-Hamilton theorem)

Let $P_A(t) := \det(A_n - tI_n)$ denote the (scalar) characteristic polynomial of $A \in \mathbb{R}^{n\times n}$, then $P_A(A) = 0_{n\times n}$.

Matrix polynomials, cont'd

- The Cayley-Hamilton theorem guarantees that, for any matrix $A \in \mathbb{R}^{n\times n}$, there is a polynomial p of degree no greater than n such that $p(A) = 0$.

  A polynomial whose value is zero at the matrix is called the **annihilating polynomial**.
- Since $p(A) = 0$ implies $\alpha p(A) = 0$ for all $\alpha \in \mathbb{C}$, we may always normalize a polynomial so that its highest-order term is 1. Such polynomials are called **monic polynomials**.

Theorem (Minimum polynomial of a matrix)

- *For a matrix $A \in \mathbb{R}^{n\times n}$, there exists a unique monic polynomial q_A of* ***minimum degree****, no greater than n, that annihilates the matrix A, i.e., $q_A(A) = 0_{n\times n}$.*
- *The unique monic polynomial q_A of minimum degree that annihilates the matrix A is called the* ***minimal polynomial*** *of A.*

- *Similar matrices* have the *same minimal polynomial*.

Krylov subspaces and matrix polynomials

- All Krylov subspace methods introduced for the solving of linear systems construct iterates of the form $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ where, we recall that

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, \ldots, A^{m-1}r_0\}$$

so that, for every such iterate x_m, there exists a polynomial p_{m-1} of degree $m-1$ such that

$$x_m = x_0 + p_{m-1}(A)r_0.$$

Moreover, for the residual associated to such iterates, we have

$$r_m := b - Ax_m = r_0 - Ap_{m-1}(A)r_0$$

so that there exists a polynomial of degree no greater than m, which we denote by φ_m, such that

$$r_m = \varphi_m(A)r_0.$$

We refer to φ_m as the **residual polynomial**.

Conjugate gradient squared (CGS) method

- While both the BiCG and QMR methods offer alternatives to solve non-symmetric linear systems on the basis of short-recurrence relations, they do both require to be able to compute $x \mapsto A^T x$.

  The CGS method (Sonneveld, 1989) was introduced as a means to to approximate the solution of non-symmetric linear systems, on the basis on short-recurrence relations, without the need to be able to evaluate $x \mapsto Ax$.
- The CGS method is derived from the perspective of BiCG iterates, that is,

$$x_j \in x_0 + \mathcal{K}_j(A, r_0) \text{ such that } r_j := b - Ax_j \perp \mathcal{K}_j(A^T, \tilde{r_0})$$

  for which we saw that, there exists a **residual polynomial** φ_j of degree no greater than j, and such that

$$r_j = \varphi_j(A) r_0.$$

  Without loss of generality, we assume $\varphi_j(0) = 1$.

Sonneveld, P. (1989). CGS: A fast Lanczos-type solver for nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 10 , 36–52.

Conjugate gradient squared (CGS) method, cont'd$_1$

- Furthermore, there exists another polynomial ψ_j of degree no greater than j such that the BiCG search direction p_{j+1} is given by
$$p_{j+1} = \psi_j(A) r_0.$$
- The BiCG dual vectors $\tilde{r}_j$ and $\tilde{p}_{j+1}$ being updated after the same schemes as those of the vectors r_j and p_{j+1}, respectively, except with A^T instead of A, we then have
$$\tilde{r}_j = \varphi(A^T)\tilde{r}_0 \text{ and } \tilde{p}_{j+1} = \psi_j(A^T)\tilde{r}_0 \text{ for } j = 1, 2, \ldots, m.$$
- The diagonal and super-diagonal components of the tridiagonal, α_j and β_j, respectively, formed by the BiCG iteration, can then be recast as follows:
$$\alpha_j = \frac{(r_{j-1}, \tilde{r}_{j-1})}{(Ap_j, \tilde{p}_j)} = \frac{(\varphi_{j-1}(A)r_0, \varphi_{j-1}(A^T)\tilde{r}_0)}{(A\psi_{j-1}(A)r_0, \psi_{j-1}(A^T)\tilde{r}_0)} = \frac{(\varphi^2_{j-1}(A)r_0, \tilde{r}_0)}{(A\psi^2_{j-1}(A)r_0, \tilde{r}_0)},$$
$$\beta_j = \frac{(r_j, \tilde{r}_j)}{(r_{j-1}, \tilde{r}_{j-1})} = \frac{(\varphi_j(A)r_0, \varphi_j(A^T)\tilde{r}_0)}{(\varphi_{j-1}(A)r_0, \varphi_{j-1}(A^T)\tilde{r}_0)} = \frac{(\varphi^2_j(A)r_0, \tilde{r}_0)}{(\varphi^2_{j-1}(A)r_0, \tilde{r}_0)}$$
which indicates that it is possible to compute x_{j+1} and r_{j+1} without any evaluation of $x \mapsto A^T x$.

