Lecture 17

Conditioning of LSQ

Learning Outcomes

- **Understand and Apply Pseudo-Inverse:**
  - Acquire knowledge on the concept of pseudo-inverse or Moore-Penrose inverse, $A^\dagger := (A^TA)^{-1}A^T$, and its application in solving Least Squares problems, particularly understanding the conditions under which it exists and its implications in solutions.
- **Comprehend Matrix Conditioning:**
  - Develop an understanding of the condition number of a matrix, $\kappa(A)$, its significance in the context of matrix norms, and its impact on the stability and accuracy of solutions to linear systems, especially in relation to singular values.
- **Implement and Evaluate LSQ Solutions using SVD:**
  - Learn to implement Least Squares solutions using Singular Value Decomposition (SVD) and evaluate the computational cost and stability of such solutions in comparison to other methods, gaining insights into the trade-offs between stability and computational efficiency.

Analyses of rectangular matrices

Recall the normal equation (7.1) $A^TAx = A^Tb$ for a solution to LSQ.
Introducing the pseudo-inverse or Moore-Penrose inverse

$$A^\dagger := (A^TA)^{-1}A^T$$

the solution is just $x = A^\dagger b$, provided $A^\dagger$ exists. This is the case if and only if A has full rank which is equivalent to A^TA being regular.

Definition 17.1. [3.7] *The condition number of a matrix $A \in \mathbb{C}^{m\times n}$ with respect to a norm $\|\cdot\|$ is*

$$\kappa(A) = \begin{cases} \|A\|\,\|A^\dagger\| & \text{if } A \text{ has full rank,} \\ \infty & \text{otherwise.} \end{cases}$$

We remark that $A^\dagger = A^{-1}$ if $m = n$ so that the above definition is consistent with the previous one for $n \times n$ matrices.

Lemma 17.2. *If A has full rank: $\kappa_2(A) = \sigma_1/\sigma_n$ where σ_1 and σ_n are the biggest and smallest singular value, respectively.*

Proof. From Corollary 16.1 we know that $\|A\|_2 = \sigma_1$. Writing $A = U\Sigma V^T$ for a SVD, a short calculation results in a SVD

$$A^\dagger = V\tilde{\Sigma}\tilde{U}^T \tag{17.1}$$

with $\tilde{\Sigma} = \mathrm{diag}(\frac{1}{\sigma_n}, \ldots, \frac{1}{\sigma_1})$ so that $\|A^\dagger\|_2 = \frac{1}{\sigma_n}$. □

However, conditioning gives an insight about how sensitive the solution is to a small change in the data. For the linear least square problem, the problem is stated as,

$$\text{find } x \in \mathbb{R}^n \text{ such that } \|Ax - b\|_2 \text{ is minimal.}$$

But what is the impact of a minor change to the right hand side vector b? In another word,

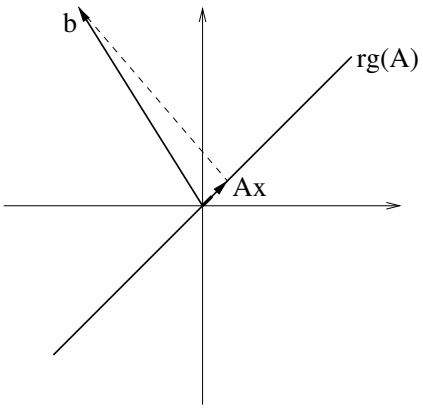
$$\text{find } x \in \mathbb{R}^n \text{ such that } \|Ax - (b + \Delta b)\|_2 \text{ is minimal.}$$

A consequence of this change in the data, $(b + \Delta b)$ is a change in the solution, $(x + \Delta x)$.

