5 Least squares problems and orthogonalization

So far we dealt with linear systems that have \( n \) equations for \( n \) unknowns. In many real-life scenarios, this is not the case, and we have more equations than variables. Usually in this case there is no true solution that fulfils all equations. We wish only to get the best vector that approximates all the equations in some “optimal” way. In other words, we wish to find a vector that minimizes the discrepancy of the equations in some norm. In this course we will focus of the \( \ell_2 \) norm, which is the most popular and easy-to-handle norm. In this section, we will consider only real-valued problems.

5.1 Least squares minimization - full rank case

[This is a rehearsal of material that was taught earlier in the course]

Let \( A \in \mathbb{R}^{m \times n} \) be of rank \( n \) with \( m \geq n \), \( x \in \mathbb{R}^n \) and \( b \in \mathbb{R}^m \). We wish to find \( x \) that minimizes the discrepancy vector \( r(x) = Ax - b \) in a least squares sense, that is minimize \( \|r(x)\|_2 \). Since we are interested in the minimizer \( x \) and not in the minimal value of \( r(x) \), we may minimize the squared objective \( \|r(x)\|_2^2 \), since it will lead to the same solution. We get the problem

\[
\hat{x} = \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2, \tag{42}
\]

where \( \arg \min \{ f \} \) denotes the argument that minimizes the objective \( f \) rather than the minimal value of \( f \) as in \( \min \{ f \} \). If \( m = n \) and \( A \) is full rank, then there is a solution such that \( Ax = b \) (which is \( A^{-1}b \)), and the minimization will turn \( r(x) \) to be 0. So will be the case if \( b \) is in the range of \( A \). In any case, if \( r(x) = 0 \) can be achieved or not, the least squares problem is well-defined and has an optimal solution.

5.1.1 Least squares through derivatives of linear and quadratic functions

Now we will see how to solve the least squares minimization. This will be done by zeroing the first derivative of 42. It is known that an extremum point of function \( f(x) \) is a point
least squares problems and orthogonalization

where the first derivative of $f$, i.e., its gradient, is equal to the zero vector.

\[
\nabla f = \begin{bmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2} \\
\vdots \\
\frac{\partial f}{\partial x_n}
\end{bmatrix} = 0.
\]

We will see the following examples that will help us to get the gradient of (42).

1. As a simple example, consider the linear function $f(x) = v^\top x$ for two vectors $x, v \in \mathbb{R}^n$. $f(x) = \sum_{i=1}^{n} x_i v_i$, and therefore, $\frac{\partial f}{\partial x_i} = v_i$. This means that $\nabla f = v$.

2. Consider the quadratic function $f(x) = x^\top x$ for $x \in \mathbb{R}^n$. $f(x) = \sum_{i=1}^{n} x_i^2$, and therefore, $\frac{\partial f}{\partial x_i} = 2x_i$. This means that $\nabla f = 2x$. We can get the same result using the formula $(uv)' = u'v + v'u$ and applying it on $f(x) = x^\top x$, each time referring to one variable $x$ as a constant, using the previous linear example.

3. Consider the function $f(x) = v^\top A^\top A x$ for $v, x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$. The vector $v^\top A^\top A$ is just a constant vector, so like the first linear case, we will get $\nabla f = A^\top Av$.

4. Consider the quadratic function $f(x) = x^\top A^\top A x$ for $x \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$. This time, we will twice refer to the vector $x^\top A^\top A$ as a constant vector, and take the derivative with respect to the other one. Similarly to the previous cases we will get $\nabla f = 2A^\top Ax$.

Rewriting our LS problem in Eq. (42) in a more explicit form we get

\[
f(x) = \|Ax - b\|_2^2 = (Ax - b)^\top (Ax - b) = x^\top A^\top A x - 2b^\top A x + b^\top b.
\]

By the rules above we get that

\[
\nabla f(x) = 2A^\top Ax - 2A^\top b,
\]

and setting $\nabla f(x) = 0$ will lead to the system

\[
A^\top Ax = A^\top b,
\]
which is called the normal equation. Assuming that the matrix \((A^\top A)\) is non-singular, the minimizer of the LS will be
\[
\hat{x} = (A^\top A)^{-1}A^\top b.
\]
We assumed above that \(A\) is full rank and hence the matrix \(A^\top A\) is invertible. However, if the matrix is singular, then we may have infinitely many solutions, and each one satisfying the normal equations is a critical point of the objective in the LS problem. As mentioned before, by zeroing the gradient of \(f\) we found an extremum point of \(f\). In the case of a convex quadratic \(f\) like the one here, any point that zeroes the gradient is a global minimum.