Conjugate gradient squared (CGS) method, cont'd$_2$

- The problem we are left with is to find update formulae for

$$\boxed{\varphi_j^2(A)r_0} \text{ and } \boxed{\psi_j^2(A)r_0}.$$

- The update formula for the BiCG residual is recast into

$$\begin{aligned} r_j =& r_{j-1} - \alpha_j A p_j \\ \varphi_j(A)r_0 =& \varphi_{j-1}(A)r_0 - \alpha_j A \psi_{j-1}(A) r_0 \end{aligned}$$

which, as it holds irrespective of r_0, leads to

$$\varphi_j(A) = \varphi_{j-1}(A) - \alpha_j A \psi_{j-1}(A) \text{ where } \varphi_0(A) = \psi_0(A) = I_n. \quad (6)$$

Irrespective of the polynomial p, we have $Ap(A) = p(A)A$, so that

$$\varphi_j^2(A) = \varphi_{j-1}^2(A) + \alpha_j^2 A^2 \psi_{j-1}^2(A) - 2\alpha_j A \varphi_{j-1}(A)\psi_{j-1}(A). \quad (7)$$

- Similarly, from the update formula for the BiCG search direction, we get

$$\begin{aligned} p_{j+1} =& r_j + \beta_j p_j \\ \psi_j(A) r_0 =& \varphi_j(A) r_0 + \beta_j \psi_{j-1}(A) r_0 \\ \psi_j(A) =& \varphi_j(A) + \beta_j \psi_{j-1}(A) \quad (8) \end{aligned}$$

so that we obtain $\psi_j^2(A) = \varphi_j^2(A) + \beta_j^2 \psi_{j-1}^2(A) + 2\beta_j \varphi_j(A)\psi_{j-1}(A)$. (9)

Conjugate gradient squared (CGS) method, cont'd$_3$

- The cross-term of Eq. (7) is developed as follows using Eq. (8):

$$\begin{aligned}\varphi_{j-1}(A)\psi_{j-1}(A) &= \varphi_{j-1}(A)(\varphi_{j-1}(A) + \beta_{j-1}\psi_{j-2}(A)) \\ &= \varphi_{j-1}^2(A) + \beta_{j-1}\varphi_{j-1}(A)\psi_{j-2}(A). \end{aligned} \tag{10}$$

  Using Eqs. (6) and (8), we get the following expression for the cross-term of Eq. (9):

$$\begin{aligned}\varphi_{j}(A)\psi_{j-1}(A) &= (\varphi_{j-1}(A) - \alpha_j A\psi_{j-1}(A))\psi_{j-1}(A) \\ &= \varphi_{j-1}(A)\psi_{j-1}(A) - \alpha_j A\psi_{j-1}^2(A) \\ &= \varphi_{j-1}(A)(\varphi_{j-1}(A) + \beta_{j-1}\psi_{j-2}(A)) - \alpha_j A\psi_{j-1}^2(A) \\ &= \varphi_{j-1}^2(A) + \beta_{j-1}\varphi_{j-1}(A)\psi_{j-2}(A) - \alpha_j A\psi_{j-1}^2(A) \end{aligned} \tag{11}$$

  where $\beta_0 := 0$.

