STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2023
LECTURE 10

1. Gram–Schmidt orthogonalization

- suppose $A \in \mathbb{C}^{n\times n}$ is square and full-rank
- so all the column vectors of A are linearly independent
- consider the QR factorization

$$A = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_n \end{bmatrix} \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ & & r_{nn} \end{bmatrix} = QR$$

- from this matrix equation, we get

$$\begin{aligned} \mathbf{a}_1 &= r_{11}\mathbf{q}_1 \\ \mathbf{a}_2 &= r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2 \\ &\vdots \\ \mathbf{a}_n &= r_{1n}\mathbf{q}_1 + r_{2n}\mathbf{q}_2 + \cdots + r_{nn}\mathbf{q}_n \end{aligned}$$

- and from which we can deduce an algorithm
- first note that $\mathbf{a}_1 = r_{11}\mathbf{q}_1$, and so

$$r_{11} = \|\mathbf{a}_1\|_2, \quad \mathbf{q}_1 = \frac{1}{\|\mathbf{a}_1\|_2}\mathbf{a}_1$$

- next, from $\mathbf{a}_2 = r_{12}\mathbf{q}_1 + r_{22}\mathbf{q}_2$ we get

$$r_{12} = \mathbf{q}_1^*\mathbf{a}_2, \quad r_{22} = \|\mathbf{a}_2 - r_{12}\mathbf{q}_1\|_2, \quad \mathbf{q}_2 = \frac{1}{r_{22}}(\mathbf{a}_2 - r_{12}\mathbf{q}_1)$$

- in general, we get

$$\mathbf{a}_k = \sum_{j=1}^{k} r_{jk}\mathbf{q}_j$$

- and hence

$$\mathbf{q}_k = \frac{1}{r_{kk}}\left[\mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j\right], \quad r_{jk} = \mathbf{q}_j^*\mathbf{a}_k$$

- note that $r_{kk} \neq 0$: since $\mathbf{a}_1, \dots, \mathbf{a}_n$ are linearly independent and so

$$\mathbf{a}_k \notin \operatorname{span}\{\mathbf{a}_1, \dots, \mathbf{a}_{k-1}\} = \operatorname{span}\{\mathbf{q}_1, \dots, \mathbf{q}_{k-1}\}$$

  and so

$$\mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j \neq \mathbf{0}$$

Date: November 2, 2023, version 1.0. Comments, bug reports: `lekheng@uchicago.edu`.

and so

$$r_{kk} = \left\| \mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk} \mathbf{q}_j \right\|_2 \neq 0 \tag{1.1}$$

- this is the *Gram–Schmidt* algorithm, there are two ways to see it
  - given a list of linearly independent vectors $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{C}^n$, it produces a list of orthonormal vectors $\mathbf{q}_1, \dots, \mathbf{q}_n$ that spans the same subspace
  - given a matrix $A \in \mathbb{C}^{n\times n}$ of full rank, it produces a QR factorization $A = QR$
- so we have established the existence of QR
- in fact, it is clear that if we started from a list of linearly independent vectors $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{C}^m$ where $n \le m$ or equivalently a matrix $A \in \mathbb{C}^{m\times n}$ of full column rank $\operatorname{rank}(A) = n \le m$, the Gram–Schmidt algorithm would still produce a list of orthogonormal vectors $\mathbf{q}_1, \dots, \mathbf{q}_n$ or equivalently a matrix $Q \in \mathbb{C}^{m\times n}$ with orthonormal columns
- the only difference is that the algorithm would terminate at step n when it runs out of input vectors
- note that this is a special QR factorization since $r_{kk} > 0$ for all $k = 1, \dots, n$ (because r_{kk} is chosen to be a norm)
- in fact, requiring $r_{kk} > 0$ gives us uniqueness (not just uniqueness up to unimodular scaling)
- now what if $A \in \mathbb{C}^{m\times n}$ is not full rank, i.e., $\mathbf{a}_1, \dots, \mathbf{a}_n$ are not linearly independent
- in this case Gram–Schmidt could fail since r_{kk} in (1.1) can now be 0
- we need to modify Gram–Schmidt so that it finds a subset of $\mathbf{a}_1, \dots, \mathbf{a}_n$ that is linearly independent
- this is equivalent to finding a permutation matrix Π so that the first $r = \operatorname{rank}(A)$ columns of $A\Pi$ are linearly independent
- this can be done adaptively and corresponds to column pivoting
- we will discuss this later when we discuss Givens and Householder QR algorithms, which are what used in practice
- the truth is that Gram–Schmidt is a numerically unstable algorithm
- for example, if $\mathbf{a}_1$ and $\mathbf{a}_2$ are almost parallel, then $\mathbf{a}_2 - r_{12}\mathbf{q}_1$ is almost zero and roundoff error becomes significant
- because of such numerical instability the computed $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k$ gradually lose their orthogonality
- however it is not difficult to fix Gram–Schmidt by *reorthogonalization*, essentially by applying Gram–Schmidt a second time to the output of the first round of Gram–Schmidt $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k$
- in exact arithmetic, $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_k$ is already orthogonal and applying Gram–Schmidt a second time has no effect
- but in the presence of rounding error, reorthogonalization has real effect — making the output of the second round orthogonal
- the nice thing is that there is no need to do a third round of Gram–Schmidt — twice suffices (for subtle reasons)