Moreover $\|b\|_2^2 = \|b - Ax\|_2^2 + \|Ax\|_2^2 \geq \|Ax\|_2^2$ which motivates to introduce the angle $\theta \in [0, \frac{\pi}{2}]$ via

$$\cos(\theta) = \frac{\|Ax\|_2}{\|b\|_2}.$$

To provide a geometric interpretation, this is the angle between the b and the range of A:



Theorem 17.1. *Assume that $x \neq 0$ solves* LSQ *for data (A, b) and $x + \Delta x$ for $(A, b + \Delta b)$. Then*

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \frac{\kappa_2(A)}{\eta \cos(\theta)} \frac{\|\Delta b\|_2}{\|b\|_2}$$

where $\eta := \|A\|_2\|x\|_2/\|Ax\|_2 \geq 1$.

Proof. Assume that A has full rank (otherwise the assertion is trivial). From the assumptions we furthermore have that $\Delta x = A^\dagger \Delta b$. Therefore

$$\frac{\|\Delta x\|_2}{\|x\|_2} \leq \frac{\|A^\dagger\|_2\|\Delta b\|_2}{\|x\|_2} = \frac{\kappa_2(A)\|\Delta b\|_2}{\|A\|_2\|x\|_2} = \frac{\kappa_2(A)\|\Delta b\|_2}{\eta\|Ax\|_2} = \frac{\kappa_2(A)}{\eta\cos(\theta)} \frac{\|\Delta b\|_2}{\|b\|_2}.$$

□

Imagine now that b is (almost) orthogonal to range(A) which means that the solution is close to zero, $\|x\|$ is small. A small error in the data b then may lead to a small absolute deviation in the solution but can also lead to a big relative error in the solution. Example:

$$A = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad b = \begin{pmatrix} -1+\delta \\ 1 \end{pmatrix} \quad \rightsquigarrow \quad x = \frac{\delta}{2}.$$

Now, think of a small error in the data b, $\Delta b = (\epsilon, 0) \rightsquigarrow \Delta x = \epsilon/2$. Hence, $\|\Delta x\|_2/\|x\|_2 = \epsilon/\delta$ may be large.

Solving LSQ using the SVD [7.3]

Algorithm 9 LSQ_SVD (solving LSQ using a SVD)

input: $A \in \mathbb{R}^{m\times n}$ with $m \geq n = \text{rank}(A)$, $b \in \mathbb{R}^m$.
output: $x \in \mathbb{R}^n$ minimiser of $g(x) = \frac{1}{2}\|Ax - b\|_2^2$.

1: compute the reduced SVD $A = \hat{U}\hat{\Sigma}V^T$
2: compute $y = \hat{U}^T b$
3: solve $\hat{\Sigma} z = y$
4: return $x = Vz$

The thus computed x satisfies

$$A^T A x = V\hat{\Sigma}^T \underbrace{\hat{U}^T\hat{U}}_{=I} \hat{\Sigma}V^T x = V\hat{\Sigma}^T\hat{\Sigma} z = V\hat{\Sigma}^T y = V\hat{\Sigma}^T\hat{U}^T b = A^T b$$

which is the normal equation (7.1) and, hence, x indeed solves LSQ.
The cost of **LSQ_SVD** is dominated by computing a reduced SVD of A as the subsequent steps essentially are matrix-vector multiplications only. According to Trefethen and Bau we have,

$$C_{\textbf{LSQ\_SVD}}(m,n) = 2mn^2 + 11n^3 + \Theta(mn + n^2) \quad \text{as } m, n \to \infty.$$

For the technical details consult "Lloyd Trefethen, David Bau Numerical Linear Algebra, SIAM, 1997. Lecture 31, page 234" if interested. This is much more expensive that that for solving the normal equation with, for instance, GEPP (or Cholesky, a special version for positive definite matrices). The benefit is better stability. Example:

$$A = \begin{pmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{pmatrix}, \quad b = \begin{pmatrix} 2 \\ \delta \\ \delta \end{pmatrix} \quad \rightsquigarrow \quad x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

When using the normal equation we encounter problems for much larger values of δ than when employing the method based on the SVD.