**Example 12** (Algebraic Least Squares). Suppose that we need to minimize \(\|Ax - b\|_2\) with
\[
A = \begin{bmatrix}
-1 & -1 & 1 \\
1 & 3 & 3 \\
-1 & -1 & 5 \\
1 & 3 & 7 \\
\end{bmatrix}, \quad b = \begin{bmatrix}
0 \\
23 \\
15 \\
39 \\
\end{bmatrix}.
\]

Using the normal equations, we will form \(A^\top A\), and \(A^\top b\), and solve \(A^\top A x = A^\top b\):
\[
A^\top A = \begin{bmatrix}
4 & 8 & 4 \\
8 & 20 & 24 \\
4 & 24 & 84 \\
\end{bmatrix}, \quad A^\top b = \begin{bmatrix}
47 \\
171 \\
417 \\
\end{bmatrix}, \quad x = (A^\top A)^{-1}A^\top b = \begin{bmatrix}
0.375 \\
3.75 \\
3.875 \\
\end{bmatrix}.
\]

Note that the residual norm in this example comes out
\[
r = Ax - b = \begin{bmatrix}
-0.25 \\
0.25 \\
0.25 \\
-0.25 \\
\end{bmatrix}.
\]

Here we got an equal error (in magnitude) in all equations, but that will not always be the case. Also, not surprisingly,
\[
A^\top r = \begin{bmatrix}
0 \\
0 \\
0 \\
\end{bmatrix}.
\]
That is, the residual is orthogonal to the columns of $A$. That is by the construction of the normal equations

$$A^\top (Ax - b) = 0.$$  

Note that on a computer, the vector of zeros there will be vector of small values due to round-off errors. In this example, in double precision, it happened to be on our computer:

$$A^\top r = \begin{bmatrix} -0.6 \\ -2.0 \\ -5.8 \end{bmatrix} \cdot 1e - 14.$$  

This is a "numerical zero".

### 5.1.2 Weighted least squares

A lot of times we wish to give more emphasis to certain equations in our least squares minimization, because there is a reason to believe that there is less noise there, for example (that is not the only case, though). This can be done with a weighed norm. Let $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. We wish to find $x$ that minimizes the discrepancy vector $r(x) = Ax - b$ in a weighted least squares sense, that is minimize

$$\|r\|_W^2 = r^\top W r,$$

where $W \in \mathbb{R}^{m \times m}$ is a positive definite matrix, usually a diagonal matrix of positive weights. We get the problem

$$\hat{x} = \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_W^2,$$

and if we do similar steps to the ones we’ve done before we get

$$\hat{x} = (A^\top W A)^{-1} A^\top W b.$$
Example 13 (Algebraic Weighted Least Squares). Suppose that we need to minimize $\|Ax-b\|_2$ with (exactly as before)

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 23 \\ 15 \\ 39 \end{bmatrix}. $$

Suppose now that we want to approximate the first equation in 10 approximately times the accuracy (we are more certain about this equation). We will set

$$W = \begin{bmatrix} 10 \\ 1 \\ 1 \\ 1 \end{bmatrix}. $$

Using the normal equations, we will form $A^\top WA$, and $A^\top Wb$, and solve $A^\top WAx = A^\top Wb$:

$$A^\top WA = \begin{bmatrix} 13 & 17 & -5 \\ 17 & 29 & 15 \\ -5 & 15 & 93 \end{bmatrix}, \quad A^\top Wb = \begin{bmatrix} 47 \\ 171 \\ 417 \end{bmatrix}, \quad \hat{x} = (A^\top WA)^{-1}A^\top Wb = \begin{bmatrix} -0.097 \\ 3.97 \\ 3.84 \end{bmatrix}. $$

Note that the residual norm in this example comes out

$$r = Ax - b = \begin{bmatrix} -0.032 \\ 0.32 \\ 0.32 \\ -0.32 \end{bmatrix}. \quad (46) $$

It is clear that the accuracy in which the first equation is satisfied is 10 times larger than the rest now.
5. Least squares problems and orthogonalization

5.1.3 Regularized least squares

We’ve seen before that if the matrix $A$ in the least squares problem is not full rank, then the matrix $A^\top A$ may be singular (or ill-conditioned), and in that case, we may have infinitely many solutions satisfying the equation $A^\top A x = A^\top b$. In many applications we do not control the matrix $A$ - it’s an input of the problem, like a data matrix from an unreliable source. Still, we wish to have a well-defined problem with a single solution. This can be done by adding a regularization term, and each regularization that we choose will influence the solution in its own way. Here, we will demonstrate a pretty common and simple Tikhonov regularization

$$\hat{x} = \arg \min_{x \in \mathbb{R}^n} \| A x - b \|_2^2 + \lambda \| C x \|_2^2,$$

where $C$ is a matrix, usually chosen as the identity $I$, and $\lambda > 0$ is a small parameter that balances between the original LS equation and the regularization. The Tikhonov regularization keeps the problem a standard least squares problem. Indeed, the problem above can be rewritten as

$$\hat{x} = \arg \min_{x \in \mathbb{R}^n} \left\| \begin{bmatrix} A \\ \sqrt{\lambda} C \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2^2,$$

and after following the above steps, the minimizer is given by

$$\hat{x} = (A^\top A + \lambda C^\top C)^{-1} A^\top b.$$ 

The matrix $A^\top A + \lambda C^\top C$ is guaranteed to be invertible if $C$ is full rank, which is why $C$ is often chosen as the identity $I$.