2. Back substitution and tridiagonal solve

- *backsolve* or *back substitution* refers to a simple, intuitive way of solving linear systems of the form $R\mathbf{x} = \mathbf{b}$ or $L\mathbf{x} = \mathbf{b}$ where R is upper-triangular and L is lower-triangular

- take $R\mathbf{x} = \mathbf{b}$ for illustration

$$\begin{bmatrix} r_{11} & \cdots & r_{1,n-1} & r_{1n} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & r_{n-1,n-1} & r_{n-1,n} \\ 0 & \cdots & 0 & r_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

- start at the bottom and work our way up

$$\begin{aligned} r_{nn}x_n &= b_n \\ r_{n-1,n}x_n + r_{n-1,n-1}x_{n-1} &= b_{n-1} \\ &\vdots \\ r_{11}x_1 + r_{12}x_2 + \cdots + r_{1n}x_n &= b_1 \end{aligned}$$

- we get

$$\begin{aligned} x_n &= b_n/r_{nn} \\ x_{n-1} &= (b_{n-1} - r_{n-1,n}x_n)/r_{n-1,n-1} \\ x_{n-2} &= (b_{n-2} - r_{n-2,n-1}x_{n-1} - r_{n-2,n}x_n)/r_{n-2,n-2} \\ &\vdots \\ x_1 &= (b_1 - r_{12}x_2 - r_{13}x_3 - \cdots - r_{1n}x_n)/r_{11} \end{aligned}$$

- this requires that $r_{kk} \neq 0$ for all $k = 1, \dots, n$, which is guaranteed if R is nonsingular
- back substitution in the above form is sometimes called *backward substitution* to distinguish it from *forward substitution*, which is for the case $L\mathbf{x} = \mathbf{b}$

$$\begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

  with

$$\begin{aligned} x_1 &= b_1/l_{11} \\ x_2 &= (b_2 - l_{21}x_1)/l_{22} \\ x_3 &= (b_3 - l_{31}x_1 - l_{32}x_2)/l_{33} \\ &\vdots \\ x_n &= (b_n - l_{n1}x_1 - l_{n2}x_2 - \cdots - l_{n,n-1}x_{n-1})/l_{nn} \end{aligned}$$

- it is easy to solve $A\mathbf{x} = \mathbf{b}$ if
  - A is unitary or orthogonal (includes permutation matrices)
  - A is upper- or lower-triangular (includes diagonal matrices)
  - $A\mathbf{x} = \mathbf{b}$ with such A can be solved with $O(n^2)$ flops
  - if A represents a special orthogonal matrix like the discrete Fourier or wavelet transforms, then $A\mathbf{x} = \mathbf{b}$ can in fact be solved in $O(n \log n)$ flops using algorithms like fast Fourier or fast wavelet transforms
- if A is not one of these forms, we factorize A into a product of matrices of these forms
- take QR factorization for example
- given $A \in \mathbb{C}^{n\times n}$ nonsingular and $\mathbf{b} \in \mathbb{C}^n$
  - step 1: find QR factorization $A = QR$
  - step 2: form $\mathbf{b} = Q^*\mathbf{b}$

