Chapter 4

Krylov subspace methods

The iterative methods in the previous chapter are only using the knowledge of the previous iterate to build the next one. Instead, it seems preferable to include more directions to improve the approximation of the solution to the linear system $Ax_* = b$. This idea is formalised in the framework of the projection processes and in this setting, we will see that the Krylov subspace methods emerge as a natural candidate for these projection processes. The celebrated conjugate gradient algorithm and GMRES are two instances of projection processes based on Krylov subspaces.

4.1 Projection process

4.1.1 Definition and well-posedness of the projection process

Definition 4.1. *Let $\mathscr{C}_k$ and $\mathscr{S}_k$ be k-dimensional linear subspaces of $\mathbb{C}^n$, $A \in \mathbb{C}^{n\times n}$ and $x^{(0)} \in \mathbb{C}^n$. We say that $x^{(k)} \in \mathbb{C}^n$ is the result of a projection process if there exists $z^{(k)} \in \mathscr{S}_k$ such that*

$$\begin{cases} x^{(k)} = x^{(0)} + z^{(k)} \\ r^{(k)} = b - Ax^{(k)} \perp \mathscr{C}_k. \end{cases} \tag{4.1}$$

We say that the projection process is well-defined if $x^{(k)}$ exists and is uniquely defined.

We immediately notice that $r^{(k)} = r^{(0)} - Az^{(k)}$, hence the condition can also be phrased as $Az^{(k)} \perp r^{(0)} + \mathscr{C}_k$. The goal is to establish natural conditions on $\mathscr{C}_k$ and $\mathscr{S}_k$ under which the projection process is well-defined. We will call $\mathscr{S}_k$ the *search space* and $\mathscr{C}_k$ the *constraint space*.

Proposition 4.2. *Let $(c_1, \dots, c_k)$ (resp. $(s_1, \dots, s_k)$) be a basis of $\mathscr{C}_k$ (resp. $\mathscr{S}_k$) and let $C_k = [c_1; \dots; c_k]$ and $S_k = [s_1; \dots; s_k]$. The projection process is well-defined if and only if $C_k^* A S_k$ is invertible.*

Proof. By definition of the projection process, we can write $x^{(k)} = x^{(0)} + S_k t_k$ for a vector $t_k \in \mathbb{C}^k$. By the orthogonal constraint, we have $r^{(k)} \perp \mathscr{C}_k$, which means that $C_k^* r^{(k)} = 0$. But $r^{(k)} = r^{(0)} - AS_k t_k$, thus we find that t_k solves $C_k^* A S_k t_k = C_k^* r^{(0)}$. t_k is uniquely defined for all $r^{(0)}$ if and only if $C_k^* A S_k$ is invertible. □

Under the assumption that the projection process is well-defined, we can wonder whether there are conditions such that the norm of the residual $r^{(k)}$ is bounded by the norm of the initial one $\|r^{(0)}\|$. We have

$$\begin{aligned} r^{(k)} &= r^{(0)} - AS_k t_k \\ &= r^{(0)} - AS_k (C_k^* AS_k)^{-1} C_k^* r^{(0)} \\ &= \big(\mathrm{id} - AS_k (C_k^* AS_k)^{-1} C_k^*\big) r^{(0)}. \end{aligned}$$

By a simple calculation, we see that the operator $P_k = AS_k(C_k^*AS_k)^{-1}C_k^*$ is a projection. If we require $\|r^{(k)}\| \leq \|r^{(0)}\|$ for any $r^{(0)}$, then we need P_k to be an orthogonal projector. In that case, a possible choice is to take $C_k = AS_k$.

If the matrix A is Hermitian positive-definite, we can also investigate whether there is another choice by looking at the operator norm of the error $x^{(k)} - x_*$:

$$\begin{aligned} \|x^{(k)} - x_*\|_A &= \|A^{1/2}(x^{(k)} - x_*)\| \\ &= \|A^{1/2}(x^{(0)} + S_k t_k - x_*)\| \\ &= \|A^{1/2}(x^{(0)} + S_k (C_k^* AS_k)^{-1} C_k^* r^{(0)} - x_*)\| \\ &= \|A^{1/2}(x^{(0)} + S_k (C_k^* AS_k)^{-1} C_k^* (Ax_* - Ax^{(0)}) - x_*)\| \\ &= \big\|\big(\mathrm{id} - A^{1/2} S_k (C_k^* AS_k)^{-1} C_k^* A^{1/2}\big)(A^{1/2}x^{(0)} - A^{1/2}x_*)\big\|. \end{aligned}$$

The matrix $Q_k = A^{1/2}S_k(C_k^*AS_k)^{-1}C_k^*A^{1/2}$ is also a projection, which is not orthogonal in general. Again a natural choice to ensure that Q_k is an orthogonal projector is to set $C_k = S_k$.

In both cases, we will show that such choices for the constraint space lead to a well-defined projection process.

Proposition 4.3. *Suppose that A is invertible and let $\mathscr{S}_k$ be a k-dimensional subspace of $\mathbb{C}^n$. Then*

1. *if $\mathscr{C}_k = A\mathscr{S}_k$, then the projection process is well-defined;*
2. *if A is Hermitian positive-definite and $\mathscr{C}_k = \mathscr{S}_k$, then the projection process is well-defined.*

Proof. 1. By Proposition 4.2, it is enough to check that $C_k^*AS_k$ is invertible where $C_k = [c_1; \ldots; c_k]$ and $S_k = [s_1; \ldots; s_k]$, for some basis $(c_1, \ldots, c_k)$ (resp. $(s_1, \ldots, s_k)$) of $\mathscr{C}_k$ (resp. $\mathscr{S}_k$). Let $y \in \mathbb{C}^k$ such that $C_k^*AS_k y = 0$, then $AS_k y \perp \mathscr{C}_k = A\mathscr{S}_k$. Hence $AS_k y \in A\mathscr{S}_k \cap A\mathscr{S}_k^\perp$, hence $AS_k y = 0$, thus $y = 0$ since A is invertible and S_k is full-rank.

2. Again it suffices to check that $C_k^*AS_k$ is invertible. Let $y \in \mathbb{C}^k$ such that $S_k^*AS_k y = 0$, then $y^*S_k^*AS_k y = 0$, thus $\|S_k y\|_A = 0$. Since S_k is full-rank, $y = 0$.

□

4.1.2 Krylov subspace methods

Looking at the residual or the norm of the error gives a condition on the constraint space. It remains to see how to wisely pick the search space $\mathscr{S}_k$.

Proposition 4.4. *Suppose that the projection process (4.1) is well-defined. Assume that* $r^{(0)} \in \mathscr{S}_k$ *and* $A\mathscr{S}_k = \mathscr{S}_k$. *Then we have* $r^{(k)} = 0$.

This means that the projection process gives the exact solution to the linear system $Ax_* = b$, if $\mathscr{S}_k$ is a stable subspace under A.

Proof. By definition, we have $r^{(k)} = r^{(0)} - Az^{(k)}$, where $z^{(k)} \in \mathscr{S}_k$. Since $r^{(0)} \in \mathscr{S}_k$ and $A\mathscr{S}_k = \mathscr{S}_k$, $r^{(k)} \in A\mathscr{S}_k$. Thus there is $y \in \mathbb{C}^k$ such that $r^{(k)} = AS_k y$. By definition, $r^{(k)} \perp \mathscr{C}_k$, hence $C_k^* r^{(k)} = 0$, so $C_k^* AS_k y = 0$. Because the projection process is well-defined, this means that $C_k^* AS_k$ is invertible hence $y = 0$ and $r^{(k)} = 0$. □

From the previous result, it seems reasonable to start with $\mathscr{S}_1 = \mathrm{Span}(r^{(0)})$ and work with nested sequences spaces $\mathscr{S}_1 \subset \mathscr{S}_2 \subset \dots$. Since we want to find a stable subspace under A, it is natural to introduce the Krylov subspaces.

Definition 4.5. *Let* $v \in \mathbb{C}^n, A \in \mathbb{C}^{n\times n}$ *and* $k \in \mathbb{N}$. *We call* $\mathcal{K}_k(A, v) = \mathrm{Span}(v, Av, A^2v, \dots, A^{k-1}v)$ *the Krylov subspace.*

Proposition 4.6. *With the notation of Definition 4.5, the following assertions are true*

1. $\mathcal{K}_k(A, v) \subset \mathcal{K}_{k+1}(A, v)$;
2. *there is an integer* $d \in \mathbb{N}$ *such that*

$$\begin{cases} \mathcal{K}_{d+1}(A, v) = \mathcal{K}_d(A, v) \\ \mathcal{K}_{j-1}(A, v) \neq \mathcal{K}_j(A, v), \quad \forall 1 \leq j \leq d \end{cases}$$

   The integer d *is called the grade of* v *with respect to* A;

3. *for* $j \leq d$, *where* d *is the grade of* v *with respect to* A, $\dim \mathcal{K}_j(A, v) = j$;
4. *if* A *is invertible and* d *is the grade of* v *with respect to* A, *then* $A\mathcal{K}_d(A, v) = \mathcal{K}_d(A, v)$.

Thanks to the last property, the Krylov subspace $\mathcal{K}_k(A, r^{(0)})$ is a good candidate for the search space of the projection process.

Proof. The first three properties are clear. We have $A\mathcal{K}_d(A, v) \subset \mathcal{K}_{d+1}(A, v) = \mathcal{K}_d(A, v)$, so it suffices to show that $\dim A\mathcal{K}_d(A, v) = \dim \mathcal{K}_d(A, v)$ to have the equality. Let $(\alpha_i)_{1\leq i\leq d} \in \mathbb{C}^d$ such that $\sum_{i=1}^d \alpha_i A^i v = 0$. Since A is invertible, this means that $\sum_{i=0}^{d-1} \alpha_{i+1} A^i v = 0$. But $(A^i v)_{0\leq i\leq d-1}$ is a basis of $\mathcal{K}_d(A, v)$, thus $\alpha_i = 0$ for all $1 \leq i \leq d$. Hence $(A^i v)_{1\leq i\leq d}$ is a free family and $\dim A\mathcal{K}_d(A, v) = d = \dim \mathcal{K}_d(A, v)$. □

From this study, we have established that Krylov subspaces are good candidates for a search space for the projection process, as they guarantee a termination of the algorithm after a finite number of steps. For the constraint space, based on the norm of the residual or the error, we have identified two possible choices

1. $\mathscr{C}_k = A\mathscr{S}_k$;
2. $\mathscr{C}_k = \mathscr{S}_k$, if A is Hermitian positive-definite.

