Numerical Linear Algebra A1 Recap

26/04/2025

The lectures below refer to Trefethen's book on numerical linear algebra

1. Lecture 3 - Norms

Diclaimer: The norm's chapter has pretty abstract concepts, some of them are not very **intutive**, so try to abstract and accept they exist for now, later we will show that they are very useful.

1.1. Vector norms

Definition 1.1.1 (Norm): A **norm** is a function $\|\cdot\| : \mathbb{C}^m \to \mathbb{R}$ that satisfies 3 properties:

1. $\|x\| \geq 0$, and $\|x\| = 0 \Leftrightarrow x = 0$
2. $\|x + y\| \leq \|x\| + \|y\|$
3. $\|\alpha x\| = |\alpha| \|x\|$

We normally see the 2-norm, or the **Euclidian Norm**, that represents the **size** of a vector. Based on that, we can define a **p-norm**.

Definition 1.1.2 (p-norm): The **p-norm** of $x \in \mathbb{C}^n$ (or $\|x\|_p$) is defined as:

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{\frac{1}{p}}$$

So we can have multiple types of norms, from 1 to ∞, and we define it as well!

Definition 1.1.3 (Inifinite-norm): The **infinite norm** of $x \in \mathbb{C}^n$ (or $\|x\|_\infty$) is defined as:

$$\|x\|_\infty = \max |x_i|$$

You probably wondering "why would I need something like this"? But trust me, it will be useful in the future! There is a pretty useful type of norm (According to the book), that is called the **weighted norm**.

Definition 1.1.4 (Weighted norm): The **weighted norm** of $x \in \mathbb{C}^n$ is:

$$\|x\|_W = \|Wx\| = \left(\sum_{i=1}^n |w_{ii} x_i|^p\right)^{\frac{1}{p}}$$

Where W is a **diagonal matrix** and p is an arbitrary number

1.2. Matrix Norms

WHAT?? MATRICES HAVE NORMS???? Yes, my young Padawan! The book tells that we could see a matrix as a vector in a $m \times n$ space, and we could use any mn-norm to measure it, but some norms are more useful than the ones already discussed.

Definition 1.2.1 (Induced norm): Given $A \in \mathbb{C}^{m \times n}$, the induced norm $\|A\|_{m \to n}$ is the smallest integer for wich the inequality holds:

$$\|Ax\|_m \leq C \|x\|_n$$

In other words:

$$\|A\|_{m \to n} = \sup_{x \neq 0} \frac{\|Ax\|_m}{\|x\|_n}$$

This definition may seem stupid and useless by now, but it will be very useful when we see about errors and conditioning.

A useful norm we may say is the ∞-norm of a Matrix

Definition 1.2.2 (Infinite norm of a Matrix): Given $A \in \mathbb{C}^{m\times n}$, if a_j is the j^{th} row of A, $\|A\|_\infty$ is defined by:

$$\|A\|_\infty = \max_{1\le i\le m}\|a_i\|_1$$

1.3. Cauchy-Schwarz & Hölder Inequalities

When we are using norms, usually it is difficult to compute p-norms with high values of p, so we manage them using inequalities! A very useful inequality is the Hölder inequality:

Definition 1.3.1 (Hölder Inequality): Given $1 \le p, q \le \infty$, and $\frac{1}{p}+\frac{1}{q}=1$, then, for any vectors x, y:

$$|x^* y| \le \|x\|_p \|y\|_q$$

and the Cauchy-Schwarz inequality is a special case where $p = q = 2$

1.4. Bounding $\|AB\|$

We can bound $\|AB\|$ as we do with vector norms

Theorem 1.4.1: Given $A \in \mathbb{C}^{l\times m}$,$B \in \mathbb{C}^{m\times n}$ and $x \in \mathbb{C}^n$, then the induced norm of AB must satisfy:

$$\|AB\|_{l\to n} \le \|A\|_{l\to m}\|B\|_{m\to n}$$

Proof: $\|ABx\|_l \le \|A\|_{l\to m}\ \|Bx\|_m \le \|A\|_{l\to m}\ \|B\|_{m\to n}\ \|x\|_n$ □

1.5. Generalization of Matrices Norms

We saw that a norm follows 3 properties, we define a general matrix norm the same way!!