Conjugate gradient squared (CGS) method, cont'd$_4$

- We are now equipped to develop the update formulae of $\varphi_j^2(A)$ and $\psi_j^2(A)$:
  - First, using Eq. (6), $\phi_0(A) = \psi_0(A) = I_n$ and Eq. (8), we obtain:

$$\begin{cases} \varphi_1^2(A) = (\varphi_0(A) - \alpha_1 A\psi_0(A))^2 = (I_n - \alpha_1 A)^2 \\ \varphi_1(A)\psi_0(A) = \varphi_1(A) = \varphi_0(A) - \alpha_1 A\psi_0(A) = I_n - \alpha_1 A \\ \psi_1^2(A) = (\varphi_1(A) + \beta_1\psi_0(A))^2 = (\varphi_1(A) + \beta_1 I_n)^2 \end{cases}.$$

  - Then using Eqs. (7) with Eq. (10), Eq. (11), and Eq. (9), respectively, for $j = 2, 3, \ldots, m$, we get:

$$\begin{cases} \varphi_j^2(A) = \varphi_{j-1}^2(A) + \alpha_j^2 A^2 \psi_{j-1}^2(A) \\ \qquad\qquad -2\alpha_j A\left(\varphi_{j-1}^2(A) + \beta_{j-1}\varphi_{j-1}(A)\psi_{j-2}(A)\right) \\ \varphi_j(A)\psi_{j-1}(A) = \varphi_{j-1}^2(A) + \beta_{j-1}\varphi_{j-1}(A)\psi_{j-2}(A) - \alpha_j A\psi_{j-1}^2(A) \\ \psi_j^2(A) = \varphi_j^2(A) + \beta_j^2\psi_{j-1}^2(A) + 2\beta_j\varphi_j(A)\psi_{j-1}(A) \end{cases}.$$

Conjugate gradient squared (CGS) method, cont'd$_5$

- Let us define

$$\hat{r}_j := \varphi_j^2(A)r_0, \;\; \hat{p}_{j+1} := \psi_j^2(A)r_0 \;\text{ and }\; \hat{q}_j := \varphi_j(A)\psi_{j-1}(A)r_0.$$

Using the update formulae from the last slide, we get

$$\begin{aligned}
\hat{r}_j =& \varphi_{j-1}^2(A)r_0 + \alpha_j^2 A^2 \psi_{j-1}^2(A)r_0 \\
& - 2\alpha_j A \left(\varphi_{j-1}^2(A) + \beta_{j-1}\varphi_{j-1}(A)\psi_{j-2}(A)\right) r_0 \\
=& \hat{r}_{j-1} + \alpha_j^2 A^2 \hat{p}_j - 2\alpha_j A \left(\hat{r}_{j-1} + \beta_{j-1}\hat{p}_{j-1}\right) \\
=& \hat{r}_{j-1} + \alpha_j A \left(\alpha_j A \hat{p}_j - 2\hat{r}_{j-1} - 2\beta_{j-1}\hat{p}_{j-1}\right).
\end{aligned}$$

As well as,

$$\begin{aligned}
\hat{q}_j =& \varphi_j(A)\psi_{j-1}(A)r_0 \\
=& \varphi_{j-1}^2(A)r_0 + \beta_{j-1}\varphi_{j-1}(A)\psi_{j-2}(A)r_0 - \alpha_j A \psi_{j-1}^2(A)r_0 \\
=& \hat{r}_{j-1} + \beta_{j-1}\hat{q}_{j-1} - \alpha_j A \hat{p}_j.
\end{aligned}$$

and

$$\begin{aligned}
\hat{p}_{j+1} =& \varphi_j^2(A)r_0 + \beta_j^2 \psi_{j-1}^2(A)r_0 + 2\beta_j\varphi_j(A)\psi_{j-1}(A)r_0 \\
=& \hat{r}_j + \beta_j^2 \hat{p}_j + 2\beta_j \hat{q}_j.
\end{aligned}$$

Conjugate gradient squared (CGS) method, cont'd$_6$

- Still using the update formulae for $\varphi_j^2(A)$ and $\psi_j^2(A)$, we get:

$$\alpha_j = \frac{(\varphi_{j-1}^2(A)r_0, \tilde{r}_0)}{(A\psi_{j-1}^2(A)r_0, \tilde{r}_0)} = \frac{(\hat{r}_{j-1}, \tilde{r}_0)}{(A\hat{p}_j, \tilde{r}_0)}$$

  as well as

$$\beta_j = \frac{(\varphi_j^2(A)r_0, \tilde{r}_0)}{(\varphi_{j-1}^2(A)r_0, \tilde{r}_0)} = \frac{(\hat{r}_j, \tilde{r}_0)}{(\hat{r}_{j-1}, \tilde{r}_0)}.$$