We will next learn about a method in between the two presented methods with respect to stability and cost.

Lecture 18

QR factorisation

Learning Outcomes

- **Understand and Apply QR Factorization:**
  - Gain knowledge on QR factorization and its significance in transforming a matrix to upper triangular form. Understand the process of Gram-Schmidt orthonormalization and its application in creating orthonormal column vectors.
- **Implement LSQ Solutions using QR Factorization:**
  - Learn to implement Least Squares solutions using QR factorization, understanding the computational steps involved and how the method relates to solving normal equations in the context of Least Squares problems.
- **Explore Advanced Factorization Techniques:**
  - Delve into advanced techniques like Householder reflections for computing QR factorization, understanding their practical applications and advantages in terms of stability and computational efficiency over basic methods like Gram-Schmidt.

Gram Schmidt is a process for converting a set of basis vector $a_1, a_2, \ldots, a_n$ into an orthonormal set of vectors.

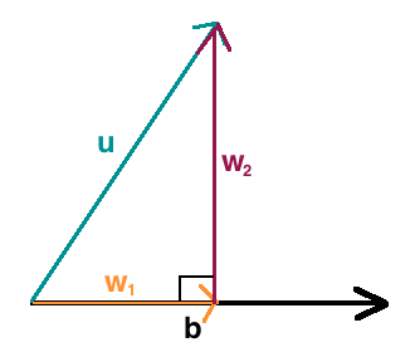
- Orthogonal vectors. The vectors $v_1, v_2, \ldots, v_n$ are orthogonal if,

$$\langle v_i, v_j \rangle = 0 \quad \forall i \neq j.$$

- Normal vectors

$$\|v_i\|_2 = 1 \quad \forall i = 1, 2, \ldots, n$$

Gram Schmidt process is based on the orthogonal projection concept which states that a vector u can be decomposed with respect to another vector b, given that $b \neq 0$.
Consider the two vectors u, and b. The vector u can be written as a sum of two vectors that are respectively perpendicular to one another, that is $u = w_1 + w_2$ with w_1 being in the direction of b and $w_2 \perp w_1$.



The two components w_1 and w_2 can then be computed as follows,

$$\begin{aligned} w_1 &= \text{proj}_b u = \frac{u.b}{\|b\|_2^2} b \\ w_2 &= u - w_1 \\ &= u - \frac{u.b}{\|b\|_2^2} b \end{aligned}$$

Gram Schmidt process: consider a set of non-orthogonal vectors $u_1, u_2, \ldots, u_n$, then

- Step I: $v_1 = u_1$
- Step II: $v_2 = u_2 - \frac{\langle u_2, v_1\rangle}{\|v_1\|_2^2} v_1$
- Step III: $v_3 = u_3 - \frac{\langle u_3, v_1\rangle}{\|v_1\|_2^2} v_1 - \frac{\langle u_3, v_2\rangle}{\|v_2\|_2^2} v_2$
- Step k: $v_k = u_k - \sum_{i=1}^{k} \frac{\langle u_k, v_i\rangle}{\|v_i\|_2^2} v_i$

The obtained vectors $v_1, v_2, \ldots, v_n$ are orthogonal vectors. To get the orthonormalised vectors of $v_1, v_2, \ldots, v_n$,

$$q_1 = \frac{v_1}{\|v_1\|_2}, q_2 = \frac{v_2}{\|v_2\|_2}, \ldots, q_n, \frac{v_n}{\|v_n\|_2}$$

Exercise: Compute the orthonormal set of vectors for $u_1 = [2, -2, 1]^T, u_2 = [2, 0, 2]^T, u_3 = [1, 2, 1]^T$