Definition 1.5.1: Given matrices A and B, a norm $\|\cdot\| : \mathbb{C}^{m\times n} \to \mathbb{R}^+$ is a function that follow these 3 properties:

1. $\|A\| \ge 0$, and $\|A\| = 0 \Leftrightarrow A = 0$
2. $\|A + A\| \le \|A\| + \|B\|$
3. $\|\alpha A\| = |\alpha|\|A\|$

the most important one is the **Frobenius Norm**, defined as:

Definition 1.5.2: Given a matrix $A \in \mathbb{C}^{m\times n}$, its **Frobenius Norm** is defined as:

$$\|A\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2\right)^{\frac{1}{2}} = \sqrt{tr(A^* A)} = \sqrt{tr(AA^*)}$$

Theorem 1.5.1: $\|AB\|_F \le \|A\|_F\ \|B\|_F$

Proof: $\|AB\|_F = \left(\sum_{i=1}^m \sum_{j=1}^n |c_{ij}|^2\right)^{\frac{1}{2}} \le \left(\sum_{i=1}^m \sum_{j=1}^n \left(\|a_i\|_2 \|b_j\|_2\right)^2\right)^{\frac{1}{2}} = \left(\sum_{i=1}^m \|a_i\|_2^2 \sum_{j=1}^n \|b_j\|_2^2\right)^{\frac{1}{2}} = \|A\|_F\|B\|_F$ □

2. Lecture 4 and 5 - The SVD

Quick Disclaimer: When we start talking about the factorization itself, we are going to talk about matrices in $\mathbb{C}^{m \times n}$ with $m \geq n$, because it's the most common when we talk about real problems, rarely is the situations with more variables then equations

Aaaaaah, the SVD, why it exists? What does it mean? Remember that, in Linear Algebra, when we have a base of a Vector Space and a Linear Transformation, we know how the Linear Transformation affects **every** vector on that Vector Space? No? Let me refresh your memmory:

Theorem 2.1: Given $\{a_j\}$ $(1 \leq j \leq n)$ being the base of a Vector Space and T a Linear Transformation in that space, we know how T affects **every** vector on that space

Proof: Being v a vector on the described Vector Space, we know v can be expressed as

$$v = \alpha_1 a_1 + \ldots + \alpha_n a_n$$

Applying T on v

$$T(v) = T(\alpha_1 a_1 + \ldots + \alpha_n a_n) \Rightarrow T(v) = \alpha_1 T(a_1) + \ldots + \alpha_n T(a_n)$$

This implies that, if we know a base of the Vectorial Space, we know how T affects every vector on the space

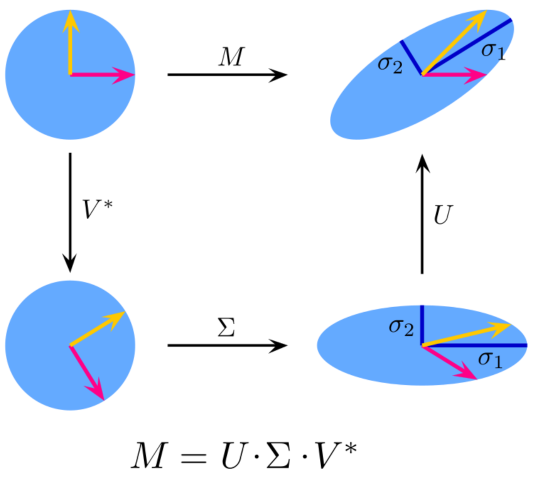
□

RIGHT! Memory refreshed, why did I say that? Remember that matrices are linear transformations? So if we have a base $\{s_j\}$ of a Vectorial Space S, we can know what happens to every linear combination of $\{s_j\}$ if we apply A on it right? Right!

We can resume the operations we do in vectors in two: **stretch**, and **rotate**, so basically, when we apply a linear transformation on a vector, we are rotating it, then stretching it.

Okay, but why am I saying that? Where the hell is the S.V.D? Well, I basically already described the S.V.D to you! When we apply A as a linear transformation, if we do the operations described early, do you agree we can decompose A as a matrix product of Orthogonal Matrices and Diagonal Matrices? What? Why? When? Wait, young padawan! Remember I said a linear transformation can be resumed in stretching and rotating vectors? Do you remember what kind of matrices do EXACTLY what I said? Yes, orthogonal matrices do rotations and diagonal matrices do stretching.