In both cases, we have already proved that the projection process is well-defined. Taking $\mathscr{S}_k = \mathcal{K}_k(A, r^{(0)})$, the first choice yields the GMRES algorithm and the second choice the conjugate gradient algorithm. We collect all these results and state the mathematical characterisation of both algorithms in the next theorem.

Theorem 4.7. *Let $A \in \mathbb{C}^{n\times n}$ be an invertible matrix, $b \in \mathbb{C}^n$ and $x_* \in \mathbb{C}^n$ be the solution to $Ax_* = b$. Let $x^{(0)} \in \mathbb{C}^n$ and $r^{(0)} = b - Ax^{(0)}$. Assume that $r^{(0)}$ has a grade $d \geq 1$ with respect to A. Let $x^{(k)}$ be defined by the projection process (4.1):*

$$\begin{cases} x^{(k)} & = x^{(0)} + z^{(k)} \text{ where } z^{(k)} \in \mathscr{S}_k \\ r^{(k)} & = b - Ax^{(k)} \perp \mathscr{C}_k. \end{cases}$$

1. *(Characterisation of the conjugate gradient algorithm) If A is Hermitian and positive-definite, $\mathscr{S}_k = \mathscr{C}_k = \mathcal{K}_k(A, r^{(0)})$ for all $1 \leq k \leq d$, then the projection process is well-defined for every $1 \leq k \leq d$ and $x^{(d)} = x_*$. Moreover we have the following characterisation for all iterates $x^{(k)}$, $1 \leq k \leq d$*

$$x^{(k)} - x_* \perp_A \mathcal{K}_k(A, r^{(0)}), \quad \text{and} \quad \|x^{(k)} - x_*\|_A = \min_{x\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|x - x_*\|_A. \tag{4.2}$$

2. *(Characterisation of GMRES) If $\mathscr{S}_k = \mathcal{K}_k(A, r^{(0)})$ and $\mathscr{C}_k = A\mathcal{K}_k(A, r^{(0)})$, then the projection process is well-defined for every $1 \leq k \leq d$ and $x^{(d)} = x_*$. Moreover we have the following characterisation for all residuals $r^{(k)}$, $1 \leq k \leq d$*

$$r^{(k)} \perp A\mathcal{K}_k(A, r^{(0)}), \quad \text{and} \quad \|r^{(k)}\| = \min_{x\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|b - Ax\|. \tag{4.3}$$

Proof. The well-posedness is given by Proposition 4.2 and the termination of the projection process is obtained by combining Proposition 4.4 and Proposition 4.6.

We now turn to the mathematical characterisations (4.2) and (4.3). These characterisations rely on rephrasing the orthogonalisation with respect to the constraint space $\mathscr{C}_k$ as an orthogonal projection with respect to a new scalar product.

1. by definition of the projection process, we have $r^{(k)} = b - Ax^{(k)} = A(x_* - x^{(k)}) \perp \mathscr{C}_k = \mathcal{K}_k(A, r^{(0)})$. Hence we have, using that A defines a scalar product

   $$x^{(k)} - x_* \perp A\mathcal{K}_k(A, r^{(0)}) \Leftrightarrow x^{(k)} - x_* \perp_A \mathcal{K}_k(A, r^{(0)}) \Leftrightarrow z^{(k)} + x^{(0)} - x_* \perp_A \mathcal{K}_k(A, r^{(0)}).$$

   Since $z^{(k)}$ belongs to the subspace $\mathscr{S}_k = \mathcal{K}_k(A, r^{(0)})$, we have that

   $$\|z^{(k)} + x^{(0)} - x_*\|_A = \min_{z\in\mathcal{K}_k(A,r^{(0)})} \|z + x^{(0)} - x_*\|_A,$$

   which is exactly Equation (4.2).

2. by definition of the projection process, we have $r^{(k)} = b - Ax^{(k)} \perp \mathscr{C}_k = A\mathcal{K}_k(A, r^{(0)})$. This is equivalent to $A^*A(x^{(k)} - x_*) \perp \mathcal{K}_k(A, r^{(0)})$. Since A^*A defines a scalar product, we have by the same reasoning as before

   $$\|x^{(k)} - x_*\|_{A^*A} = \min_{z\in\mathcal{K}_k(A,r^{(0)})} \|z + x^{(0)} - x_*\|_{A^*A}.$$

   Using that for any $y \in \mathbb{C}^n$, $\|y\|^2_{A^*A} = \langle y, A^*Ay\rangle = \|Ay\|^2$, we have (4.3).

□

The mathematical characterisations (4.2) and (4.3) will be central in establishing the practical algorithms and the convergence rates of both algorithms.

4.2 The conjugate gradient algorithm

The CG algorithm is an iterative method to solve $Ax_* = b$ when A is Hermitian positive-definite. This algorithm has several properties that make the algorithm efficient numerically:

- it has a short-term recurrence that makes it cheap to implement;
- it has a well-understood convergence behaviour based on the spectrum of the matrix A.

Such features are not shared with the GMRES algorithm as it will be exposed in Section 4.3.

In order to see that the CG algorithm has a short-term recurrence, it is natural to look at the Gram-Schmidt process to build an orthogonal basis to $\mathcal{K}_k(A, r^{(0)})$. This is the goal of the Arnoldi algorithm.

4.2.1 The Arnoldi algorithm

For this algorithm, we are not going to assume that A is Hermitian positive-definite. We simply require A to be invertible. The Arnoldi algorithm is simply a Gram-Schmidt process for $\mathcal{K}_k(A, v) = \mathrm{Span}(v, Av, \ldots, A^{k-1}v)$.

Algorithm 4.1 Arnoldi algorithm

function ARNOLDI(A, v, k)
  $v_1 = \frac{v}{\|v\|}$
  for $j = 1, \ldots, k$ **do**
    for $i = 1, \ldots, j$ **do**
      $h_{ij} = \langle v_i, Av_j \rangle$[1]
    end for
    $\widehat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
    $h_{j+1,j} = \|\widehat{v}_{j+1}\|$
    if $h_{j+1,j} \neq 0$ **then**
      $v_{j+1} = \frac{\widehat{v}_{j+1}}{h_{j+1,j}}$
    end if
  end for
  return $(v_1, \ldots, v_k)$
end function

The Arnoldi algorithm breaks down if in the course of the algorithm $h_{j+1,j} = 0$. As we are going to show, it does not happen if $j \leq d$ where d is the grade of v with respect to A.

Proposition 4.8. *Let $v \in \mathbb{C}^n$ be of grade d with respect to A. Then the following assertions are true*

1. *the Arnoldi algorithm 4.1 is well-posed for $k \leq d$ (*i.e. *$h_{j+1,j} \neq 0$ for $j \leq d-1$), moreover for all $j \leq d-1$, $(v_1, \ldots, v_j)$ is an orthonormal basis of $\mathcal{K}_j(A, v)$;*

[1] the convention used is $\langle x, y \rangle = \sum_{i=1}^{n} x_i^* y_i$

2. *let* $V_k = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix} \in \mathbb{C}^{n \times k}$ *and let* $H_{kk} = \begin{bmatrix} h_{11} & \dots & \dots & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & & \\ 0 & & h_{k,k-1} & h_{kk} \end{bmatrix} \in \mathbb{C}^{k \times k}$ *then*

$$AV_k = V_k H_{kk} + h_{k+1,k} v_{k+1} e_k^T, \tag{4.4}$$

and

$$V_k^* A V_k = H_{kk}; \tag{4.5}$$

3. *if A is Hermitian, then H_{kk} is tridiagonal with real entries.*

Remark 4.9. *Matrices of the form* $\begin{bmatrix} h_{11} & \dots & \dots & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & & \\ 0 & & h_{k,k-1} & h_{kk} \end{bmatrix} \in \mathbb{C}^{k \times k}$ *are called* upper Hessenberg.

Proof. 1. by definition of the grade of v, $(v, Av, \dots, A^{j-1}v)$ is a basis of the Krylov space $\mathcal{K}_j(A, v)$. Since $(v_1, \dots, v_j)$ are obtained from a Gram-Schmidt process of $(v, Av, \dots, A^{j-1}v)$

2. by definition of the algorithm, at each step $j \leq d$ we have

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i = V_{j+1} \begin{bmatrix} h_{1j} \\ \vdots \\ h_{j+1,j} \end{bmatrix}.$$

Thus

$$\begin{aligned} AV_k &= V_{k+1} \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1k} \\ h_{21} & h_{22} & \dots & \\ 0 & h_{32} & \dots & \vdots \\ \vdots & & \ddots & \\ 0 & & & h_{k+1,k} \end{bmatrix} \\ &= V_k H_{kk} + h_{k+1,k} v_{k+1} e_k^T. \end{aligned}$$

The second identity (4.5) follows from the orthogonality of $(v_j)_{1 \leq j \leq k+1}$.

3. we have $V_k^* A V_k = H_{kk}$. The matrix H_{kk} is upper Hessenberg and $V_k^* A V_k$ is Hermitian if A is Hermitian, hence H_{kk} is tridiagonal. It remains to show that the entries of H_{kk} are real. We have $h_{j+1,j} = \|\widehat{v}_{j+1}\|$ and $h_{jj} = \langle v_j, Av_j \rangle$, thus the entries are real.

□

The Arnoldi algorithm has a remarkable simplification when A is Hermitian. It is not necessary to reorthogonalise the vectors Av_j against $(v_i)_{1 \leq i \leq j-2}$, by the property above. The resulting algorithm is called the Hermitian Lanczos algorithm.

The three-term recurrence in the Hermitian Lanczos algorithm is the reason why the CG algorithm has also a short term recurrence.

Algorithm 4.2 Hermitian Lanczos algorithm

function HERMITIANLANCZOS(A, v, k)
  $v_1 = \frac{v}{\|v\|}$
  for $j = 1, \dots, k$ **do**
    $h_{jj} = \langle v_j, Av_j \rangle$
    $\widehat{v}_{j+1} = Av_j - h_{jj}v_j - h_{j-1,j}v_{j-1}$
    $h_{j+1,j} = \|\widehat{v}_{j+1}\|$
    if $h_{j+1,j} \neq 0$ **then**
      $v_{j+1} = \frac{\widehat{v}_{j+1}}{h_{j+1,j}}$
    end if
  end for
  return $(v_1, \dots, v_k)$
end function

4.2.2 The practical CG algorithm

From the Hermitian Lanczos algorithm, we will at first derive the three-term recurrence of the CG algorithm.