QR factorisation: Let $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ and denote the column vectors by a_i, $i = 1, \ldots, n$.
To orthonormalise the columns of A one may proceed as follows:

$$\begin{aligned}
\hat{q}_1 &:= a_1 && q_1 := \frac{\hat{q}_1}{r_{1,1}} \quad \text{with } r_{1,1} := \|\hat{q}_1\|_2,\\
\hat{q}_2 &:= a_2 - \underbrace{\langle q_1, a_2\rangle}_{=:r_{1,2}} q_1 && q_2 := \frac{\hat{q}_2}{r_{2,2}} \quad \text{with } r_{2,2} := \|\hat{q}_2\|_2,\\
\hat{q}_3 &:= a_3 - \underbrace{\langle q_1, a_3\rangle}_{=:r_{1,3}} q_1 - \underbrace{\langle q_2, a_3\rangle}_{=:r_{2,3}} q_2 && q_3 := \frac{\hat{q}_3}{r_{3,3}} \quad \text{with } r_{3,3} := \|\hat{q}_3\|_2,\\
&\vdots && \vdots\\
\hat{q}_n &:= a_n - \sum_{j=1}^{n-1} \underbrace{\langle q_j, a_n\rangle}_{=:r_{j,n}} q_j && q_n := \frac{\hat{q}_n}{r_{n,n}} \quad \text{with } r_{n,n} := \|\hat{q}_n\|_2,
\end{aligned}$$

This is the so-called Gram-Schmidt orthonormalisation. Equivalently to the above formulas, we can write

$$a_i = \sum_{k=1}^{i} q_k r_{ik}, \quad i = 1, \ldots, n, \qquad \Leftrightarrow \quad A = \hat{Q}\hat{R} \tag{18.1}$$

where

$$\hat{Q} = (q_1, \ldots, q_n) \in \mathbb{R}^{m\times n} \quad \text{and} \quad \hat{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ 0 & r_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & r_{n-1,n} \\ 0 & \cdots & 0 & r_{n,n} \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

A factorisation of the form (18.1) with $\hat{Q}$ having orthonormal column vectors and $\hat{R}$ upper triangular is called reduced QR factorisation.

Exercise: Compute the QR factorisation for the following matrix,

$$A = \begin{pmatrix} 2 & 2 \\ 0 & \sqrt{5}/5 \\ 1 & 0 \end{pmatrix}$$

Full rank QR factorisation: Extending $\{q_1, \ldots, q_n\}$ by vectors $\{q_{n+1}, \ldots, q_m\}$ to an orthonormal basis of $\mathbb{R}^m$, defining $Q = (q_1, \ldots, q_m) \in \mathbb{R}^{m \times m}$ and extending $\hat{R}$ with an $(m-n) \times n$ block of zeros to

$$R := \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \in \mathbb{R}^{m \times n}$$

we obtain a

$$\text{QR factorisation:} \quad A = QR. \tag{18.2}$$

Exercise: Compute the full rank QR factorisation for the following matrix,

$$A = \begin{pmatrix} 2 & 2 \\ 0 & \sqrt{5}/5 \\ 1 & 0 \end{pmatrix}$$

Theorem 18.1. *Every matrix $A \in \mathbb{R}^{m \times n}$ with $m \geq n$ can be factorised in the form $A = QR$ with $Q \in \mathbb{R}^{m \times m}$ orthogonal and $R \in \mathbb{R}^{m \times n}$ upper triangular.*

Solving LSQ using the QR factorisation [7.2]

Algorithm 10 LSQ_QR (solving LSQ using a QR factorisation)

input: $A \in \mathbb{R}^{m \times n}$ with $m \geq n = \operatorname{rank}(A)$, $b \in \mathbb{R}^m$.
output: $x \in \mathbb{R}^n$ minimiser of $g(x) = \frac{1}{2}\|Ax - b\|_2$.