Now we can introduce that classic visualization of how S.V.D works, imagine a orthonormal base in $\mathbb{R}^2$, looks what happens if we apply A on it:



(Change the M in the image for A)

Based on that, we can define that, given $A \in \mathbb{C}^{m \times n}$:

$$Av_j = \sigma_j u_j$$

Where v_j and u_j are from two different orthonormal bases and $\sigma_j \in \mathbb{C}$.

2.1. Reduced Form

We can rewrite this equation as a matrix product!

$AV = \hat{U}\hat{\Sigma}$

Where

$$V = \begin{pmatrix} | & & | \\ v_1 & \dots & v_n \\ | & & | \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix}, U = \begin{pmatrix} | & & | \\ u_1 & \dots & u_n \\ | & & | \end{pmatrix}$$

This is known as the **reduced** SVD factorization. We can see that V is a square orthogonal matrix (For Av_j be a valid multiplication, $v_j \in \mathbb{C}^n$), so we can rewrite A as:

$$A = \hat{U}\hat{\Sigma}V^*$$

2.2. Full SVD

Okay, if $v_j \in \mathbb{C}^n$ and $Av_j = \sigma_j u_j$, then $u_j \in \mathbb{C}^m$! That means, beside the u vectors we added in $\hat{U}$, we have $m - n$ more orthonormal vectors to the columns of $\hat{U}$, finding those vectors, we can build another matrix U that the columns are a orthonormal base of $\mathbb{C}^m$, that means the new matrix U is orthogonal!

$$V = \begin{pmatrix} | & & | \\ v_1 & \dots & v_n \\ | & & | \end{pmatrix}, U = \begin{pmatrix} | & & | \\ u_1 & \dots & u_m \\ | & & | \end{pmatrix}$$

Nice! But what about the $\hat{\Sigma}$ matrix? How does it change? Well, we want to maintain V and U as we wanted right? Well, the thing we did was add columns to $\hat{U}$, so, in the multiplication, we only need those columns to disappear, how we do that? Multiplying by 0! So, before, $\hat{\Sigma}$ was a square matrix with the single values on the diagonal, specifically, n single values. If we added $m - n$ vectors on U, we can add $m - n$ zeros on $\hat{\Sigma}$, so our new matrix multiplication is

$$A = U\Sigma V^*$$

$$\begin{pmatrix} | & & | \\ a_1 & \dots & a_n \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ u_1 & \dots & u_m \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_n & \\ & & & 0 \\ & & & \ddots \end{pmatrix} \begin{pmatrix} -v_1- \\ \dots \\ -v_n- \end{pmatrix}$$

2.3. Formal Definition

Definition 2.3.1: Given $A \in \mathbb{C}^{m \times n}$ with $m \geq n$, the Singular Value Decomposition of A is:

$$A = U\Sigma V^*$$

where $U \in \mathbb{C}^{m \times m}$ is unitary, $V \in \mathbb{C}^{n \times n}$ is unitary and $\Sigma \in \mathbb{C}^{m \times n}$ is diagonal. For **convinience**, we denote:

$$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots \geq \sigma_n$$

Where σ_j is the j-th entry of Σ

Okay, we saw a intuitive method for seeing that every matrix has this decomposition, but how do we prove it mathematically?

Theorem 2.3.1: Every $A \in \mathbb{C}^{m \times n}$ matrix has a S.V.D decomposition

Proof: Let $\{v_j\}$ be an orthonormal base of $\mathbb{C}^n$, $\{u_j\}$ an orthonormal base of $\mathbb{C}^m$, $Av_j = \sigma_j u_j$, U_1 and V_1 be unitary matrices of columns $\{u_j\}$ and $\{v_j\}$ respectively and that, for every matrix with less than m lines and n columns the factorization is valid:

$$A = U_1 S V_1^* \Leftrightarrow U_1^* A V_1 = S$$

Então temos $S = \begin{pmatrix} \sigma_1 & w^* \\ 0 & B \end{pmatrix}$ onde σ_1 é 1×1, w^* é $1 \times (n-1)$ e B é $(m-1) \times (n-1)$. Beleza, mas o que é w? Bem, podemos chegar nesse resultado fazendo umas manipulações com $\left\| \begin{pmatrix} \sigma_1 & w^* \\ 0 & B \end{pmatrix} \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2$:

$$\left\| \begin{pmatrix} \sigma_1 & w^* \\ 0 & B \end{pmatrix} \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2^2 \geq \sigma_1^2 + w^* w$$