**Example 14** (Regularized least squares). Suppose that we need to minimize $\| A x - b \|_2$ with

$$A = \begin{bmatrix} -1 & -1 & -1 \\ 1 & 3 & 2 \\ -1 & -1 & -1 \\ 1 & 3 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} -2.999 \\ 5.999 \\ -3.001 \\ 6.001 \end{bmatrix}.$$

As you may notice, the matrix $A$ is singular - the third column is an average of the first two. The vector $b$ is the sum of all columns with a little bit of noise. That is, $x = [1, 1, 1]^\top$ is a reasonable solution. However, the matrix $A^\top A$ is singular, hence we cannot solve the system.
with standard LU factorization.

What we’ll do is add Tikhonov regularization to solve the problem. That is, we will solve

$$\hat{x} = \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \lambda \|x\|_2^2,$$

with $\lambda = 0.01$, which is chosen arbitrarily in this example. The solution will be

$$\hat{x} = (A^\top A + \lambda I)^{-1}A^\top b.$$

We get:

$$A^\top A = \begin{bmatrix} 4 & 8 & 6 \\ 8 & 20 & 14 \\ 6 & 14 & 10 \end{bmatrix}, \quad A^\top b = \begin{bmatrix} 18 \\ 42 \\ 30 \end{bmatrix}, \quad \hat{x} = (A^\top A + \lambda I)^{-1}A^\top b \approx \begin{bmatrix} 0.99 \\ 1.003 \\ 0.998 \end{bmatrix}.$$

You may also see an example from an image processing application in the appendix 5.8.

### 5.2 Orthogonalization using Gram Schmidt and QR factorization

Assume that we have a set of vectors $\{a_i\}_{i=1}^n \in \mathbb{R}^m$, $m \geq n$. We know that if the vectors $a_i$ are linearly independent, then they span the whole space $\mathbb{R}^n$. Suppose that we have all these vectors as columns of a matrix $A = [a_1 | a_2 | \ldots | a_n]$. If $a_i$ are linearly independent, then $A$ is full rank. However, if some of the vectors $a_i$ are almost linearly dependent, then $A$ is almost singular, and performing operations like inverting $A$ may involve high numerical errors.

Back to our least squares problem, we noticed that the solution is achieved when $A^\top (Ax - b) = 0$. That is, when the residual $Ax - b$ is orthogonal to the columns of $A$. In fact, this also means that for the solution $x$ the residual vector $Ax - b$ is not in the span of the subspace defined by $A$’s columns. This means that the least squares approximation is all about the span of $A$’s columns rather than $A$ itself. Again, if the columns of $A$ are close to being linearly dependent, then we may have large numerical errors (that will be introduced when we solve the linear system with the matrix $A^\top A$). We will now see how to solve a LS problem using an orthogonalization process, that will make the calculation more numerically stable.
5.2.1 Gram Schmidt orthogonalization

Assume that we are given with a set of linearly independent vectors \( \{a_i\}_{i=1}^n \), and we wish to build a set of orthogonal vectors \( \{q_i\}_{i=1}^n \), based on linear combinations of \( a_i \)'s. The orthogonality is based on some inner product \( \langle \cdot, \cdot \rangle \). One process for doing this is called the Gram Schmidt algorithm.

- Step 1: \( q_1 = a_1 \).
- Step 2: \( q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\langle q_1, q_1 \rangle} q_1 \).
  
  We indeed see that \( \langle q_2, q_1 \rangle = \langle a_2, q_1 \rangle - \frac{\langle a_2, q_1 \rangle}{\langle q_1, q_1 \rangle} \langle q_1, q_1 \rangle = 0 \)
- Step 3: \( q_3 = a_3 - \frac{\langle a_3, q_1 \rangle}{\langle q_1, q_1 \rangle} q_1 - \frac{\langle a_3, q_2 \rangle}{\langle q_2, q_2 \rangle} q_2 \).
  
  We indeed see that \( \langle q_3, q_1 \rangle = \langle a_3, q_1 \rangle - \frac{\langle a_3, q_1 \rangle}{\langle q_1, q_1 \rangle} \langle q_1, q_1 \rangle - \frac{\langle a_3, q_2 \rangle}{\langle q_2, q_2 \rangle} \langle q_2, q_1 \rangle = 0 \)
  and we also see that \( \langle q_3, q_2 \rangle = \langle a_3, q_2 \rangle - \frac{\langle a_3, q_1 \rangle}{\langle q_1, q_1 \rangle} \langle q_1, q_2 \rangle - \frac{\langle a_3, q_2 \rangle}{\langle q_2, q_2 \rangle} \langle q_2, q_2 \rangle = 0 \)
- Step \( i \): \( q_i = a_i - \sum_{j=1, \ldots, i-1} \frac{\langle a_i, q_j \rangle}{\langle q_j, q_j \rangle} q_j \).
  
  Similarly to before we will have that \( \langle q_i, q_j \rangle = 0 \) for \( j = 1, \ldots, i - 1 \).

Let us now rewrite Steps 1 to 3 above in a slightly different way \( \| q_i \| = \sqrt{\langle q_i, q_i \rangle} \):

- Step 1: \( q_1 = a_1 \).
- Step 2: \( q_2 = a_2 - \frac{\langle a_2, q_1 \rangle}{\| q_1 \|^2} q_1 \).
- Step 3: \( q_3 = a_3 - \frac{\langle a_3, q_1 \rangle}{\| q_1 \|^2} q_1 - \frac{\langle a_3, q_2 \rangle}{\| q_2 \|^2} q_2 \).