- step 3: backsolve $R\mathbf{x} = \mathbf{y}$ to get $\mathbf{x}$

- this may be viewed as the basic impetus for matrix factorizations like LU, Cholesky, QR, SVD, EVD
- actually to the above list, we could also add
  - A is bidiagonal/tridiagonal (or banded, i.e., $a_{ij} = 0$ if $|i-j| > b$ for some *bandwidth* $b \ll n$)
  - A is Toeplitz or Hankel, i.e., $a_{ij} = a_{i-j}$ or $a_{ij} = a_{i+j}$ — constant on the diagonals or the opposite diagonals
  - A is semiseparable
  - $A\mathbf{x} = \mathbf{b}$ with bidiagonal or tridiagonal A can be solved in $O(n)$ flops
  - $A\mathbf{x} = \mathbf{b}$ with Toeplitz or Hankel A can be solved in $O(n^2 \log n)$ flops
  - these are often called structured matrices
- for example, a tridiagonal system

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}$$

may be solved by first computing

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & i = 1, \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & i = 2, 3, \dots, n-1, \end{cases}$$

and

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & i = 1, \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & i = 2, 3, \dots, n, \end{cases}$$

followed by back substitution

$$\begin{aligned} x_n &= d'_n, \\ x_i &= d'_i - c'_i x_{i+1}, \qquad i = n-1, n-2, \dots, 1 \end{aligned}$$

- exercise: prove that the above algorithm indeed gives you a solution
- in this course we will just restrict ourselves to unitary and triangular factors
- but we will discuss a general principle for solving linear systems and least squares problems based on rank-retaining factorizations that works with any structured matrices

3. RANK-RETAINING FACTORIZATIONS

- let $A \in \mathbb{C}^{m\times n}$ with $\operatorname{rank}(A) = r$, a *rank-retaining factorization* is a factorization of A into

$$A = GH$$

where $G \in \mathbb{C}^{m\times r}$ and $H \in \mathbb{C}^{r\times n}$ and

$$\operatorname{rank}(G) = \operatorname{rank}(H) = r$$

  - example: condensed SVD $A = U\Sigma V^*$, $U \in \mathbb{C}^{m\times r}$, $\Sigma \in \mathbb{C}^{r\times r}$, $V \in \mathbb{C}^{n\times r}$ where we could pick $G = U\Sigma$ and $H = V^*$ or $G = U$ and $H = \Sigma V^*$

- example: condensed QR $A\Pi = QR$, $Q \in \mathbb{C}^{m\times r}$, $R \in \mathbb{C}^{r\times n}$, where we could pick $G = Q$ and $H = R\Pi^{\mathsf{T}}$
- example: condensed LU $\Pi_1 A \Pi_2 = LU$, $L \in \mathbb{C}^{m\times r}$, $U \in \mathbb{C}^{r\times n}$, where we could pick $G = \Pi_1^{\mathsf{T}} L$ and $H = U\Pi_2^{\mathsf{T}}$

- easy facts: if $A = GH$ is rank-retaining, then
  (i) $G^*G \in \mathbb{C}^{r\times r}$ is nonsingular
  (ii) $HH^* \in \mathbb{C}^{r\times r}$ is nonsingular
  (iii) $\operatorname{im}(A) = \operatorname{im}(G)$
  (iv) $\ker(A^*) = \ker(G^*)$
  (v) $\ker(A) = \ker(H)$
  (vi) $\operatorname{im}(A^*) = \operatorname{im}(H^*)$
- prove these as exercises

