Stat 607: Numerical analysis of the least squares problem

These notes were prepared using [2] and [1] which I would recommend for further reading. Throughout this class, all vector $u \in \mathbb{R}^m$ are column vectors. If A is a matrix, $A_{\star,j}$ (or $A_{\cdot,j}$) denotes the j-th column of A.

1. Introduction

Let $X \in \mathbb{R}^{m\times m}$ $m \geq 2$ be a matrix and $y \in \mathbb{R}^m$ a column vector. Solving linear system is one of the most fundamental computational problem in modern science. We want to find $b \in \mathbb{R}^m$ such that

$$\text{(E)} \qquad Xb = y. \tag{1}$$

If X is non-singular, then we know that (E) admits a unique solution $\hat{b} = X^{-1}y$. But in practice we never compute $\hat{b}$ that way, because it is more expensive (and unnecessary) to compute the inverse matrix X^{-1}. Rather we solve (E) directly typically using Gaussian elimination of the QR decomposition. The most common approach being Gaussian elimination. This corresponds to doing a LU decomposition of X.

Definition 1.1. X admits a LU decomposition if there exists a lower triang. matrix L and an upper triang. matrix U such that

$$X = LU.$$

Clearly from an LU decomposition, we can solve (E) easily by back-substitution. In practice (out of necessity or for increased stability) one often introduces pivoting, which consists in swapping rows or columns. If X is symmetric positive definite, we can exploit symmetry in the LU decomposition to obtain a special decomposition known as the Cholesky decomposition.

Definition 1.2. A symmetric positive definite matrix X admits a Cholesky decomposition if there exists a lower triang. matrix R such that

$$X = RR'.$$

We will study these matrix decompositions and how to derive a solution to problem (E).

Suppose now that $X \in \mathbb{R}^{m\times n}$ with $m \geq n$. In this case Problem (E) is overdetermined and typically has no solution. Instead we try to find b that solve

$$\text{(P)} \qquad \min_b \|Xb - y\|_2 , \tag{2}$$

where $\|x\|_2 := \sqrt{\sum_{i=1}^m x_i^2}$ is the Euclidean norm. (P) is the *least squares problem*. The least squares problem is well known in Statistics since the least squares is one of our core inference procedures.

It is easy to show (see below) that a solution to (P) must satisfy the so-called normal equation

$$X'Xb = X'y. \tag{3}$$

Call P the orthogonal projector on the linear space spanned by the column of X, and define $r = y - Py$, so that $y = Py + r$. Since $\langle u, r\rangle = 0$ for any $u \in \text{span}(x_1, \ldots, x_n)$, we see that:

$$\|Xb - y\|_2^2 = \|Xb - Py - r\|_2^2 = \|Xb - Py\|_2^2 + \|r\|_2^2 .$$

Since $\|r\|_2$ does not depend on b, minimizing $\|Xb - y\|_2^2$ in b is equivalent to minimizing $\|Xb - Py\|_2^2$ and we can always choose b so as to make this latter term equal to zero, since $Py \in \text{span}(x_1, \ldots, x_n)$. Thus: a solution to tProblem (P) always exist. Call it $\hat{b}$ (if it is not unique, call the chosen one $\hat{b}$). $\hat{b}$ satisfies:

$$X\hat{b} = Py. \tag{4}$$

Also, since $y - Py$ is orthogonal to X, $X'(y - Py) = 0$ which also writes $X'(y - X\hat{b})$ and we get back the normal equation (3).

If X is full-rank then $X'X$ is invertible and a solution to (P) exists and is unique and given by $\hat{b} = (X'X)^{-1}X'y$. Again this solution is rarely computing by taking inverse. There are three main approaches for solving (P).

- We can solve (P) by solving the normal equation (3) using the Cholesky decomposition as discussed above. Indeed, assuming X is full-rank, the Cholesky decomposition of $X'X$ gives $X'X = RR'$ where R is lower triangular matrix. Then we can solve the least squares problem by solving (in w) by back-substitution the linear system $Rw = X'y$ and then solving in b by back-substitution the linear system $R'b = w$. This approach tends to be the fastest but is often unstable. The reason: the matrix $X'X$ tends to be more ill-conditioned than the original matrix X. This can be measured by calculating the condition number of $X'X$

  $$\kappa_2(X'X) = \|X'X\|_2\|(X'X)^{-1}\|_2 = \kappa_2(X)^2.$$

  More generally, the conditioning number of a matrix A wrt the matrix norm $\|\cdot\|_p$ is

  $$\kappa_p(A) = \|A\|_p\|A^{-1}\|_p.$$

  For $p = 2$, the SVD decomposition shows that $\kappa_2(A) = \frac{\overline{\sigma}(A)}{\underline{\sigma}(A)}$, where $\overline{\sigma}(A)$ (resp. $\underline{\sigma}(A)$) is the largest (resp. smallest) singular value of A.
- A more common approach for solving (P) is through the QR decomposition.

Definition 1.3. Let $X \in \mathbb{R}^{m\times n}$ a matrix. We say that X admits a QR decomposition if

$$X = QR,$$

where $Q \in \mathbb{R}^{m\times n}$ is a matrix with orthogonal columns ($Q'Q = I_n$) and $R \in \mathbb{R}^{n\times n}$ is a square matrix upper triangular ($r_{ij} = 0$ for $i > j$).

If $X = QR$ is the QR decomposition of X, then $P = QQ'$. This is not hard to see: QQ' is an othogonal projector whose range is the linear space spanned by the columns of Q which is the as span$(x_1, \ldots, x_n)$. From (4), we get $QR\hat{b} = Py$. Premutiplying by Q' and noting that $Q'Q = I_n$, we get: $R\hat{b} = Q'Py = Q'QQ'y = Q'y$. We are then left with:

$$R\hat{b} = Q'y.$$

This equation can then be solved easily by back-substitution as R is upper triangular.

- There is yet another approach for solving the least squares problems via the SVD (singular value decomposition).

  Definition 1.4. Let $X \in \mathbb{R}^{m\times n}$ a matrix. We say that X admits a SVD decomposition if

  $$X = U\Sigma V', \quad \text{or} \quad X = \hat{U}\hat{\Sigma}V',$$

  where $V \in \mathbb{R}^{n\times n}$ and $\hat{U} \in \mathbb{R}^{m\times m}$ are an orthogonal matrices, $U \in \mathbb{R}^{m\times n}$ has orthogonal columns, and $\Sigma \in \mathbb{R}^{m\times n}$ and $\hat{\Sigma} \in \mathbb{R}^{n\times n}$ are diagonal matrices with nonnegative diagonal elements arranged in nonincreasing order called singular values of X.

  Thus V is nonsingular. As before $P = UU'$ and the equation (4) becomes $\Sigma V'b = U'y$ and b is trivially obtained as $b = V\Sigma^{-1}U'y$. This algorithm is often more stable than the two previously mentioned strategy.

We will explore these different decompositions and compare these different strategies in solving (P).

Direct methods described above require $O(mn^2)$ operations to solve (P) and can become computationally expensive to use when $m \geq n$ are very large. Iterative methods attempt to solve (E) and (P) only approximately by solving iteratively lower dimensional problems. These methods are also better suited to exploit sparsity structures. With the large interest of high-dimensional data analysis in Statistics, exposure to these iterative methods can prove useful.

1.1. Condition numbers and stability

Let X, Y be two normed spaces. We use $\|\cdot\|$ to denote the norms on X, Y and $\mathcal{L}(X, Y)$ (the space of linear maps from X to Y). So when we write $\|x\|$, the element x determines which

norm we are referring to. Let $f:\ X \to Y$ be a function. We say that f is an ill-posed or ill-conditioned problem if small perturbation of x result in large variation of $f(x)$. f is well-posed or well-conditioned otherwise. We can make this idea precise by using the condition number of f at x defined as

$$\kappa(f,x) \stackrel{\text{def}}{=} \lim_{r\downarrow 0} \sup_{\|u\|\le r} \frac{\|f(x+u)-f(x)\| \,/\, \|f(x)\|}{\|u\| \,/\, \|x\|}. \tag{5}$$

Assume that f is differentiable with derivative $J(x)$ at x. Recall that the derivation $J(x)$ of f is the linear map $J(x):\ X \to Y$ such that $f(x+u) = f(x) + J(x)\cdot u + o(\|u\|)$. In this case, we define the norm of $J(x)$ as $\|J(x)\| = \sup_{\|u\|\le 1} \|J(x)\cdot u\|$. It follows that

$$\kappa(f,x) = \|J(x)\| \frac{\|x\|}{\|f(x)\|}.$$

Example 1.1. Let $f(x) = \sqrt{x}$ from $[,\infty) \to [0,\infty)$. Then, for any $x \ge 0$, $\kappa(f,x) = \frac{1}{2\sqrt{x}} \frac{x}{\sqrt{x}} = 0.5$.

Example 1.2. For any $m \ge 1$, we equip $\mathbb{R}^m$ with the norm ℓ^p, for some $p \ge 1$: $\|x\|_p \stackrel{\text{def}}{=} (\sum_{i=1}^m |x_i|^p)^{1/p}$ if $1 \le p < \infty$, and $\|x\|_\infty \stackrel{\text{def}}{=} \max_{1\le i\le m} |x_i|$. A popular choice is $p = 2$ which is the Euclidean norm.

Let $A \in \mathbb{R}^{m\times n}$. Consider $f:\ \mathbb{R}^n \to \mathbb{R}^m$ defined as $f(x) = Ax$. Then $J(x) = \frac{d}{dx} f(x) = A$. Thus $\kappa(f,x) = \|A\|_p \frac{\|x\|_p}{\|Ax\|_p}$, where the norm of A is defined as

$$\|A\|_p \stackrel{\text{def}}{=} \sup_{\|x\|_p \le 1} \|Ax\|_p.$$

Assume that $m = n$ and A is non-singular. Then $x = A^{-1}Ax$ which implies that $\|x\|_p = \|A^{-1}Ax\|_p \le \|A^{-1}\|_p \|Ax\|_p$ and implies that $\kappa(f,x) \le \|A^{-1}\|_p \|A\|_p$. We define the condition number of the matrix A as

$$\kappa_p(A) \stackrel{\text{def}}{=} \|A^{-1}\|_p \|A\|_p.$$

$\kappa_p(A)$ plays a fundamental role in matrix numerical analysis.

Example 1.3. Let $\mathcal{M}_m$ be the space of $m \times m$ matrices equipped with the norm $\|A\|_p$, and fix $b \in \mathbb{R}^m$. Consider $f:\ \mathcal{M}_m \to \mathbb{R}^m$ defined as $f(A) = A^{-1}x$. This map is differentiable and its derivative is the linear map $J(A):\ \mathcal{M}_m \to \mathbb{R}^m$ defined by $J(A)M = -A^{-1}MA^{-1}b$. Therefore

$$\kappa(f,A) = \|J(A)\| \frac{\|A\|_p}{\|A^{-1}b\|} = \left(\sup_{\|M\|_p \le 1} \|A^{-1}MA^{-1}b\|_p \right) \frac{\|A\|_p}{\|A^{-1}b\|} \le \|A\|_p \|A^{-1}\|_p.$$

Again we find out that the ill-posedness of this problem is controlled by the ill-posedness of the matrix A as defined by its condition number $\kappa_p(A)$.

The condition number of a matrix A admits a simpler and more intuitive form when we use the Euclidean norm ($p = 2$).

Proposition 1.1. *Let $A \in \mathcal{M}_m$ assumed non-singular. Then*

$$\kappa_2(A) = \frac{\sigma_1}{\sigma_m},$$

where σ_1 is the largest singular value of A and σ_m its smallest singular value.

Proof. This follows from the SVD decomposition of A and was proved in class. In fact we will come back to this when studying the SVD decomposition. □

We end this discussion of stability with a result that relates the condition number of a problem with the behavior of numerical algorithm that tries to compute that problem. The main message is that even with the best possible algorithm, rounding error can greatly affect the computation of ill-posed problems. Thus let $f: X \to Y$ be a problem, and let $\tilde{f}(x)$ be a computer implementation of $f(x)$. This include the computer representation of x, the application of a particular algorithm to compute $f(x)$ and the computer representation of the solution denoted $\tilde{f}(x)$.

Definition 1.5. We say that a computer implementation $\tilde{f}(x)$ of a problem $f(x)$ is stable if there exists $\tilde{x} \in X$ that satisfies $\frac{\|\tilde{x}-x\|}{\|x\|} = O(\epsilon_{\mathsf{mac}})$, such that $\tilde{f}(x) = f(\tilde{x})$.

In other words, the algorithm gives an output that is the correct answer for a closely related problem. We can also view this definition as saying that rounding error aside, the algorithm computes the exact solution to the problem. The constant in $O(\epsilon_{\mathsf{mac}})$ does not depend on x.

Proposition 1.2. *If a computer implementation $\tilde{f}(x)$ of a problem $f(x)$ is stable then*

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = \kappa(f, x)\, O\left(\epsilon_{\mathsf{mac}}\right).$$

2. Chapter 1. Linear equations: the LU and Cholesky decompositions

2.1. Gaussian elimination

Let $X \in \mathbb{R}^{m\times m}$ and $y \in \mathbb{R}^m$. We are interested in solving our Problem (E): $Xb = y$. We assume that X is non-singular. Gaussian elimination is the idea of successively transforming the equation $Xb = y$ into an upper triangular system of linear equations that can be easily solved by back substitution. It corresponds to finding $L_1, L_2, \ldots, L_{m-1}$ lower triangular matrices such that

$$L_{m-1} \cdots L_1 X = U,$$

where U is upper triangular. This factorization results in what is known as a LU decomposition of X

$$X = LU, \;\; L = L_1^{-1} \cdots L_{m-1}^{-1},$$

where L is lower triangular and U upper triangular. The **LU** factorization of X allows to easily solve Problem (E). Indeed, we get $Ub = L^{-1}y = L_{m-1} \cdots L_1 y$, which is easily solved by back substitution.

Lemma 2.1. *1. If $a, b \in \mathbb{R}^m$ are such that $a'b = 0$, then $(I - ab')^{-1} = (I + ab')$.*

2. Let $a_k, b_k \in \mathbb{R}^m$ $1 \le k \le n$ such that $a_j' b_i = 0$ whenever $i < j$. Then

$$\prod_{k=1}^{n} (I_m + a_k b_k') = I_m + \sum_{k=1}^{n} a_k b_k'.$$

Proof. By simple verification for 1., and by recursion for 2. □

Remark 2.1. This implies the recursion $\prod_{k=1}^{n}(I + a_k b_k') = \prod_{k=1}^{n-1}(I + a_k b_k') + a_n b_n'$.

For an arbitrary matrix $A \in \mathbb{R}^{m\times m}$. For $1 \le k \le m-1$, assume $A_{k,k} \neq 0$ and let $\ell_k(A) \in \mathbb{R}^m$ defined as

$$\ell_k(A) = \frac{1}{A_{k,k}} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ A_{k+1,k} \\ \vdots \\ A_{m,k} \end{bmatrix}.$$

Then define $L_k(A) = I_m - \ell_k(A) e_k'$, where e_j is the j-th Cartesian unit vector of $\mathbb{R}^m$. The matrices $L_k(A)$ are the building blocks of Gaussian elimination. We have the following properties.

Lemma 2.2.

$$L_k(A)A_{\cdot,k} = \begin{bmatrix} A_{k,1} \\ \vdots \\ A_{k,k} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

And for any vector $u \in \mathbb{R}^m$ such that $u = (u_1, \ldots, u_{k-1}, 0 \ldots, 0)'$, we have $L_k(A)u = u$. More generally, for $u \in \mathbb{R}^m$, $L_k(A)u = u - u_k \ell_k(A)$.

Back to matrix X, the above shows that $L_1(X)X$ is a matrix where the first column is $(X_{1,1}, 0 \ldots, 0)'$. Setting $X_1 = L_1(X)X$, $L_2(X_1)X_1$ has the same first column as X_1 but the second column is zero below the diagonal, etc... until we obtain $X_{m-1} = L_{m-1}(X_{m-2})X_{m-2}$ which is upper triangular.

Algorithm 2.1 (Gaussian elimination without pivot)**.** *Set $U = X$, and $M = I_m$. For $j = 1$ to $m - 1$:*

1. *Define $\ell_j = (0. \ldots, 0, U_{j+1,j}/U_{j,j}, \ldots, U_{m,j}/U_{j,j})'$, and $L_j = I_m - \ell_j e_j'$.*
2. *Set $M = L_j M$, and $U = L_j U$.*

When Algorithm 2.1 succeeds, we get:

$$L_{m-1}L_{m-1} \cdots L_1 X = U \quad \text{is upper triangular.}$$

If $M = L_{m-1}L_{m-1} \cdots L_1$, it is clear that M is lower-triang. and Lemma 2.1 show that $L = M^{-1} = \prod_{j=1}^{m-1}(I_m + \ell_j e_j') = I_m + \sum_{k=1}^{m-1} \ell_j e_j'$. From this we get:

Proposition 2.1. *When Gaussian elimination procedure (Algorithm 2.1) carries through, it generates the LU factorization of the matrix X, $X = LU$. The matrix L is non-singular and L^{-1} is obtain from L by negating its off-diagonal terms.*

But the Gaussian elimination as described above will often fail even when X is nonsingular. For example:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

The answer to this problem is to swap the column or the rows of X. For example, in the example above, $X = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ or $X = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ will work (is already good). As we know, permutation of rows or column is equivalent to multiplying by permutation matrices to the left (for rows) or

to the right (for columns). A permutation matrix is a matrix with 1 on all its diagonal exempt at the two diagonal to be permuted. For example $P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ is a permutation matrix that will permute the two first columns or the two first rows.

So we see the idea. When in the Gaussian elimination, we encounter a 0, we swap rows (partial pivoting). Even if the current pivot is not zero, for better stability of the algorithm, it can still be beneficial to swap rows. We get the following result.

Theorem 2.1. *For every square matrix X, there exist permutation matrix P_j and lower triangular L_j such that $L_{m-1}P_{m-1}\cdots L_1P_1X = U$ is upper-triangular.*

Remark 2.2. The theorem comes from the fact that if at a given step of the LU decomposition all terms below the diagonal are zeros (which will happen for a singular matrix), no action is needed, we set the corresponding P and L matrices to I_m and move on to the next iteration.

It is possible to write $L_{m-1}P_{m-1}\cdots L_1P_1$ as $L'_{m-1}\cdots L'_1P_{m-1}\dots P_1$, for some lower-triangular matrices L'_i. Then, setting $L = \left(L'_{m-1}\cdots L'_1\right)^{-1}$, the theorem states that any matrix X admits the decomposition

$$PX = LU,$$

for a lower-triang. matrix L, an upper-triang. matrix U and a permutation matrix P.

From this result we can solve Problem (E) by solving the back-substitution the equation $Ub = My$, where $M = L_{m-1}P_{m-1}\cdots L_1P_1$.

Algorithm 2.2 (Gaussian elimination with pivot)**.** *Set $U = X$ and $M = I$. For $j = 1$ to $m-1$:*

1. *Find k such that $|U_{kj}| = \max_{j\le i\le m}|U_{i,j}|$. Set P_j as the permutation matrix that permute rows k and j. Set $U = PU$ and $M = PM$.*
2. *Define $\ell_j = (0\dots,0,U_{j+1,j}/U_{j,j},\dots,U_{m,j}/U_{j,j})'$, with $\ell_j = 0 \in \mathbb{R}^m$ if $U_{j,j} = 0$. Set $L_j = I_m - \ell_j e'_j$.*
3. *Set $U = L_jU$, and $M = L_jM$.*

Of course, in practice we do not need to compute the matrix M (only My), and we can save time in computing L_jU. This leads to the following practical algorithm for solving Problem (E).

Algorithm 2.3 (Solving $Xb = y$ by Gaussian elimination with pivot)**.** *1. Given X and y, set $U = X$. For $j = 1$ to $m-1$:*

(a) Find k such that $|U_{kj}| = \max_{j\le i\le m}|U_{i,j}|$.

(b) *Swap* $U_{j,j:m}$ *and* $U_{k,j:m}$. *Swap* y_j *and* y_k.

(c) *For* $i = j+1$ *to* m, *do the following. Set* $\ell_{i,j} = U_{i,j}/U_{j,j}$, *set* $y_i = y_i - y_j * \ell_{i,j}$, *and set* $U_{i,j+1:m} = U_{i,j+1:m} - U_{j,j+1:m} * \ell_{i,j}$.

2. *Set* $b_m = y_m/U_{m,m}$. *For* $k = m-1$ *to* 1. *Set* $b_k = (y_k - \sum_{j=k+1}^{m} U_{k,j}b_j)/U_{k,k}$.

The workload of the algorithm is dominated by the calculations $U_{i,j+1:m} = U_{i,j+1:m} - U_{j,j+1:m} * \ell_{i,j}$ in step 1.(c). Thus the number of operations in solving the equation $Xb = y$ by Gaussian elimination is a big O of

$$\sum_{j=1}^{m-1}\sum_{i=j+1}^{m} 2(m-j) = 2\sum_{j=1}^{m-1}(m-j)^2 = O\left(\frac{2m^3}{3}\right).$$

2.2. The Cholesky decomposition

The Cholesky factorization is one of the most useful in Statistics as we deal a lot with symmetric matrices. $X \in \mathbb{R}^{m\times m}$ is called symmetrix if $X' = X$. A symmetric matrix X is called semi-definite positive if $u'Xu \geq 0$ for any column vector u. If in addition $u'Xu = 0$ if and only if $u = 0$, we call X positive definite.

Lemma 2.3. *If* X *is a symmetric positive definite matrix and* $A \in \mathbb{R}^{m\times n}$, $m \geq n$ *is a matrix of full rank, then* $A'XA$ *is also symmetric positive definite. In particular,* $x_{ii} > 0$ *for all* $i \in \{1,\ldots,m\}$ *and any sub-matrix* $(X_{ij})_{i,j\in I}$, $I \subseteq \{1,\ldots,m\}$ *is also symmetric and positive definite.*

Theorem 2.2. *Any symmetric positive definite matrix admits a decomposition*

$$X = RR',$$

where R *is a lower triangular matrix. The decomposition is unique if we require* $R_{ii} > 0$. *Such decomposition is called a Cholesky decomposition of* X.

Proof. By Lemma 2.3, $X_{11} > 0$. Starting with Gaussian elimination on X, and with a slight modification of the matrix L_1, we get:

$$\begin{pmatrix} 1/\sqrt{X_{11}} & 0 \\ -\frac{X_{2:m,1}}{X_{11}} & I_{m-1} \end{pmatrix} X = \begin{pmatrix} \sqrt{X_{11}} & X'_{2:m,1}/\sqrt{X_{11}} \\ 0 & X_{2:m,2:m} - X_{2:m,1}X'_{2:m,1}/X_{11} \end{pmatrix}$$

Set $R_1 = \begin{pmatrix} 1/\sqrt{X_{11}} & 0 \\ -\frac{X_{2:m,1}}{X_{11}} & I_{m-1} \end{pmatrix}$. Now, since X is symmetric, we can continue the Gaussian elimination to the right and get the block matrix:

$$R_1XR'_1 = \begin{pmatrix} 1 & 0 \\ 0 & X_{2:m,2:m} - X_{2:m,1}X'_{2:m,1}/X_{11} \end{pmatrix}.$$

Set $X^{(1)} = X_{2:m,2:m} - X_{2:m,1}X'_{2:m,1}/X_{11}$. By Lemma 2.3, $R_1 X R_1'$ is positive definite and so is $X^{(1)}$. Therefore the same decomposition can be carried over: with $R_{(2)} = \begin{pmatrix} 1/\sqrt{X_{11}^{(1)}} & 0 \\ -\frac{X_{2:(m-1),1}^{(1)}}{X_{11}^{(1)}} & I_{m-2} \end{pmatrix}$, we have

$$R_{(2)} X^{(1)} R'_{(2)} = \begin{pmatrix} 1 & 0 \\ 0 & X^{(2)} \end{pmatrix}.$$

Therefore

$$\begin{pmatrix} 1 & 0 \\ 0 & R_{(2)} \end{pmatrix} R_1 X R_1' \begin{pmatrix} 1 & 0 \\ 0 & R_{(2)} \end{pmatrix}' = \begin{pmatrix} 1 & 0 \\ 0 & R_{(2)} X^{(1)} R'_{(2)} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & X^{(2)} \end{pmatrix}.$$

Continuing this way, we conclude that there exists a lower triangular matrix R such that $RXR' = I_m$. Thus

$$X = R^{-1}(R^{-1})'.$$

□

In practice, we find the Cholesky decomposition of X by solving directly the equation $RR' = X$. Indeed this equation implies that for any $1 \le j \le m$,

$$X_{\cdot,j} = \sum_{k=1}^{j} R_{j,k} R_{\cdot,k}.$$

Thus, for $1 \le j \le m$,

$$R_{j,j} = \sqrt{X_{j,j} - \sum_{k=1}^{j-1} R_{j,k}^2}, \quad \text{and for} \quad i > j, \;\; R_{i,j} = \left(X_{i,j} - \sum_{k=1}^{j-1} R_{j,k} R_{ik} \right) / R_{j,j},$$

where it is understood that $\sum_a^b \cdot = 0$ if $a > b$.

Algorithm 2.4 (Cholesky factorization)**.** *Given X.*
For $j = 1, \ldots, m$:

1. $R_{j,j} = \sqrt{X_{j,j} - \sum_{k=1}^{j-1} R_{jk}^2}$,
   (a) *For $i = j+1, \ldots, m$,* $R_{i,j} = \left(X_{i,j} - \sum_{k=1}^{j-1} R_{j,k} R_{ik} \right) / R_{j,j}$.

The number of operations is dominated by the inner loop, each iteration of which takes $2j$ operations. Thus the computing time of Cholesky decomposition is big O of

$$\sum_{j=1}^{m} \sum_{i=j+1}^{m} 2j = 2 \sum_{j=1}^{m} j(m-j) = O\left(2 \left(\frac{m^3}{2} - \frac{m^3}{3} \right) \right) = O\left(\frac{m^3}{3} \right).$$

Thus Cholesky decomposition is 2 times faster than Gaussian elimination.

3. Chapter 2. Direct Methods for the least squares problem

Suppose now that $X \in \mathbb{R}^{m\times n}$ with $m \geq n$. In this case Problem (E) is overdetermined and typically has no solution. Instead we try to find b that solves

$$\text{(P)} \qquad \min_b \|Xb - y\|_2 , \tag{6}$$

where $\|x\|_2 := \sqrt{\sum_{i=1}^m x_i^2}$ is the Euclidean norm. (P) is the *least squares problem.*

3.1. Gram-Schmidt orthogonalization

Consider the following problem:

Problem (O): *For $X \in \mathbb{R}^{m\times n}$, with columns $X_{\star,1}, \ldots, X_{\star,n}$, find orthonormal vectors $q_1, \ldots, q_n \in \mathbb{R}^m$ such that span$(X_{\star,1}, \ldots, X_{\star,j}) \subseteq$ span$(q_1, \ldots, q_j)$ for $j = 1, \ldots, n$.*

The first important point is to realize that Problem (O) is equivalent to the existence of a QR decomposition for X (Definition 1.3 above).

Proposition 3.1. *Problem (O) is equivalent to the question of existence of a QR decomposition for X.*

Proof. $X = QR$ means precisely $X_{\star,j} = \sum_{k=1}^n Q_{\star,k} R_{k,j}$ for each $j = 1, \ldots, n$. So, if Q has orthogonal columns and R is upper triang., then $X = QR$ is equivalent to $X_{\star,j} \in \text{span}(Q_{\star,1}, \ldots, Q_{\star,j})$ for each $j = 1, \ldots, n$. □

Problem (O) is easily solved by the Gram-Schmidt orthogonalization. Suppose that X is full rank. Take $q_1 = \frac{X_{\star,1}}{\|X_{\star,1}\|}$. Suppose that we have orthonormal $q_1, \ldots, q_j$ such that $\text{span}(q_1, \ldots, q_j) = \text{span}(X_{\star,1}, \ldots, X_{\star,j})$ for $j < n$. Then define:

$$v_{j+1} = X_{\star,j+1} - \sum_{k=1}^{j} \langle X_{\star,j+1}, q_k \rangle \, q_k, \tag{7}$$

Then check that $\langle v_{j+1}, q_k \rangle = 0$ for $k = 1, \ldots, j$. Moreover $\|v_{j+1}\|_2$ cannot be 0, otherwise (7) will contradict the assumption that X is full rank. Then define $q_{j+1} = \frac{v_{j+1}}{\|v_{j+1}\|_2}$. Then $q_1, \ldots, q_{j+1}$ are orthonormal and clearly $\text{span}(q_1, \ldots, q_{j+1}) = \text{span}(X_{\star,1}, \ldots, X_{\star,j+1})$. Problem (O) is therefore solved. By Proposition 3.1, we have just proved:

Theorem 3.1. *Any matrix $X \in \mathbb{R}^{m\times n}$ of full rank column has a QR decomposition. The decomposition is unique if we insist that $r_{ii} > 0$ for $i = 1, \ldots, n$.*

Actually more is true:

Theorem 3.2. *Any matrix $X \in \mathbb{R}^{m\times n}$ has a QR decomposition.*

Proof. The theorem is true if X is full rank (shown above). Suppose X is not full rank. Then there exists steps j in the Gram-Schmidt orthogonalization for which v_j as in (7) will be 0. In which case, choose z arbitrary provided $v_j' = z - \sum_{k=1}^{j-1} \langle z, q_k\rangle\, q_k \neq 0$. Then set $q_j = v_j'/\left\|v_j'\right\|_2$. Verify that we still have $\mathsf{span}(X_{\star,1},\ldots,X_{\star,j}) = \mathsf{span}(q_1,\ldots,q_{j-1}) \subset \mathsf{span}(q_1,\ldots,q_j)$. □

Notice from (7) that

$$X_{\star,j} = \|X_{\star,j} - \sum_{k=1}^{j-1} \langle X_{\star,j}, q_k\rangle\, q_k\| q_j + \sum_{k=1}^{j-1} \langle X_{\star,j}, q_k\rangle\, q_k.$$

Compare with $X_{\star,j} = \sum_{k=1}^{j} R_{k,j} Q_{\star,k}$, we identify that $R_{jj} = \|X_{\star,j} - \sum_{k=1}^{j-1} \langle X_{\star,j}, q_k\rangle\, q_k\|$ and $R_{kj} = \langle X_{\star,j+1}, q_k\rangle$, for $k < j$. This provides the details for understanding the following algorithm.

Algorithm 3.1 (QR by Gram-Schmidt)**.** *Given X.*
For $j = 1,\ldots,n$:

1. v=$X_{\star,j}$. *For $i = 1$ to $j-1$.*
   (a) $R_{i,j} = \langle X_{\star,j}, Q_{\star,i}\rangle$.
   (b) $v = v - \langle X_{\star,j}, Q_{\star,i}\rangle\, Q_{\star,i}$.
2. $Q_{\star,j} = \frac{v}{\|v\|}$.
3. $R_{j,j} = \|v\|$.

3.2. Modified Gram-Schmidt orthogonalization

The G-S procedure is best understood in terms of projection. The procedure works as follows. First normalize $X_{\star,1}$ to get $q_1 = Q_{\star,1}$. Then project $X_{\star,2}$ on q_1 and normalize the residuals to obtain q_2. At step j, project $X_{\star,j}$ on the linear span $\mathsf{span}(q_1,\ldots,q_{j-1})$ and normalize the residuals to get q_j. The next lemma explains that the residual in projecting $X_{\star,j}$ on the linear span $\mathsf{span}(q_1,\ldots,q_{j-1})$ can also be obtained as follows. Project $X_{\star,j}$ on $\mathsf{span}(q_1)$ and form the residual. Then project the residual on q_2 and form new residual that you project on q_3 etc... This approach leads to a slightly different algorithm for carrying the G-S orthogonalization which turns out to be more stable.

Lemma 3.1. *Let $q_1,\ldots,q_n$ be orthonormal vectors. Then:*

$$I - \sum_{k=1}^{n} q_k q_k' = \prod_{k=1}^{n} \left(I - q_k q_k'\right). \tag{8}$$

This implies among other things that for any $j > 2$ and $x \in \mathbb{R}^m$,

$$\langle q_j, x\rangle = \left\langle q_j, \prod_{i=1}^{j-1}(I - q_i q_i')x \right\rangle. \tag{9}$$

Proof. Easily check by induction upon noticing that $(I - q_\ell q_\ell')q_j q_j' = q_j q_j'$, for $\ell < j$. □

Now, take v_j the residu of the projection of x_j on $\text{span}(q_1, \ldots, q_{j-1})$ as given in (7). We can actually rewrite as:

$$v_j = X_{\star,j} - \sum_{k=1}^{j-1} \langle X_{\star,j}, q_k\rangle\, q_k = X_{\star,j} - \sum_{k=1}^{j-1} q_k q_k' X_{\star,j} = \left(I - \sum_{k=1}^{j-1} q_k q_k'\right) X_{\star,j} \quad = \quad \prod_{k=1}^{j-1} \left(I - q_k q_k'\right) X_{\star,j},$$

where the last equality uses Lemma 3.1. The modifed version of the algorithm consists in computing v_j through formula $\prod_{k=1}^{j-1}(I - q_k q_k')\, X_{\star,j}$ which is numerically more stable than $X_{\star,j} - \sum_{k=1}^{j-1}\langle X_{\star,j}, q_k\rangle\, q_k$. Finally, notice from (9) that $\langle q_j, X_{\star,i}\rangle = \left\langle q_j, \prod_{k=1}^{j-1}(I - q_k q_k')\, X_{\star,i}\right\rangle$. for $j < i$.

Algorithm 3.2 (QR by Modified Gram-Schmidt)**.** *Given X.*
For $j = 1, \ldots, n$:

1. $R_{j,j} = \|X_{\star,j}\|$.
2. $Q_{\star,j} = X_{\star,j}/R_{j,j}$. *For $i = j+1$ to n.*
   (a) $R_{j,i} = \langle X_{\star,i}, Q_{\star,j}\rangle$.
   (b) $X_{\star,i} = X_{\star,i} - \langle X_{\star,i}, Q_{\star,j}\rangle\, Q_{\star,j}$.

In both cases the number of operations in dominated by the inner loop. So for large m and n, the number of operations to perform the QR decomposition by Gram-Schmidt is

$$\sum_{j=1}^{n}\sum_{i=j+1}^{n}(4m-1) = O\left(2mn^2\right).$$

3.3. The Householder triangularization

As above let $X \in \mathbb{R}^{m\times n}$ be a matrix. Here we will mix the notations x_j and $X_{1:m,j}$ to denote the j-th column of X. Moreover we will write $X_{i:j,l:k}$ to denote the corresponding submatrix of X. In the Householder triangularization, we build the QR decomposition in another special way. We left-multiply X by orthogonal matrix $Q_1, \ldots, Q_n \in \mathbb{R}^{m\times m}$ such that the end result $Q_n \cdots Q_1 X = R \in \mathbb{R}^{m\times n}$ is a upper triangular matrix. Since orthogonal matrices are non-singular, defining $Q = Q_1' \cdots Q_n'$, we get a QR decomposition of X:

$$X = QR.$$

Notice that this QR decomposition is slightly different from the previous one. Here Q is a unitary matrix (square orthogonal) and R is $\mathbb{R}^{m\times n}$. But the two forms are equivalent.

The question is then: can we and how to build orthogonal matrices $Q_1,\dots,Q_n$ such that $Q_n\cdots Q_1X = R$ is a upper triangular matrix. It is not hard to see that the following choices work: take Q_1 orthogonal such that $Q_1x_1 = \|x_1\|\,e_m$, where x_1 is the first column of X, $e_m = (1,0\dots,0)'\in\mathbb{R}^m$. For $j=2,\dots,n$, take

$$Q_j = \begin{pmatrix} I_{j-1} & 0 \\ 0 & F_j \end{pmatrix}, \tag{10}$$

where $F_j\in\mathbb{R}^{(m-j+1)\times(m-j+1)}$ is such that $F_jX_{j:m,j} = \|X_{j:m,j}\|\,e_{m-j+1}$. The notation $X_{j:m,j}$ denotes the vector of $\mathbb{R}^{m-j+1}$ formed by taking the elements $X_{j,j},\dots,X_{m,j}$ of the matrix X. Note that F_j is doing on $X_{j:m,j}$ exactely what Q_1 is doing on x_1, the first column of X. Q_1 and the F_j can be built using the following lemma:

Lemma 3.2. *Let $x\in\mathbb{R}^m$ and define $v_1 = x-\|x\|_2\,e_m$ (resp. $v_2 = x+\|x\|_2\,e_m$) and $Q_i = I-2\frac{v_iv_i'}{v_i'v_i}$. Then Q_i is a symmetric orthogonal matrix and $Q_1x = \|x\|_2\,e_m$ and $Q_2x = -\|x\|_2\,e_m$.*

This Lemma tells us precisely, given $X_{j:m,j}$, how to build F_j such that $F_jX_{j:m,j} = \|X_{j:m,j}\|\,e_{m-j+1}$: by taking $F_j = I_{m-j+1}-2\frac{vv'}{v'v}$, with $v = X_{j:m,j}-\|X_{j:m,j}\|_2\,e_{m-j+1}$ or $v = X_{j:m,j}+\|X_{j:m,j}\|_2\,e_{m-j+1}$. The choice $+/-$ is not innocuous. A good choice is to take $v = X_{j:m,j}+sign(X_{j,j})\,\|X_{j:m,j}\|_2\,e_{m-j+1}$ where $sign(x) = 1$ if $x\geq 0$ and -1 otherwise.

Algorithm 3.3. *[QR factoriation by Householder] Given X:*
For $j=1$ to n:

1. $x = X_{j:m,j}$; $v = x + sign(x_1)\,\|x\|_2\,e$.
2. $X_{j:m,j:n} = \left(I-\frac{2}{v'v}vv'\right)X_{j:m,j:n}$

If needed, we can easily recover the matrices Q_j (using (10)) and obtain $Q = Q_1'\cdots Q_n'$ by multiplication. But often this is not necessary. Suppose that we only want to calculate $Q'y = Q_n\cdots Q_1y$ for some vector $y\in\mathbb{R}^m$. This will be the case if we are trying to solve the least squares problem. The following algorithm generates $Q'y$:

Algorithm 3.4. *Given y: For $j=1$ to n:*

- $y_{j:m} = y_{j:m} - \frac{2}{v'v}(vv')y_{j:m}$, *where v is the same as in the Algorithm 3.3.*

4. Chapter 3. Singular value decomposition

4.1. The SVD

So far we have seen the LU and QR factorization. The LU factorization is mostly a trick factorization to solve linear system as it does not reveal much about the structure of X. The QR gives more insight particularly in understanding the linear space spanned by the columns of X. One of the most useful matrix decomposition that gives much information about X is the singular value decomposition (SVD). A matrix $X \in \mathbb{R}^{m\times n}$ admits a SVD if there exist orthogonal matrix $U \in \mathbb{R}^{m\times m}$, orthogonal matrix $V \in \mathbb{R}^{n\times n}$ and diagonal matrix $\Sigma \in \mathbb{R}^{m\times n}$ with non-negative diagonal elements in nonincreasing order such that $X = U\Sigma V'$. The diagonal elements of Σ are called singular values of X, the columns of U are called left-singular vectors and the columns of V are called right singular vectors.

Theorem 4.1. *Any matrix $X \in \mathbb{R}^{m\times n}$ admits a SVD.*

Proof. In class, I gave a proof along the lines of what is done in the course's textbook. Here is another proof adapted from [**?**].

Since $X'X$ is symmetric semi-positive definite, all its eigenvalues are nonnegative real numbers. Let $\sigma_1^2 > \cdots > \sigma_r^2 > 0$ be the positive eigenvalues of $X'X$ and $\sigma_{r+1} = \cdots = \sigma_m = 0$. Let $(v_1, \ldots, v_n)$ an orthonormal basis of $\mathbb{R}^n$, made of eigenvectors of $X'X$. Set $S = \mathsf{diag}(\sigma_1, \ldots, \sigma_r)$, $V_1 = (v_1, \ldots, v_r)$, and $V_2 = (v_{r+1}, \ldots, v_n)$. Then $X'XV_1 = V_1S^2$, so that $S^{-1}V_1'(X'X)V_1S^{-1} = I_r$. Now, if we set $U_1 = XV_1S^{-1}$, then we can rewrite this as $U_1'U_1 = I_r$. Similarly, we have $(X'X)V_2 = 0$, so that $V_2'(X'X)V_2 = 0$. Hence $XV_2 = 0$. Choose U_2 so that (U_1, U_2) is an orthonormal basis of $\mathbb{R}^m$. Then

$$U'XV = \begin{pmatrix} U_1' \\ U_2' \end{pmatrix} (XV_1 \;\; XV_2) = \begin{pmatrix} U_1'XV_1 & U_1'XV_2 \\ U_2'XV_1 & U_2'XV_2 \end{pmatrix} = \begin{pmatrix} S & 0 \\ U_2'U_1S & 0 \end{pmatrix} = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix}.$$

So that $X = U\Sigma V'$, as needed. □

A SVD has many useful consequences that are briefly recalled here. Many of these results where derived in class.

Proposition 4.1. *$\mathsf{Rank}(X)$ (the rank of X) is equal to the number of non-zeros singular values of X and the SVD provides a very robust approach for computing the rank of a matrix.*

Recall that $\|X\|_2 \stackrel{\text{def}}{=} \sup_{u\neq 0} \frac{\|Xu\|_2}{\|u\|_2}$ and $\|X\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n X_{ij}^2} = \sqrt{\mathsf{Tr}(X'X)}$. $\|X\|_2$ is called the ℓ^2-norm, or the spectral norm of X, whereas $\|X\|_F$ is called the Frobenius norm of X.

Proposition 4.2. *$\|X\|_2 = \sigma_1$ and $\|X\|_F = \sqrt{\sum_{k=1}^r \sigma_k^2}$.*

Proposition 4.3. *The singular values of X are the square root of the eigenvalues of $X'X$ or XX'. Furthermore, if X is symmetric, the singular values of X are the absolute values of the eigenvalues of X.*

Proposition 4.4. *Any rank r matrix $X \in \mathbb{R}^{m\times n}$ can be written as the sum of r rank-one matrices:*

$$X = \sum_{j=1}^{r} \sigma_j u_j' v_j'.$$

Furthermore, for $1 \leq q \leq r$, $\sum_{j=1}^{q} \sigma_j u_j' v_j'$ solves

$$\min_{B\in\mathbb{R}^{m\times n},\ \mathsf{Rank}(B)\leq q} \|X - B\|_\star,$$

where $\|\cdot\|_\star$ is either the spectral norm of the Frobenius norm.

4.2. SVD computation

We now detail the standard two-steps approach used for computing SVD. Most software use this general approach, although specific implementations are often more sophisticated than what will be discussed. This two-step approach is a modification of ideas used for computing eigenvalues. So we focus the discussion on the task of computing eigenvalues and eigenvectors of a given matrix and we will indicate the modifications needed to solve the SVD problem.

Let $X \in \mathbb{R}^{m\times m}$ be a matrix. We say that X admits a Schur factorization if there exist an orthogonal matrix X and an upper triangular matrix R such that

$$X = QRQ'.$$

Thus the Schur factorization somewhat relaxes the eigenvalue decomposition by allow R to be upper triangular. But notice that in the Schur decomposition, Q is assumed orthogonal. It is useful to think of this decomposition as follows: there exists another orthonormal basis with change of basis matrix Q' such that the linear transform represented by X has matrix R is this new basis. We say that X and R are similar: they represent the same linear transformation. This implies that λ is eigenvalue of X with eigenvector u, if (and only if) λ is eigenvalue of R with eigenvector $Q'u$.

Notice also that as an upper triangular matrix, the eigenvalues of R are given by its diagonal: $\mathsf{det}(\lambda I_m - R) = \prod_{i=1}^{m}(\lambda - R_{ii}) = 0$ iif $\lambda = R_{ii}$, $i \in \{1,\ldots,m\}$. The real usefulness of the Schur factorization comes from the following.

Theorem 4.2. *Any matrix $X \in \mathbb{R}^{m\times m}$ admits a Schur decomposition.*

Currently, the predominant strategy for deriving the Schur factorization of a matrix X is to reduce X to a Hessenberg matrix (a matrix with zeros below the first sub-diagonal) or a tridiagonal matrix (a symmetric Hessenberg matrix). This step is carried on using Householder reflectors. In a second step, an iterative algorithm known as the QR algorithm is used to compute the eigenvalues and eigenvectors of the Hessenberg matrix.

4.2.1. Reduction to Hessenberg matrix

Definition 4.1. A matrix $H \in \mathbb{R}^{m\times m}$ is a Hessenberg matrix if for all $1 \leq i, j \leq m$, $H_{ij} = 0$ whenever $i > j + 1$.

Given $X \in \mathbb{R}^{m\times m}$, consider the matrix $Q_1 = \begin{pmatrix} I_1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix}$, where $\tilde{Q}_1 = I_{m-1} - 2v_1v_1'/(v_1'v_1)$, where $v_1 = x+\mathsf{sign}(x_1)\|x\|_2 e$, where $x = X_{2:m,1}$. In other words, Q_1 is a Householder reflector that "operates" on $X_{2:m,1}$. See Section 3.3 for basic properties of Householder's reflectors. It follows that

$$Q_1'X = \begin{pmatrix} I_1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix}\begin{pmatrix} X_{11} & X_{1,2:m} \\ X_{2:m,1} & X_{2:m,2:m} \end{pmatrix} = \begin{pmatrix} X_{11} & X_{1,2:m} \\ \star & \\ 0 & \\ \vdots & \tilde{Q}_1X_{2:m,2:m} \\ 0 & \end{pmatrix}.$$

where $\star = \mathsf{sign}(X_{2,1})\|X_{2:m,1}\|_2$. Therefore

$$Q_1'XQ_1 = \begin{pmatrix} I_1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix}\begin{pmatrix} X_{11} & X_{1,2:m} \\ X_{2:m,1} & X_{2:m,2:m} \end{pmatrix}\begin{pmatrix} I_1 & 0 \\ 0 & \tilde{Q}_1 \end{pmatrix} = \begin{pmatrix} X_{11} & X_{1,2:m}\tilde{Q}_1 \\ \star & \\ 0 & \\ \vdots & \tilde{Q}_1\left(X_{2:m,2:m}\right)\tilde{Q}_1 \\ 0 & \end{pmatrix}.$$

The result is that the first column of $Q_1'XQ_1$ is like the first column of a Hessenberg matrix. Now, suppose that for some $j \geq 2$, the first $j-1$ columns of X are like the first $j-1$ columns of a Hessenberg matrix. This means precisely this: $X_{k\ell} = 0$ for $1 \leq \ell \leq j-1$ and $k > \ell + 1$. Then we can repeat the process above to transform X into a matrix where the first j columns are like the first j columns of a Hessenberg matrix. By continuing this process for $j = 1$ to $m-2$, we reduce X into an Hessenberg matrix (that is $X_{k\ell} = 0$ for $k > \ell + 1$, $1 \leq \ell \leq m-2$).

To do the work at the j-th step, we use $Q_j = \begin{pmatrix} I_j & 0 \\ 0 & \tilde{Q}_j \end{pmatrix}$, where $\tilde{Q}_j = I_{m-j} - 2v_j v_j'/(v_j' v_j)$, where $v_j = x + \mathsf{sign}(x_1)\|x\|_2 e$, where $x = X_{j+1:m,j}$. Therefore

$$Q_j' X Q_j = \begin{pmatrix} X_{1:j,1:j} & X_{1:j,j+1:m}\tilde{Q}_j \\ \begin{pmatrix} 0_{j+1:m,1:j-1} & \tilde{Q}_j X_{j+1:m,j} \end{pmatrix} & \tilde{Q}_j \left(X_{j+1:m,j+1:m}\right) \tilde{Q}_j \end{pmatrix}.$$

This leads to the following algorithm

Algorithm 4.1. *[Reduction to Hessenberg matrix by Householder] Given X:*
For $j = 1$ to $m-2$:

1. $x = X_{j+1:m,j}$, $v = x + \mathsf{sign}(x_1)\,\|x\|_2\, e$, *and* $v_j = v/\|v\|_2$.
2. $X_{j+1:m,j:m} = X_{j+1:m,j:m} - 2v_j \left(v_j' X_{j+1:m,j:m}\right)$.
3. $X_{1:m,j+1:m} = X_{1:m,j+1:m} - 2\left(X_{1:m,j+1:m} v_j\right) v_j'$.

This algorithm leads to the decomposition $X = QHQ'$, where $Q = Q_1 \cdots Q_{m-2}$, and H is Hessenberg. Of course, in practice the matrix Q_j are never formed in the algorithm. Instead the vectors $v_1, \ldots, v_{m-2}$ are saved and one can always compute Qu or $Q'u$ if need be. For large m, the number of operations performed in Algorithm 4.1 is O of

$$\sum_{j=1}^{m-2} 5(m-j)(m-j+1) + 5m(m-j) = O\left(\frac{25}{6}m^3\right).$$

This count is slightly higher than the count $\sum_{j=1}^{m-2} 4(m-j)(m-j+1) + 4m(m-j) = O\left(\frac{10}{3}m^3\right)$ reported in [2].

Remark 4.1. If X is symmetric, H is a tri-diagonal matrix (as a symmetric Hessenberg matrix). It is then possible to use symmetry to reduce the computational load in Algorithm 4.1. This is left as an exercise

Example 4.1. The function hess in matlab performs a reduction to Hessenberg form. Here is an example.

```
 X=rand(4,4)
X =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
>> hess(X) %reduction to Hessenberg form
```

```
ans =
    0.8147   -1.2135   -0.8559    0.1663
   -1.2926    0.8399    1.2997    0.0668
    0         0.6987   -0.0233   -0.3394
    0         0         0.0323   -0.4196
```

Consider the following example where X is symmetric.

```
>> U=rand(4,4);X=U'*U
X =
    2.5647    1.8781    1.9452    1.0231
    1.8781    2.0246    1.4695    1.0652
    1.9452    1.4695    1.7409    1.1118
    1.0231    1.0652    1.1118    0.9585
>> hess(X)
ans =
    0.4642   -0.0653         0         0
   -0.0653    0.2674    0.7101         0
         0    0.7101    5.5986   -1.8486
         0         0   -1.8486    0.9585
```

Golub-Kahan bidiagonalization

If we have the freedom of choosing two orthogonal matrices as in $X = UHV'$ instead of $X = QHQ'$, it is possible to reduce further H into a bi-diagonal matrix. This is the Golub-Kahan bidiagonalization process. We have the added flexibility that X can be rectangular. The Golub-Kahan bidiagonalization process is the Step 1 in computing singular values, the same way the reduction to Hessenberg (Algorithm 4.1) is the Step 1 in computing eigenvalues. We will not insist too much on this algorithm.

Let $X \in \mathbb{R}^{m\times n}$. The Golub-Kahan proceeds as follows. The derivation is similar to the derivation of Algorithm 4.1. But more efficient implementation is possible.

Algorithm 4.2. *[Golub-Kahan bidiagonalization] Given X:*
For $j = 1$ to $n-2$:

1. *Get $u_j = X_{j:m,j} + \mathsf{sign}(X_{jj})\|X_{j:m,j}\|_2 e$, and set $u_j = u_j/\|u_j\|_2$.*
2. *$X_{j:m,j:n} = X_{j:m,j:n} - 2\left(u_j' X_{j:m,j:n}\right) u_j$.*
3. *Get $v_j = X'_{j,j+1:n} + \mathsf{sign}(X_{j,j+1})\|X_{j,j+1:n}\|_2 e$, and set $v_j = v_j/\|v_j\|_2$.*
4. *$X_{1:m,j:n} = X_{1:m,j:n} - 2\left(X_{1:m,j:m} v_j\right) v_j'$.*

Repeat Step 1-2 for $j = n-1$ and $j = n$.

4.2.2. The QR algorithm

For most small to moderate size matrices, the eigenvalue problem is solved using an algorithm known as the QR algorithm. The algorithm was developed in the late 50's early 60's. In its simplest form, the algorithm is very easy to implement. Throughout this section, we restrict the analysis to symmetric matrices to ensure that the eigenvalues are real. Let $X \in \mathbb{R}^{m\times m}$ be a symmetric matrix with eigenvalues $\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_m|$, and eigenvectors $q_1, \ldots, q_m$.

Algorithm 4.3 (Pure QR algorithm)**.** *Set $X_0 = X$. For $k = 1, 2, \ldots$.*

1. *Perform a QR decomposition of X_{k-1}: $Q_k R_k = X_{k-1}$. That is, do a QR decomposition of X_{k-1}.*
2. *Set $X_k = R_k Q_k$.*

Here is a MATLAB example that implements the algorithm.

Example 4.2.
```
>> U=rand(4,4);X=U'*U;
    0.7204    1.0790    1.0045    1.1399
    1.0790    1.8197    1.6081    1.8114
    1.0045    1.6081    1.5011    1.5855
    1.1399    1.8114    1.5855    2.0747
>> lambda=eig(X)'; %find the eigenvalues of X
    0.0138    0.0643    0.2026    5.8352
>> A=X; for i = 1:4 [q,r]=qr(A); A=r*q; end
>> r
    5.8352   -0.0000   -0.0000    0.0000
         0    0.1908    0.0639   -0.0002
         0         0    0.0682    0.0000
         0         0         0    0.0138
```

We see that in 4 iterations, the diagonal of R is already pretty close to the eigenvalues of X.

The natural question is? what is really going on in the QR algorithm? Perhaps the simplest idea underlying the QR algorithm is the the so-called power iteration. This is a fundamental idea that has been exploited is many areas of mathematics: as $n \to \infty$, $A^n x$, behaves like $\lambda_1^n \langle x, q_1 \rangle q_1$, where λ_1 is the largest (in absolute value) eigenvalue of X with associated eigenvector q_1. For us, A is a matrix. The same idea is also used in Markov chain theory, where A is a transition kernel

operating on a spaces of measures or on a space of functions. In our case here this leads to the following algorithm.

Algorithm 4.4 (Power iteration algorithm)**.** *Let* $v^{(0)} \in \mathbb{R}^m$*. For* $k = 1, 2 \ldots$*,*

1. *Set* $w = Xv^{(k-1)}$*, and* $v^{(k)} = w/\|w\|_2$*.*
2. *Set* $\lambda_k = \{v^{(k)}\}^\intercal X v^{(k)}$*.*

Theorem 4.3. *Suppose that* $\lambda_1| > |\lambda_2|$ *and* $q_1' v^{(0)} \neq 0$*. Then as* $n \to \infty$*,*

$$\|v^{(n)} - \epsilon q_1\|_2 = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^n\right), \quad \textit{and} \quad |\lambda_n - \lambda_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2n}\right),$$

for some $\epsilon \in \{1, -1\}$*.*

Proof. We have $v^{(n)} = Xv^{(n-1)}/\|Xv^{(n-1)}\|_2$. It is easy to check that $v^{(n)} = X^2 v^{(n-2)}/\|X^2 v^{(n-2)}\|_2 = X^n v^{(0)}/\|X^n v^{(0)}\|_2$. Now, $(q_1, \ldots, q_m)$ is an orthonormal basis of $\mathbb{R}^m$, so that $v^{(0)} = \sum_{i=1}^m \alpha_i q_i$. Notice that $\alpha_1 = q_1' v^{(0)} \neq 0$, and we have

$$X^n v^{(0)} = \sum_{i=1}^m \alpha_i X^n q_i = \sum_{i=1}^m \alpha_i \lambda_i^n q_i = (\lambda_1)^n \left(\alpha_1 q_1 + \sum_{i=2}^m \alpha_i \left(\frac{\lambda_i}{\lambda_1}\right)^n q_i\right).$$

This easily implies that

$$\left\| \frac{X^n v^{(0)}}{\|X^n v^{(0)}\|_2} - \mathsf{sign}(\alpha_1) q_1 \right\|_2 = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^n\right).$$

In order to deal with λ_n, we introduce the function $r : \mathbb{R}^m \setminus \{0_m\} \to \mathbb{R}$ defined as $r(x) = \frac{x'Xx}{\|x\|_2^2}$. This function is also known as the Rayleigh quotient of X at x. It is easy to verify that r is twice continuously differential everywhere on $\mathbb{R}^m \setminus \{0_m\}$, and its gradient is given by $\nabla r(x) = \frac{2(Xx - r(x)x)}{\|x\|_2^2}$. Hence, $x \in \mathbb{R}^m$ is an eigenvector for X if and only if $\nabla r(x) = 0$, and the eigenvalue is $\lambda = r(x)$. Furthermore, if q is an eigenvector for X with eigenvalue λ, by Taylor expansion,

$$r(x) = r(q) + \nabla r(q)(x - q) + \frac{1}{2}(x - q)' \nabla^2 r(q_\star)(x - q) = \lambda + \frac{1}{2}(x - q)' \nabla^2 r(q_\star)(x - q),$$

for some $q_\star$ on the linear segment from x to q. Now we apply this Taylor expansion around $\mathsf{sign}(\alpha_1) q_1$, and by noting that $\lambda_n = \{v^{(n)}\}^\intercal X v^{(n)} = r(v^{(n)})$, get

$$|\lambda_n - \lambda_1| = \left|\frac{1}{2}\left(v^{(n)} - \mathsf{sign}(\alpha_1) q_1\right)' \nabla^2 r(q_\star) \left(v^{(n)} - \mathsf{sign}(\alpha_1) q_1\right)\right| = O\left(\left\|v^{(n)} - \mathsf{sign}(\alpha_1) q_1\right\|_2^2\right).$$

□

We see that in power iteration, although $X^n v^{(0)}$ might not converge to anything, $X^n v^{(0)}/\|X^n v^{(0)}\|_2$ converges to q_1. A more fruitful way to look at the power iterations is in terms of the one-dimensional linear sub-space generated by $X^n v^{(0)}$. This viewpoint is the correct one if we want

to generalized the idea to higher dimension. For U, V two p-dimensional linear sub-spaces of $\mathbb{R}^m$, we define the distance between U and V as

$$\mathsf{d}(U,V) \stackrel{\text{def}}{=} \sqrt{\sum_{i=1}^{p} \mathsf{d}^2(u_i, V)},$$

where $\{u_1, \ldots, u_p\}$ is an orthonormal basis of U, and where $\mathsf{d}(u,V) \stackrel{\text{def}}{=} \min_{v\in V} \|u - v\|_2$. If $\{v_1, \ldots, v_p\}$ is an orthonormal basis for V, then $\mathsf{d}(u,V) = \|u - p_V(u)\|_2 = \|u - \sum_{j=1}^{p}(u'v_j)v_j\|_2$. So that

$$\mathsf{d}(U,V) = \sqrt{p - \sum_{i=1}^{p}\sum_{j=1}^{p}(u_i'v_j)^2}.$$

This latter expression makes it easy to see that $\mathsf{d}(A,B)$=$\mathsf{d}(A,B)$. It is also easy to verify that this is indeed a distance and that the definition does not depend on the orthonormal basis chosen. With this definition, we can check that

$$\mathsf{d}\left(\left\langle X^n v^{(0)} \right\rangle, \langle q_1 \rangle\right) = \sqrt{1 - \frac{\lambda_1^{2n}\alpha_1^2}{\sum_{j=1}^{m}\lambda_j^{2n}\alpha_j^2}} \to 0,$$

as $n \to \infty$, under the assumption that $\alpha_1 \neq 0$ and $|\lambda_1| > |\lambda_2|$.

Now, suppose that we start with $V^{(0)}$ a matrix with p columns $v_1^{(0)}, \ldots v_p^{(0)}$. Now define $V^{(n)} \stackrel{\text{def}}{=} X^n V^{(0)}$. We know from above that for each column of $v_j^{(0)}$, $X^n v_j^{(0)}$ converges to q_1. As the result, the matrix $V^{(n)}$ becomes more and more ill-conditioned as $n \to \infty$. But surprisingly, the following fundamental result holds and is the key to understanding the behavior of the QR algorithm. We omit the details.

Theorem 4.4. *Assume that $\mathbb{R}^m$ has an orthonormal basis of eigenvectors of X with eigenvalues $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_p| > |\lambda_{p+1}| \geq |\lambda_{p+1}| \geq \cdots \geq |\lambda_m|$. Under some additional regularity conditions on $V^{(0)}$,*

$$\lim_{n\to\infty} \mathsf{d}\left(\left\langle V^{(n)} \right\rangle, \langle q_1, \ldots, q_p \rangle\right) = 0.$$

The result suggest the following algorithm known as simultaneous iteration

Algorithm 4.5 (Simultaneous iteration)**.** *Let $Q^{(0)} = I_m$. For $k = 1, 2 \ldots$,*

1. *Set $Z^{(k)} = XQ^{(k-1)}$, and*
2. *Set $Q^{(k)}R^{(k)} = Z^{(k)}$ (do a QR decomposition).*

The following result was derived in class. The proof is simple.

Proposition 4.5. *In simultaneous iteration,*

$$X^n = Q^{(n)}\underline{R}^{(n)},$$

where $\underline{R}^{(n)} = R^{(n)} \cdots R^{(1)}$.

In view of this result and Theorem 4.4, we can conclude that in simultaneous iteration, the columns of the resulting matrix $Q^{(n)}$ should be approximately the eigenvectors of X. Finally we come to the QR algorithm (Algorithm 4.3).

Theorem 4.5. *From the QR algorithm given in Algorithm 4.3, define* $\underline{R}^{(n)} = R^{(n)} \cdots R^{(1)}$, *and* $\underline{Q}^{(n)} = Q^{(1)} \cdots Q^{(n)}$. *Then*

$$X^n = \underline{Q}^{(n)} \underline{R}^{(n)}, \quad \textit{and} \quad X^{(n)} = \{\underline{Q}^{(n)}\}^{\mathsf{T}} X \underline{Q}^{(n)}.$$

Proof. From the algorithm, we see that $X^{(n)} = \{Q^{(n)}\}^{\mathsf{T}} X^{(n-1)} Q^{(n)}$. This recursive formula easily leads to $X^{(n)} = \{\underline{Q}^{(n)}\}^{\mathsf{T}} X \underline{Q}^{(n)}$. The first identity can be obtained by induction. It is trivially true for $n = 1$. Assume it is true for $n-1$. We have $X^n = X X^{n-1}$. But $X^{(n)} = \{\underline{Q}^{(n)}\}^{\mathsf{T}} X \underline{Q}^{(n)}$. This implies that $\{\underline{Q}^{(n)}\} X^{(n)} = X \underline{Q}^{(n)}$, so that

$$\begin{aligned} X^n = X X^{n-1} = X Q^{(n-1)} R^{(n-1)} = \{\underline{Q}^{(n-1)}\} X^{(n-1)} R^{(n-1)} = \{\underline{Q}^{(n-1)}\} Q^{(n)} R^{(n)} R^{(n-1)} \\ = \underline{Q}^{(n)} \underline{R}^{(n)}. \end{aligned}$$

□

This result tells us that the linear space spanned by $X^n I_m$ is the same as the linear space spanned by $Q^{(n)}$. We can combine this with Theorem 4.4 to get.

Corollary 4.1. *If The regularity of Theorem 4.4 hold, then* $\lim_{n\to\infty} d\left(\left\langle \underline{Q}^{(n)} \right\rangle, \langle q_1, \ldots, q_m \rangle\right) = 0$, *and the diagonal element of* $X^{(n)}$ *converges to the eigenvalues of* X.

Remark 4.2. The QR algorithm as described in Algorithm 4.3 is not used in practice. The reason is that its convergence can be slow if the eigenvalues of X are not well separated. Accelerated version of the algorithm can be obtained by introducing the so-called shift operator which replaces X by $X - \rho I_m$. For well chosen ρ, better convergence can be obtained. We will not discuss the QR algorithm with shift. See for example the textbook [2].

Example 4.3. Here is a MATLAB illustration with $m = 5$.

```
>> U = rand(m,m);
X=U'*U;
[eigv,eigval]=eig(X)
eigv =
   -0.4429    0.3135    0.4704   -0.6461    0.2585
    0.3204   -0.3633   -0.5018   -0.6127    0.3716
```

```
   -0.4329    0.3659   -0.5191    0.3184    0.5549
   -0.1605   -0.6654    0.4086    0.2954    0.5265
    0.6986    0.4393    0.3007    0.1365    0.4582
eigval =
    0.0327         0         0         0         0
         0    0.0629         0         0         0
         0         0    0.2912         0         0
         0         0         0    0.6927         0
         0         0         0         0    5.8698

%We do a QR algo. with K=m=5 iterations.
K=m;
X0=X;Q=eye(m);
for i=1:K
    [q,r]=qr(X0);
    X0=r*q;
    Q=Q*q; %accumulate the q
end
Q =
    0.2586   -0.6683    0.4384   -0.3237   -0.4354
    0.3717   -0.5874   -0.5313    0.3706    0.3117
    0.5549    0.3434   -0.5028   -0.3763   -0.4240
    0.5265    0.2751    0.4223    0.6616   -0.1762
    0.4582    0.1217    0.3071   -0.4224    0.7088
X0 =
    5.8698    0.0003    0.0000    0.0000    0.0000
    0.0003    0.6917    0.0196   -0.0000   -0.0000
    0.0000    0.0196    0.2921    0.0001    0.0000
    0.0000   -0.0000    0.0001    0.0629   -0.0007
    0.0000   -0.0000    0.0000   -0.0007    0.0327
```

4.3. Back to SVD

The eigenvalues methods discussed are adapted for SVD. Given $X \in \mathbb{R}^{m\times n}$, one possible strategy is to form $\Lambda = X'X$. Then if the SVD for X is $X = U\Sigma V'$, then we check that $\Lambda = V\Sigma^2 V'$. Then the QR algorithm can be applied on Λ to get V and Σ. From that, we deduce U by solbing $XV = U\Sigma$. Typically the QR algorithm is not directly applied on Λ. Rather a Hessenberg reduction or a bi-diagonal reduction of Λ is done first, the result of which is fed to the QR algorithm. The main reason for this strategy is that the QR algorithm can be implemented more efficiently on Hessenberg matrices. But this strategy is often not used because, as we have seen, the matrix $X'X$ will tend to be more ill-conditioned than X.

Another strategy that avoids $X'X$, is to introduce the matrix $T = \begin{pmatrix} 0 & X' \\ 0 & X \end{pmatrix}$ and to observe that if $X = U\Sigma V'$, then

$$T \begin{pmatrix} V & V \\ U & -U \end{pmatrix} = \begin{pmatrix} V & V \\ U & -U \end{pmatrix} \begin{pmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{pmatrix}.$$

Therefore finding the eigenvalues/vectors for T reveals the SVD decomposition of X. This is a more common strategy. Here again, the matrix is first reduced to a bi-diagonal form. More details can be obtained in [2].

5. Chapter 4: Introduction to iterative methods

5.1. Sparse matrices

A sparse matrix is defined somewhat vaguely as a matrix with few non-zero entries. Proper handling of sparse matrices is key in taking advantage of the iterative methods described here. We take a very quick look at ideas and methods for handling sparse matrices. The key idea is to find efficient methods for storing the non-zero elements of sparse matrices and be able to perform common matrix operations.

Let $X \in \mathbb{R}^{m\times n}$ be a sparse matrix with $n_\star$ non-zero entries. There are many storage format for sparse matrices. The simplest storage scheme for sparse matrices is the coordinate format. This requires three vectors (A_X, I_X, J_X). $A_X \in \mathbb{R}^{n_\star}$ and contains the non-zero elements of X in any order. $I_X \in \mathbb{R}^{n_\star}$ contains the row indi ces of the elements of A_X and $J_X \in \mathbb{R}^{n_\star}$ contains the column indices of the elements of X.

Example 5.1. Let

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 0 & 0 \\ 6 & 0 & 7 \end{pmatrix}.$$

Then $A_X = (1, 3, 6, 7)$, $I_X = (1, 2, 3, 3)$, $J_X = (1, 1, 1, 3)$

A more efficient storage is the so-called compressed sparse row format (CSR format). In the CSR format X is represented by three vectors $A_X \in \mathbb{R}^{n_\star}$, $J_X \in \mathbb{R}^{n_\star}$ and $I_X \in \mathbb{R}^{m+1}$. A_X holds the non-zero elements listed by row. J_X holds their corresponding column indices and $I_X(i)$ is the position in A_X (and J_X) where the i-th row of X starts. Conveniently, we set $I_X(m+1) = n_\star + 1$.

Example 5.2. For X as in the above example, the CSR format of X is $A_X = (1, 3, 6, 7)$, $J_X = (1, 1, 1, 3)$, $I_X = (1, 2, 3, 4)$.

The CSR format is preferred because a number of matrix algebra can be efficiently performed.

For $b \in \mathbb{R}^n$, suppose we wan to compute $[Xb](i) = \sum_{j=1}^{n} X(i, j)b(j)$. Given $i \in \{1, \ldots, m\}$, the range $I_X(i) \cdots I_X(i + 1) - 1$ holds the indices of the non-zero elements of $X(i, \cdot)$. Hence, the following returns the vector $y = Xb$.

Algorithm 5.1 (Computing $y = Xb$ for X in CSR format)**.** *For $i = 1$ to m,*

1. $K_1 = I_X(i)$, $K_2 = I_X(i + 1) - 1$
2. $y(i) = \sum_{\ell=K_1}^{K_2} A_X(\ell)b\left(J_X(\ell)\right)$.

Another useful operation is solving the linear system $Xb = y$ assuming that X upper-triangular.

Algorithm 5.2 (Solving $Xb = y$ for $X \in \mathbb{R}^{m\times m}$ in CSR format)**.** *Set* $b = 0$*. For* $i = m$ *to* 1*,*

1. $K_1 = I_X(i)$, $K_2 = I_X(i+1) - 1$
2. $Z = \sum_{\ell=K_1}^{K_2} A_X(\ell) b\left(J_X(\ell)\right)$.
3. $b(i) = \left(y(i) - Z\right)/A_X(K_1)$.

5.2. Iterative methods: an overview

Consider solving the linear system $Xb = y$ for $X \in \mathbb{R}^{m\times m}$ and $b \in \mathbb{R}^m$. We can solve this iteratively as follows. Given an initial solution $b_0 \in \mathbb{R}^m$, the residual is $r_0 = y - Xb_0$. We can form the next solution as $b_1 = b_0 + \alpha_1 r_0$, for some stepsize $\alpha_1 > 0$. More generally, given b_n, we form

$$b_{n+1} = b_n + \alpha_{n+1} r_n, \quad \text{where} \quad r_n = y - Xb_n.$$

Notice that $r_{n+1} = y - Xb_{n+1} = y - Xb_n - \alpha_{n+1} X r_n = (I - \alpha_{n+1} X) r_n$, and we have

$$r_n = \prod_{i=1}^{n} \left(I - \alpha_i X\right) r_0.$$

The following is an easy consequence

Proposition 5.1. *If* $\{\alpha_n,\ n \geq 1\}$ *is such that* $\sum -\log \|I - \alpha_n X\|_2 = \infty$*, then* $\|r_n\|_2 \to\ 0$*.*

Notice that $b_{n+1} = b_n + \alpha_{n+1} r_n = \sum_{i=0}^{n} \alpha_{i+1} r_i$. Since $r_i = \prod_{j=1}^{i} \left(I - \alpha_j X\right) r_0 \in \mathsf{span}(r_0, Xr_0, \ldots, X^i r_0)$, we conclude that $b_n \in \mathsf{span}(r_0, Xr_0, \ldots, X^n r_0)$.

Definition 5.1. $\mathcal{K}_n(X, r_0) \stackrel{\text{def}}{=} \mathsf{span}\left(r_0, Xr_0 \ldots, X^{n-1} r_0\right)$ is called a n-dimensional Krylov sub-space of $\mathbb{R}^m$.

Thus $b_n \in \mathcal{K}_{n+1}(X, r_0)$, with $r_0 = y - Xb_0$. Typically, $b_0 = 0$, so that $r_0 = y$. In general, the type of recursive methods $b_{n+1} = b_n + \alpha_{n+1} r_n$ described in the proposition above are rarely used in practice. Instead, a strategy that turns out to be more promising is to build sequence $\{b_n,\ n \geq 1\}$ such that $b_n \in \mathcal{K}_n(X, r_0)$, and b_n is chosen according to some optimality condition. Another argument for searching the solution b_n in a Krylov sub-space is as follows. The minimal polynomial p of $X \in \mathbb{R}^{m\times m}$ is the monic polynomial of minimal degree such that $p(X) = 0$. For example, if X has eigenvalues $\lambda_1, \ldots, \lambda_m$ with eigenvectors $q_1, \ldots, q_m$ that form an orthonormal basis of $\mathbb{R}^m$, then the minimal polynomial of X is $p(t) = \prod_{j=1}^{m}(t - \lambda_j)$. If p has degree d, then it can be written $p(t) = \sum_{j=0}^{d} \alpha_j t^j$. Hence,

$$0 = p(X) = \alpha_0 I_m + \left(\alpha_1 + \sum_{j=2}^{d} \alpha_j X^{j-1}\right) X.$$

Therefore, if X is non-singular and $\alpha_0 \neq 0$, then $X^{-1} = -\frac{1}{\alpha_0}\left(\alpha_1 + \sum_{j=2}^{d} \alpha_j X^{j-1}\right)$, which means that $X^{-1}y \in \mathcal{K}_d(X, y)$. Thus, it makes sense to search for the solution of the equation $Xb = y$ in Krylov spaces. This general approach has given rise to a large variety of methods.

1. Ritz-Galerkin method: choose b_n such that $b_n \in \mathcal{K}_n(X, r_0)$ such that $y - Xb_n \perp \mathcal{K}_n(X, r_0)$.
2. Petrov-Galerkin method: choose b_n such that $b_n \in \mathcal{K}_n(X, r_0)$ and $y - Xb_n$ is orthogonal to some sub-space of $\mathcal{K}_n(X, r_0)$.
3. The minimum norm error method: Find $b_n \in \mathcal{K}_n(X, r_0)$ such that $\|b_n - b_\star\|_2$ is minimum.
4. The minimum norm residual method: Find $b_n \in \mathcal{K}_n(X, r_0)$ that solves $\min_{b \in \in \mathcal{K}_n(X, r_0)} y - Xb\|_2$.

There is a vast literature on iterative method for solving linear system. Many of these methods can be easily extended to solve the least squares problem. In this introduction, we focus on the minimum norm residual method, more precisely on the Generalized Minimum residual method (GMRES).

5.3. Krylov sub-spaces, Arnoldi's and Lanczos algorithms

Fix $X \in \mathbb{R}^{m \times m}$ and $r_0 \in \mathbb{R}^m$. Denote $u_i = X^{i-1} r_0$, and $U^{(k)}$ the $m \times k$ matrix with columns $u_1, \ldots, u_k$. A task of fundamental importance is to construct an orthonormal basis for $\mathcal{K}_k(X, r_0)$. As we know, this is equivalent to deriving the QR factorization of $U^{(k)}$: $U^{(k)} = Q^{(k)} R^{(k)}$. The following result allows us to construct $Q^{(k)}$ recursively. Let q_j denotes the j-th column of $Q^{(k)}$. In the theorem $e_k = (0, \ldots, 0, 1)' \in \mathbb{R}^k$.

Theorem 5.1. *Assume $U^{(k)} = Q^{(k)} R^{(k)}$, for $k \in \{1, \ldots, n\}$, where $Q^{(k)} \in \mathbb{R}^{m \times k}$ has orthogonal columns and $R^{(k)} \in \mathbb{R}^{k \times k}$ is upper-triangular. Then there exists a Hessenberg matrix $H^{(k)}$ such that*

$$XQ^{(k)} = Q^{(k)} H^{(k)} + \alpha_{k+1} q_{k+1} e_k', \tag{11}$$

where $\alpha_{k+1} = H^{(k+1)}_{k+1,k}$, $H^{(k)} = \{Q^{(k)}\}^\intercal X Q^{(k)}$. In particular,

$$Xq_k = Q^{(k)} H^{(k)}_{1:k,k} + \alpha_{k+1} q_{k+1}.$$

Proof. As a matter or re-writing things, we have

$$XU^{(k)} = U^{(k)} B^{(k)} + u_{k+1} e_k',$$

where $B^{(k)} \in \mathbb{R}^{k \times k}$ is $B^{(k)}_{i+1,i} = 1$ and $B^{(k)}_{i,j} = 0$ otherwise. Hence $XQ^{(k)} R^{(k)} = Q^{(k)} R^{(k)} B^{(k)} + u_{k+1} e_k'$, and

$$XQ^{(k)} = Q^{(k)} R^{(k)} B^{(k)} \{R^{(k)}\}^{-1} + \frac{1}{R^{(k)}_{kk}} u_{k+1} e_k'.$$

Now, if $U^{(k+1)} = Q^{(k+1)}R^{(k+1)}$, then $Q^{(k+1)} = [Q^{(k)}, q_{k+1}]$ and $R^{(k+1)} = \begin{pmatrix} R^{(k)} & \tilde{r} \\ & R^{(k+1)}_{k+1,k+1} \end{pmatrix}$, then $u_{k+1} = Q^{(k)}\tilde{r} + R^{(k+1)}_{k+1,k+1}q_{k+1}$, which gives:

$$\begin{aligned} XQ^{(k)} &= Q^{(k)}R^{(k)}B^{(k)}\{R^{(k)}\}^{-1} + \frac{1}{R^{(k)}_{kk}}\left(Q^{(k)}\tilde{r} + R^{(k+1)}_{k+1,k+1}q_{k+1}\right)e_k' \\ &= Q^{(k)}\left(R^{(k)}B^{(k)}\{R^{(k)}\}^{-1} + \frac{1}{R^{(k)}_{k,k}}\tilde{r}e_k'\right) + \frac{R^{(k+1)}_{k+1,k+1}}{R^{(k)}_{k,k}}q_{k+1}e_k' = Q^{(k)}H^{(k)} + \alpha_{k+1}q_{k+1}e_k', \end{aligned}$$

where $H^{(k)} = R^{(k)}B^{(k)}\{R^{(k)}\}^{-1} + \frac{1}{R^{(k)}_{k,k}}\tilde{r}e_k'$. By orthonormality of $Q^{(k)}$ and q_{k+1}, we see that $\{Q^{(k)}\}^{\mathsf{T}}XQ^{(k)} = H^{(k)}$, which together with $Xq_k = Q^{(k)}H^{(k)}_{1:k,k} + \alpha_{k+1}q_{k+1}$, implies that $q_{k+1}Xq_k = \alpha_{k+1}$.

It remains to show that $H^{(k)} = R^{(k)}B^{(k)}\{R^{(k)}\}^{-1} + \frac{1}{R^{(k)}_{k,k}}\tilde{r}e_k'$ is Hessenberg. It is enough to show that $\tilde{H}^{(k)} = R^{(k)}B^{(k)}\{R^{(k)}\}^{-1}$ is Hessenberg. But this is a matter of easy verification. □

The result allows us to construct recursively the vectors $Q^{(k)} = (q_1, \ldots, q_k)$. Indeed, given $(q_1, \ldots, q_k)$, we compute Xq_k and then compute $q_1'Xq_k$, $q_2'Xq_k, \ldots, q_k'Xq_k$. Then q_{k+1} is obtained by normalizing the vector

$$Xq_k - \sum_{\ell=1}^{k} H^{(k)}_{\ell,k}q_k = Xq_k - \sum_{\ell=1}^{k}\{q_\ell'Xq_k\}q_k.$$

This leads to an efficient algorithm for building the vectors q_j known as Arnoldi's algorithm.

Algorithm 5.3 (Arnoldi's algorithm)**.** *Set* $q_1 = r_0/\|r_0\|_2$.
For $j = 1$ *to* $n-1$*:*

1. $t = Xq_j$.
   For $i = 1$ *to* j
   (a) $h_{ij} = t'q_i$.
   (b) $t = t - h_{ij}q_i$.
2. $h_{j+1,j} = \|t\|_2$, *and* $q_{j+1} = t/h_{j+1,j}$.

If X is symmetric, then $H^{(k)} = \{Q^{(k)}\}^{\mathsf{T}}XQ^{(k)}$ is also symmetric and thus tri-diagonal. Therefore equation $Xq_k = Q^{(k)}H^{(k)}_{1:k,k} + \alpha_{k+1}q_{k+1}$ becomes

$$Xq_k = H^{(k)}_{k-1,k}q_{k-1} + H^{(k)}_{k,k}q_k + H^{(k+1)}_{k+1,k}q_{k+1},$$

and this greatly simplifies Arnoldi's algorithm. The resulting algorithm is known as the Lanczos iterations, which is a substantial improvement over the Arnoldi's method in terms of computing time (we spare the inner for loop) and in terms of storage (we only need to keep q_{j-1} and q_j).

5.4. The Generalized Minimum Residual method

Theorem 5.1 allows to give a quick description of GMRES. Equation (11) implies the recursion

$$XQ^{(k)} = Q^{(k+1)} H^{(k+1)}_{1:k+1,1:k}.$$

At the k-th iteration of the GMRES method, we seek to find $b_k \in \mathcal{K}_k(X, r_0)$ that solves $\min_{b\in\mathcal{K}_k(X,r_0)} \|y - Xb\|_2$. We can write $b \in \mathcal{K}_k(X, r_0)$ as $b = Q^{(k)}z$, for $z \in \mathbb{R}^k$. We assume that $b_0 = 0$, so that $r_0 = y$. In that case, $v_1 = r_0/\|r_0\|_2$. Set $\rho = \|r_0\|_2$. Then

$$\|Xb - y\|_2 = \left\|XQ^{(k)}z - \rho Q^{(k+1)}e_1\right\|_2 = \left\|Q^{(k+1)}H^{(k+1)}_{1:k+1,1:k}z - \rho Q^{(k+1)}e_1\right\|_2 = \left\|H^{(k+1)}_{1:k+1,1:k}z - \rho e_1\right\|_2.$$

So that it suffices to solve the k-dimensional least squares problems

$$\min_{z\in\mathbb{R}^k} \left\|H^{(k+1)}_{1:k+1,1:k}z - \rho e_1\right\|_2,$$

and given a solution z_k, form $b_k = Q^{(k)}z_k$.

Example 5.3. We consider the following MATLAB simulation example. The matrix $X \in \mathbb{R}^{m\times m}$ and $b_\star \in \mathbb{R}^m$ have Gaussian entries.

```
m=200;n=m;
X=randn(m,n);
b_star=randn(n,1);
sigma=0.1;
y=X*b_star;
```

Then we implement the Generalized Minimum Residual method using the Arnoldi's algorithm.

```
%%%%%%Implement a GMRES using Arnoldi's algorithm

K=180; %number of iterations

m=length(y);    b=zeros(n,1);    r0=y-X*b;    rho=norm(r0);
Q=zeros(m,K+1);    Q(:,1)=r0 / rho;     H=zeros(K+1,K);

error=zeros(K,1);     error(1) = norm(y-X*b)/norm(y);

for j=1:K
    t=X*Q(:,j);
```

```
    for i=1:j
        H(i,j)=t'*Q(:,i);
        t=t-H(i,j)*Q(:,i);
    end
    H(j+1,j)=norm(t);
    Q(:,j+1)=t/H(j+1,j);
    if j>1
        V=H(1:j,1:(j-1));
        [QV,RV]=qr(V,0);
        mu=rho*[1;zeros(j-1,1)];
        b=RV\(QV'*mu);
        b=Q(:,1:(j-1))*b;
        error(j)=norm(y-X*b)/norm(y);
    end
end
```

After $K = 180$ iterations, we obtain a solution b such that

```
norm(y-X*b)/norm(y)
ans =
    0.1208
```

Compare with the exact solution:

```
b_0=X\y
norm(y-X*b_0)/norm(y)
ans =
   3.2041e-15
```

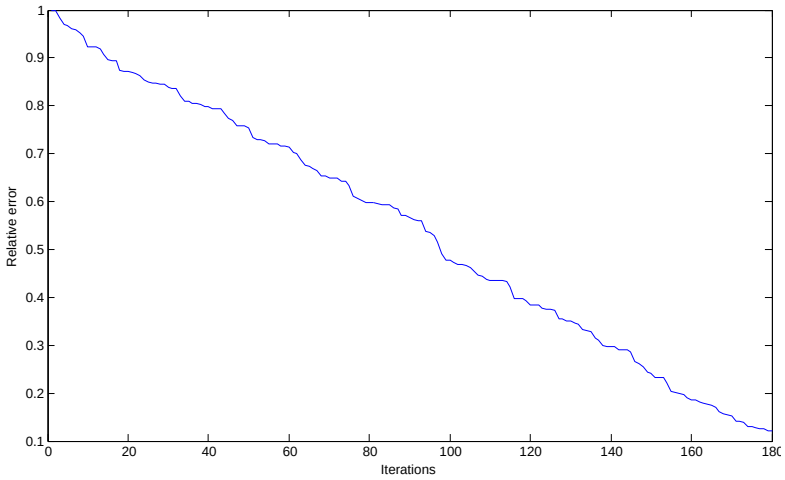

Figure 1: Relative error $\|y - Xb_n\|_2/\|y\|$ of GMRES versus the number of iterations performed.

Remark 5.1. The resolution of the least squares problem $\min_{z\in\mathbb{R}^k} \left\| H^{(k+1)}_{1:k+1,1:k} z - \rho e_1 \right\|_2$ can be done more efficiently than the above implementation. Since $H^{(k)}$ are Hessenberg matrices, Givens rotations can be used to implement a much faster QR decomposition of $H^{(k+1)}_{1:k+1,1:k}$, as done in HW3.

Remark 5.2. The GMRES algorithm can sometimes be used to solve the least squares problem by solving the normal equation $(X'X)b = X'y$, and in this case we use the Lanczos iterations to carry the QR of the Krylov space vectors. But of course, as we have seen in HW2, this approach should be avoided if the matrix X is severely ill-conditioned.

Remark 5.3. The above is a very short introduction to iterative methods for linear systems. Much more can be said on the topic and systematic discussion can be found in the two references listed below.

References

[1] Saad, Y. (2003). *Iterative Methods for sparse Linear Systems, 2nd Edition.* SIAM, Philadelphia.

[2] Trefethen, L. N. and Bau, D. (1997). *Numerical linear algebra.* SIAM, Philadelphia.