Solving Linear Systems with Iterative Methods

5 Solving Linear Systems with Iterative Methods

5.1 Splitting Methods

Assumption in this section: $A \in \mathbb{R}^{n \times n}$ is invertible, so that $x^* = A^{-1}b$ is the unique solution.

5.1.1 Motivation and Overview

Problem: *The matrix A can be very large* ($n \geq 10^5$)!

- If A can be stored, direct methods (such as LU, QR) are very slow or even not feasible due to large byproducts.
- Often A cannot be stored, so that direct methods are not an option.

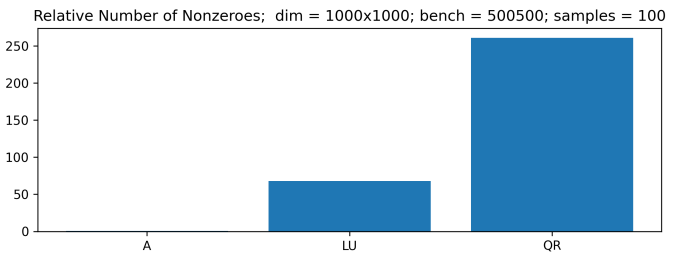
Example:

- A float with double precision (standard) needs 8 bytes of memory.
- Considering 3d measurements with just 100 measurements in each dimension.
  $\Rightarrow 100 \cdot 100 \cdot 100 = 10^6$ measurements
- Discretization methods (e.g., finite element method) for physical models (e.g., heat diffusion) interrelate these measurements.
  $\Rightarrow$ gives a matrix $A \in \mathbb{R}^{n \times n}$ with $n = 10^6$
  $\Rightarrow n \cdot n = 10^{12}$ numbers have to be stored
  $\Rightarrow 10^{12}$ bytes $= 1$ **Terabyte** of memory has to be allocated in RAM
- A standard PC nowadays has 8–32 Gbytes of RAM (fast memory).

What if the matrix is *sparse* (= many 0 entries = redundancy)?
→ We only need to store nonzero entries and their coordinates (see, e.g., CSR from previous sheets).
→ **But:** Direct methods may still produce *dense* ($\neq$ sparse) byproducts.

```
    A = genRandomSymSparse(D)
    # NONZEROES OF A
    nonzeroesA += [np.count_nonzero(A)]
    # NONZEROS OF LU
    lu, piv = linalg.lu_factor(A)
    nonzeroesLU += [np.count_nonzero(lu)]
    # NONZEROS OF QR
    Q, R = linalg.qr(A)
    nonzeroesQR += [(np.count_nonzero(Q)+np.count_nonzero(R))]

# For sym. matrices we only need to store lower or upper triangle
bench = int(D*(D+1)*0.5)
# Average
nonzeroesA = np.round((np.array(nonzeroesA, dtype=float).sum()/N)/bench*100,2)
nonzeroesLU = np.round((np.array(nonzeroesLU, dtype=float).sum()/N)/bench*100,2)
nonzeroesQR = np.round((np.array(nonzeroesQR, dtype=float).sum()/N)/bench*100,2)
```



Key idea: *We do not need the full matrix, but only some matrix-vector product* $x \mapsto S_A x$

Approach: Using this product we define a sequence $\{x^0, x^1, x^2, \ldots\} \subset \mathbb{R}^n$ such that

- $x^k \to x^* \in \mathbb{R}^n$ for $k \to \infty$
- x^* solves the linear equation $Ax = b$
- x^{k+1} is a better approximation to x^* than x^k

Standard classes of such iterations:

(1) **Linear iterations (splitting methods)** $\leftarrow$

(2) **Krylov subspace methods**

Common advantages:

- Even a few iteration steps may yield good results in stark contrast to LU (Gaussian Elimination), which has to be performed to the bitter end.
- The matrix A or its decomposition does not need to be stored! Only a **matrix-vector product** $S_A x$ has to be provided.
- In general, the overall computational complexity is lower than with *direct* methods (LU, QR,...).

$\rightarrow$ Therefore: Iterative (=*indirect*) methods are to be preferred, when the matrix is large (and sparse).

C.F. Gauß in a letter to Gerling from 1823
(https://gdz.sub.uni-goettingen.de/id/PPN23601515X?tify)

Fast jeden Abend mache ich eine neue Auflage des Tableaus, wo immer leicht nachzuhelfen ist. Bei der Einförmigkeit des Messungsgeschäfts gibt dies immer eine angenehme Unterhaltung; man sieht dann auch immer gleich, ob etwas zweifelhaftes eingeschlichen ist, was noch wünschenswerth bleibt, etc. Ich empfehle Ihnen diesen Modus zur Nachahmung. Schwerlich werden Sie je wieder direct eliminiren, wenigstens nicht, wenn Sie mehr als 2 Unbekannte haben. Das indirecte Verfahren lässt sich halb im Schlafe ausführen, oder man kann während desselben an andere Dinge denken.