4. General principle for linear systems and least squares

- we will discuss a general principle for solving linear systems and least squares problems via matrix factorization
- given $A \in \mathbb{C}^{m\times n}$ and $\mathbf{b} \in \mathbb{C}^m$, two of the most common problems are
  - if $A\mathbf{x} = \mathbf{b}$ is consistent and A is full column rank, we want the unique solution
  - if $A\mathbf{x} = \mathbf{b}$ is inconsistent and A is full column rank, we want the unique least squares solution
- the trouble is that when A is rank deficient, i.e., not full rank, then the solution is not unique and so we want the minimum length solution instead
  - if $A\mathbf{x} = \mathbf{b}$ is consistent and A is rank deficient, we want the minimum length solution
$$\min\{\|\mathbf{x}\|_2 : A\mathbf{x} = \mathbf{b}\} \tag{4.1}$$
  - if $A\mathbf{x} = \mathbf{b}$ is inconsistent and A is rank deficient, we want the minimum length least squares solution
$$\min\{\|\mathbf{x}\|_2 : \mathbf{x} \in \operatorname{argmin}\|\mathbf{b} - A\mathbf{x}\|_2\} \tag{4.2}$$
- if we can solve the min length versions then we can solve the full column rank versions, so let's focus on the min length version

5. Min length linear systems via rank-retaining factorization

- we start from the consistent case: $\mathbf{b} \in \operatorname{im}(A)$ and so $\mathbf{b} = A\mathbf{x}$ for some $\mathbf{x} \in \mathbb{C}^n$
  - recall the Fredholm alternative that we proved in the homework:
$$\mathbb{C}^n = \operatorname{im}(A^*) \oplus \ker(A)$$
  - $\mathbf{x} \in \mathbb{C}^n$ can be written uniquely as
$$\mathbf{x} = \mathbf{x}_0 + \mathbf{x}_1, \quad \mathbf{x}_0 \in \ker(A),\ \mathbf{x}_1 \in \operatorname{im}(A^*),\ \mathbf{x}_0^*\mathbf{x}_1 = 0$$
  - since
$$\mathbf{b} = A\mathbf{x} = A\mathbf{x}_0 + A\mathbf{x}_1 = A\mathbf{x}_1$$
    $\mathbf{x}_1$ is also a solution to the linear system
  - by Pythagoras theorem
$$\|\mathbf{x}\|_2^2 = \|\mathbf{x}_0\|_2^2 + \|\mathbf{x}_1\|_2^2 \ge \|\mathbf{x}_1\|_2^2$$
  - so for a minimum length solution we set $\mathbf{x}_0 = \mathbf{0}$, i.e., the minimum length solution is given by $\mathbf{x} = \mathbf{x}_1$
- now we will see how to find $\mathbf{x}_1$ using a rank-retaining factorization
$$A = GH \tag{5.1}$$

– since $\mathbf{x}_1 \in \operatorname{im}(A^*) = \operatorname{im}(H^*)$ by easy fact (vi), so for some $\mathbf{v} \in \mathbb{C}^r$,

$$\mathbf{x}_1 = H^*\mathbf{v} \tag{5.2}$$

– by easy fact (iii), $\mathbf{b} \in \operatorname{im}(A) = \operatorname{im}(G)$ and so for some $\mathbf{s} \in \mathbb{C}^r$,

$$\mathbf{b} = G\mathbf{s} \tag{5.3}$$

– so upon substituting (5.1), (5.2), (5.3), $A\mathbf{x}_1 = \mathbf{b}$ becomes

$$GHH^*\mathbf{v} = G\mathbf{s}$$

– now multiply by G^* to get

$$(G^*G)HH^*\mathbf{v} = (G^*G)\mathbf{s}$$

– by easy fact (i), G^*G is nonsingular and so

$$HH^*\mathbf{v} = \mathbf{s}$$

– by easy fact (ii), HH^* is nonsingular and so

$$\mathbf{v} = (HH^*)^{-1}\mathbf{s}$$

– plugging back into (5.2), we get

$$\mathbf{x}_1 = H^*(HH^*)^{-1}\mathbf{s} \tag{5.4}$$

- this gives an algorithm for solving the minimum length linear system (4.1)
  – step 1: compute rank retaining factorization $A = GH$
  – step 2: solve $G\mathbf{s} = \mathbf{b}$ for $\mathbf{s} \in \mathbb{C}^r$
  – step 3: solve $HH^*\mathbf{v} = \mathbf{s}$ for $\mathbf{v} \in \mathbb{C}^r$
  – step 4: compute $\mathbf{x}_1 = H^*\mathbf{v}$
- this works because

$$A\mathbf{x}_1 = GH\mathbf{x}_1 = GHH^*\mathbf{v} = G(HH^*)(HH^*)^{-1}\mathbf{s} = G\mathbf{s} = \mathbf{b}$$