- For the sake of brevity, let $u_j := \hat{r}_j + \beta_j \hat{q}_j$, so that we have:

$$\begin{cases} \hat{q}_j = u_{j-1} - \alpha_j A\hat{p}_j, \\ \hat{r}_j = \hat{r}_{j-1} + \alpha_j A(\alpha_j A\hat{p}_j - 2u_{j-1}) \\ \quad = \hat{r}_{j-1} + \alpha_j A(u_{j-1} - \hat{q}_j - 2u_{j-1}) \\ \quad = \hat{r}_{j-1} - \alpha_j A(\hat{q}_j + u_{j-1}), \\ \hat{p}_{j+1} = u_j + \beta_j^2 \hat{p}_j + \beta_j \hat{q}_j. \end{cases}$$

Conjugate gradient squared (CGS) method, cont'd$_7$

- If the BiCG method converges, then $\|r_j\|_2 = \|\varphi_j(A)r_0\|_2$ tends to zero. Then, one might expect that $\|\hat{r}_j\|_2 = \|\varphi_j^2(A)r_0\|_2$ tends faster to zero. Hence, in an attempt to accelerate convergence, the CGS iterate x_j is defined so as to yield

$$b - Ax_j = \hat{r}_j.$$

Given our update formula for $\hat{r}_j$, we get:

$$\begin{aligned}
b - Ax_j &= \hat{r}_{j-1} - \alpha_j A(\hat{q}_j + u_{j-1}) \\
Ax_j &= b - \hat{r}_{j-1} + \alpha_j A(\hat{q}_j + u_{j-1}) \\
Ax_j &= b - (b - Ax_{j-1}) + \alpha_j A(\hat{q}_j + u_{j-1}) \\
Ax_j &= Ax_{j-1} + \alpha_j A(\hat{q}_j + u_{j-1})
\end{aligned}$$

so that

$$\boxed{x_j = x_{j-1} + \alpha_j(\hat{q}_j + u_{j-1})}.$$

Conjugate gradient squared (CGS) method, cont'd$_8$

- Eventually, we obtain the following algorithm:

Algorithm 12 CGS: $(x_0, \varepsilon) \mapsto x_j$

$r_0 := b - Ax_0$
Pick $\tilde{r}_0$ such that $(r_0, \tilde{r}_0) \neq 0$ ▷ E.g., $\tilde{r}_0 = r_0$
$\hat{p}_1 := r_0$; $\hat{r}_0 := r_0$; $u_0 := r_0$
for $j = 1, 2 \ldots$ **do**
    $\alpha_j := (\hat{r}_{j-1}, \tilde{r}_0)/(A\hat{p}_j, \tilde{r}_0)$
    $\hat{q}_j := u_{j-1} - \alpha_j A\hat{p}_j$
    $x_j := x_{j-1} + \alpha_j(\hat{q}_j + u_{j-1})$
    $\hat{r}_j := \hat{r}_{j-1} - \alpha_j A(\hat{q}_j + u_{j-1})$
    if $\|\hat{r}_j\|_2 < \varepsilon\|b\|_2$ **then** Stop
    $\beta_j := (\hat{r}_j, \tilde{r}_0)/(\hat{r}_{j-1}, \tilde{r}_0)$
    $u_j := \hat{r}_j + \beta_j\hat{q}_j$
    $\hat{p}_{j+1} := u_j + \beta_j^2\hat{p}_j + \beta_j\hat{q}_j$

- A CGS iteration entails two matrix-vector products, which is similar to BiCG, the difference being that CGS does not need to evaluate $x \mapsto A^T x$.
- When it converges, CGS often does so about twice as fast as BiCG.

Conjugate gradient squared (CGS) method, cont'd₉

- However, as the residual polynomial is squared, i.e., $\hat{r}_j = \varphi_j^2(A)r_0$ where $r_j = \varphi_j(A)r_0$, if the residual r_j increases in BiCG, then it does so even more significantly in CGS.

  As a result, CGS convergence curves can exhibit important oscillations, sometimes leading to numerical instability.