And now, with some simple arithmetics we can get:

\[
\begin{align*}
    a_1 &= \| q_1 \| \frac{q_1}{\| q_1 \|} \\
    a_2 &= \| q_2 \| \frac{q_2}{\| q_2 \|} + \langle a_2, \frac{q_1}{\| q_1 \|} \rangle \frac{q_1}{\| q_1 \|} \\
    a_3 &= \| q_3 \| \frac{q_3}{\| q_3 \|} + \langle a_3, \frac{q_2}{\| q_2 \|} \rangle \frac{q_2}{\| q_2 \|} + \langle a_3, \frac{q_1}{\| q_1 \|} \rangle \frac{q_1}{\| q_1 \|} \\
\end{align*}
\]

(47)

The triangular shape in Eq. (47) above gives rise to a matrix factorization that is based on orthogonalization with respect to the standard inner product \( \langle u, v \rangle = u^\top v \).
5. Least squares problems and orthogonalization

5.2.2 The QR factorization

The process above is used to transform a set of vector arranged in a matrix
\[ A = \begin{bmatrix} a_1 & a_2 & \ldots & a_n \end{bmatrix} \]
into a set of orthogonal and normalized vectors, arranged in a matrix
\[ Q = \begin{bmatrix} q_1 & q_2 & \ldots & q_n \end{bmatrix}. \]
With minimal effort, we can set this process into a factorization
\[ A = QR, \]
where \( Q \) is an orthogonal matrix \((Q^TQ = I)\), and \( R \) is an upper triangular matrix, defined by the calculated constants in the orthogonalization process. This factorization is called a QR factorization and is a very useful tool, especially in the context of least squares. Algorithm 7 is the “economic” QR factorization using Gram-Schmidt for a matrix \( A \in \mathbb{R}^{m \times n} \) with \( m \geq n \).

```plaintext
Algorithm: Gram Schmidt QR

# A = [a_1 | a_2 | \ldots | a_n] ∈ \mathbb{R}^{m \times n}
Initialize: R = 0^{n \times n}, R_{1,1} = \|a_1\|_2, q_1 = \frac{a_1}{R_{1,1}}

for i = 2, ..., n do
    q_i ← a_i
    for j = 1 : i - 1 do
        R_{j,i} = q_j^T a_i
        q_i ← q_i - R_{j,i} q_j
    end
    R_{i,i} = \|q_i\|_2
    q_i ← \frac{q_i}{R_{i,i}}
end
```

Algorithm 7: QR factorization using Gram Schmidt.

It turns out, however, that the Gram-Schmidt process above can be numerically unstable and introduce high numerical errors. This phenomenon can be fixed with a small almost unnoticeable algorithmic change. A key difference between the two algorithms is that the \( R_{j,i} \)'s must be computed sequentially in MGS because \( R_{j,i} \) depends on \( q_i \) which is changing, while in GS they can be computed in parallel because \( R_{j,i} \) do not depend on \( q_i \).
Algorithm 8: QR factorization using Modified Gram Schmidt.

Example 15 (Economic QR factorization). Assume we have the following vectors:

\[
\{a_i\}_{i=1}^{3} = \left\{ \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 3 \\ -1 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \right\},
\]

or the equivalently, the following matrix

\[
A = \begin{bmatrix}
-1 & -1 & 1 \\
1 & 3 & 3 \\
-1 & -1 & 5 \\
1 & 3 & 7 \\
\end{bmatrix}.
\]

We will now decompose the \( A \) into a QR factorization.

Initialization:

\[
\|a_1\|_2 = 2 \Rightarrow q_1 = (-1/2, 1/2, -1/2, 1/2)^T, \quad R_{1,1} = 2.
\]
Iteration $i = 2$

\[
\begin{align*}
z &= a_2 - (a_2^\top q_1)q_1 = a_2 - R_{1,2}q_1 = a_2 - 4q_1 = (1, 1, 1)^T \\
R_{2,2} &= \|z\| = 2 \\
q_2 &= \frac{z}{R_{2,2}} = (1/2, 1/2, 1/2, 1/2)^T
\end{align*}
\]

Iteration $i = 3$

\[
\begin{align*}
z &= a_3 - (a_3^\top q_1)q_1 - (a_3^\top q_2)q_2 = a_3 - R_{1,3}q_1 - R_{2,3}q_2 \\
&= a_3 - 2q_1 - 8q_2 = (-2, -2, 2, 2)^T. \\
R_{3,3} &= \|z\| = 4 \\
q_3 &= \frac{z}{R_{3,3}} = (-1/2, -1/2, 1/2, 1/2)^T
\end{align*}
\]

Finally we get:

\[
Q = \frac{1}{2} \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, R = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}
\]

**Remark 9** (Full QR factorization). The QR factorization above is the “economic” QR where for a matrix $A \in \mathbb{R}^{m \times n}$ with $m > n$ we find $Q \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$. Alternatively, we can also find an equivalent factorization such that $Q \in \mathbb{R}^{m \times m}$ and $R \in \mathbb{R}^{m \times n}$. Some packages compute it by default. To achieve this we can complement $Q$ with the missing subspace of size $m - n$, such that $Q$ would be orthogonal sized $m \times m$, and its columns would span $\mathbb{R}^m$, and placing zeros on the bottom $m - n$ rows of $R$. This is an equivalent factorization, which has mostly theoretical value and not practical.
5.3 Least Squares solution using QR factorization