- note that the system in steps 2 and 3 involve a full-rank G and a nonsingular HH^* — both have unique solutions
- example: if $A\Pi = QR$ is the condensed QR, then with $G = Q$ and $H = R\Pi^{\mathsf{T}}$
  – step 2: $Q\mathbf{s} = \mathbf{b}$ is easy to obtain via

$$Q^*Q\mathbf{s} = Q^*\mathbf{b}$$

  and so $\mathbf{s} = Q^*\mathbf{b}$
  – step 3: $R\Pi^{\mathsf{T}}\Pi R^*\mathbf{v} = \mathbf{s}$ is also easy to obtain via two backsolves

$$\begin{cases} R\mathbf{y} = \mathbf{s} \\ R^*\mathbf{v} = \mathbf{y} \end{cases}$$

- example: if $A = U\Sigma V^*$ is the condensed SVD, then with $G = U$ and $H = \Sigma V^*$
  – step 2: $U\mathbf{s} = \mathbf{b}$ is easy to obtain via

$$U^*U\mathbf{s} = U^*\mathbf{b}$$

  and so $\mathbf{s} = U^*\mathbf{b}$
  – step 3: $\Sigma V^* V\Sigma\mathbf{v} = \mathbf{s}$ is just

$$\Sigma^2\mathbf{v} = \mathbf{s}$$

  or

$$\begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_r^2 \end{bmatrix} \begin{bmatrix} z_1 \\ \vdots \\ z_r \end{bmatrix} = \begin{bmatrix} s_1 \\ \vdots \\ s_r \end{bmatrix}$$

and so for $k = 1, \dots, r$,

$$z_k = s_k/\sigma_k^2$$

- note that (5.4) is in terms of $\mathbf{s}$, if we want an analytic expression, it should involve only quantities we know, i.e., $\mathbf{b}, G, H$
- to express $\mathbf{s}$ in terms of quantities we know, we just multiply (5.3) by G^* to get

$$G^*G\mathbf{s} = G^*\mathbf{b}$$

and using fact (i) to get

$$\mathbf{s} = (G^*G)^{-1}G^*\mathbf{b}$$

- with this and (5.4), we get an analytic expression for the minimum length solution

$$\mathbf{x}_1 = H^*(HH^*)^{-1}(G^*G)^{-1}G^*\mathbf{b} \tag{5.5}$$

6. Min length least squares via rank-retaining factorization

- we now consider the inconsistent case: $\mathbf{b} \notin \operatorname{im}(A)$
  - this time we use the other part of the Fredholm alternative:

$$\mathbb{C}^m = \ker(A^*) \oplus \operatorname{im}(A)$$

  - any $\mathbf{b} \in \mathbb{C}^m$ can be written uniquely as

$$\mathbf{b} = \mathbf{b}_0 + \mathbf{b}_1, \quad \mathbf{b}_0 \in \ker(A^*),\ \mathbf{b}_1 \in \operatorname{im}(A),\ \mathbf{b}_0^*\mathbf{b}_1 = 0$$

  - since $\mathbf{b}_1 - A\mathbf{x} \in \operatorname{im}(A)$, it must also be orthogonal to $\mathbf{b}_0$ and by Pythagoras

$$\|\mathbf{b} - A\mathbf{x}\|_2^2 = \|\mathbf{b}_0 + \mathbf{b}_1 - A\mathbf{x}\|_2^2 = \|\mathbf{b}_0\|_2^2 + \|\mathbf{b}_1 - A\mathbf{x}\|_2^2 \geq \|\mathbf{b}_0\|_2^2$$