Bi-conjugate gradient stabilized (BiCGSTAB) method

- The CGS method, which is based on squaring the BiCG residual polynomial, i.e., $\hat{r}_j := \varphi_j^2(A)r_0$, is prone to substantial build-up of rounding error, possibly even overflow.
- The BiCGSTAB method (van der Vorst, 1992) is a variant of CGS developed to remedy unwanted oscillations, hence the name of BiCG stabilized.

  BiCGSTAB iterates are defined so as to yield a residual of the form

$$r_j = \phi_j(A)\varphi_j(A)r_0$$

  where φ_j is, still, the *residual polynomial of* the *BiCG* method, and ϕ_j is a *new j-th degree polynomial* introduced to remedy those potentially spurious oscillations, and defined as follows:

$$\phi_0(A) = I_n \text{ and } \phi_j(A) = (I_n - \omega_j A)\phi_{j-1}(A) \text{ for } j = 1, 2, \ldots$$

  where ω_j is chosen so as to minimize the residual norm.

van der Vorst, H. A. (1992). Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 13, 631–644.

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_1$

Then, the search direction is defined as

$$p_{j+1} = \phi_j(A)\psi_j(A)r_0 \;\text{ for }\; j = 1, 2, \dots$$

where the polynomial ψ_j is the *search direction polynomial of CGS*.
We thus have the following update formulae:

$$\begin{cases} \varphi_j(A) = \varphi_{j-1}(A) - \alpha_j A\psi_{j-1}(A) \\ \psi_j(A) = \varphi_j(A) + \beta_j\psi_{j-1}(A) \\ \phi_j(A) = (I_n - \omega_j A)\phi_{j-1}(A) \end{cases} \quad \text{for } j = 1, 2, \dots \tag{12}$$

where $\varphi_0(A) = \psi_0(A) = \phi_0(A) = I_n$.

- We can then develop the following update formula for the polynomial of the BiCGSTAB residual:

$$\begin{aligned} \phi_j(A)\varphi_j(A) &= (I_n - \omega_j A)\phi_{j-1}(A)\left(\varphi_{j-1}(A) - \alpha_j A\psi_{j-1}(A)\right) \\ &= (I_n - \omega_j A)\left(\phi_{j-1}(A)\varphi_{j-1}(A) - \alpha_j A\phi_{j-1}(A)\psi_{j-1}(A)\right). \end{aligned} \tag{13}$$

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_2$

- From $r_j = \phi_j(A)\varphi_j(A)r_0$, Eq. (13) and $p_{j+1} = \phi_j(A)\psi_j(A)r_0$, we get the following residual update formula:

$$\begin{aligned} r_j &= (I_n - \omega_j A)\left(\phi_{j-1}(A)\varphi_{j-1}(A) - \alpha_j A\phi_{j-1}(A)\psi_{j-1}(A)\right) r_0 \\ &= (I_n - \omega_j A)\left(\phi_{j-1}(A)\varphi_{j-1}(A)r_0 - \alpha_j A\phi_{j-1}(A)\psi_{j-1}(A)r_0\right) \\ &= (I_n - \omega_j A)\left(r_{j-1} - \alpha_j A p_j\right). \end{aligned}$$

- From $p_{j+1} = \phi_j(A)\psi_j(A)r_0$, $r_j = \phi_j(A)\varphi_j(A)r_0$ and Eq. (12), we get the following expression for the update of the search direction:

$$\begin{aligned} p_{j+1} &= \phi_j(A)\left(\varphi_j(A) + \beta_j\psi_{j-1}(A)\right) r_0 \\ &= \phi_j(A)\varphi_j(A)r_0 + \beta_j\phi_j(A)\psi_{j-1}(A)r_0 \\ &= r_j + \beta_j(I_n - \omega_j A)\phi_{j-1}(A)\psi_{j-1}(A)r_0 \\ &= r_j + \beta_j(I_n - \omega_j A)p_j. \end{aligned}$$

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_3$

- Similarly as for BiCG and CGS, we have

$$\alpha_j = \frac{(\varphi_{j-1}(A)r_0, \varphi_{j-1}(A^T)\tilde{r}_0)}{(A\psi_{j-1}(A)r_0, \psi_{j-1}(A^T)\tilde{r}_0)} \text{ and } \beta_j = \frac{(\varphi_j(A)r_0, \varphi_j(A^T)\tilde{r}_0)}{(\varphi_{j-1}(A)r_0, \varphi_{j-1}(A^T)\tilde{r}_0)}.$$

However, unlike with CGS, we do not intend to compute the squared polynomials $\varphi_j^2(A)$ and $\psi_j^2(A)$. We proceed as follows.