. . . . . .

→ He already mentions an iterative method which was later coined *Gauß-Seidel method*.

→ In general, one may need a lot of iteration steps but the aim is to keep the iteration instruction simple and fast.

5.1.2 A General Framework: Linear Fixed Point Iteration

Linear iterations are of the form

$$x^{k+1} = Mx^k + Nb$$

with $M, N \in \mathbb{R}^{n\times n}$. The matrix M is called the **iteration matrix** and motivates the adjective "linear".

We first derive a general convergence result and then relate M and N to the system $Ax = b$.

Convergence analysis

Definition 5.1 (*Spectral radius*) *Let $M \in \mathbb{R}^{n\times n}$. Then the largest eigenvalue of M in magnitude is called the **spectral radius of** M and denoted by $\rho(M)$, more precisely*

$$\rho(M) := \max\{|\lambda_1|, \ldots, |\lambda_n|\}.$$

Theorem 5.2 (*Fixed point iteration*) *Let $M, N \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. If $\rho(M) < 1$, then the sequence*

$$x^{k+1} = Mx^k + Nb$$

converges for any starting point x^0 and its limit $x^* \in \mathbb{R}^n$ is a fixed point of the affine linear function $x \mapsto Mx + Nb$, i.e.,

$$x^* = Mx^* + Nb.$$

Proof for the special case that M is symmetric:

5.1.3 Splitting Methods

We now apply Theorem 5.2 to the linear system $Ax = b$ by reformulating it as a fixed point problem. We also explain the idea of *preconditioning*.

What is a good preconditioner?

How to construct a preconditioner?

General scheme:

$$x^{k+1} = (I - NA)x^k + Nb = x^k - N(Ax^k - b)$$

Preconditioner N	**Iteration Matrix** $M = I - NA$	**Iteration Instruction** $x^{k+1} = x^k - N(Ax^k - b)$	**Method Name**
θI	$M_{Rich} = I - \theta A$	$x^{k+1} = x^k - \theta(Ax^k - b)$	(relax.) **Richardson**
θD^{-1}	$M_{Jac} = I - \theta D^{-1}A$ $= (1-\theta)I - \theta D^{-1}(L+U)$	$x^{k+1} = x^k - \theta D^{-1}(Ax^k - b)$ *Element-based:* $x_i^{k+1} = (1-\theta)x_i^k + \frac{\theta}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}\, x_j^{k+1}\right)$	(weighted) **Jacobi**
$(L+D)^{-1}$	$M_{GS} = I - (L+D)^{-1}A$ $= (L+D)^{-1}U$	$x^{k+1} = x^k - (L+D)^{-1}(Ax^k - b)$ *Element-based:* $x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^k\right)$	**Gauß-Seidel**
$\theta(\theta L + D)^{-1}$	$M_{SOR} = I - \theta(\theta L + D)^{-1}A$ $= (\theta L + D)^{-1}((1-\theta)D - \theta U)$	$x^{k+1} = x^k - \theta(\theta L + D)^{-1}(Ax^k - b)$ *Element-based:* $x_i^{k+1} = (1-\theta)x_i^k$ $+ \frac{\theta}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}\, x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}\, x_j^k\right)$	**Successive Over–Relaxation (SOR)**

What remains: When does $\rho(M) < 1$ hold?

Idea: Derive some (possibly easy-to-compute) conditions which are sufficient for $\rho(M) < 1$
$\rightarrow$ As usual, we often need to assume some properties for A (e.g., symmetry).

Method	Condition
(relax.) Richardson $M_{Rich} = I - \theta A$	With relaxation ($\theta \neq 1$): If A is symmetric and positive definite (spd), then: $\rho(M_{Rich}) < 1 \quad \Leftrightarrow \quad 0 < \theta < \frac{2}{\lambda_{max}(A)}$
(weighted) Jacobi $M_{Jac} = I - \theta D^{-1} A$	No relaxation ($\theta = 1$): If A is strictly diagonally dominant (i.e., $\lvert a_{ii}\rvert > \sum_{i \neq j} \lvert a_{ij}\rvert$), then $\rho(M_{Jac}) < 1$ With relaxation ($\theta \neq 1$): If A is spd, then: $\rho(M_{Jac}) < 1 \quad \Leftrightarrow \quad 0 < \theta < \frac{2}{\lambda_{max}(D^{-1}A)}$
Gauß-Seidel $M_{GS} = I - (L + D)^{-1} A$	• If A is strictly diagonally dominant, then $\rho(M_{GS}) < 1$ • If A is spd, then $\rho(M_{GS}) < 1$
Successive Over–Relaxation (SOR) $M_{SOR} = I - \theta(\theta L + D)^{-1} A$	If A is spd, then $\rho(M_{SOR}) < 1$ for $0 < \theta < \mathbf{2}$