  - so for a least squares solution, we must have

$$\|\mathbf{b}_1 - A\mathbf{x}\|_2^2 = 0$$

  i.e.,

$$A\mathbf{x} = \mathbf{b}_1 \tag{6.1}$$

  - this is always consistent since $\mathbf{b}_1 \in \operatorname{im}(A)$ and we proceed as in the consistent case to get from (5.5),

$$\mathbf{x}_1 = H^*(HH^*)^{-1}(G^*G)^{-1}G^*\mathbf{b}_1 \tag{6.2}$$

  - but by easy fact (iv), $\ker(A^*) = \ker(G^*)$ and so

$$G^*\mathbf{b} = G^*(\mathbf{b}_0 + \mathbf{b}_1) = G^*\mathbf{b}_0 + G^*\mathbf{b}_1 = G^*\mathbf{b}_1 \tag{6.3}$$

  - in other words, the $\mathbf{b}_1$ in (6.2) may be replaced by $\mathbf{b}$ and we get

$$\mathbf{x}_1 = H^*(HH^*)^{-1}(G^*G)^{-1}G^*\mathbf{b} \tag{6.4}$$

- note that there is no difference in the expression (5.5) for minimum length linear system and the expression (6.4) for minimum length least squares and the four-step algorithm presented earlier works for minimum length least squares without change
- (6.4) should never be used as is, instead it should be used to construct an algorithm as in the previous section
- exercise: construct an algorithm using (6.4) to get the minimum length solution to a least squares problem (4.2)
- a consequence of (6.4) is that given a rank-retaining factorization $A = GH$, the Moore–Penrose pseudoinverse of A is given by

$$A^\dagger = H^*(HH^*)^{-1}(G^*G)^{-1}G^* \tag{6.5}$$

- example: if $A = U\Sigma V^*$ is the condensed SVD, then $A^\dagger = V\Sigma^{-1}U^*$ since (6.5) with $G = U$ and $H = \Sigma V^*$ yields
$$A^\dagger = V\Sigma(\Sigma V^* V\Sigma)^{-1}(U^*U)^{-1}U^* = V\Sigma\Sigma^{-2}U^* = V\Sigma^{-1}U^*$$
- example: if $A\Pi = QR$ is the condensed QR, then $A^\dagger = \Pi R^*(RR^*)^{-1}Q^*$ since (6.5) with $G = Q$ and $H = R\Pi^\mathsf{T}$ yields
$$A^\dagger = \Pi R^*(R\Pi^\mathsf{T}\Pi R^*)^{-1}(Q^*Q)^{-1}Q^* = \Pi R^*(RR^*)^{-1}Q^*$$

7. other uses of qr

- the QR decomposition for a square matrix may be used to determine the magnitude of determinant
$$|\det(A)| = |\det(QR)| = |\det(Q)||\det(R)| = |\det(R)| = \prod_{k=1}^{n}|r_{kk}|$$
- we used two facts: determinant of unitary matrix must have absolute value 1, determinant of triangular (upper or lower) matrix is just product of diagonal elements
- the rank-retaining QR decomposition may be used to determine orthonormal bases for the fundamental subspaces
$$A\Pi = [Q_1, Q_2]\begin{bmatrix} R_1 & S \\ 0 & 0 \end{bmatrix}$$
- the columns of Q_1 form an orthonormal basis for $\operatorname{im}(A)$ (follows from Gram–Schmidt) and the columns of Q_2 form an orthonormal basis for $\ker(A^*)$
- if we need orthonormal bases for $\operatorname{im}(A^*)$ and $\ker(A)$, we find the rank-retaining QR factorization of A^*
- this is a cheaper way than SVD to obtain orthonormal bases for the fundamental subsapces

8. full rank least squares problem

- the general method for a rank-retaining factorization works for matrices of any rank but there are better alternatives to solve least squares problem when the coefficient matrix A has full column rank
- this case is particularly important and common we want to say more about it
- here we seek to minimize $\|A\mathbf{x} - \mathbf{b}\|_2$ where $A \in \mathbb{C}^{m\times n}$ has $\operatorname{rank}(A) = n \le m$ and $\mathbf{b} \in \mathbb{C}^m$
- such problems *always* have unique solution $\mathbf{x}^*$ (why?)
- so there is no question of finding a min length solution — since there's only one solution in this case, we don't get to choose
- we consider three methods:
  (1) QR factorization
  (2) normal equation
  (3) augmented system
- mathematically they all give the same solution (i.e., in exact arithmetic) but they have different numerical properties
- so one has to know all three since each is good/bad under different circumstances