- First, from the update formulae for ϕ_j and ψ_j in Eq. (12), we have

$$\varphi_j(A^T) = -\alpha_j A^T \varphi_{j-1}(A^T) + \varphi_{j-1}(A^T) - \alpha_j \beta_{j-1} A^T \psi_{j-2}(A^T),$$

which implies that the highest-order term of $\varphi_j(A^T)$ is the same as that of $-\alpha_j A^T \varphi_{j-1}(A^T)$. Thus, proceeding by induction, we find that this term is

$$(-1)^j \alpha_j \alpha_{j-1} \cdots \alpha_1 (A^T)^j.$$

- Let us then restate the orthogonality of BiCG residuals with their duals as follows:

$$\big(\varphi_i(A)r_0, \varphi_j(A^T)\tilde{r}_0\big) = 0 \text{ for } i \neq j.$$

As this holds for all $j \neq i$, this implies $\big(\varphi_i(A)r_0, (A^T)^j\tilde{r}_0\big) = 0$ for $i \neq j$.

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_4$

As a result, the only term of $\varphi_j(A^T)$ which contributes to the non-zero part of $\left(\varphi_j(A)r_0, \varphi_j(A^T)\tilde{r}_0\right)$ is the highest-order one. Thus, we have:

$$\left(\varphi_j(A)r_0, \varphi_j(A^T)\tilde{r}_0\right) = (-1)^j \alpha_j \alpha_{j-1} \cdots \alpha_1 \left(\varphi_j(A)r_0, (A^T)^j \tilde{r}_0\right). \quad (14)$$

- Secondly, from the update formula of ϕ_j in Eq. (12), we have:

$$\phi_j(A^T) = (I_n - \omega_j A)\phi_{j-1}(A^T) = -\omega_j A^T \phi_{j-1}(A^T) + \phi_{j-1}(A^T),$$

  which indicates that the highest-order term of $\phi_j(A^T)$ is the same as that of $-\omega_j A^T \phi_{j-1}(A^T)$. Thus, by induction again, we get that this term is

$$(-1)^j \omega_j \omega_{j-1} \cdots \omega_1 (A^T)^j.$$

- As we have previously stated that $\left(\varphi_i(A)r_0, (A^T)^j \tilde{r}_0\right) = 0$ for all $i \neq j$, we have that the only term of $\phi_j(A^T)$ which contributes to the non-zero part of $\left(\varphi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right)$ is the highest-order one. Therefore, we have:

$$\left(\varphi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right) = (-1)^j \omega_j \omega_{j-1} \cdots \omega_1 \left(\varphi_j(A)r_0, (A^T)^j \tilde{r}_0\right). \quad (15)$$

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_5$

- Now, by combining Eqs. (14) and (15), we obtain

$$\left(\varphi_j(A)r_0, \varphi_j(A^T)\tilde{r}_0\right) = \frac{\alpha_j\alpha_{j-1}\cdots\alpha_1}{\omega_j\omega_{j-1}\cdots\omega_1}\left(\varphi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right). \tag{16}$$

Consequently, using Eq. (16), the formula for the β_j can be recast as follows:

$$\begin{aligned}
\beta_j &= \frac{(\varphi_j(A)r_0, \varphi_j(A^T)\tilde{r}_0)}{(\varphi_{j-1}(A)r_0, \varphi_{j-1}(A^T)\tilde{r}_0)} \\
&= \frac{\alpha_j}{\omega_j} \cdot \frac{\left(\varphi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right)}{(\varphi_{j-1}(A)r_0, \phi_{j-1}(A^T)\tilde{r}_0)} \\
&= \frac{\alpha_j}{\omega_j} \cdot \frac{(\phi_j(A)\varphi_j(A)r_0, \tilde{r}_0)}{(\phi_{j-1}(A)\varphi_{j-1}(A)r_0, \tilde{r}_0)} \\
&= \frac{\alpha_j}{\omega_j} \cdot \frac{(r_j, \tilde{r}_0)}{(r_{j-1}, \tilde{r}_0)}.
\end{aligned}$$