1: compute a reduced QR factorisation $A = \hat{Q}\hat{R}$
2: compute $y = \hat{Q}^T b \in \mathbb{R}^n$
3: solve $\hat{R}x = y$ with **BS**

The thus computed x satisfies

$$A^T A x = \hat{R}^T \underbrace{\hat{Q}^T \hat{Q}}_{=I} \hat{R}x = \hat{R}^T y = \hat{R}^T \hat{Q}^T b = A^T b$$

which is the normal equation (7.1) and, hence, x indeed solves LSQ.
However, in practice, Gram-Schmidt is not used for computing the (reduced) QR factorisation, mainly for stability reasons (though there are stabilised variants). Alternative (often better) methods employing reflections or rotations are utilised instead.

Householder reflections

In the Gram Schmidt algorithm, the columns a_j of the matrix A are successively reduced in length as components in the directions of $q_1, \ldots, q_j$ are subtracted leaving a small vector if a_j was almost in the span of the first j columns of A. This leads to numerical instability. Application of a unitary transformation to a matrix or vector inherently preserves length. Thus, it would be beneficial if the QR factorization can be implemented as the successive application of unitary transformations. The Householder QR factorization accomplishes this.
For a given $A \in \mathbb{C}^{m\times n}$ a sequence of unitary matrices, $Q_1 Q_2 \ldots Q_n$ can be computed such that,

$$Q_n Q_{n-1} \ldots Q_1 A = R$$

where $R \in \mathbb{C}^{n\times n}$ is an upper triangular matrix. This yields,

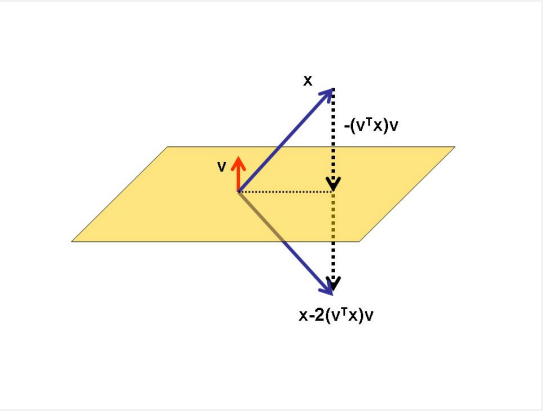
$$A = \underbrace{Q_1 Q_2 \ldots Q_n}_{Q} R$$

which is closely related to the QR factorisation of A. However note that here Q itself is unitary since the product of unitary matrices is unitary.

$$A \xrightarrow{Q_1\cdot()} \underbrace{\begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & & & \vdots \\ 0 & * & \cdots & \cdots & * \end{pmatrix}}_{=:R^{(1)}} \xrightarrow{Q_2\cdot()} \underbrace{\begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ \vdots & 0 & * & \cdots & * \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{pmatrix}}_{=:R^{(2)}} \to \cdots \to \underbrace{\begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ 0 & * & \cdots & \cdots & * \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & * \\ \vdots & & & & 0 \\ \vdots & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}}_{=:R^{(n)}} \tag{18.3}$$

For the orthogonal matrices, reflections may be used.

Householder reflection it acts like a mirroring with respect to the subspace orthogonal to a vector $v \in \mathbb{C}^n$ which is normally a unit vector, ($\|v\|_2 = 1$).



The objective here is to compute a matrix that performs the transformation that reflects a vector x through that mirror. This transformation can be computed as follows,

$$H = I - 2vv^*$$

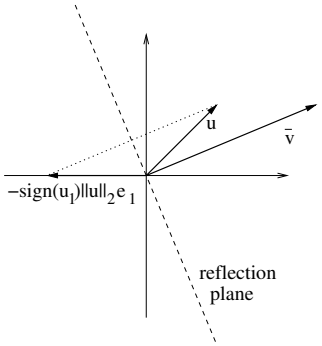
The reflection of the vector x through the mirror is then given by,

$$(I - 2vv^*)x$$

Consider the vector $u^{(k)} := (r_{k,k}^{(k-1)}, \ldots, r_{m,k}^{(k-1)})^T \in \mathbb{R}^{m-k+1}$. In step k, we want to reflect it to $-\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1$ where e_1 denotes the first standard basis vector in $\mathbb{R}^{m-k+1}$. An appropriate reflection matrix is given by