9. full rank least squares via qr

- the first approach is to take advantage of the fact that the 2-norm is invariant under orthogonal transformations, and seek an orthogonal matrix Q such that the transformed problem
$$\min \|A\mathbf{x} - \mathbf{b}\|_2 = \min \|Q^*(A\mathbf{x} - \mathbf{b})\|_2$$
is "easy" to solve

- we could use the QR factorization of A

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}\begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R$$

- then $Q_1^* A = R$ and

$$\begin{aligned} \min \|A\mathbf{x} - \mathbf{b}\|_2 &= \min \|Q^*(A\mathbf{x} - \mathbf{b})\|_2 \\ &= \min \|(Q^*A)\mathbf{x} - Q^*\mathbf{b}\|_2 \\ &= \min \left\| \begin{bmatrix} R \\ 0 \end{bmatrix}\mathbf{x} - Q^*\mathbf{b} \right\|_2 \end{aligned}$$

- if we partition

$$Q^*\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

  then

$$\min \|A\mathbf{x} - \mathbf{b}\|_2^2 = \min \left\| \begin{bmatrix} R \\ 0 \end{bmatrix}\mathbf{x} - \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \right\|_2^2 = \min \|R\mathbf{x} - \mathbf{c}\|_2^2 + \|\mathbf{d}\|_2^2$$

- therefore the minimum is achieved by the vector $\mathbf{x}$ such that $R\mathbf{x} = \mathbf{c}$ and therefore

$$\min_{\mathbf{x}\in\mathbb{C}^n} \|A\mathbf{x} - \mathbf{b}\|_2 = \|\mathbf{d}\|_2$$

10. FULL RANK LEAST SQUARES VIA NORMAL EQUATION

- the second approach is to define

$$\varphi(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|_2^2$$

  which is a differentiable function of $\mathbf{x}$
- we can minimize $\varphi(\mathbf{x})$ by noting that $\nabla\varphi(\mathbf{x}) = A^*(A\mathbf{x} - \mathbf{b})$ which means that $\nabla\varphi(\mathbf{x}) = \mathbf{0}$ if and only if

$$A^*A\mathbf{x} = A^*\mathbf{b} \tag{10.1}$$

- this system of equations is collectively called the *normal equation*, and were used by Gauss to solve least squares problems
- we saw at least two other ways to derive (10.1) in the homeworks
- most people believe that it is a bad idea to solve the normal equations to get the least squares solution but this is not always the case
- first the bad stuff about normal equation:
  - it can be ill-conditioned: the linear system $A^*A\mathbf{x} = A^*\mathbf{b}$ has condition number $\kappa_2(A^*A) = \kappa_2(A)^2$ double in order of magnitude
  - it can be unstable: for example, if

$$A = \begin{bmatrix} 1 & 1 \\ \delta & 0 \end{bmatrix}, \qquad A^\mathsf{T}A = \begin{bmatrix} 1+\delta^2 & 1 \\ 1 & 1 \end{bmatrix},$$

    and δ is so small that your computer rounds off $1+\delta^2$ to 1, then you end up with a rank-deficient matrix

$$\mathrm{fl}(A^\mathsf{T}A) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{10.2}$$

  - the issue here is not that δ^2 is so small that it underflows but that $1+\delta^2$ cannot be stored in the mantissa and will be rounded to 1
- now the good stuff about normal equation:

- statisticians often use the normal equation because in many statistical problems, the measurement errors in A are much larger than the roundoff errors and so the latter type of errors are relatively insignificant
- if $n \ll m$ then A^*A is $n \times n$, so the normal equation involves much less arithmetic when and A^*A requires much less storage than solving $\min \|A\mathbf{x} - \mathbf{b}\|_2^2$ via the QR method

- the normal equation is very useful in proofs and derivations — it is perfectly fine using it as a mathematical tool, all the bad stuff has to do with using it in numerical computations
- if you have to numerically solve the normal equation for A of full column rank, the matrix A^*A is positive definite and so you should apply Cholesky factorization