One interpretation of orthogonal matrices is that they are multi-dimensional rotation matrices, which do not change the lengths of vectors. This is easy to show: assume that $Q \in \mathbb{R}^{n \times n}$ is an orthogonal matrix ($Q^T Q = I$), and that $v \in \mathbb{R}^n$ is a vector, then multiplying $Q$ with $v$ preserves the $\ell_2$ norm of $v$:

$$\|Qv\|_2^2 = v^T Q^T Q v = v^T v = \|v\|_2^2.$$  \hfill (48)

This is an important property that we’re about to exploit.

Back to our least squares problem

$$\hat{x} = \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2.$$

This time we will see another approach for solving the problem, which is also more numerically stable. We will later see, that the condition number of $A^T A$ is the square of the condition number of $A$. Therefore, solving the normal equations may be ill-conditioned. Suppose that $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, and suppose that we have an economic QR factorization $A = QR$, where $Q \in \mathbb{R}^{m \times n}$ and $R \in \mathbb{R}^{n \times n}$. Note that

$$A^T A = (QR)^T (QR) = R^T Q^T QR = R^T R.$$

We know that the solution is:

$$\hat{x} = (A^T A)^{-1} A^T b = (R^T R)^{-1} R^T Q^T b = R^{-1} (R^T)^{-1} R^T Q^T b = R^{-1} Q^T b,$$

where here we used the fact that $R$ is square and invertible (full rank).

**Example 16** (Least squares via QR). *Suppose that we need to minimize $\|Ax - b\|_2$ with

$$A = \begin{bmatrix} -1 & -1 & 1 \\ 1 & 3 & 3 \\ -1 & -1 & 5 \\ 1 & 3 & 7 \end{bmatrix}, \quad b = \begin{bmatrix} -1 \\ 23 \\ 15 \\ 39 \end{bmatrix}. $$
We will now solve the minimization by two equivalent approaches:

1. **Using the normal equations, we will form** $A^\top A$, and $A^\top b$, and solve $A^\top A x = A^\top b$:

   $$
   A^\top A = \begin{bmatrix} 4 & 8 & 4 \\ 8 & 20 & 24 \\ 4 & 24 & 84 \end{bmatrix}, \quad A^\top b = \begin{bmatrix} 48 \\ 172 \\ 416 \end{bmatrix}, \quad \hat{x} = (A^\top A)^{-1} A^\top b = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}.
   $$

   The solution of the normal equation is done using LU or Cholesky factorizations.

2. **Alternatively, we can use the QR factorization (found in the previous example):**

   $$
   Q = \frac{1}{2} \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad R = \begin{bmatrix} 2 & 4 & 2 \\ 0 & 2 & 8 \\ 0 & 0 & 4 \end{bmatrix}.
   $$

   Now we solve $Rx = Q^\top b$:

   $$
   Q^\top b = \begin{bmatrix} 24 \\ 38 \\ 16 \end{bmatrix}, \quad x = R^{-1} Q^\top b = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}.
   $$
5.4 Spectral Decomposition - reminder and some facts

**Theorem 18** (Spectral Decomposition). Let $A \in \mathbb{C}^{n \times n}$, and $Av_i = \lambda_i v_i$ for $i = 1, \ldots, n$. If the vectors $\{v_i\}$ are linearly independent then $A$ is diagonalizable such that

$$
A = V \Lambda V^{-1}
$$

$$
\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)
$$

$$
V = [v_1|v_2|\ldots|v_n].
$$

This is just a writing in matrix form. Note that $Av_i = \lambda_i v_i$ for $i = 1, \ldots, n$ can be written as $AV = V\Lambda$ is matrix form.

**Theorem 19** (Normal and Hermitian Spectral Decomposition). Let $A \in \mathbb{C}^{n \times n}$ be normal (or Hermitian) then there exist a unitary matrix $U \in \mathbb{C}^{n \times n}$ such that

$$
U^*U = I
$$

$$
A = U\Lambda U^*
$$

$$
\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)
$$

$$
U = [u_1|u_2|\ldots|u_n],
$$

where $u_i \in \mathbb{C}^n$ are the eigenvectors of $A$, and $\lambda_i$ are the eigenvalues of $A$. If $A$ is normal then $\lambda_i \in \mathbb{C}$, and if Hermitian then $\lambda_i \in \mathbb{R}$ (real numbers). If $A$ is real and symmetric, **all the matrices and eigenvalues above are real-valued**.
%% MATLAB code for demonstrating eigenvalues and eigenvectors:

```
>> A = randn(3)
[V,D] = eig(A)
norm(A - V*D*inv(V),'fro')

B = A'*A % B is now a symmetric matrix
[U,D] = eig(B)
norm(B - U*D*U','fro')

%% Output:
A =
   0.3129  -0.1649   1.1093
  -0.8649   0.6277  -0.8637
  -0.0301   1.0933   0.0774

V =
   0.8215   0.0000   0.0000
  0.2570   0.0000   0.0000
 -0.5090   0.0000   0.0000

D =
  -0.4261   0.0000   0.0000
  0.0000  0.7220   0.0000
  0.0000  0.0000  3.1475

% This means: A = V*D*inv(V)
ans =
     2.3265e-15

B =
   0.8468  -0.6273   1.0917
  -0.6273   1.6164  -0.6404
   1.0917  -0.6404   1.9824

U =
   0.8717   0.1041   0.4788
  0.1724   0.8496  -0.4985
 -0.4587   0.5171   0.7227

D =
   0.1483   0.0000   0.0000
   0.0000   1.1498   0.0000
   0.0000   0.0000   3.1475

% This means: B = U*D*U'
ans =
     3.5804e-15
```
5.5 SVD - Singular Value Decomposition

In this section we will deal with the singular value decomposition, which is one of the more important and useful matrix factorizations in linear algebra.

**Theorem 20.** Let \( A \in \mathbb{R}^{m \times n} \), then there exist orthogonal matrices \( U \in \mathbb{R}^{m \times m} \) and \( V \in \mathbb{R}^{n \times n} \), such that \( V^TV = I_n \), \( U^TU = I_m \) and

\[
A = U\Sigma V^T, \quad \Sigma = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_p) \in \mathbb{R}^{m \times n} \quad p = \min(m, n),
\]

where \( \sigma_1 \geq \sigma_2 \geq \sigma_3 \geq ... \geq \sigma_p \geq 0 \). This is called the full SVD factorization of \( A \).

You may see the proof in Section 5.9.2.

**Remark 10.** Similarly to the “economic” and full versions of QR, the SVD factorization also has “economic” and full factorizations, where the economic factorization is more useful in practice, and the full one is mostly used in theoretical results. For start, note that in the full SVD factorization, the matrix \( \Sigma \) may actually be rectangular with blocks of zeros. For example, if \( m > n \) then the bottom \( m - n \) rows in \( \Sigma \) are zeros. Given a matrix \( A \in \mathbb{R}^{m \times n} \), let \( p = \min(m, n) \). In the economic SVD we find orthogonal matrices \( U \in \mathbb{R}^{m \times p} \), and \( V \in \mathbb{R}^{p \times n} \), and a diagonal matrix \( \Sigma \in \mathbb{R}^{p \times p} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_p) \). This is the economic SVD decomposition, and in this case \( \Sigma \) is square and diagonal.

**Remark 11 (Complex SVD - for general knowledge).** In the case of \( A \in \mathbb{C}^{m \times n} \), there is a complex SVD decomposition, where the matrices \( U \) and \( V \) are unitary (complex and orthogonal), but \( \Sigma \) remains real with a positive diagonal.

**Remark 12 (Computing SVD - for general knowledge).** Computing the SVD decomposition is considered to be a rather expensive and complicated task and we will not address it in this course. Generally, there are different algorithms for computing the SVD, and in all of them, maintaining the accuracy of the computation is necessary. In general, it is not numerically “safe” to compute \( A^TA \), since we square the condition number. In essence, the SVD of a full matrix \( A \) starts with a Bi-orthogonalization process (See Appendix). Then, \( A^TA \) is essentially transformed to be tri-diagonal and we can compute its determinant efficiently to find its eigenvalues.
5. Least squares problems and orthogonalization

**Least squares via SVD** Suppose again that we are solving our least squares problem \( A \in \mathbb{R}^{m \times n}, m \geq n \)

\[
\hat{x} = \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2.
\]

and this time assume that we have the (economic) SVD decomposition of \( A: A = U\Sigma V^T \) \((U \in \mathbb{R}^{m \times n} \text{ and } V \in \mathbb{R}^{n \times n} \text{ are orthogonal matrices}). Similarly to the QR case,

\[
\begin{align*}
A^TA &= (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^TU^T U\Sigma V^T = V\Sigma^T \Sigma V^T. \\
A^Tb &= (U\Sigma V^T)^Tb = V\Sigma^T U^T b.
\end{align*}
\]

We get an equation:

\[
V\Sigma^T \Sigma V^T \hat{x} = V\Sigma^T U^T b.
\]

Since \( V \) and \( \Sigma \) are square and invertible, we can “divide from the left”

\[
\Sigma V^T \hat{x} = U^T b,
\]

and applying a change of variables \( \hat{y} = V^T \hat{x} \) we get a diagonal system

\[
\Sigma \hat{y} = U^T b.
\]

After solving this equation, our solution is \( \hat{x} = (V^T)^{-1} \hat{y} = V \hat{y} \).

**Example 17** (Least squares via the SVD). *Suppose that we need to minimize \( \|Ax - b\|_2 \) with*

\[
A = \begin{bmatrix}
-1 & -1 & 1 \\
1 & 3 & 3 \\
-1 & -1 & 5 \\
1 & 3 & 7 \\
\end{bmatrix}, \quad b = \begin{bmatrix}
-1 \\
23 \\
15 \\
39 \\
\end{bmatrix}.
\]

*We will now solve this problem using a third approach (in addition to the two in the previous example), via the economic SVD decomposition which is the default in Julia’s svd().*
A = [-1.0 -1.0 1.0 ; 1.0 3.0 3.0 ; -1 -1 5.0 ; 1 3 7];

b = [ -1.0 ; 23 ; 15 ; 39];

(U,S,V) = svd(A);

U

4x3 Array{Float64,2}:
0.0574049 -0.410658 0.760306
0.402203 0.509243 0.573502
0.450295 -0.732826 -0.100999
0.795093 0.187076 -0.287803

S # Julia returns only a vector of singular values, not a diagonal matrix
3-element Array{Float64,1}:
9.61534
3.91982
0.424511

V

3x3 Array{Float64,2}:
0.0717183 0.469358 -0.88009
0.320757 0.824639 0.465924
0.944442 -0.31571 -0.0914083

Utb = U'*b;

Utb

3-element Array{Float64,1}:
46.9563
8.42681
-0.309061

y = Utb./S; # here we invert Sigma

y

3-element Array{Float64,1}:
4.88348
2.14979
-0.728041

x = V*y;

x

3-element Array{Float64,1}:
2.0
3.0
4.0
5.6 Minimum norm solution to rank-deficient LS and the pseudo inverse

So far we’ve looked at a full-rank version of the least-squares problem, where the matrix $A \in \mathbb{R}^{m \times n}$, with $m \geq n$, is full rank (rank $n$), which also means that $A^\top A$ is full rank and invertible. This means that there is a unique solution to the normal equation. Also, the matrix $R$ in the $QR$ factorization of $A$ is invertible.

Now we consider the LS problem

$$\hat{x} = \arg \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2,$$

where $\text{rank}(A) < n$, and $A^\top A$ is singular, and the normal equation

$$A^\top Ax = A^\top b,$$

has infinitely many solutions. One way to see that is to place the SVD decomposition $A = U\Sigma V^\top$ ($U \in \mathbb{R}^{m \times n}$, $\Sigma \in \mathbb{R}^{n \times n}$,$V \in \mathbb{R}^{n \times n}$) in the normal equation (similarly as before):

$$V\Sigma^2 V^\top x = V\Sigma U^\top b.$$

After multiplying by $V^\top$ from the left, this equation can also be written as:

$$\Sigma^2 y = \hat{b},$$

for $\hat{b} = U^\top b$, and $y = V^\top x$. If there is a zero diagonal entry in $\Sigma$, there will also be a zero in the right-hand-side, indicating that this equation has infinitely many solutions.

Out of all these solutions, in many cases we wish to find the one with minimum $\ell_2$ norm. Since $\Sigma$ is diagonal, $y_i = \hat{b}_i/\sigma_i$ for the entries where $\sigma_i \neq 0$ and $y_i$ can get any value in the entries where $\sigma_i = 0$. Obviously, the solution $y$ with the lowest $\ell_2$ norm is the one with zeros where $\sigma_i = 0$, and so will be the solution $x$ since $\|x\|_2 = \|Vy\|_2 = \|y\|_2$.

To sum up, if $A$ is full rank, then the solution to the least squares problem is

$$\hat{x} = (A^\top A)^{-1} A^\top b = V(\Sigma)^{-1} U^\top = A^\dagger b.$$
The matrix $A^\dagger = (A^TA)^{-1}A^T = V\Sigma^{-1}U^T$ is also called the Moore-Penrose pseudoinverse of $A$. In the case where $A$ is not full rank, we will define $A^\dagger$ to be $V(\Sigma)^\dagger U^T$ where

$$(\Sigma^\dagger)_{ii} = \begin{cases} (\Sigma_{ii})^{-1} & \Sigma_{ii} \neq 0 \\ 0 & \Sigma_{ii} = 0 \end{cases}$$

**Remark 13.** We can get the minimum norm LS solution using the pseudo inverse by taking a different approach. Consider the regularized problem $\hat{x} = \arg\min \|Ax - y\|_2^2 + \lambda\|x\|_2^2$ for $\lambda > 0$. In this problem, the normal equation is always solvable, and because we penalize the norm of the solution, we will always get a minimum norm solution to the LS problem, if the matrix $A$ is rank deficient. The pseudo inverse will be achieved for $\lambda \to 0$.

### 5.7 Best low-rank approximation of matrices

Another important role of the SVD factorization is answering the following question: Given a matrix $A \in \mathbb{R}^{m \times n}$, what is the matrix $X \in \mathbb{R}^{m \times n}$ of rank $p < \min(n,m)$ that best approximates $A$ in Frobenius norm?

For starts, $X \in \mathbb{R}^{m \times n}$ will be a low-rank matrix, and we will be able to represent it as a sum of $p$ outer products. To understand this, consider the following example, where the following outer product leads to a rank-one matrix:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ 2 & 3 & 4 \\ 2 & 3 & 4 \end{bmatrix}.$$

Generally, if we have the sets $\{v_i\}_{i=1}^p \in \mathbb{R}^n$, and $\{u_i\}_{i=1}^p \in \mathbb{R}^m$, and $p \leq \min(m,n)$, then the sum

$$\sum_{i=1}^p u_i v_i^\top \in \mathbb{R}^{m \times n}$$

is a matrix of rank $p$ at most. So, our approximation $X$ will have to be defined as a sum of outer products. It happens that such a sum is exactly what we get if we look at the SVD,
because by the definition of matrix multiplication (we assume here that $A$ is full rank):

$$A = U \Sigma V^T = \sum_{i=1}^{\min(m,n)} \sigma_i u_i v_i^T,$$

so if $\Sigma$ will have only $p$ non-zero values, this sum will be of $p$ outer-products.

Next, we will recall that just like for the $\ell_2$ norm, we have that for every matrix $X$ and orthogonal matrix $V$

$$\|VX\|_F^2 = \text{trace}(X^TV^TVX) = \text{trace}(X^TX) = \|X\|_F^2,$$

and this will hold for every matrix $V$ satisfying $V^TV = I$.

Back to our problem of approximating $A$. Let’s that $A = U \Sigma_A V^T$ is the full SVD decomposition where $U$ and $V$ are square. Then, for example, $\|UX\|_F = \|U^TX\|_F = \|X\|_F$. Following this:

$$\|A - X\|_F = \|U \Sigma_A V^T - X\|_F = \|\Sigma_A - U^TXV\|_F.$$

So, it is clear that $U^TXV$ should be diagonal with entries as close as possible to $\Sigma_A$. This will be achieved is we set $X = U \Sigma_X V^T$, and then we will get:

$$\|A - X\|_F = \|\Sigma_A - \Sigma_X\|_F.$$

Both $\Sigma_A, \Sigma_X$ are diagonal. Which $p$ diagonal entries will we choose to be non-zeros to best approximate $A$? The largest ones of course!
Example 18 (Low rank approximations via the SVD). First, we will generate a random matrix $A$ (that fits our purposes) and form its SVD decomposition.

```
 julia> B = rand(5,3)*diagm([1;0.1;0.001])*rand(3,3)
 5x3 Array{Float64,2}:
 0.610097  0.371672  0.437624
 0.270363  0.163615  0.192502
 0.710042  0.425165  0.500611
 0.472682  0.31145  0.366526
 0.0786736 0.0559478  0.0653355

 julia> (U,s,V) = svd(B);

 julia> U
 5x3 Array{Float64,2}:
 -0.559522  0.135717  -0.10364
 -0.247125  0.106349   0.28395
 -0.645929  0.451951   0.0593005
 -0.450122  -0.825446  -0.269576
 -0.0776604 -0.290945  0.912383

 julia> s
 3-element Array{Float64,1}:
 1.4973  
 0.0281634  
 0.000312524  

 julia> V
 3x3 Array{Float64,2}:
 -0.725098  0.68863  0.00468583  
 -0.445839  -0.474613  0.758927  
 -0.524844  -0.548207  -0.651159  

 julia> vecnorm(B - U*diagm(s)*V') # vecnorm is one way to get the Frobenius norm in Julia.
 9.7223783957695e-16
```

Now we take a few largest elements of $\Sigma$ and form a low-rank approximation of $A$ by multiplying it with $U$ and $V$. As we take more singular values, the approximation improves (in Frobenius norm).
# recall B:
```
5x3 Array{Float64,2}:
0.610097 0.371672 0.437624
0.270363 0.163615 0.192502
0.710042 0.425165 0.500611
0.472682 0.31145  0.366526
0.0786736 0.0559478 0.0653355
```
# here we take only the largest element of Sigma (form a rank-1 approximation).
```
julia> s1 = copy(s); s1[2:end] = 0.0;
```
```
5x3 Array{Float64,2}:
0.607465 0.37351  0.439698
0.2683 0.164969 0.194202
0.701276 0.431192 0.507601
0.488691 0.30048  0.353727
0.0843149 0.0518424 0.0610292
```
# here we take the two largest elements of Sigma (form a rank-2 approximation).
```
julia> s2 = copy(s); s2[end] = 0.0;
```
```
5x3 Array{Float64,2}:
0.610097 0.371696 0.437603
0.270363 0.164969 0.19256
0.710041 0.42515  0.500623
0.472683 0.311514 0.366471
0.0786723 0.0557314 0.0655212
```
# here we compare the norms of the two approximations:

# the rank-one approximation
```
julia> vecnorm(B - U*diagm(s1)*V')/vecnorm(A)
0.0188073
```

# the rank-two approximation
```
julia> vecnorm(B - U*diagm(s2)*V')/vecnorm(A)
0.000208688134648313
```
Example 19 (Low rank approximations of images). The following matlab code demonstrates low-rank approximation of images:

```matlab
%Im = imread('cameraman.tif');
Im = imread('corn.tif',3);
Im = double(Im)/255;
[U,S,V] = svd(Im,'econ');
pmax = min(size(Im));
figure()
S_I = diag(S);
plot(S_I,'LineWidth',2)
pause
S_X = zeros(pmax,1);

for k = [1,5,10,15,20,30,40,50,60,70,80,90,100]
    S_X(1:k) = S_I(1:k);
    X = U*diag(S_X)*V';
imshow(X);
title([num2str(k),'-rank approximation']);
pause(2);
end
# See more in: https://www.youtube.com/watch?v=_lY74pXW1S8
```