Let $(v_1, \dots, v_d)$ be the family of orthonormal vectors obtained by the Hermitian Lanczos algorithm applied to $\mathcal{K}_d(A, r^{(0)})$. We are going to exploit the tridiagonal structure of the matrix $T_k = V_k^* A V_k, k = 1, \dots, d$.

Lemma 4.10. *There exist* $(\mu_1, \dots, \mu_{d-1}) \in \mathbb{R}^{d-1}$ *and* $(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^d$ *such that for all* $1 \leq k \leq d$, *we have* $T_k = L_k \Lambda_k L_k^T$ *where* $L_k = \begin{bmatrix} 1 & & & \\ \mu_1 & 1 & & \\ & \ddots & \ddots & \\ & & \mu_{k-1} & 1 \end{bmatrix}$ *and* $\Lambda_k = \operatorname{diag}(\lambda_1, \dots, \lambda_k)$.

Proof. The matrix T_k is tridiagonal and positive-definite, so it has a unique LU factorisation $T_k = L_k U_k$. Since T_k is Hermitian and invertible, we can factorise the diagonal elements of U_k. Using the uniqueness of the LU factorisation, we have a unique factorisation of T_k of the form

$$T_k = L_k \Lambda_k L_k^T,$$

where $L_k = \begin{bmatrix} 1 & & & \\ \mu_1^{(k)} & 1 & & \\ & \ddots & \ddots & \\ & & \mu_{k-1}^{(k)} & 1 \end{bmatrix}$ and $\Lambda_k = \operatorname{diag}(\lambda_1^{(k)}, \dots, \lambda_k^{(k)})$. It remains to show that $(\lambda_j^{(k)})$ and $(\mu_j^{(k)})$ are independent of k. This is done by noticing that

$$\begin{aligned} T_{k+1} &= \left[\begin{array}{c|c} T_k & \alpha_k \\ \hline \alpha_k & \beta_{k+1} \end{array}\right] = \left[\begin{array}{c|c} L_k & \\ \hline \mu_k & 1 \end{array}\right] \left[\begin{array}{c|c} \Lambda_k & \\ \hline & \lambda_{k+1} \end{array}\right] \left[\begin{array}{c|c} L_k^T & \mu_k \\ \hline & 1 \end{array}\right] \\ &= \begin{bmatrix} L_k \Lambda_k L_k^T & \mu_k L_k \Lambda_k e_k \\ \mu_k e_k^T \Lambda_k L_k & \lambda_{k+1} \end{bmatrix}. \end{aligned}$$

By identification, the claim is proved. □

Proposition 4.11. *Let $r^{(0)}$ be of grade d with respect to A Hermitian positive-definite. Let $(v_1, \dots, v_d)$ be the vectors obtained by the Hermitian Lanczos algorithm. With the notation of Lemma 4.10, there are coefficients $(c_k)_{1\leq k\leq d}$ defined iteratively such that the CG iterates $(x^{(k)})_{1\leq k\leq d}$ and $(r^{(k)})_{1\leq k\leq d}$ are defined by*

$$\begin{cases} \hat{p}_k = v_{k+1} - \mu_k \hat{p}_{k-1} \\ x^{(k)} = x^{(k-1)} + c_k \hat{p}_{k-1} \\ r^{(k)} = r^{(k-1)} - c_k A \hat{p}_{k-1} \end{cases} \tag{4.6}$$

where $\hat{p}_{-1} = 0$ and $c_0^{(0)} = 0$.

Proof. By definition of the CG algorithm, we have

$$\begin{cases} x^{(k)} = x^{(0)} + z^{(k)}, \quad z^{(k)} \in \mathcal{K}_k(A, r^{(0)}) \\ r^{(k)} = b - Ax^{(k)} \perp \mathcal{K}_k(A, r^{(0)}). \end{cases} \tag{4.7}$$

We know that for each $k \leq d$, $(v_1, \dots, v_k)$ is a basis of $\mathcal{K}_k(A, r^{(0)})$ so $x^{(k)} = x^{(0)} + V_k t_k$, $t_k \in \mathbb{C}^k$ and $V_k = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix}$. Hence $r^{(k)} = r^{(0)} - AV_k t_k$. By orthogonality, we have $V_k^* r^{(k)} = 0$, thus $V_k^* r^{(0)} - V_k^* A V_k t_k = 0$ and $t_k = (V_k^* A V_k)^{-1} V_k^* r^{(0)}$. Plugging this in $x^{(k)}$ and using Lemma 4.10 yield

$$\begin{aligned} x^{(k)} &= x^{(0)} + V_k (V_k^* A V_k)^{-1} V_k^* r^{(0)} \\ &= x^{(0)} + V_k L_k^{-T} \Lambda_k^{-1} L_k^{-1} V_k^* r^{(0)}. \end{aligned}$$

Let $\widehat{P}_k = V_k L_k^{-T} = \begin{bmatrix} \hat{p}_0, \dots, \hat{p}_{k-1} \end{bmatrix}$. $\widehat{P}_k$ solves $\widehat{P}_k L_k^T = V_k$ so the columns of $\widehat{P}_k$ satisfies

$$\begin{bmatrix} \hat{p}_0, \dots, \hat{p}_{k-1} \end{bmatrix} \begin{bmatrix} 1 & \mu_1 & & \\ & \ddots & \ddots & \\ & & \ddots & \mu_{k-1} \\ & & & 1 \end{bmatrix} = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix}$$

$$\begin{bmatrix} \hat{p}_0, \hat{p}_1 + \mu_1 \hat{p}_0, \dots, \hat{p}_{k-1} + \mu_{k-1} \hat{p}_{k-2} \end{bmatrix} = \begin{bmatrix} v_1, \dots, v_k \end{bmatrix},$$

which is the first item in Equation (4.6).

We have

$$\begin{aligned} x^{(k)} &= x^{(0)} + V_k L_k^{-T} \Lambda_k^{-1} L_k^{-1} V_k^* r^{(0)} \\ &= x^{(0)} + \widehat{P}_k \Lambda_k^{-1} \widehat{P}_k^* r^{(0)} \\ &= x^{(0)} + \begin{bmatrix} \widehat{P}_{k-1}, & \hat{p}_{k-1} \end{bmatrix} \left[\begin{array}{c|c} \Lambda_{k-1}^{-1} & \\ \hline & \lambda_k^{-1} \end{array}\right] \begin{bmatrix} \widehat{P}_{k-1}^* \\ \hat{p}_{k-1}^* \end{bmatrix} r^{(0)} \\ &= x^{(0)} + \widehat{P}_{k-1} \Lambda_{k-1}^{-1} \widehat{P}_{k-1}^* r^{(0)} + \frac{\hat{p}_{k-1}^* r^{(0)}}{\lambda_k} \hat{p}_{k-1} \\ &= x^{(k-1)} + \frac{\hat{p}_{k-1}^* r^{(0)}}{\lambda_k} \hat{p}_{k-1}, \end{aligned}$$

which is the second item in (4.6) with $c_k = \frac{\hat{p}_{k-1}^* r^{(0)}}{\lambda_k}$.

For the last item, we use the definition of $r^{(k)}$ and the expression of $x^{(k)}$

$$r^{(k)} = b - Ax^{(k)} = r^{(k-1)} - c_k A\hat{p}_{k-1}.$$

□

Since the Arnoldi vectors $(v_j)_{1\leq j\leq d}$ can be generated using a three-term recurrence, the CG algorithm derived from Equation (4.6) can be rephrased in a three-term recurrence. There is however a way to get rid of the Arnoldi vectors and rewrite the CG algorithm into a two-term recurrence, which is the standard way to implement CG. This is based on the following observations on the vectors $\hat{p}_j$ and $r^{(j)}$.

Remark 4.12. *1. the vectors $(\hat{p}_j)_{0\leq j\leq d-1}$ define an A-orthogonal basis (*i.e. $\langle \hat{p}_i, A\hat{p}_j\rangle = 0$ *if $i \neq j$): we have*

$$\widehat{P}_d^* A \widehat{P}_d = L_d^{-1} V_d^* A V_d L_d^{-T} = \Lambda_d.$$

2. $\hat{p}_k \in \mathrm{Span}(r^{(k)}, \hat{p}_{k-1})$ for all $0 \leq k \leq d-1$. Since $\hat{p}_k = v_{k+1} - \mu_{k-1}\hat{p}_{k-1}$, it is sufficient to prove that $r^{(k)}$ and v_{k+1} are colinear:

$$\begin{aligned} r^{(k)} &= b - Ax^{(k)} = b - A(x^{(0)} + V_k(V_k^*AV_k)^{-1}V_k^* r^{(0)}) \\ &= r^{(0)} - AV_k(V_k^*AV_k)^{-1}V_k^* r^{(0)} \\ &= r^{(0)} - V_k V_k^* r^{(0)} - h_{k+1,k} v_{k+1} e_k^T (V_k^*AV_k)^{-1} V_k^* r^{(0)} \\ &= -h_{k+1,k} v_{k+1} e_k^T (V_k^*AV_k)^{-1} V_k^* r^{(0)}, \end{aligned}$$

where we have used that $AV_k = V_k(V_k^*AV_k) + h_{k+1,k}v_{k+1}e_k^T$ by Proposition 4.8 and $V_kV_k^*r^{(0)} = r^{(0)}$ since $V_kV_k^*$ is the orthogonal projection onto $\mathrm{Span}(v_1, \ldots, v_k)$ *and $v_1 = \frac{r^{(0)}}{\|r^{(0)}\|}$.*

3. we have $r^{(k)} \perp r^{(j)}$ for any $j < k$: by definition of $r^{(j)}$, we have $r^{(j)} = r^{(0)} - Az^{(j)}$ with $z^{(j)} \in \mathcal{K}_j(A, r^{(0)})$, thus $r^{(j)} \in \mathcal{K}_{j+1}(A, r^{(0)})$ but $r^{(k)} \perp \mathcal{K}_k(A, r^{(0)}) \supset \mathcal{K}_{j+1}(A, r^{(0)})$.

The idea is to generate the sequences $(x^{(k)}), (p_k), (r^{(k)})$ of the CG algorithm (p_k and $\hat{p}_k$ are colinear) starting with $p_0 = r^{(0)}$ and using that

- $p_k = r^{(k)} + \omega_k p_{k-1}$ where ω_k is chosen such that $p_k \perp_A p_{k-1}$;
- $r^{(k)} = r^{(k-1)} - \alpha_{k-1} A p_{k-1}$ and α_{k-1} is set such that $r^{(k)} \perp r^{(k-1)}$.