11. QR factorization versus normal equation

- it is not clear cut whether QR or NE is better
- for the QR method, we work directly with $A \in \mathbb{C}^{m\times n}$ and do not need to form $A^*A \in \mathbb{C}^{n\times n}$ explicitly so we don't face the problem in (10.2)
- if we do a careful error analysis
  - normal equation produces a solution $\widehat{\mathbf{x}}_{\mathrm{NE}}$ with relative error
$$\frac{\|\mathbf{x} - \widehat{\mathbf{x}}_{\mathrm{NE}}\|}{\|\mathbf{x}\|} \le \gamma_{\mathrm{NE}}\kappa(A)^2\left(1 + \frac{\|\mathbf{b}\|}{\|A\|\|\mathbf{x}\|}\right)\mathsf{u}$$
  - QR method avoids produces a solution $\widehat{\mathbf{x}}_{\mathrm{QR}}$ with relative error
$$\frac{\|\mathbf{x} - \widehat{\mathbf{x}}_{\mathrm{QR}}\|}{\|\mathbf{x}\|} \le 2\gamma_{\mathrm{QR}}\kappa(A)\mathsf{u} + \gamma_{\mathrm{QR}}\kappa(A)^2\frac{\|\mathbf{b} - A\mathbf{x}\|}{\|A\|\|\mathbf{x}\|}\mathsf{u}$$
- everything above is with respect to the 2-norm, u is unit roundoff,[1] γ_{NE} and γ_{QR} are slow growing functions of m, n (therefore constants if we fix m, n)
- even though the QR method avoids forming A^*A, it does not avoid $\kappa(A)^2$ entirely
- the QR method is appealing if $\|\mathbf{b} - A\mathbf{x}\|$ is small, which is more often than not the case since the most common reason for wanting to solve $\min\|A\mathbf{x} - \mathbf{b}\|_2$ is when we expect $A\mathbf{x} \approx \mathbf{b}$ (e.g., linear regression)
- if we do a careful flop count
  - normal equation forms $C = A^*A$ and $\mathbf{c} = A^*\mathbf{b}$, Cholesky factorizes $C = R^*R$, backsolves $R^{\mathsf{T}}\mathbf{y} = \mathbf{c}$ and $R\mathbf{x} = \mathbf{y}$:
$$n^2\left(m + \frac{n}{3}\right)$$
  - QR method does Householder QR $A = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$, unitary transform $\mathbf{c} = Q^*\mathbf{b}$, backsolves $R\mathbf{x} = \mathbf{c}$:
$$2n^2\left(m - \frac{n}{3}\right)$$
- flop counts are similar if $m \approx n$ but normal equation is twice as fast if $m \gg n$
- we will discuss Householder QR and Cholesky factorization in the future
- assuming a dense A, the following table summarizes the relative merits of normal equation (NE) method, QR method, and the SVD method (in lecture 5)

accuracy:	NE	$<$	QR	$<$	SVD
speed:	NE	$>$	QR	$>$	SVD

- for very ill-conditioned problems, the SVD method is recommended

[1] Recall $\mathsf{u} = \varepsilon_{\mathrm{machine}}/2$ and around 10^{-16} (double), 10^{-19} (extended), 10^{-35} (quadruple).

12. FULL RANK LEAST SQUARES VIA AUGMENTED SYSTEM

- we can cast the normal equation in another form
- let $\mathbf{r} = \mathbf{b} - A\mathbf{x}$ be the residual
- now by the normal equation

$$A^*\mathbf{r} = A^*\mathbf{b} - A^*A\mathbf{x} = \mathbf{0}$$

- and so we obtain the *augmented system*

$$\begin{aligned} \mathbf{r} + A\mathbf{x} &= \mathbf{b} \\ A^*\mathbf{r} &= \mathbf{0} \end{aligned}$$

- in matrix form, we get

$$\begin{bmatrix} I & A \\ A^* & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}$$

- this is often a large system since the coefficient matrix has dimension $(m+n) \times (m+n)$, but it preserves the structure and sparsity of A