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_6$

- In order to find an adequate formula for α_j, we now work on simplifying

$$\left(A\psi_{j-1}(A)r_0, \psi_{j-1}(A^T)\tilde{r}_0\right).$$

From the update formula of ψ_j given in Eq. (12), we get:

$$\psi_j(A^T) = \varphi_j(A^T) + \beta_j\psi_{j-1}(A^T),$$

which indicates that the highest-order term of $\psi_j(A^T)$ is the same as that of $\varphi_j(A^T)$. We recall this term is

$$(-1)^j\alpha_j\alpha_{j-1}\cdots\alpha_1(A^T)^j.$$

- We then restate the A-orthogonality of BiCG search directions with their duals as follows:

$$\left(A\psi_i(A)r_0, \psi_j(A^T)\tilde{r}_0\right) = 0 \;\text{ for }\; i \neq j.$$

As this holds for all $j \neq i$, this implies $\left(A\psi_i(A)r_0, (A^T)^j\tilde{r}_0\right) = 0$ for $i \neq j$.

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_7$

Therefore, the only term of $\psi_j(A^T)$ which contributes to the non-zero part of $\left(A\psi_j(A)r_0, \psi_j(A^T)\tilde{r}_0\right)$ is the highest order. Thus, we have:

$$\left(A\psi_j(A)r_0, \psi_j(A^T)\tilde{r}_0\right) = (-1)\alpha_j\alpha_{j-1}\cdots\alpha_1\left(A\psi_j(A)r_0, (A^T)^j\tilde{r}_0\right). \quad (17)$$

- Analogously, we can show that

$$\left(A\psi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right) = (-1)\omega_j\omega_{j-1}\cdots\omega_1\left(A\psi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right). \quad (18)$$

- Then, upon combining Eqs. (17) and (18), we obtain:

$$\left(A\psi_j(A)r_0, \psi_j(A^T)\tilde{r}_0\right) = \frac{\alpha_j\alpha_{j-1}\cdots\alpha_1}{\omega_j\omega_{j-1}\cdots\omega_1}\left(A\psi_j(A)r_0, \phi_j(A^T)\tilde{r}_0\right). \quad (19)$$

- Finally, an update formula for α_j is obtained as follows by combining Eqs. (14), (15) and (19):

$$\begin{aligned}\alpha_j &= \frac{(\varphi_{j-1}(A)r_0, \varphi_{j-1}(A^T)\tilde{r}_0)}{(A\psi_{j-1}(A)r_0, \psi_{j-1}(A^T)\tilde{r}_0)} \\ &= \frac{(\varphi_{j-1}(A)r_0, \phi_{j-1}(A^T)\tilde{r}_0)}{(A\psi_{j-1}(A)r_0, \phi_{j-1}(A^T)\tilde{r}_0)}\end{aligned}$$

so that

$$\alpha_j = \frac{(\phi_{j-1}(A)\varphi_{j-1}(A)r_0, \tilde{r}_0)}{(A\phi_{j-1}(A)\psi_{j-1}(A)r_0, \tilde{r}_0)} = \frac{(r_{j-1}, \tilde{r}_0)}{(Ap_{j-1}, \tilde{r}_0)}.$$

- In summary, we have obtained the following updating formula:

$$\begin{cases} r_j = (I_n - \omega_j A)(r_{j-1} - \alpha_j A p_j) \text{ where } \alpha_j = (r_{j-1}, \tilde{r}_0)/(Ap_j, \tilde{r}_0) \\ p_{j+1} = r_j + \beta_j (I_n - \omega_j A) p_j \text{ where } \beta_j = \alpha_j (r_j, \tilde{r}_0)/\left(\omega_j (r_{j-1}, \tilde{r}_0)\right) \end{cases}$$

for $j = 1, 2 \ldots$ where $p_1 := r_0$.