Compared to Equation (4.6), the choice of the constants α_{k-1} and ω_k has to be justified:

- α_{k-1} ensures that $r^{(k)} \perp r^{(k-1)}$ as by A-orthogonality of (p_j), we have

$$\langle r^{(k-1)}, Ap_{k-1}\rangle = \langle p_{k-1} - \omega_{k-1}p_{k-2}, Ap_{k-1}\rangle = \langle p_{k-1}, Ap_{k-1}\rangle.$$

- ω_k ensures that $p_k \perp_A p_{k-1}$

$$-\frac{\langle r^{(k)}, Ap_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle} = -\frac{\langle r^{(k)}, r^{(k-1)} - r^{(k)}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle}\frac{1}{\alpha_{k-1}} = \frac{\|r^{(k)}\|^2}{\|r^{(k-1)}\|^2} = \omega_k.$$

Algorithm 4.3 Conjugate-gradient algorithm

function CG($A, b, x^{(0)}, \varepsilon_{\text{tol}}$)
  $p_0 = r^{(0)} = b - Ax^{(0)}$, $k = 0$
  while $\|r^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    $\alpha_{k-1} = \frac{\|r^{(k-1)}\|^2}{\langle p_{k-1} A p_{k-1} \rangle}$
    $x^{(k)} = x^{(k-1)} + \alpha_{k-1} p_{k-1}$
    $r^{(k)} = r^{(k-1)} - \alpha_{k-1} A p_{k-1}$
    $\omega_k = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}$
    $p_k = r^{(k)} + \omega_k p_{k-1}$
  end while
  return $x^{(k)}$
end function

The cost of implementing the CG algorithm is a single matrix vector multiplication at each step, and the storage of the vectors $x^{(k)}, r^{(k)}$ and p_k.

The properties of the CG sequences $(x^{(k)})$, (p_k) and $(r^{(k)})$ which have been discussed above is summarised in the theorem below.

Theorem 4.13. *Let $A \in \mathbb{C}^{n\times n}$ a Hermitian, positive-definite matrix, $x^{(0)} \in \mathbb{C}^n$ such that $r^{(0)} = b - Ax^{(0)}$ is of grade d with respect to A.*

Then the sequence $(x^{(k)})$ defined by Algorithm 4.3 is the conjugate gradient algorithm characterised by $\|x^{(k)} - x_*\|_A = \min_{z\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|z - x_*\|_A$ where x_* is the solution to $Ax_* = b$.

The algorithm stops after d iterations with the exact solution: $x^{(d)} = x_*$. The family of residuals $(r^{(j)})_{0\le j\le k-1}$ defines an orthogonal basis of $\mathcal{K}_k(A, r^{(0)})$ for each $1 \le k \le d$ and $(p_j)_{0\le j\le k-1}$ is an A-orthogonal basis of $\mathcal{K}_k(A, r^{(0)})$ for each $1 \le k \le d$.

4.2.3 Convergence of the CG algorithm

Using the mathematical characterisation of the CG algorithm, we can estimate the speed of convergence of the CG algorithm.

Recall that $x^{(k)}$ is defined by $\|x^{(k)} - x_*\|_A = \min_{z\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|z - x_*\|_A$. Let $z \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})$, by definition of the Krylov space $\mathcal{K}_k(A, r^{(0)})$, we can write

$$z = x^{(0)} + \sum_{i=0}^{k-1} \zeta_i A^i r^{(0)}, \quad (\zeta_i)_{0\le i\le k-1} \in \mathbb{C}^k,$$

thus

$$z - x_* = x^{(0)} - x_* + \sum_{i=0}^{k-1} \zeta_i A^{i+1}(x^{(0)} - x_*) = \phi(A)(x^{(0)} - x_*),$$

where ϕ is a polynomial such that $\phi(0) = 1$ and $\deg \phi \le k$.

The minimisation problem becomes

$$\begin{aligned}\|x^{(k)} - x_*\|_A &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)(x^{(0)} - x_*)\|_A \\ &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|A^{1/2}\phi(A)(x^{(0)} - x_*)\| \\ &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)A^{1/2}(x^{(0)} - x_*)\| \\ &\leq \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)\|\|x^{(0)} - x_*\|_A.\end{aligned}$$

Since A is Hermitian and positive-definite, there is a unitary matrix U and a diagonal matrix Λ with positive entries such that $A = U\Lambda U^*$. The matrix norm of $\phi(A)$ is then $\|\phi(A)\| = \max_{1\leq i\leq n} |\phi(\lambda_i)|$. This gives a convergence result for the CG algorithm.

Theorem 4.14. *Let $x^{(k)}$ be the k-th iterate of the CG algorithm with A. Let $0 < \lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of A. Then we have*

$$\|x^{(k)} - x_*\|_A \leq \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\leq i\leq n} |\phi(\lambda_i)| \ \|x^{(0)} - x_*\|_A. \tag{4.8}$$

Remark 4.15. *If $k = n$, by picking ϕ as the Lagrange interpolation polynomial such that $\phi(0) = 1$ and $\phi(\lambda_i) = 0$ for all $1 \leq i \leq n$, we prove that CG stops after n iterations.*

It appears that it is convenient to relax $\max_{1\leq i\leq n} |\phi(\lambda_i)|$ to $\max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)|$ in order to explicit a convergence rate of the CG algorithm.

Corollary 4.16. *Let $x^{(k)}$ be the k-th iterate of the CG algorithm with A and $\mathrm{cond}_2(A)$ the condition number of A with respect to the 2-norm. Then we have*

$$\|x^{(k)} - x_*\|_A \leq 2\Big(\frac{\sqrt{\mathrm{cond}_2(A)} - 1}{\sqrt{\mathrm{cond}_2(A)} + 1}\Big)^k \ \|x^{(0)} - x_*\|_A. \tag{4.9}$$

Proof. We have $\|x^{(k)} - x_*\|_A \leq \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)| \ \|x^{(0)} - x_*\|_A$ and we use the fact that the min-max problem has an explicit solution given by the rescaled Chebyshev polynomial [2]

$$\chi_k(\lambda) = \frac{T_k\Big(\frac{\lambda_1+\lambda_n-2\lambda}{\lambda_n-\lambda_1}\Big)}{T_k\Big(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\Big)} = \frac{\cos\Big(k\arccos\Big(\frac{\lambda_1+\lambda_n-2\lambda}{\lambda_n-\lambda_1}\Big)\Big)}{T_k\Big(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\Big)}.$$

Then

$$\min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)| = \frac{1}{T_k\Big(\frac{\lambda_n+\lambda_1}{\lambda_n-\lambda_1}\Big)}.$$

[2] the Chebyshev polynomial of the first kind are defined by $T_k(\cos(\theta)) = \cos(k\theta)$, for $\theta \in [0, \pi]$.

Let $\kappa = \frac{\lambda_n}{\lambda_1} = \mathrm{cond}_2(A)$, then

$$\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1} = \frac{\kappa+1}{\kappa-1} = \frac{1}{2}\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} + \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big).$$

We invoke another property[3] of the Chebyshev polynomials T_k

$$T_k\Big(\frac{x+\frac{1}{x}}{2}\Big) = \frac{1}{2}(x^k + x^{-k}), \quad \forall x \in \mathbb{R}.$$

So we deduce

$$T_k\Big(\frac{\kappa+1}{\kappa-1}\Big) = \frac{1}{2}\Big(\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^k + \Big(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big)^k\Big) \geq \frac{1}{2}\Big(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big)^k.$$

Thus we obtain

$$\min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{\lambda_1\leq\lambda\leq\lambda_n} |\phi(\lambda)| \leq 2\Big(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\Big)^k.$$

□

Remark 4.17. *For the steepest gradient algorithm, we had*

$$\|x_{\mathrm{SG}}^{(k)} - x_*\|_A \leq \Big(\frac{\mathrm{cond}_2(A)-1}{\mathrm{cond}_2(A)+1}\Big)^k \|x^{(0)} - x_*\|_A.$$

Asymptotically, the convergence rate obtained for the CG algorithm is much better than the one for the steepest gradient, but still sensitive to an ill-conditioned matrix A.

Remark 4.18. *We have proved that for $0 < a < b$, we have*

$$T_m\Big(\frac{a+b}{b-a}\Big) \geq \frac{1}{2}\Big(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\Big),$$

with $\kappa = \frac{b}{a}$.

In the case where A has clustered eigenvalues $0 < \lambda_1 \leq \cdots \leq \lambda_{n-\ell} \ll \lambda_{n-\ell+1} \leq \ldots$, we can improve the previous estimate by considering another relaxation of the min-max problem. For $k \geq \ell$, we can choose $\phi \in \mathbb{C}^k[X], \phi(0) = 1$ as $\phi(\lambda) = q(\lambda)\widetilde{\phi}(\lambda)$, where $\widetilde{\phi}$ is a polynomial of degree at most $k-\ell$ with $\widetilde{\phi}(0) = 1$, and $q(\lambda) = \prod_{i=n-\ell+1}^{n}(1 - \frac{\lambda}{\lambda_i})$ *i.e.* the polynomial of degree ℓ such that $q(0) = 1$ and $q(\lambda_i) = 0$ for $n-\ell+1 \leq i \leq n$. Now using that $|q(\lambda)| \leq 1$ on $[0, \lambda_{n-\ell+1}]$, we have

$$\begin{aligned}
\min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\leq i\leq n} |\phi(\lambda_i)| &\leq \max_{1\leq i\leq n} |q(\lambda_i)| \min_{\substack{\widetilde{\phi}\in\mathbb{C}^{k-\ell}[X]\\ \widetilde{\phi}(0)=1}} \max_{1\leq i\leq n-\ell} |\widetilde{\phi}(\lambda_i)| \\
&\leq \min_{\substack{\widetilde{\phi}\in\mathbb{C}^{k-\ell}[X]\\ \widetilde{\phi}(0)=1}} \max_{\lambda_1\leq\lambda\leq\lambda_{n-\ell}} |\widetilde{\phi}(\lambda)| \\
&\leq 2\left(\frac{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}}-1}{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}}+1}\right)^{k-\ell}.
\end{aligned}$$

[3] by definition, the equation is true for $x = e^{i\theta}$, thus it extends to any complex number.

The corresponding convergence rate is then

$$\|x^{(k)} - x_*\|_A \leq 2 \left(\frac{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}} - 1}{\sqrt{\frac{\lambda_{n-\ell}}{\lambda_1}} + 1} \right)^{k-\ell} \|x^{(0)} - x_*\|_A. \tag{4.10}$$

As $\frac{\lambda_n}{\lambda_1} \gg \frac{\lambda_{n-\ell}}{\lambda_1}$, the previous estimate is much better than (4.9). This explains the good convergence properties of the CG algorithm in practice (see Figure 4.1).

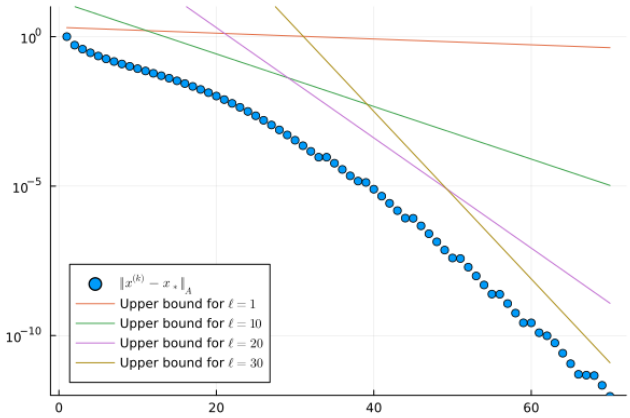

Figure 4.1: CG convergence rate compared to various upper bounds (4.10)

4.2.4 Preconditioned conjugate gradient algorithm

It is often advised to use a preconditioner to solve $Ax_* = b$ to reduce the number of iterations of the solver. A good preconditioner $M \in \mathbb{C}^{n\times n}$ is an invertible matrix such that $\mathrm{cond}_2(M^{-1}A) \ll \mathrm{cond}_2(A)$. Then solving $M^{-1}Ax_* = M^{-1}b$ is significantly easier than the original system. In our case, even if M is Hermitian, positive-definite, $M^{-1}A$ is in general not Hermitian. It is necessary to adapt the CG algorithm in order to incorporate the preconditioner. If we assume that M is Hermitian, positive-definite, we can write the Cholesky decomposition of $M = EE^*$, where $E \in \mathbb{C}^{n\times n}$ is a lower triangular matrix with positive entries. Instead of solving $M^{-1}Ax_* = M^{-1}b$, we can look at the symmetrised system

$$E^{-1}AE^{-*}\tilde{x}_* = E^{-1}b \tag{4.11}$$

Note that we have $x_* = E^{-*}\tilde{x}_*$. For the preconditioned linear system (4.11), the CG algorithm is the following (see Algorithm 4.4).

It is possible to simplify Algorithm 4.4 and get rid of the Cholesky matrices E and E^*. To do so, we are going to work with the variables $x^{(k)} = E^{-*}\tilde{x}^{(k)}$ and $r^{(k)} = E\tilde{r}^{(k)}$ and introduce a new variable $d_k = E^{-*}\tilde{p}_k$. Note that since $\tilde{r}^{(k)} = E^{-1}b - E^{-1}AE^{-*}\tilde{x}^{(k)}$, we have that $r^{(k)} = b - AE^{-*}\tilde{x}^{(k)} = b - Ax^{(k)}$.

We now reexpress the quantities appearing in the transformed CG algorithm 4.4 in the variables $x^{(k)}$, $r^{(k)}$ and d_k:

- $\|\tilde{r}^{(k)}\|^2 = \langle \tilde{r}^{(k)}, \tilde{r}^{(k)} \rangle = \langle E^{-1}r^{(k)}, E^{-1}r^{(k)} \rangle = \langle r^{(k)}, E^{-*}E^{-1}r^{(k)} \rangle = \langle r^{(k)}, M^{-1}r^{(k)} \rangle$

Algorithm 4.4 Transformed conjugate-gradient algorithm

function TCG($A, b, \tilde{x}^{(0)}, \varepsilon_{\text{tol}}, E$)
  $\tilde{p}_0 = \tilde{r}^{(0)} = E^{-1}b - E^{-1}AE^{-*}\tilde{x}^{(0)}$, $k = 0$
  while $\|\tilde{r}^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    $\alpha_{k-1} = \frac{\|\tilde{r}^{(k-1)}\|^2}{\langle \tilde{p}_{k-1}, E^{-1}AE^{-*}\tilde{p}_{k-1}\rangle}$
    $\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \alpha_{k-1}\tilde{p}_{k-1}$
    $\tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \alpha_{k-1}E^{-1}AE^{-*}\tilde{p}_{k-1}$
    $\omega_k = \frac{\|\tilde{r}_k\|^2}{\|\tilde{r}_{k-1}\|^2}$
    $\tilde{p}_k = \tilde{r}^{(k)} + \omega_k\tilde{p}_{k-1}$
  end while
  return $E^{-*}\tilde{x}^{(k)}$
end function

- $\tilde{x}^{(k)} = \tilde{x}^{(k-1)} + \alpha_{k-1}\tilde{p}_{k-1} \Leftrightarrow x^{(k)} = x^{(k-1)} + \alpha_{k-1}E^{-*}\tilde{p}_{k-1} = x^{(k-1)} + \alpha_{k-1}d_{k-1}$
- $\tilde{r}^{(k)} = \tilde{r}^{(k-1)} - \alpha_{k-1}E^{-1}AE^{-*}\tilde{p}_{k-1} \Leftrightarrow r^{(k)} = r^{(k-1)} - \alpha_{k-1}AE^{-*}\tilde{p}_{k-1} = r^{(k-1)} - \alpha_{k-1}Ad_{k-1}$
- $\tilde{p}_k = \tilde{r}^{(k)} + \omega_k\tilde{p}_{k-1} \Leftrightarrow d_k = E^{-*}\tilde{r}^{(k)} + \omega_k d_{k-1} = M^{-1}r^{(k)} + \omega_k d_{k-1}$.

It is thus possible to rewrite Algorithm 4.4 without E or E^*.

Algorithm 4.5 Preconditioned conjugate-gradient algorithm

function PCG($A, b, x^{(0)}, \varepsilon_{\text{tol}}, M$)
  $r^{(0)} = b - Ax^{(0)}$, $d_0 = M^{-1}r^{(0)}$, $k = 0$
  while $\|r^{(k)}\| > \varepsilon_{\text{tol}}$ **do**
    $k = k + 1$
    $\alpha_{k-1} = \frac{\langle r^{(k-1)}, M^{-1}r^{(k-1)}\rangle}{\langle d_{k-1}Ad_{k-1}\rangle}$
    $x^{(k)} = x^{(k-1)} + \alpha_{k-1}d_{k-1}$
    $r^{(k)} = r^{(k-1)} - \alpha_{k-1}Ad_{k-1}$
    $\omega_k = \frac{\langle r^{(k)}, M^{-1}r^{(k)}\rangle}{\langle r^{(k-1)}, M^{-1}r^{(k-1)}\rangle}$
    $d_k = M^{-1}r^{(k)} + \omega_k d_{k-1}$
  end while
  return $x^{(k)}$
end function

Compared to the CG algorithm, we need an additional linear solve of the system $My = r^{(k)}$ at each step of the preconditioned CG algorithm. Usually M has a simple structure (*i.e.* diagonal or block-diagonal) such that the linear solve is cheap compared to the total cost of the preconditioned CG algorithm.

Remark 4.19. *We can check that the iterates that are produced by the preconditioned CG algorithm satisfy:*

- $(r^{(k)})$ *are* M^{-1}*-orthogonal,* i.e. $\forall\, i \neq j, \langle r^{(i)}, M^{-1}r^{(j)}\rangle = 0$*;*
- (d_k) *are* A*-orthogonal,* i.e. $\forall\, i \neq j, \langle d_i, Ad_j\rangle = 0$.

4.2.5 Conjugate gradient algorithm in the XXI$^{\text{st}}$ century

The CG algorithm has become the reference method to solve linear problems with hermitian positive-definite matrices, as it combines all the advantages of a numerical method. The numerical convergence is fast, and is fully understood from a theoretical point of view. The algorithm is numerically stable, and requires minimal memory. Finally it is straightforward to use any preconditioner with CG.

4.3 GMRES

The generalised minimal residual (GMRES) algorithm is a popular iterative method to solve the linear system $Ax_* = b$ when A is invertible and non-Hermitian.

Contrary to the CG algorithm studied previously, GMRES does not have a short-term recurrence. This stems from the fact that we do not have the simplification of the Arnoldi algorithm for general matrices.

4.3.1 The mathematical characterisation and the minimisation problem

Recall that GMRES is mathematically characterised by

$$\begin{cases} x^{(k)} = x^{(0)} + z^{(k)}, \quad z^{(k)} \in \mathcal{K}_k(A, r^{(0)}) \\ r^{(k)} = b - Ax^{(k)} \perp A\mathcal{K}_k(A, r^{(0)}), \end{cases}$$

or equivalently

$$\|r^{(k)}\| = \min_{z \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})} \|b - Az\|. \tag{4.12}$$

Let $(v_1, \dots, v_k)$ be the Arnoldi vectors forming an orthonormal basis of $\mathcal{K}_k(A, r^{(0)})$ and satisfying

$$\begin{cases} AV_k = V_{k+1}\underline{H}_{kk} \\ v_1 = \dfrac{r^{(0)}}{\|r^{(0)}\|}, \end{cases}$$

with

$$V_k = [v_1, \dots, v_k], \quad \text{and} \quad \underline{H}_{kk} = \begin{bmatrix} h_{11} & \dots & & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & \ddots & \\ & & h_{k,k-1} & h_{kk} \\ & & & h_{k+1,k} \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}.$$

A vector $z \in x^{(0)} + \mathcal{K}_k(A, r^{(0)})$ can be expressed as $z = x^{(0)} + V_k t_k$, for some $t_k \in \mathbb{C}^k$. The minimisation problem (4.12) becomes

$$\begin{aligned} \|r^{(k)}\| &= \min_{t_k \in \mathbb{C}^k} \|r^{(0)} - AV_k t_k\| \\ &= \min_{t_k \in \mathbb{C}^k} \|\|r^{(0)}\| V_{k+1} e_1 - V_{k+1}\underline{H}_{kk} t_k\| \\ &= \min_{t_k \in \mathbb{C}^k} \|V_{k+1}(\|r^{(0)}\| e_1 - \underline{H}_{kk} t_k)\| \\ &= \min_{t_k \in \mathbb{C}^k} \|\|r^{(0)}\| e_1 - \underline{H}_{kk} t_k\|, \end{aligned} \tag{4.13}$$

where we have used that V_{k+1} has orthonormal columns. The last equation is a mean square minimisation problem. The standard way to solve such a problem is to write the so-called *QR factorisation* of $\underline{H}_{kk}$.

4.3.2 The QR factorisation

Theorem 4.20. *Let $H \in \mathbb{C}^{m\times n}$. Then there exist $Q \in \mathbb{C}^{m\times m}$ unitary (*i.e. $Q^*Q = QQ^* = \mathrm{id}_m$*) and $R \in \mathbb{C}^{m\times n}$ upper-triangular such that $H = QR$. Such a factorisation is called a QR factorisation of H.*

Proof. The theorem is proved by induction on the dimension n.

For $n = 1$, $H \in \mathbb{C}^{m\times 1}$, so we can pick $Q = [\frac{H}{\|H\|} \quad Q^\perp]$ where $Q^\perp$ has columns which are an orthonormal basis of $\{H\}^\perp$. Then $H = Q\begin{bmatrix}\|H\| \\ 0\end{bmatrix}$.

Suppose that for any $G \in \mathbb{C}^{m\times n}$, we can write its QR factorisation and let $H \in \mathbb{C}^{m\times(n+1)}$. Write $H = [H_1 \quad v]$, with $H_1 \in \mathbb{C}^{m\times n}$ and $v \in \mathbb{C}^m$. By the induction hypothesis, we have $H_1 = Q_1R_1$ where $Q_1 \in \mathbb{C}^{m\times m}$ is unitary and $R_1 \in \mathbb{C}^{m\times n}$ is upper-triangular. Let

$$w = Q_1^*v \in \mathbb{C}^m, \quad \text{and} \quad w_{n+1:m} = \begin{bmatrix} w_{n+1} \\ \vdots \\ w_m \end{bmatrix} = Q_2R_2,$$

where $w_{n+1:m} = Q_2R_2$ is a QR factorisation of $w_{n+1:m}$. Then setting $Q = Q_1\begin{bmatrix}\mathrm{id}_n & 0 \\ 0 & Q_2\end{bmatrix}$ and $R = \left[\begin{array}{c|c} R_1 & \begin{matrix} w_{1:n} \\ R_2 \end{matrix}\end{array}\right]$, we check that

$$QR = Q_1\begin{bmatrix}(R_1)_{1:n} & w_{1:n} \\ 0 & Q_2R_2\end{bmatrix} = Q_1\left[\begin{array}{c|c} R_1 & \begin{matrix} w_{1:n} \\ w_{n+1:m} \end{matrix}\end{array}\right] = \begin{bmatrix} Q_1R_1 & Q_1w \end{bmatrix} = H.$$

□

In general, the QR factorisation of a matrix H is not unique. We are going to give a few properties of the QR factorisation.

Proposition 4.21. *Let $H \in \mathbb{C}^{m\times n}$ be a full-rank matrix and $H = QR$ a QR factorisation of H. Denote by $(h_1, \dots, h_n)$ (resp. $(q_1, \dots, q_m)$) the columns of H (resp. of Q). Then for $1 \le k \le n$, we have*

$$\mathrm{Span}(h_1, \dots, h_k) = \mathrm{Span}(q_1, \dots, q_k), \quad \text{and } r_{kk} \neq 0.$$

Moreover if $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$, $Q_1 \in \mathbb{C}^{m\times n}$ and $Q_2 \in \mathbb{C}^{m\times(m-n)}$ and $R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$ with $R_1 \in \mathbb{C}^{n\times n}$, then $H = Q_1R_1$.

Proof. For the first part of the statement, we simply compare the columns of H and QR. For all $1 \le k \le n$, we have $h_k = \sum_{i=1}^k q_i r_{ik}$, thus we have $\mathrm{Span}(h_1, \dots, h_k) \subset \mathrm{Span}(q_1, \dots, q_k)$ but $(h_1, \dots, h_k)$ is a free family since H is full-rank. Hence we have $\mathrm{Span}(h_1, \dots, h_k) = \mathrm{Span}(q_1, \dots, q_k)$. If $r_{kk} = 0$, this would mean that $h_k \in \mathrm{Span}(q_1, \dots, q_{k-1}) = \mathrm{Span}(h_1, \dots, h_{k-1})$ which is in contradiction with H being full-rank.

The last statement is a consequence of R being upper-triangular. We have $R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$, and $H = Q_1 R_1$ follows. □

The last identity is called the thin QR decomposition of H and under the assumption that H is full-rank, we can show that there is a unique characterisation of the thin QR decomposition.

Theorem 4.22. *Let $H \in \mathbb{C}^{m\times n}$ be a full-rank matrix. Then there exists a unique $Q_1 \in \mathbb{C}^{m\times n}$ with orthonormal columns and $R_1 \in \mathbb{C}^{n\times n}$ upper triangular with positive real entries on the diagonal such that $H = Q_1 R_1$.*

Proof. The existence of Q_1 and R_1 is guaranteed by the proposition above. For the uniqueness, we compute $H^*H = R_1^* Q_1^* Q_1 R_1 = R_1^* R_1$. This is the Cholesky decomposition of H^*H which is unique. Q_1 is then given by $Q_1 = HR_1^{-1}$. □

Remark 4.23. *For the QR factorisation, R has the same shape as H but for the thin QR factorisation, it is Q which has the same shape as H. The full QR factorisation is rarely needed in practice, as it contains redundant information on the matrix H.*

4.3.3 The GMRES algorithm

We are now solving the minimisation (4.13) by using a QR factorisation of $\underline{H}_{kk} = Q_k R_k$, $Q_k \in \mathbb{C}^{(k+1)\times(k+1)}$ and $R_k \in \mathbb{C}^{(k+1)\times k}$. In this case, Equation (4.13) becomes

$$\min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|e_1 - \underline{H}_{kk} t_k\| = \min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|e_1 - Q_k R_k t_k\| = \min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|Q_k^* e_1 - R_k t_k\|.$$

R_k is upper triangular, so we have $R_k = \begin{bmatrix} \widetilde{R}_k \\ 0 \end{bmatrix}$ and denoting $\|r^{(0)}\|Q_k^* e_1 = \begin{bmatrix} g_k \\ \gamma_{k+1} \end{bmatrix}$ with $g_k \in \mathbb{C}^k, \gamma_{k+1} \in \mathbb{C}$, we see that $t_k \in \mathbb{C}^k$ solves $\widetilde{R}_k t_k = g_k$. Moreover we have

$$\|r^{(k)}\| = \min_{t_k\in\mathbb{C}^k} \|\|r^{(0)}\|e_1 - \underline{H}_{kk} t_k\| = |\gamma_{k+1}|. \tag{4.14}$$

It remains to implement efficiently a QR factorisation of $\underline{H}_{kk}$. To this end, we are going to use the fact that $\underline{H}_{kk}$ is an upper-Hessenberg matrix and that we can do a simple update of the QR factorisation of $\underline{H}_{k-1,k-1}$. Indeed we have

$$\underline{H}_{kk} = \begin{bmatrix} \underline{H}_{k-1,k-1} & h^{(k)} \\ 0 & h_{k+1,k} \end{bmatrix} = \begin{bmatrix} Q_{k-1}R_{k-1} & h^{(k)} \\ 0 & h_{k+1,k} \end{bmatrix},$$

where $h_{k+1,k} \in \mathbb{R}$ (see Proposition 4.8) and $h^{(k)} \in \mathbb{C}^k$. Consider Q_k defined by

$$Q_k = \begin{bmatrix} Q_{k-1} & 0 \\ 0 & 1 \end{bmatrix} \Omega_k, \tag{4.15}$$

where $\Omega_k \in \mathbb{C}^{(k+1)\times(k+1)}$ is some unitary matrix fixed later. Then we have

$$\Omega_k^* Q_k^* \underline{H}_{kk} = \Omega_k^* \begin{bmatrix} R_{k-1} & Q_{k-1}^* h^{(k)} \\ 0 & h_{k+1,k} \end{bmatrix}.$$

We simply need Ω_k^* to cancel the $(k+1,k)$ entry of the above matrix. Let $\tilde{h}^{(k)} = Q_{k-1}^* h^{(k)} \in \mathbb{C}^k$, and let

$$\Omega_k^* = \begin{bmatrix} \mathrm{id}_{k-1} & & \\ & c_k^* & s_k \\ & -s_k & c_k, \end{bmatrix} \tag{4.16}$$

with

$$c_k^* = \frac{(\tilde{h}_k^{(k)})^*}{\sqrt{|h_k^{(k)}|^2 + h_{k+1,k}^2}}, \quad \text{and} \quad s_k = \frac{h_{k+1,k}}{\sqrt{|h_k^{(k)}|^2 + h_{k+1,k}^2}}. \tag{4.17}$$

By a matrix multiplication, we can check that $R_k = \Omega_k^* Q_k^* \underline{H}_{kk}$ is upper triangular, and R_k is given by

$$R_k = \begin{bmatrix} R_{k-1} & \tilde{h}_{1:k-1}^{(k)} \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}. \tag{4.18}$$

We are now in position to write the GMRES algorithm 4.6.

Algorithm 4.6 GMRES

function GMRES($A, b, x^{(0)}, \varepsilon_{\mathrm{tol}}$)
  $r^{(0)} = b - Ax^{(0)}$, $k = 0$
  while $\|r^{(k)}\| > \varepsilon_{\mathrm{tol}}$ **do**
    $k = k + 1$
    Compute v_k of the Arnoldi algorithm 4.1 for A with $v = r^{(0)}$
    Update Q_k according to Eq. (4.15), (4.16) and (4.17)
    Compute $\begin{bmatrix} g_k \\ \gamma_{k+1} \end{bmatrix} = \|r^{(0)}\| Q_k^* e_1$
    Set $\|r^{(k)}\| = |\gamma_{k+1}|$
  end while
  Compute $t_k = \widetilde{R}_k^{-1} g_k$, where $R_k = \begin{bmatrix} \widetilde{R}_k \\ 0 \end{bmatrix}$ and R_k given by Eq. (4.18)
  return $x^{(0)} + V_k t_k$
end function

Note that in GMRES, only the last approximation $x^{(k)}$ to the solution x_* to the linear equation is computed. Indeed, at each iteration, we just need to estimate the residual which is given by Equation (4.14). Concerning the cost of GMRES, at each step, one step of Arnoldi algorithm has to be performed, which costs one matrix-vector multiplication, and k scalar products. The matrix Q_k needs to be updated, but this cost is negligeable. However, in terms of storage cost, all the Arnoldi vectors $(v_1, \ldots, v_k)$ have to be kept for each iteration. This is a serious limitation to the algorithm and in practice, a full GMRES by keeping all the Arnoldi vectors is not advisable, especially if the convergence is slow.

4.3.4 Restarted GMRES

The idea is to limit the number of Arnoldi vectors to K and restart a GMRES run from the latest GMRES iteration.

Algorithm 4.7 Restarted GMRES(K)

function GMRES($A, b, x^{(0)}, \varepsilon_{\text{tol}}, K$)
  $r^{(0)} = b - Ax^{(0)}$
  while $\|r^{(0)}\| > \varepsilon_{\text{tol}}$ **do**
    Compute $(v_1, \dots, v_K)$ the Arnoldi vectors of Algorithm 4.1 for A with $v = r^{(0)}$
    Update Q_K according to Eq. (4.15),(4.16) and (4.17)
    Compute $\begin{bmatrix} g_K \\ \gamma_{K+1} \end{bmatrix} = \|r^{(0)}\| Q_K^* e_1$
    Set $\|r^{(0)}\| = |\gamma_{K+1}|$
    Compute $t_K = \widetilde{R}_K^{-1} g_K$, where $R_K = \begin{bmatrix} \widetilde{R}_K \\ 0 \end{bmatrix}$ and R_K given by Eq. (4.18)
    $x^{(0)} = x^{(0)} + V_K t_K$
  end while
  return $x^{(0)}$
end function

The storage cost of the restarted GMRES scales as the number of Arnoldi vectors stored at each step of the algorithm. Note that contrary to GMRES, we have no guarantee that the algorithm converges after a finite number of iterations, although the residuals are still nonincreasing, due to the mathematical characterisation of GMRES (4.3).

Remark 4.24. *Let $A \in \mathbb{C}^{n\times n}$ be given by*

$$A = \begin{bmatrix} 0 & \dots & 0 & \alpha_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & \alpha_{n-2} \\ & & 1 & \alpha_{n-1} \end{bmatrix}, \tag{4.19}$$

where $(\alpha_0, \dots, \alpha_{n-1}) \in \mathbb{C}^n$. The characteristic polynomial of A is given by $P(\lambda) = \det(\lambda \,\mathrm{id} - A) = \lambda^n - \sum_{k=0}^{n-1} \alpha_k \lambda^k$, thus the coefficients can be chosen to have any eigenvalue distribution. Then for any $b \in \mathbb{C}^n$, the restarted GMRES with $x^{(0)} = A^{-1}(b - e_1)$ does not converge, except if $K = n$. In fact, the residual is constant $r^{(k)} = r^{(0)}$ for $k \le n - 1$. This shows that there is no hope to give an accurate characterisation of the convergence of GMRES solely based on the spectrum of the matrix.

In practice, it is customary to take $K = 20$ and in general, a larger K improves the convergence of the restarted GMRES algorithm.

Remark 4.25. *The latter comment is a general advice but counterexamples exist to this rule of thumb. Let $A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix}$ and $b = \begin{bmatrix} 2 \\ -4 \\ 1 \end{bmatrix}$, then for $x^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$, restarted GMRES converges after three steps for $K = 1$ but does not converge for $K = 2$.*

4.3.5 Convergence of GMRES

Convergence results on GMRES are less powerful than for the CG algorithm. An attempt consists in following the same steps as in the convergence estimate of the CG algorithm:

$$\begin{aligned}\|r^{(k)}\| &= \min_{z\in x^{(0)}+\mathcal{K}_k(A,r^{(0)})} \|b-Az\| \\ &= \min_{\tilde{z}\in\mathcal{K}_k(A,r^{(0)})} \|r^{(0)}-A\tilde{z}\| \\ &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \|\phi(A)r^{(0)}\|.\end{aligned}$$

Since A is no longer Hermitian, we have to resort to another decomposition of the matrix A, namely the Jordan decomposition which is recalled in the next proposition.

Proposition 4.26. *Let $A\in\mathbb{C}^{n\times n}$ and $(\lambda_1,\dots,\lambda_r)$ be the distinct eigenvalues of A. For $1\le\ell\le r$, let $J_{\lambda_\ell}=\begin{bmatrix}\lambda_\ell & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1\\ & & & \lambda_\ell\end{bmatrix}\in\mathbb{C}^{n_\ell\times n_\ell}$ be the Jordan blocks, where $\sum_{\ell=1}^r n_\ell=n$. Then there exists $Y\in\mathbb{C}^{n\times n}$ invertible such that*

$$A=Y\begin{bmatrix}J_{\lambda_1} & & \\ & \ddots & \\ & & J_{\lambda_r}\end{bmatrix}Y^{-1}. \tag{4.20}$$

This is the Jordan decomposition *of A and if the columns of Y are of norm 1, it is unique up to the permutations of the Jordan blocks and rotations in the Jordan blocks.*

Using the Jordan decomposition of A, we have

$$\begin{aligned}\|r^{(k)}\| &= \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \left\|Y\begin{bmatrix}\phi(J_{\lambda_1}) & & \\ & \ddots & \\ & & \phi(J_{\lambda_r})\end{bmatrix}Y^{-1}r^{(0)}\right\| \\ &\le \|Y\|\|Y^{-1}\|\|r^{(0)}\| \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\le\ell\le r}\|\phi(J_{\lambda_\ell})\|.\end{aligned}$$

Proposition 4.27. *Let $A=Y\begin{bmatrix}J_{\lambda_1} & & \\ & \ddots & \\ & & J_{\lambda_r}\end{bmatrix}Y^{-1}$ be a Jordan decomposition of A, and $r^{(k)}$ be the k-th residual of the GMRES algorithm 4.6. Then*

$$\|r^{(k)}\|\le\|Y\|\|Y^{-1}\|\|r^{(0)}\| \min_{\substack{\phi\in\mathbb{C}^k[X]\\ \phi(0)=1}} \max_{1\le\ell\le r}\|\phi(J_{\lambda_\ell})\|.$$

If Y is ill-conditioned, the bound given is meaningless. Consider the matrix $A=\mathrm{tridiag}(-\alpha,\alpha,-\frac{1}{\alpha})\in\mathbb{R}^{n\times n}$, with $\alpha\in\mathbb{R},\alpha>1$. The eigenvalues of A are $\alpha-2\cos\left(\frac{j\pi}{n+1}\right)$ for $1\le j\le n$ and the associated eigenvectors are $y_j=\frac{Dz_j}{\|Dz_j\|}$ for $1\le j\le n$ where (z_j) is some orthonormal basis of $\mathbb{R}^n$ and $D=\mathrm{diag}(\alpha,\dots,\alpha^n)$. The conditioning of Y then scales as α^n, but $\|r^{(k)}\|\le\|r^{(0)}\|$.

4.3.6 Beyond GMRES?

The bottleneck of GMRES is the Arnoldi algorithm and in particular, the absence of a short recurrence in the Arnoldi algorithm for a general matrix A.

A natural question to ask is whether the Arnoldi algorithm is a good starting point to derive a short-term iterative linear solver. In other words, for a given matrix A, is there an $(s+1)$-term recurrence of the form

$$\begin{cases} x^{(k)} = x^{(k-1)} + \alpha_{k-1} p^{(k-1)} \\ p^{(k)} = Ap^{(k-1)} - \sum_{j=0}^{s-2} \beta_{k-1,j} p^{(k-1-j)}, \end{cases} \tag{4.21}$$

with $x^{(0)} \in \mathbb{C}^n$ and $p^{(0)} = r^{(0)} = b - Ax^{(0)}$ which stops after $m \leq n$ iterations at the exact solution x_* to $Ax_* = b$, for a well-chosen set of coefficients $(\alpha_k)_{0\leq k\leq m-1}$ and $(\beta_{k-1,j})_{\substack{0\leq k\leq m-1\\ 0\leq j\leq s-2}}$?

By Eq. (4.21), the error $e_k = x^{(k)} - x_*$ satisfies $e_k = e_{k-1} + \alpha_{k-1} p^{(k-1)}$. Hence by iteration,

$$e_k \in e_0 + \mathrm{Span}(p^{(0)}, \dots, p^{(k-1)}).$$

The error e_k is thus minimised when $e_k \perp \mathrm{Span}(p^{(0)}, \dots, p^{(k-1)})$, thus $\langle e_k, p^{(j)} \rangle = 0$ for all $0 \leq j \leq k-1$. This gives in particular for $j = k-1$

$$\alpha_{k-1} = \frac{\langle p^{(k-1)}, e_{k-1} \rangle}{\langle p^{(k-1)}, p^{(k-1)} \rangle},$$

and for $0 \leq j \leq k-2$, we have

$$0 = \langle p^{(j)}, e_k \rangle = \langle p^{(j)}, e_{k-1} \rangle + \alpha_{k-1} \langle p^{(j)}, p^{(k-1)} \rangle = \alpha_{k-1} \langle p^{(j)}, p^{(k-1)} \rangle.$$

This means that if $\alpha_{k-1} \neq 0$, $\langle p^{(j)}, p^{(k-1)} \rangle = 0$, thus the coefficients $(\beta_{k-1,j})_{\substack{0\leq k\leq m-1\\ 0\leq j\leq s-2}}$ have to be set to

$$\beta_{k-1,j} = \frac{\langle p^{(k-1-j)}, Ap^{(k-1)} \rangle}{\langle p^{(k-1-j)}, p^{(k-1-j)} \rangle}.$$

This motivates to restrict the search of a short-term iterative linear solver to the following class of sequences.

Definition 4.28. *The sequences $(x^{(k)}), (p^{(k)})$ defined by Equation (4.21) such that for any $b \in \mathbb{C}^n$, there is $m \leq n$ such that $x^{(m)}$ solves $Ax^{(m)} = b$ and $\langle p^{(j)}, p^{(k)} \rangle = 0$ for $0 \leq j \neq k \leq m$ are called an $(s+1)$-term CG method.*

We have a characterisation of the class of matrices which have an $(s+1)$-term CG method.

Theorem 4.29 (Faber, Manteuffel (1984))**.** *An $(s+1)$-term CG method exists for the matrix A if and only if either*

1. *the minimal polynomial of A has degree less than s;*
2. *A^* is a polynomial of degree less than $s-2$ in A.*

Notice that for $s+1 = 3$ and assuming additionally that A is Hermitian positive-definite, the CG algorithm 4.3 is a solution to the Faber-Manteuffel theorem.

Proof. We are only going to prove the converse, as the proof of the implication is quite involved.

First, assume that the minimal polynomial of A has degree less than s. Let $(x^{(k)}), (p^{(k)})$ be the sequences defined by Eq. (4.21). By induction, we see that for all $1 \leq k \leq s$, $(p^{(0)}, \dots, p^{(k-1)})$ is a basis of $\mathrm{Span}(r^{(0)}, Ar^{(0)}, \dots, A^{k-1}r^{(0)})$. By induction, we also have that $x^{(k)} = x^{(0)} + \sum_{j=0}^{k-1} \alpha_j p^{(j)}$, thus we can choose $(\alpha_j)_{0\leq j\leq s-1}$ and $(\beta_{k,j})_{\substack{0\leq k\leq s-1\\ 0\leq j\leq s-2}}$ such that

$$\begin{cases} \|b - Ax^{(k)}\| = \min\limits_{z^{(k)} \in \mathrm{Span}(p^{(0)}, \dots, p^{(k-1)})} \|r^{(0)} - Az^{(k)}\| \\ p^{(k)} \perp \mathrm{Span}(p^{(k-s+1)}, \dots, p^{(k-1)}). \end{cases}$$

In particular for $k = s$, we have that

$$\begin{aligned} \|b - Ax^{(s)}\| &= \min_{z^{(s)} \in \mathrm{Span}(p^{(0)}, \dots, p^{(s-1)})} \|r^{(0)} - Az^{(s)}\| \\ &= \min_{\phi \in \mathbb{C}^{s-1}[X]} \|r^{(0)} - \phi(A)r^{(0)}\| \\ &= \min_{\substack{\phi \in \mathbb{C}^{s}[X] \\ \phi(0)=1}} \|\phi(A)r^{(0)}\|. \end{aligned}$$

Choosing ϕ as the minimal polynomial such that $\phi(0) = 1$ shows that $Ax^{(s)} = b$.

Now assume that A^* is a polynomial of degree less than $s-2$ of A. We will first prove by induction that if $(p^{(j)})_{0\leq j\leq k}$ are orthogonal vectors, then $(p^{(j)})_{0\leq j\leq k+1}$ are also orthogonal.

For the initialisation, we know that for $k \leq s-1$, choosing $(\beta_{k-1,j})_{0\leq j\leq s-2}$ such that

$$\beta_{k-1,j} = \frac{\langle p^{(j)}, Ap^{(k-1)} \rangle}{\langle p^{(k-1-j)}, p^{(k-1-j)} \rangle}$$

ensures that $\langle p^{(i)}, p^{(j)} \rangle = 0$ for $0 \leq i \neq j \leq s-1$.

For the induction step, let us assume that $(p^{(j)})_{0\leq j\leq k}$ are orthogonal vectors. We already know that by setting $(\beta_{k,j})_{0\leq j\leq s-2}$ such that

$$\beta_{k,j} = \frac{\langle p^{(k-j)}, Ap^{(k)} \rangle}{\langle p^{(k-j)}, p^{(k-j)} \rangle},$$

we have that $p^{(k+1)} \perp \mathrm{Span}(p^{(k-s+2)}, \dots, p^{(k)})$. For $i \leq k-s+1$, we have from Eq. (4.21)

$$\langle p^{(i)}, p^{(k+1)} \rangle = \langle p^{(i)}, Ap^{(k)} \rangle - \sum_{j=0}^{s-2} \beta_{k,j} \langle p^{(i)}, p^{(k-j)} \rangle.$$

By the induction assumption, $\langle p^{(i)}, p^{(k-j)} \rangle = 0$ for $0 \leq j \leq s-2$. Now $\langle p^{(i)}, Ap^{(k)} \rangle = \langle A^* p^{(i)}, p^{(k)} \rangle$. By assumption, $A^* = q_{s-2}(A)$ for some polynomial $q_{s-2} \in \mathbb{C}^{s-2}[X]$. Thus $A^* p^{(i)} = q_{s-2}(A)p^{(i)} \in \mathrm{Span}(p^{(i)}, \dots, p^{(i+s-2)})$. But $i+s-2 \leq k-1$, thus using the induction assumption $\langle q_{s-2}(A)p^{(i)}, p^{(k)} \rangle = 0$. This shows that $\langle p^{(i)}, p^{(k+1)} \rangle = 0$.

Notice that $\mathrm{Span}(p^{(0)}, \dots, p^{(k)}) \subset \mathrm{Span}(r^{(0)}, \dots, A^k r^{(0)})$ but by orthogonality of the vectors $(p^{(0)}, \dots, p^{(k)})$, we deduce that $\mathrm{Span}(p^{(0)}, \dots, p^{(k)}) = \mathrm{Span}(r^{(0)}, \dots, A^k r^{(0)})$. Since $x^{(k)} = x^{(0)} + \sum_{i=0}^{k-1} \alpha_i p^{(i)}$, by selecting $\alpha_i = -\langle p^{(i)}, x^{(0)} - x_* \rangle$, we have that $(x^{(k)})$ converges to x_* in at most n steps. This finishes the proof. □

The Faber-Manteuffel theorem essentially states that for a general matrix A, there is no equivalent to the conjugate-gradient algorithm solely based on the Krylov space $\mathcal{K}_k(A, r^{(0)})$. This means that we need to look for another construction to define a short-term recurrence for the iterative method.

Going back to the essence of the conjugate-gradient and GMRES methods, it is reasonable to look for alternatives to the Arnoldi/Lanczos algorithms that would give a short-term recurrence for any matrix.

One way to achieve such a short recurrence is to relax the orthogonality constraint on the Arnoldi vectors $(v_1, \dots, v_k)$ while remaining a basis of $\mathcal{K}_k(A, v)$. The new constraint is to build a family of vectors $(w_1, \dots, w_k)$ which is a basis of $\mathcal{K}_k(A^*, w)$ and also a dual family to $(v_1, \dots, v_k)$, *i.e.* $\forall 1 \leq i, j \leq k, \langle w_i, v_j \rangle = \delta_{ij}$. This gives the non-Hermitian Lanczos algorithm 4.8.

Algorithm 4.8 Non-Hermitian Lanczos algorithm

function NonHermitianLanczos(A, v, w, k)
  $v_0 = w_0 = 0$, $\beta_1 = \delta_1 = 0$
  $v_1 = \frac{v}{\|v\|}$, $w_1 = \frac{w}{\langle v_1, w \rangle}$
  for $j = 1, \dots, k$ **do**
    $\gamma_j = \langle w_j, A v_j \rangle$
    $\widehat{v}_{j+1} = A v_j - \gamma_j v_j - \beta_j v_{j-1}$
    $\widehat{w}_{j+1} = A^* w_j - \gamma_j w_j - \delta_j w_{j-1}$
    $\delta_{j+1} = \|\widehat{v}_{j+1}\|$
    if $\delta_{j+1} = 0$ **then**
      Stop
    end if
    $\beta_{j+1} = \langle v_{j+1}, \widehat{w}_{j+1} \rangle$
    if $\beta_{j+1} = 0$ **then**
      Stop
    end if
    $w_{j+1} = \frac{\widehat{w}_{j+1}}{\beta_{j+1}}$
  end for
  return $(v_1, \dots, v_k), (w_1, \dots, w_k)$
end function

For the non-Hermitian Lanczos vectors, we have the following properties.

Proposition 4.30. *If there is no breakdown in Algorithm 4.8,* i.e. *for all* $1 \leq j \leq k-1$ $\widehat{v}_{j+1} \neq 0, \widehat{w}_{j+1} \neq 0, \beta_{j+1} \neq 0$, *then we have*

$$\begin{cases} A V_k = V_k T_k + \delta_{k+1} v_{k+1} e_k^T \\ A^* W_k = W_k T_k + \beta_{k+1} w_{k+1} e_k^T \\ W_k^* A V_k = T_k \\ W_k^* V_k = \mathrm{id}_k, \end{cases} \tag{4.22}$$

where $V_k = \begin{bmatrix} v_1 & \dots & v_k \end{bmatrix}$, $W_k = \begin{bmatrix} w_1 & \dots & w_k \end{bmatrix}$ *and* $T_k = \begin{bmatrix} \gamma_1 & \beta_2 & & \\ \delta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_k \\ & & \delta_k & \gamma_k \end{bmatrix}$.

It is possible to derive an iterative method as a projection process, using the non-Hermitian Lanczos to define a basis for the Krylov subspace. This yields the biconjugate gradient method (BiCG) which by construction enjoys a short recurrence but does not preserve the relations between the search space and the constraint space for the residuals. The non-Hermitian Lanczos algorithm has two types of breakdowns:

- when $\widehat{v}_{j+1} = 0$ or $\widehat{w}_{j+1} = 0$: this is related to a stagnation of the corresponding subspace, which means that we have reached convergence in the projection process;
- when $\widehat{v}_{j+1} \neq 0, \widehat{w}_{j+1} \neq 0$ but $\langle \widehat{v}_{j+1}, \widehat{w}_{j+1} \rangle = 0$: this is called a serious breakdown as we do not have a stable Krylov subspace under A, and so we do not have reached convergence in the iterative algorithm. This can happen for well-conditioned matrices which indicate that it can happen generically.

Remark 4.31. *Let* $A = \begin{bmatrix} 5 & 1 & -1 \\ -5 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$, $v_1 = \begin{bmatrix} 0.6 \\ -1.4 \\ 0.3 \end{bmatrix}$ *and* $w_1 = \begin{bmatrix} 0.6 \\ 0.3 \\ -0.1 \end{bmatrix}$. *The eigenvalues of* A *are* $1, 2$ *and 3 and* $v_2 = \frac{1}{3}\begin{bmatrix} 1.5 \\ -2.5 \\ 1.5 \end{bmatrix}$, $w_2 = \frac{1}{3}\begin{bmatrix} 1.8 \\ 0.6 \\ -0.8 \end{bmatrix}$. *The vectors* v_2 *and* w_2 *are orthogonal although the matrix* A *is well-conditioned.*

4.3.7 GMRES in the XXI$^{\text{st}}$ century

GMRES has become the method of choice for solving nonhermitian linear problems. However, compared to CG, GMRES suffer from some drawbacks:

- theoretically, there is no fully satisfying explanation of the convergence behaviour of GMRES as the upper bounds are often much more pessimistic than what is actually observed;
- in practice, one would often use restarted GMRES (or one of its variants), but again, the behaviour of the algorithm with respect to the restart parameter is rather unclear.

Excellent references on that topic are the monograph [LS12] or the book by Y. Saad [Saa03].