Remark: For many matrix classes (e.g., symmetric), we find

$$\rho(M_{GS}) \leq \rho(M_{Jac}) \leq \rho(M_{Rich}) \qquad (\theta = 1).$$

Example

Final Remarks

- Using the Richardson iteration we only need the evaluation of the matrix vector product $x \mapsto Ax$ to solve the system $Ax = b$.
- All iterations of the form $x^{k+1} = x^k - N(Ax^k - b)$ can be seen as preconditioned Richardson iterations. Next semester you will learn a strong correspondence between the Richardson method and the gradient method, as well as preconditioned Richardson method and Newton-type methods to minimize functions of the form $f(x) := \frac{1}{2}x^\top Ax - b^\top x$.
- By looking at the element-wise formulas for the Jacobi (with $\theta = 1$) and Gauß-Seidel method (see the orange formulas in the table above) we can understand these methods as alternating methods. Consider for example the simple case $n = 2$ and write out these formulas. Then you will see that we alternatingly compute the components x_1 and x_2.
- Similar to a block LU factorization, we can consider blocks A_{ij} (= matrices) instead of numbers a_{ij} in the element-wise formulas of Jacobi and Gauß-Seidel. Thereby, one obtains the block Jacobi and block Gauß-Seidel method. These are highly related to so called additive and multiplicative Schwarz methods, respectively.
- In practice, splitting methods (or more precisely the preconditioner N) are mainly used as preconditioners for Krylov subspace methods, which we will address in the next section.

5.2 Krylov Subspace Methods

5.2.1 Krylov Subspaces

Situation:

- $A \in GL_n(\mathbb{R})$ (invertible, n typically large, but sparse)
- $b \in \mathbb{R}^n$

We want to solve $Ax = b$, but we cannot work with the matrix as a (dense) "array". Instead we only have access to the mapping

$$x \mapsto Ax \quad \text{(matrix-vector product)}.$$

Then given $b \in \mathbb{R}^n$, we can produce the vectors

$$b, Ab, A^2b, \ldots, A^kb.$$

There is not much more to consider. Let us collect all linear combinations of these vectors and give it a name:

Definition 5.3 (*Krylov *subspaces)*** *Let $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, then the set*

$$K_r(A,b) := \mathit{span}\{b, Ab, A^2b, \ldots, A^{r-1}b\}$$

is called ***Krylov Subspace*** *of order $r \geq 1$ generated by A and b.*

In order to develop an iterative scheme based on this definition, we look deeper into the Krylov subspaces and first collect some insightful observations.

---

* *named after the Russian engineer Alexei Krylov who developed the idea in a paper published around 1931.*

Remarks

i) If $b = 0$, then $A^k b = 0$ for all $A \in \mathbb{R}^{n\times n}$ and all $k \in \mathbb{N}$, so that

$$K_r(A, 0) = \{0\} \qquad \forall\ A \in \mathbb{R}^{n\times n},\ r \geq 1.$$

ii) If $b \neq 0$ and $A = I$, then $A^k b = b$ for all $k \in \mathbb{N}$, so that

$$K_r(I, b) = \text{ span}\{b\} \qquad \forall\ b \neq 0,\ r \geq 1.$$

iii) If b is an eigenvector of A to the eigenvalue $\lambda \in \sigma(A)$, then $A^k b = \lambda^k b$, so that $A^k b$ and b are linearly dependent, implying

$$K_r(A, b) = \text{ span}\{b\} \qquad \forall \text{ eigenvectors } b \text{ of } A.$$

iv) Insight from the power method:
Recall: Let $|\lambda_1| > |\lambda_2| \geq \ldots |\lambda_n|$ and $b^T v_1 \neq 0$, then $\frac{A^k b}{\|A^k b\|} \to v_1$.
Thus, for large k we have that the $A^k b$ point into a "similar direction" – more precisely, into the direction of v_1. With other words the $A^k b$ become more and more linearly dependent.

v) Dimension of Krylov subspaces:
   - Since $(K_r(A, b)) \subset \mathbb{R}^n$ is spanned by r vectors, we clearly have $\dim(K_r(A, b)) \leq \min(r, n)$.
   - For $A \in GL_n(\mathbb{R})$ one can show that

$$b, Ab, \ldots, A^{r-1} b \text{ are independent} \qquad \forall\ r \leq r_{\max},$$

where $r_{\max}$ is the maximal dimension a Krylov subspace generated by A and b can have, i.e., $r_{\max} := \max_{s\leq n}(\dim K_s(A, b))$.

vi) Next we state the crucial result, which forms the basis for the development of Krylov subspace methods:

Lemma 5.4 *Let $A \in GL_n(\mathbb{R})$ and $b \in \mathbb{R}^n$, then there exists an order $r \leq n$ so that for the solution x^* of $Ax = b$ we have*

$$x^* = A^{-1}b \in K_r(A, b).$$

In particular, we find coefficients $\beta_0, \ldots, \beta_{r-1}$, such that

$$x^* = \sum_{j=0}^{r-1} \beta_j A^j b.$$

Proof.

Idea of Krylov Subspace Methods

$\rightarrow$ In this chapter we will derive the GMRES method.

Comparison to other methods

Least squares	Krylov subspace	Splitting
$\hat{x} := \operatorname{argmin}_{x \in \mathbb{R}^n} \|Ax - b\|_2^2$ $A\hat{x} = \operatorname{argmin}_{w \in \mathsf{Im}(A)} \|w - b\|_2^2$	$x_r := \operatorname{argmin}_{x \in K_r(A,b)} \|Ax - b\|_2^2$ $Ax_r = \operatorname{argmin}_{w \in AK_r(A,b)} \|w - b\|_2^2$	$x_r := Mx_{r-1} + Nb$
$\rightsquigarrow$ Projection of b onto $\mathsf{Im}(A) = A\mathbb{R}^n = \mathbb{R}^n$	$\rightsquigarrow$ Projection of b onto $AK_r(A,b) = \mathsf{span}\{Ab, A^2b, \ldots, A^rb\} \subseteq K_{r+1}(A,b)$	
Yields exact solution, i.e., $\hat{x} = A^{-1}b$	In theory: Yields a finite sequence with $x_n = A^{-1}b$	Typically yields an infinite sequence with $x^r \xrightarrow{(k \to \infty)} A^{-1}b$

As a preparation for what follows, we will first derive an iterative method to find orthonormal bases for Krylov subspaces:

5.2.2 The Arnoldi Iteration

- **Situation:** Let us consider the r-th Krylov subspace

$$K_r(A,b) = \text{span}\{A^0 b, A^1 b, \ldots, A^{r-1} b\}$$

  with $r \leq r_{\max} = \max_{s \leq n} (\dim(K_s(A,b)))$, so that all $A^{j-1}b$ are independent and $\dim(K_r(A,b)) = r$.
- **Aim:** Find an orthonormal basis $\{q_1, \ldots, q_r\}$ of $K_r(A,b)$.
- **Idea:** Apply the Gram–Schmidt orthogonalization process to the linearly independent vectors $c_j := A^{j-1}b$, $1 \leq j \leq r$.
- **Recall Gram–Schmidt:** Let $c_1, \ldots, c_r \in \mathbb{R}^n$, where $r \leq n$, be linearly independent vectors. Then an orthonormal basis $\{q_1, \ldots, q_r\}$ of $\text{span}(c_1, \ldots, c_r) = \text{Im}(C)$ can be found by the following iterative scheme:

$$q_1 := \frac{c_1}{\|c_1\|}$$

$$\text{"substracting projections:"} \ \widehat{q}_j := c_j - \sum_{\ell=1}^{j-1} q_\ell^\top c_j \cdot q_\ell, \quad r_{\ell j} := q_\ell^\top c_j$$

$$\text{"normalization:"} \ q_j := \frac{\widehat{q}_j}{\|\widehat{q}_j\|_2}, \quad r_{jj} := \|\widehat{q}_j\|_2$$

The matrix perspective: Putting the vectors q_j and r_j (which are computed step by step) into matrices, say Q and R, then we obtain the (reduced) QR-decomposition of C, i.e., a matrix $Q = [q_1, \ldots, q_r] \in \mathbb{R}^{n \times r}$ with orthonormal columns and an upper triangular matrix $R \in \mathbb{R}^{r \times r}$ with $r_{ii} \neq 0$, so that $C = QR$, which implies (also see Lemma 1.47) $\text{Im}(C) = \text{Im}(QR) = \text{Im}(Q) = \text{span}(q_1, \ldots, q_r)$.

Now we consider the specific choice $c_j := A^{j-1}b$. By inserting these c_j into the above scheme, we obtain the so called **Arnoldi iteration**:

$$q_1 := \frac{b}{\|b\|_2} \qquad (\rightarrow \text{ orthonormal basis for } K_1(A,b) = \text{span}(b),\ b \neq 0)$$

for $j = 2, \ldots, r$:

$$\text{"substracting projections:"}\ \widehat{q}_j := Aq_{j-1} - \sum_{\ell=1}^{j-1} q_\ell^\top (Aq_{j-1}) \cdot q_\ell, \qquad h_{\ell,j-1} := q_\ell^\top (Aq_{j-1}) \quad (*)$$

$$\text{"normalization:"}\ q_j := \frac{\widehat{q}_j}{\|\widehat{q}_j\|_2}, \qquad h_{j,j-1} := \|\widehat{q}_j\| \quad (**)$$

All in all, the Arnoldi process yields an orthonormal basis for $K_r(A,b)$, so that

$$K_r(A,b) = \text{span}(b, Ab, \ldots, A^{r-1}b) = \text{span}(q_1, \ldots, q_r)$$

or with $Q_r := [q_1, \ldots, q_r] \in \mathbb{R}^{n \times r}$ in matrix form

$$K_r(A,b) = \text{Im}(Q_r).$$

Remarks:

- Rearranging **(*)** and **(**)** easily gives

$$Aq_{j-1} = \|\widehat{q}_j\|_2 \cdot q_j + \sum_{\ell=1}^{j-1} q_\ell^\top (Aq_{j-1}) \cdot q_\ell = \sum_{\ell=1}^{j} h_{\ell,j-1} q_\ell.$$

  With other words, Aq_{j-1} is a linear combination of $q_1, \ldots, q_j$.
- Also observe for $j \leq r$: The first $q_1, \ldots, q_{j-1}$ are a basis for $K_{j-1}(A,b)$. Thus, assumed these are given, then in order to find an orthonormal basis for $K_j(A,b)$ we just need to compute one more vector, namely q_j.

5.2.3 GMRES

(with Arnoldi and $x_0 = 0$)

Let us recall the general idea of Krylov subspace methods: In each iteration step we compute

$$x_r := \underset{x \in K_r(A,b)}{\operatorname{argmin}} \|Ax - b\|_2^2.$$

Now we approach this minimization problem in a specific procedure resulting in the GMRES method:

We first find an orthonormal basis $\{q_1, \ldots, q_r\}$ of $K_r(A,b)$ so that $K_r(A,b) = \operatorname{span}(q_1, \ldots, q_r) = \operatorname{Im}(Q_r)$, where $Q_r := [q_1, \ldots, q_r] \in \mathbb{R}^{n \times r}$. Then, since by definition $x_r \in K_r(A,b)$, the minimization problem can be rephrased as

$$x_r = \underset{x \in \operatorname{Im}(Q_r)}{\operatorname{argmin}} \|Ax - b\|_2^2 \overset{(x_r = Q_r c_r)}{=} Q_r \cdot \underbrace{\left(\underset{c \in \mathbb{R}^r}{\operatorname{argmin}} \|AQ_r c - b\|_2^2\right)}_{=:c_r,\ \text{standard least squares problem}}.$$

The least squares problem can then be solved with the help of the QR-decomposition of the design matrix AQ_r, say $\widetilde{Q}_r \widetilde{R}_r := AQ_r$, so that the corresponding normal equation reads as

$$\widetilde{R}_r c_r = \widetilde{Q}_r^T b \quad \text{(when is } \widetilde{R}_r \text{ invertible?)}$$

Thus, GMRES boils down to the three steps. For $1 \leq r \leq n$, do

- Step 1: Find an orthonormal basis for $K_r(A,b)$.
- Step 2: Find $\widetilde{Q}_r \widetilde{R}_r := AQ_r$.
- Step 3: Solve $\widetilde{R}_r c_r = \widetilde{Q}_r^T b$ and set $x_r := Q_r c_r$.

Crucial: We can iteratively compute step 1 and step 2, i.e., use Q_{r-1}, $\widetilde{R}_{r-1}$, $\widetilde{Q}_{r-1}$ to obtain Q_r, $\widetilde{R}_r$, $\widetilde{Q}_r$!

GMRES – Step 1 "Find an orthonormal basis for $K_r(A,b)$"

Given $Q_{r-1} = (q_1, \ldots, q_{r-1}) \in \mathbb{R}^{n\times(r-1)}$ with $\mathrm{Im}(Q_{r-1}) = K_{r-1}(A,b)$ from the previous iteration step, let us compute

$$q_r := \texttt{Arnoldi\_step}(q_1, \ldots, q_{r-1}; Aq_{r-1}).$$

(*"orthogonalizing Aq_{r-1} against all $q_1, \ldots, q_{r-1}$ by subtracting projections and normalization"*)

We recall the r-th Arnoldi iteration step in an algorithmic fashion:

Given $q_1 (:= \frac{b}{\|b\|}), \ldots, q_{r-1}$, then q_r is computed by:

$q_r :=$ **Arnoldi_step**$(q_1, \ldots, q_{r-1};\ v = Aq_{r-1})$:

for $\ell = 1, \ldots, r-1$ **do**
  $h_{\ell,r-1} = q_\ell^T v$
  $v = v - h_{\ell,r-1} q_\ell$ *(substracting projections)*
end
$h_{r,r-1} = \|v\|$
if $h_{r,r-1} \neq 0$ **then**
  $q_r = \frac{v}{h_{r,r-1}}$ *(normalization)*
  return q_r
end
return v

In matrix notation this can be summed up as follows:

$$A \cdot \underbrace{\begin{pmatrix} | & & | \\ q_1 & \cdots & q_{r-1} \\ | & & | \end{pmatrix}}_{=:Q_{r-1}\in\mathbb{R}^{n\times(r-1)}} = \underbrace{\begin{pmatrix} | & & | \\ Aq_1 & \cdots & Aq_{r-1} \\ | & & | \end{pmatrix}}_{\text{Arnoldi: } Aq_j=\sum_{\ell=1}^{j+1} h_{\ell,j}q_\ell} = \underbrace{\begin{pmatrix} | & & | & | \\ q_1 & \cdots & q_{r-1} & q_r \\ | & & | & | \end{pmatrix}}_{=:Q_r\in\mathbb{R}^{n\times r}} \underbrace{\begin{pmatrix} h_{1,1} & h_{1,2} & & \cdots & \\ h_{2,1} & h_{2,2} & & & \vdots \\ 0 & h_{3,2} & & & \\ \vdots & 0 & \ddots & h_{r-1,r-2} & h_{r-1,r-1} \\ 0 & 0 & 0 & h_{r,r-2} & h_{r,r-1} \end{pmatrix}}_{=:H_{r,r-1}\in\mathbb{R}^{r\times(r-1)}}$$

$$\overset{Q_{r-1}^T\cdot|}{\Leftrightarrow} \quad Q_{r-1}^T A Q_{r-1} = Q_{r-1}^T Q_r H_{r,r-1} = \begin{pmatrix} & | & 0 \\ I_{(r-1)\times(r-1)} & | & \vdots \\ & | & 0 \end{pmatrix} \begin{pmatrix} H_{r-1} \\ - - - \\ \text{row } r \text{ of } H_{r,r-1} \end{pmatrix} = H_{r-1}$$

- We observe that $H_{r-1} \in \mathbb{R}^{(r-1)\times(r-1)}$ has only one subdiagonal. Such matrices are called (upper) **Hessenberg matrices**.

- In particular, for $r = n+1$, we find
$$Q_n^T A Q_n = H_n \in \mathbb{R}^{n\times n}.$$
With other words, with the help of the Arnoldi iteration we can find in finitely many steps (at most n steps) an upper Hessenberg H_n, which is orthogonally similar to A and thus has the same eigenvalues. That is why one can tailor the QR-algorithm (which was an algorithm to compute eigenvalues) to matrices of Hessenberg structure. The Arnoldi iteration is therefore also considered an eigenvalue algorithm. Furthermore, if A is symmetric, so is H_n, which then becomes a tridiagonal matrix.

GMRES – Step 2 "Compute QR-decomposition $\widetilde{Q}_r\widetilde{R}_r := AQ_r$"

Again, we want to rely on the computations from the previous steps, i.e., $\widetilde{Q}_{r-1} = [\widetilde{q}_1, \ldots, \widetilde{q}_{r-1}] \in \mathbb{R}^{n\times(r-1)}$ and $\widetilde{R}_{r-1} = [\widetilde{r}_1, \ldots, \widetilde{r}_{r-1}] \in \mathbb{R}^{(r-1)\times(r-1)}$ with $\widetilde{Q}_{r-1}\widetilde{R}_{r-1} = AQ_{r-1}$. Therefore, let us first observe that

$$\widetilde{Q}_r\widetilde{R}_r \overset{!}{=} AQ_r = A \cdot [Q_{r-1}|q_r] = [AQ_{r-1}|Aq_r] = [\widetilde{Q}_{r-1}\widetilde{R}_{r-1}|Aq_r] \in \mathbb{R}^{n\times r}.$$

Let us use one Arnoldi step to orthogonalize Aq_r against $\widetilde{q}_1, \ldots, \widetilde{q}_{r-1}$ to obtain

$$\widetilde{q}_r, \widetilde{r}_r := \texttt{Arnoldi\_step}(\widetilde{q}_1, \ldots, \widetilde{q}_{r-1}; Aq_r),$$

where the vector $\widetilde{q}_r$ and the coefficients of $\widetilde{r}_r = (\widetilde{r}_{1,r}, \ldots, \widetilde{r}_{r,r})^\top \in \mathbb{R}^r$ are computed via

$$\widehat{q}_r := Aq_r - \sum_{j=1}^{r-1} \widetilde{r}_{j,r} \cdot \widetilde{q}_j, \quad \widetilde{r}_{j,r} := \widetilde{q}_j^\top (Aq_r), \quad \widetilde{q}_r := \frac{\widehat{q}_r}{\|\widehat{q}_r\|_2}, \quad \widetilde{r}_{rr} := \|\widehat{q}_r\|_2,$$

from which we can conclude

$$Aq_r = \widetilde{r}_{rr}\widetilde{q}_r + \sum_{j=1}^{r-1} \widetilde{r}_{j,r}\widetilde{q}_j = \sum_{j=1}^{r} \widetilde{r}_{j,r}\widetilde{q}_j. \quad (*)$$

Then let us define potential candidates for the sought-after QR-decomposition as follows:

$$\widetilde{Q}_r := [\widetilde{Q}_{r-1}|\widetilde{q}_r], \quad \widetilde{R}_r := \begin{pmatrix} \widetilde{R}_{r-1} & | \\ - - - & \widetilde{r}_r \\ 0 \cdots 0 & | \end{pmatrix} \in \mathbb{R}^{r\times r}.$$

By (*) we can write $Aq_r = \widetilde{Q}_r\widetilde{r}_r$ and indeed find, as desired,

$$AQ_r = (AQ_{r-1}|Aq_r) = (\widetilde{Q}_{r-1}\widetilde{R}_{r-1}|\widetilde{Q}_r\widetilde{r}_r) = \widetilde{Q}_r\widetilde{R}_r.$$

GMRES – Step 3 "Solve a triangular system"

The last step is easily performed by invoking a routine to solve the upper triangular system

$$\widetilde{R}_r c_r = \widetilde{Q}_r^T b$$

via backward substitution. Then we find our r-th iterate by setting

$$x_r := Q_r c_r.$$

5.2.4 Summary and final Remarks: GMRES with Arnoldi

i) We only need one matrix-vector product $v \mapsto Av$ in each step! This product can be delivered to the solver GMRES as some sort of black box function.

ii) **An initial guess $x_0 \neq 0$:**
Consider

$$x_r := x_0 + p_r, \quad \text{for some} \quad x_0 \neq 0,$$

then

$$Ax_r = b \quad \Leftrightarrow \quad A(x_0 + p_r) = b \Leftrightarrow \quad Ap_r = b - Ax_0 = \widehat{b}.$$

Then we could solve the auxiliary system to obtain $p := \texttt{GMRES}(A, \widehat{b})$ and set $x := x_0 + p$. The above algorithm can therefore easily be extended to allow for an initial guess x_0 other than the zero vector. Also note that the Krylov subspaces are then generated based on the initial residual $\widehat{b} = b - Ax_0$.

iii) **Preconditioning:**
Let us consider $N \in \mathrm{GL}_n(\mathbb{R})$, $N \approx A^{-1}$, where $v \mapsto Nv$ is also easy to compute. Then

$$Ax = b \quad \Leftrightarrow \quad \underbrace{NA}_{=:\widetilde{A}} x = \underbrace{Nb}_{=:\widetilde{b}} \Leftrightarrow \quad \widetilde{A}x = \widetilde{b}.$$

Then we could solve the equivalent system to obtain $x := \texttt{GMRES}(\widetilde{A}, \widetilde{b})$.

What is actually a good precondition? Quite a bit of research has been put into this question over the last decades. What one can say: Krylov subspace methods converge quickly if the eigenvalues appear in clusters away from zero.

iv) **Restarted GMRES:**

- We need to store $Q_r := (q_1, \dots, q_r)$, $\widetilde{Q}_r := (\widetilde{q}_1, \dots, \widetilde{q}_r)$ and $\widetilde{R}_r := (\widetilde{r}_1, \dots, \widetilde{r}_r)$. Thus, if the above algorithm runs many iterations, then $Q_r, \widetilde{Q}_r$ may become large (usually dense) matrices ($n \gg 1$), then the advantage of the sparsity of A is lost.
- Idea: Perform only a fixed number of iterations, say $m = 20 - 40$ and then restart `GMRES` with initial guess x_m. The resulting algorithm is sometimes coined `GMRES(m)`. You will later learn about a similar idea in the context of the L-BFGS method.
- Remark: If A is symmetric and positive definite, then one can show that we do not need the whole history $q_1, \dots, q_r$. This is the power of the CG-method which you will investigate next semester.

v) **Another perspective and variant:** Let us consider

$$\min_{x_r \in K_r(A,b)} \|Ax_r - b\|_2^2 = \min_{c_r \in \mathbb{R}^r} \|AQ_r c_r - b\|_2^2 = \min_{c_r \in \mathbb{R}^r} \|Q_{r+1} H_{r+1,r} c_r - b\|_2^2$$

$$\overset{(\#)}{=} \min_{c_r \in \mathbb{R}^r} \underbrace{\|H_{r+1,r} c_r - \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \|_2^2}_{\text{we can evaluate this residual without computing } x_r!} \qquad (*)$$

$$(\#): \; q_1 := \frac{b}{\|b\|} \;\Rightarrow\; b = q_1 \|b\| \;\Rightarrow\; Q_{r+1}^T b = \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \;\Rightarrow\; b = Q_{r+1} \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Let $\widehat{Q}_{r+1}\widehat{R}_r := H_{r+1,r}$ be a QR-decomposition, then the normal equation associated to (*) could be solved by:

$$\widehat{R}_r c_r = \widehat{Q}_{r+1}^T \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

This results in a different variant of the GMRES and the following adaptions would apply:

- In step 2: We could derive such a QR-decomposition $\widehat{Q}_{r+1}\widehat{R}_r := H_{r+1,r}$ from our arrays by setting

  $$\widehat{Q}_{r+1} := Q_{r+1}^T \widetilde{Q}_r, \quad \widehat{R}_r := \widetilde{R}_r$$

  because then

  $$\widetilde{Q}_r \widetilde{R}_r \overset{!}{=} AQ_r = Q_{r+1}H_{r+1,r} \Leftrightarrow \quad Q_{r+1}^T \widetilde{Q}_r \widetilde{R}_r = H_{r+1,r} \Leftrightarrow \quad \widehat{Q}_{r+1}\widehat{R}_r = H_{r+1,r} \quad (*).$$

- In step 3: Our normal equation could then by written as

  $$\widetilde{R}_r c_r = \widetilde{Q}_r^T b \quad \Leftrightarrow \quad \widetilde{R}_r c_r = (Q_{r+1}^T \widetilde{Q}_r)^T \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Leftrightarrow \quad \widehat{R}_r c_r = \widehat{Q}_{r+1} \begin{pmatrix} \|b\| \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

All in all we obtain the following algorithm:

INPUT: $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$
OUTPUT: approximation $x_r \in K_r(A,b)$ to the exact solution $A^{-1}b$

GMRES(A, b, $x_0 = 0$, `tol = 1e-6`, `maxiter=None`, $N = I$):
    $b = b - Ax_0$ //account for initial guess
    $A = NA$, $b = Nb$ //account for preconditioner
    //Initialization:
    $q_1 := \frac{b}{\|b\|_2}$, $Q_1 := [q_1]$
    $v := Aq_1$, $\tilde{q}_1 := \frac{v}{\|v\|_2}$, $\tilde{Q}_1 := [\tilde{q}_1]$, $\tilde{R}_1 = [\|v\|_2]$
    for $r = 2, ..., \min(n, \texttt{maxiter})$ **do**
        //STEP 1: use Arnoldi to find column q_r by orthogonalizing v against $q_1, \ldots, q_{r-1}$
        $q_r, h_{r-1} := \texttt{Arnoldi\_step}(Q_{r-1}; v)$ //we don't need h_{r-1}
        $Q_r := [Q_{r-1}, q_r]$
        $v := Aq_r$

        //STEP 2: use Arnoldi to find columns $\tilde{q}_r$, $\tilde{r}_r$ by orthogonalizing v against $\tilde{q}_1, \ldots, \tilde{q}_{r-1}$
        $\tilde{q}_r, \tilde{r}_r := \texttt{Arnoldi\_step}(\tilde{Q}_{r-1}; v)$
        $\tilde{Q}_r := [\tilde{Q}_{r-1}, \tilde{q}_r]$, $\tilde{R}_r := [\tilde{R}_{r-1}, \tilde{r}_r]$

        //STEP 3: solve auxiliary least squares problems to obtain coordinates
        $c_r := \texttt{solve\_triangular}(\tilde{R}_r, \tilde{Q}_r^\top b)$
        $x_r := Q_r c_r$
        //Attention: Evaluate the original residual here:
        if $\|N^{-1}(Ax_r - b)\|_2 < \texttt{tol}$ **then**
            break
        end
    end