- Using the update formulae found for r_j and p_{j+1}, we can find the update formula of the BiCGSTAB iterate as follows:

$$\begin{aligned} b - Ax_j &= r_j \\ b - Ax_j &= (I_n - \omega_j A)(r_{j-1} - \alpha_j A p_j) \\ b - Ax_j &= r_{j-1} - \alpha_j A p_j - \omega_j A (r_{j-1} - \alpha_j A p_j) \\ Ax_j &= b - r_{j-1} + \alpha_j A p_j + \omega_j A (r_{j-1} - \alpha_j A p_j) \\ Ax_j &= b - (b - Ax_{j-1}) + \alpha_j A p_j + \omega_j A (r_{j-1} - \alpha_j A p_j) \end{aligned}$$

so that $\boxed{x_j = x_{j-1} + \alpha_j p_j + \omega_j (r_{j-1} - \alpha_j A p_j)}$.

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd[9]

- All what remains to do is to define ω_j. As previously mentioned, our goal is to pick ω_j so as to minimize the residual norm $\|r_j\|_2$, that is

$$\omega_j = \arg\min_{\omega\in\mathbb{R}} \|(I_n - \omega A)(r_{j-1} - \alpha_j A p_j)\|_2.$$

For this, let $q_j := r_{j-1} - \alpha_j A p_j$, so that we aim at finding

$$\min_{\omega\in\mathbb{R}} \|(I_n - \omega A) q_j\|_2$$

which yields

$$\omega_j = \frac{(q_j, Aq_j)}{(Aq_j, Aq_j)}.$$

Bi-conjugate gradient stabilized (BiCGSTAB) method, cont'd$_{10}$

- Eventually, BiCGSTAB iterations are given as follows:

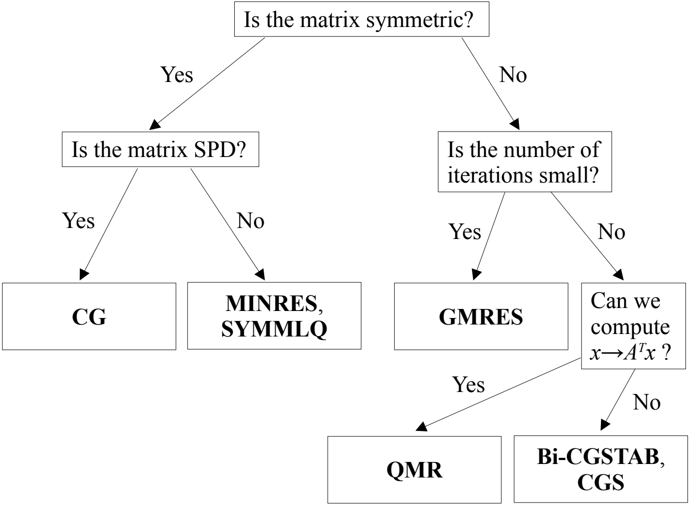
Algorithm 13 BiCGSTAB: $(x_0, \varepsilon) \mapsto x_j$

$r_0 := b - Ax_0$
Pick $\tilde{r}_0$ such that $(r_0, \tilde{r}_0) \neq 0$ ▷ E.g., $\tilde{r}_0 = r_0$
$p_1 := r_0$
for $j = 1, 2 \dots$ **do**
    $\alpha_j := (r_{j-1}, \tilde{r}_0)/(Ap_j, \tilde{r}_0)$
    $q_j := r_{j-1} - \alpha_j Ap_j$
    $\omega_j := (q_j, Aq_j)/(Aq_j, Aq_j)$
    $x_j := x_{j-1} + \alpha_j p_j + \omega_j q_j$
    $r_j := q_j - \omega_j Aq_j$
    if $\|\tilde{r}_j\|_2 < \varepsilon \|b\|_2$ **then** Stop
    $\beta_j := (\alpha_j/\omega_j) \cdot (r_j, \tilde{r}_0)/(r_{j-1}, \tilde{r}_0)$
    $p_{j+1} := r_j + \beta_j (p_j - \omega_j Ap_j)$

Summary

Flowchart of Krylov subspace-based linear iterative solvers

- The following flowchart can be used for practical solver selection:



Things we did not talk about

- Breakdowns.
- Convergence theories.
- Effects of finite precision.
- Preconditioning (Lecture 14).
- Restarting strategies (Lecture 15).
- Block variants for multiple simultaneously available right-hand sides.
- Communication-avoiding variants.

References

References

- Bai, Z. Z., & Pan, J. Y. (2021). Matrix analysis and computations. Society for Industrial and Applied Mathematics.
- Saad, Y. (2003). Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics.