Stat 607: Orthogonalization

1. Preliminaries

We review here some basic properties of projectors. A projector is a matrix $P \in \mathbb{R}^{m\times m}$ such that $P^2 = P$. If $\mathsf{Range}(P) = S$ and $\mathsf{Ker}(P) = L$ we say that P is a projector on S along L. We say that P is an orthogonal projector if $L \perp S$. A projector is orthognal if and only if $P' = P$. If P is a projector, $I - P$ is also a projector, $\mathsf{Ker}(P) = \mathsf{Range}(I - P)$, and $\mathbb{R}^m = \mathsf{Ker}(P) \oplus \mathsf{Range}(P)$. Let $Q \in \mathbb{R}^{m\times n}$ a full-rank matrix with columns $q_1, \ldots, q_n$. The orthogonal projector on the space spanned by $q_1, \ldots, 1_n$ is $Q(Q'Q)^{-1}Q'$. If $n = 1$, the orthogonal projector on the space spanned by q_1 is $q_1q_1'/\|q_1\|_2^2$. If $q_1, \ldots, q_n$ are orthonormal $P = QQ' = \sum_{i=1}^n q_iq_i'$. Finally we recall that if P is the orthogonal projector on a subspace M then for all $x \in \mathbb{R}^m$ $\|x - Px\|_2 = \min_{y\in M} \|x - y\|_2$.

Given a matrix $X \in \mathbb{R}^{m\times n}$ one is often interested in finding a matrix $Q \in \mathbb{R}^{m\times n}$ such that $\mathsf{span}(X_{\star,1}, \ldots, X_{\star,n}) = \mathsf{span}(Q_{\star,1}, \ldots, Q_{\star,n})$. Q can then be readily used to compute QQ', the projector of the space spanned by the column of X.

Definition 1.1. *We say that $X \in \mathbb{R}^{m\times n}$ 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 upper triangular ($r_{ij} = 0$ for $i > j$).

Consider the following problem:

Problem (O): *Find orthonormal vectors $q_1, \ldots, q_n \in \mathbb{R}^m$ such that*

$$span(X_{\star,1}, \ldots, X_{\star,j}) \subseteq span(q_1, \ldots, q_j), \text{ for } j = 1, \ldots, n.$$

The first important point is to realize that

Proposition 1.2. *Problem (O) is equivalent to the 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 $j = 1, \ldots, n$. □

There are several problems in statistics where we need to solve this problem.

1.1. The least squares problem.

Consider for instance the least squares problem. Given $X \in \mathbb{R}^{m\times n}$, $y \in \mathbb{R}^m$, find $b \in \mathbb{R}^n$ that solves

$$\text{(P)} \qquad \min_b \|Xb - y\|_2, \tag{1}$$

where $\|x\|_2 := \sqrt{\sum_{i=1}^m x_i^2}$ is the Euclidean norm. The least squares problem is well known in Statistics as a core inferential procedures. 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 $u \in \text{span}(X_{\star,1}, \ldots, X_{\star,n})$, 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 $Py \in \text{span}(x_1, \ldots, x_n)$. Hence a solution to Problem (P) always exists. Call it $\hat{b}$ (if it is not unique, call the chosen one $\hat{b}$). $\hat{b}$ satisfies:

$$X\hat{b} = Py. \tag{2}$$

If $X = QR$ is the QR decomposition of X the (2) gives

$$R\hat{b} = Q'y. \tag{3}$$

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 so-called normal equation

$$X'X\hat{b} = X'y. \tag{4}$$

Hence we can solve the least-squares problem either by solving the the normal equation (4), or via the QR decomposition $X = QR$ and solving the upper triangular system (3).

1.2. **Low rank approximation of a matrix.** A common problem in the applications is to find a low-rank approximation of a given matrix $X \in \mathbb{R}^{m\times n}$. As we have seen this can be solved via the singular value decomposition of X. For instance if we want a rank $q < n$ approximation of X, from the svd $X = U\Sigma V'$ of X, by Eckart-Young theorem, $X_1 = U_1\Sigma_1 V_1'$ is the best rank q approximation of X where U_1 is the first q columns of U, V_1 the first q columns of V, and Σ_1 the $q \times q$ top-left submatrix of Σ. However computing the svd of a matrix $X \in \mathbb{R}^{m\times n}$ cost at least $O(mn^2)$, and becomes expensive to do for very large matrices. In large scale applications one can build approximate low-rank approximations using Monte Carlo methods. The QR decomposition is an important component of this type of methods. The general idea is the following. Suppose that we can find a matrix $Q \in \mathbb{R}^{m\times q}$ with orthonormal columns ($Q'Q = I_q$) such that

$$QQ'X \approx X. \tag{5}$$

Then setting $B \overset{\text{def}}{=} Q'X$, and forming the svd decomposition of $B = U_1 S_1 V_1'$ leads to

$$X \approx QU_1 S_1 V_1' = \bar{U}_1 S_1 V_1'.$$

Hence $\bar{U}_1 S_1 V_1'$ is an approximate SVD of X. The cost of building this approximation is now $O\left(2mnq + mq^2\right)$.

Monte Carlo methods and the QR decomposition is commonly used to find a matrix Q that satisfies (5). Indeed let $Z_1, \ldots, Z_{p+r} \overset{i.i.d.}{\sim} \mathbf{N}(0, I_n)$. Let $\bar{X} \in \mathbb{R}^{m\times(q+r)}$ be such that the j-th column of $\bar{X}$ is XZ_j. We can think of the columns of $\bar{X}$ as random samples from the range of X. Therefore it should not be surprising that if $\bar{X} = QR$ is the QR decomposition of $\bar{X}$, the matrix Q thus obtained satisfies (5) if the rank of X is closed to q. The constant r is called the oversampling

parameter, and is used to compensate for the random search nature of the algorithm. It is usually taken very small (< 10). In the code below it is taken as 0. The computation of $\bar{X}$ and its QR decomposition is also a $O(mq^2)$ operation. Hence the whole computation of building this low-rank approximation of X is $O\left(2mnq + mq^2\right)$. The code below illustrates a comparison with the full svd approach.

```
% Generate matrix
p = 2000;   n = 10000;
A = randn(n,20);
% A = A*randn(20,p); or A*diag(1:1:20)*randn(20,p);
A = A*randn(20,p);
q = 10;

%%% Eckhart-Young-- via SVD
t1 = tic;
[u2,sig2,v2] = svd(A,0);
time2 = toc(t1);
res2 = A - u2(:,1:q)*sig2(1:q,1:q)*(v2(:,1:q)');
error_2 = norm(res2,'fro')/norm(A,'fro');

%%% Monte Carlo approx.
t1 = tic;
Z= randn(p,q);
Action = A*Z;
[Q,R] = qr(Action,0);
B = Q'*(A);
[u,sig,v] = svd(B,0);
u1 = Q*u;
time1 = toc(t1);
res = A - u1*sig*(v');
error_1 = norm(res,'fro')/norm(A,'fro');

%%Results
[error_1,error_2,time1,time2]
ans =
    0.7050    0.6731    0.1429    9.0468
```

2. The QR factorization by Gram-Schmidt

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:

$$v_{j+1} = X_{\star,j+1} - \sum_{k=1}^{j} \langle X_{\star,j+1}, q_k \rangle \, q_k, \tag{6}$$

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 (6) will contradict the assumption that X is full rank. 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 1.2, we have just proved:

Theorem 2.1. *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 2.2. *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 (6) 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 (6) 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 2.1 (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) $v = v - R_{i,j} Q_{\star,i}$.
(2) $Q_{\star,j} = \frac{v}{\|v\|}$.
(3) $R_{j,j} = \|v\|$.

2.1. **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 2.3. *Let $q_1, \ldots, q_n$ be orthonormal vectors. Then:*

$$I - \sum_{k=1}^{n} q_k q_k' = \prod_{k=1}^{n} \left(I - q_k q_k'\right). \tag{7}$$

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

$$\langle q_j, x\rangle = \left\langle q_j, \prod_{i=1}^{j-1} (I - q_i q_i')x \right\rangle. \tag{8}$$

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 $\mathrm{span}(q_1, \ldots, q_{j-1})$ as given in (6). 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 2.3. 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 (8) 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 2.2 (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) $X_{\star,i} = X_{\star,i} - R_{j,i} Q_{\star,j}$.

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

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

Remark 1. Another well-known approach to improve on the basic Gram-Schmidt scheme is to do it twice. Mathematically, the second Gram-Schmidt should return the exact same matrix as the result of the first. In practice we get an improvement in the orthogonalization.

3. QR factorization by 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 and $Q_j^{-1} = Q_j'$, 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 $Q \in \mathbb{R}^{m\times m}$ is an orthogonal matrix and $R \in \mathbb{R}^{m\times n}$. This form of the QR decomp. is called full-QR, the form obtained above, where $Q \in \mathbb{R}^{m\times n}$ with orthogonal columns and $R \in \mathbb{R}^{n\times n}$ is diagonal, is the reduced form. Clearly the two forms are equivalent.

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

$$Q_j = \begin{pmatrix} I_{j-1} & 0 \\ 0 & F_j \end{pmatrix}, \tag{9}$$

where $F_j \in \mathbb{R}^{(m-j+1)\times(m-j+1)}$ is such that $F_j X_{j:m,j} = \|X_{j:m,j}\| e_{m-j+1}$. The notation $X_{j:m,j}$ denotes the vector of $\mathbb{R}^{m-j+1}$ formed by taking the elements $X_{j,j}, \ldots, X_{m,j}$ 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 3.1. *Let $x \in \mathbb{R}^m$ and define $v_1 = x - \|x\|_2 e_m$ (resp. $v_2 = x + \|x\|_2 e_m$) and $Q_i = I - 2\frac{v_i v_i'}{v_i' v_i}$. Then Q_i is a symmetric orthogonal matrix and $Q_1 x = \|x\|_2 e_m$ and $Q_2 x = -\|x\|_2 e_m$.*

This Lemma tells us precisely, given $X_{j:m,j}$, how to build F_j such that $F_j X_{j:m,j} = \|X_{j:m,j}\| e_{m-j+1}$: 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 3.1. *[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) $X_{j:m,j:n} = \left(I - \frac{2}{v'v} vv'\right) X_{j:m,j:n}$

The total computational cost of the algorithm is of order

$$O\left(2mn^2 - \frac{2}{3}n^3\right).$$

If needed, we can easily recover the matrix Q. But often this is not necessary. Suppose that we only want to calculate $Q'y = Q_n \cdots Q_1 y$ 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 3.2. *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 3.1.*

3.1. **An example with Vandermonde matrices.** Consider the matrix $X \in \mathbb{R}^{m\times n}$, where $X_{ij} = (j/n)^{i-1}$. The following table show the values of $\|I_n - Q'Q\|_2$ where $X = QR$, for the different methods discussed above.

$m \times n$	Gram-Schmidt	Modified GS	iterated GS	Householder
6×4	2.910^{-15}	1.910^{-15}	3.210^{-16}	5.1210^{-16}
12×8	2.510^{-6}	2.810^{-12}	4.610^{-16}	1.710^{-15}
18×12	2.69	1.3610^{-9}	7.1210^{-11}	1.6510^{-15}

TABLE 1. Values of $\|I_n - Q'Q\|_2$ for the Vandermonde matrices.

4. Application to the least squares problem

We consider the least squares problem

$$\text{(P)} \qquad \min_b \|Xb - y\|_2 , \tag{10}$$

where $\|x\|_2 := \sqrt{\sum_{i=1}^m x_i^2}$ is the Euclidean norm. The least squares problem is well known in Statistics as a core inferential procedures. 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 $Py \in \text{span}(x_1, \ldots, x_n)$. Hence a solution to Problem (P) always exist. Call it $\hat{b}$ (if it is not unique, call the chosen one $\hat{b}$). $\hat{b}$ satisfies:

$$X\hat{b} = Py. \tag{11}$$

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 so-called normal equation

$$X'X\hat{b} = X'y. \tag{12}$$

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

(1) We can solve (P) by solving the normal equation (12) using the Cholesky decomposition of $X'X$ that writes (assuming X is full rank) $X'X = R'R$ where R is upper triangular. 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 more vulnerable to rounding errors.

(2) A more common approach for solving (P) is through the QR decomposition. If $X = QR$ is the QR decomposition of X, then the columns of Q span the same sub-space as the columns of X, hence the projector P is given by $P = QQ'$. From (11), 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.

(3) A third approach is based on using the singular value decomposition discussed later.

5. Some computational stability concepts

Numbers are approximated on a computer. The most widely used standard is the double precision floating point representation. It has very good coverage of small and large numbers (from $\pm 2.2310^{-308}$ to $\pm 1.7910^{308}$). However the interval $[2^j, 2^{j+1}]$ is represented as

$$2^j,\ 2^j + 2^j\frac{1}{2^{52}},\ 2^j + 2^j\frac{2}{2^{52}},\ 2^j + 2^j\frac{3}{2^{52}}, \ldots, 2^j + 2^j\frac{2^{52}}{2^{52}}.$$

The implication is that is $\mathsf{fl}(x)$ denotes the representation of x,

$$\frac{|x - \mathsf{fl}(x)|}{|x|} \leq \epsilon_{\mathsf{mach}} \stackrel{\text{def}}{=} \frac{1}{2^{52}} \approx 2.210^{-16}.$$

Hence on a double precision floating point computer there is no difference between for instance 1 and $1 + 0.5\epsilon_{\mathsf{mach}}$, or between 2^{52} and $2^{52} + 0.5$.

Let $f : \mathsf{X} \to \mathsf{Y}$ be a function. Suppose we want to compute $f(x)$. Because of the computer representation of numbers discussed above, we will end up computing $f(\tilde{x})$ in the best case scenario, where $\tilde{x}$ is the computer representation of x. How bad can the result be? In other words, how close can we expect $f(x)$ and $f(\tilde{x})$ to be? It depends whether f is well-conditioned or not. The (relative) condition number of f at x is the quantity

$$\kappa(f, x) \stackrel{\text{def}}{=} \lim_{r \to 0} \sup_{\|h\| \leq r} \frac{\frac{\|f(x+h) - f(x)\|}{\|f(x)\|}}{\frac{\|h\|}{\|x\|}}.$$

$\kappa(f, x)$ measures the relative variation of the output $f(x)$ over the relative variation of the input x. If f is continuously differentiable then $f(x + h) = f(x) + \nabla f(x) \cdot h + \|h\|\epsilon(h)$, and we get

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

The quantity $\kappa(f,x)$ tells us that if instead of $f(x)$ we end up computing $f(\tilde{x})$ for $\tilde{x}$ relatively close to x, then

$$\frac{\|f(\tilde{x})-f(x)\|}{\|f(x)\|} \leq C_x \kappa(f,x) \frac{\|\tilde{x}-x\|}{\|x\|}.$$

It small ($0 \leq \kappa(f,x) \leq 10^3$, we say that f is well-conditioned (at x) and we don't worry too much about rounding error. However, if $\kappa(f,x) > 10^{10}$, we say that f is ill-conditioned.

Example 1. Consider the problem of computing the solution of the equation $Ax = b$: $A^{-1}b$. Using the spectral norm, The condition number of computing $f(b) = A^{-1}b$ is

$$\kappa(f,b) = \|A^{-1}\|_2 \frac{\|b\|_2}{\|A^{-1}b\|_2}.$$

Since $b = AA^{-1}b$, we see that $\|b\|_2 \leq \|A\|_2 \|A^{-1}b\|_2$. Hence $\frac{\|b\|_2}{\|A^{-1}b\|_2} \leq \|A\|_2$. We conclude that

$$\kappa(f,b) \leq \|A\|_2 \|A^{-1}\|_2.$$

The quantity $\kappa(A) \stackrel{\text{def}}{=} \|A\|_2 \|A^{-1}\|_2$ is called the condition number of the matrix A (more accurately the condition number of A in the spectral norm). By taking the SVD of A is it easy to check that $\|A\|_2 \|A^{-1}\|_2$ can also be written as the ratio of the largest singular value of A divided by the smallest (which is the definition of condition number we saw in Chapter 1).

Similarly if we assume b given and consider $f(A) = A^{-1}b$, it is easily shown that

$$\kappa(f,A) \leq \|A\|_2 \|A^{-1}\|_2.$$

Hence in the problem of solving a linear system $Ax = b$, the condition number of A is a key quantity that informs whether the problem is ill-conditioned or not.

Given the problem of computing $f(x)$, let $\tilde{f}(x)$ be the computed value. Going from x to $\tilde{f}(x)$ entails representing x on a computer, choosing a particular algorithm, and performing all the instructions of the algorithm. For instance, in solving $Ax = b$, $f(b,A) = A^{-1}b$, and $\tilde{f}(b,A)$ would be for instance the output obtained using Gaussian elimination with partial pivoting. We a different $\tilde{f}$ if we decide to use Cholesky (if A is psd).

We say that the computation $\tilde{f}$ (or the algorithm $\tilde{f}$ of f is <u>accurate</u> if

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

where the constants in the big-O on the right-hand side do not depend on x. Accuracy is a very strong requirement. It typically cannot hold unless the problem is well-conditioned for all the input x. For instance, consider solving $Ax = b$. $f(b,A) = A^{-1}b$. We know that just rounding errors, the relative error of the computed solution is of order $\kappa(A) = \|A\|\|A^{-1}\|\epsilon_{\mathsf{mach}}$ and cannot be bounded in A. Accuracy is too stringent. Most computations are not accurate.

We say that $\tilde{f}$ is (backward) <u>stable</u> if for all $x \in \mathsf{X}$, there exists $\tilde{x} \in \mathsf{X}$, with $\|\tilde{x}-x\|/\|x\| = O(\epsilon_{\mathsf{mach}})$, such that $\tilde{f}(x) = f(\tilde{x})$. Here again, the constant in the big-O does not depend x. In other word a stable computation (or algorithm) is one that does not amplify rounding errors.

Put differently, a stable computation is one that computes the exact answer on a close-by input. Though, note that whether $f(\tilde{x})$ itself is close to $f(x)$, now depends on the conditioning of the problem. Hence a stable algorithm does not protect against catastrophic consequences of rounding errors if the problem is ill-conditioned. Several computations are known to be stable.

(1) Solving a triangular system by back-substitution or forward-substitution is stable.
(2) The QR decomposition by Householder decomposition is stable.
(3) Unfortunately, solving a linear system $Ax = b$ by Gaussian elimination is not theoretically stable. But seems to be the case in practice.

References