$$H_k := I_{m-k+1} - 2v^{(k)}(v^{(k)})^T \in \mathbb{R}^{(m-k+1)\times(m-k+1)}$$
$$\text{where } v^{(k)} := \frac{\hat{v}^{(k)}}{\|\hat{v}^{(k)}\|_2} \text{ with } \hat{v}^{(k)} := \text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)}$$

where I_{m-k+1} denotes the identity matrix in $\mathbb{R}^{(m-k+1)\times(m-k+1)}$.



Indeed, using that

$$\begin{aligned}\|\hat{v}^{(k)}\|_2^2 &= (\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)})^T(\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)}) \\ &= \|u^{(k)}\|_2^2 + 2\text{sign}(u_1^{(k)})u_1^{(k)}\|u^{(k)}\|_2 + \|u^{(k)}\|_2^2 = 2(\text{sign}(u_1^{(k)})u_1^{(k)}\|u^{(k)}\|_2 + \|u^{(k)}\|_2^2)\end{aligned}$$

we obtain that

$$\begin{aligned}H_k u^{(k)} &= u^{(k)} - 2\frac{\hat{v}^{(k)}}{\|\hat{v}^{(k)}\|_2}\frac{(\hat{v}^{(k)})^T u^{(k)}}{\|\hat{v}^{(k)}\|_2} \\ &= u^{(k)} - 2\frac{\hat{v}^{(k)}}{\|\hat{v}^{(k)}\|_2^2}(\text{sign}(u_1^{(k)})u_1^{(k)}\|u^{(k)}\|_2 + \|u^{(k)}\|_2^2) = u^{(k)} - \hat{v}^{(k)} = -\text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1.\end{aligned}$$

Moreover, $(H_k)^{-1} = (H_k)^T = H_k$ is orthogonal so that

$$Q_k := \begin{pmatrix} I_{k-1} & 0 \\ 0 & H_k \end{pmatrix} \in \mathbb{R}^{m\times m}$$

is orthogonal, too, and this is an appropriate matrix to be used in (18.3).

Algorithm 11 QR_HH (computing a QR factorisation with Householder reflections)

input: $A = (a_{ij})_{i,j=1}^{m,n} \in \mathbb{R}^{m\times n}$ with $m \geq n$ and full rank.
output: $Q \in \mathbb{R}^{m\times m}$ orthogonal, $R \in \mathbb{R}^{m\times n}$ upper triangular with $A = QR$.

1: $R^{(0)} := A$, $Q^{(0)} := I_m$.
2: **for** $k = 1$ to $n-1$ (to n if $m > n$) **do**
3: $u^{(k)} := (r_{k,k}^{(k-1)}, \ldots, r_{m,k}^{(k-1)})^T \in \mathbb{R}^{m-k+1}$
4: $\hat{v}^{(k)} := \text{sign}(u_1^{(k)})\|u^{(k)}\|_2 e_1 + u^{(k)} \in \mathbb{R}^{m-k+1}$
5: $v^{(k)} := \hat{v}^{(k)}/\|\hat{v}^{(k)}\|_2$
6: $H_k := I_{m-k+1} - 2v^{(k)}(v^{(k)})^T \in \mathbb{R}^{(m-k+1)\times(m-k+1)}$
7: $Q_k := \text{diag}(I_{k-1}, H_k) \in \mathbb{R}^{m\times m}$
8: $R^{(k)} := Q_k R^{(k-1)}$
9: $Q^{(k)} := Q^{(k-1)}Q_k$
10: **end for**
11: return $Q^{(n-1)}$ and $R^{(n-1)}$ ($Q^{(n)}$ and $R^{(n)}$ if $m > n$)

A reduced QR factorisation is obtained by dropping the last $m-n$ columns of Q and the last $m-n$ rows of R (which contain zeros only).