What? Why this is valid? Because:

$$\|Mx\|_2 \leq \|M\|_2 \, \|x\|_2 \Rightarrow \|M\|_2 \geq \frac{\|Mx\|_2}{\|x\|_2}$$

If we set $x = \begin{pmatrix} \sigma_1 \\ w \end{pmatrix}$ and $M = S$, then we have:

$$Mx = \begin{pmatrix} \sigma_1^2 + \|w\|^2 \\ Bw \end{pmatrix} \Rightarrow \|M\|_2 \geq \frac{|\sigma_1^2 + \|w\|^2|^2 + \|Bw\|^2}{\sigma_1^2 + \|w\|^2}$$

But notice that the numerator is always greater than the denominator, so that means

$$\frac{|\sigma_1^2 + \|w\|^2|^2 + \|Bw\|^2}{\sigma_1^2 + \|w\|^2} \geq \sigma_1^2 + \|w\|^2 = (\sigma_1^2 + w^* w)^{\frac{1}{2}} \left\| \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|$$

Now we can go back to see what w is! Well, now is easy! We know that $\|S\|_2 = \|U_1^* A V_1\|_2 = \|A\|_2 = \sigma_1$ because U_1 and V_1 are orthogonal. That means $\|S\|_2 \geq (\sigma_1^2 + \|w\|^2)^{\frac{1}{2}} \Rightarrow \sigma_1 \geq (\sigma_1^2 + \|w\|^2)^{\frac{1}{2}} \Leftrightarrow \sigma_1^2 \geq \sigma_1^2 + \|w\|^2 \Rightarrow w = 0$.

By the inductive hypothesis described in the start of the proof, we know $B = U_2 \Sigma_2 V_2^*$, so we can easily write A as

$$A = U_1 \begin{pmatrix} 1 & 0 \\ 0 & U_2 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & V_2^* \end{pmatrix}^* V_1^*$$

That is a S.V.D of A, using the base case of $m = 1$ and $n = 1$, we finish the existence's proof □

2.4. Change of Basis

Given $b \in \mathbb{C}^m$, $x \in \mathbb{C}^n$ and $A \in \mathbb{C}^{m \times n}$, $A = U\Sigma V^*$ we can get the coordinates of b on the basis of the columns of U and x in the columns of V. Just to remember:

Definition 2.4.1: Given $w \in V$ where V is a Vector Space, $\exists! x_1, ..., x_n \in \mathbb{C}$ such that $w = v_1 x_1 + ... + v_n x_n$ where $\{v_j\}$ is a base of V. The vector $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$, also denoted as $[w]_v$, is the w's **coordinate vector** in the v base

Getting back, we can express $[b]_u = U^* b$ and $[x]_v = V^* x$, but why?

Theorem 2.4.1: Given a orthonormal base $\{v_k\}$ of V and $w \in V$, then

$$([w]_v)_j = v_j^* w$$

Proof:

$$w = \alpha_1 v_1 + ... + \alpha_n v_n$$

$$v_j^* w = \alpha_1 v_j^* v_1 + ... + \alpha_n v_j^* v_n$$

Knowing that $\{v_j\}$ is a orthonormal base, the product $\alpha_i v_j^* v_i$ is equal to 0 if $j \neq i$ and equal to α_i if $j = i$, that is:

$$v_j^* w = \alpha_j$$

□

Ok, now that we remembered all these properties, we can express the relation $b = Ax$ in terms of $[b]_u$ and $[x]_v$, let's see:

$$b = Ax \Leftrightarrow U^*b = U^*Ax = U^*U\Sigma V^*x \Leftrightarrow U^*b = \Sigma V^*x$$
$$\Leftrightarrow [b]_u = \Sigma[x]_v$$

So we can reduce A to the Σ matrix and b and x to its coordinates on u base and v base

2.5. S.V.D vs Eigenvalue Decomposition

We can do something similar with the eigenvalue decomposition. Given $A \in \mathbb{C}^{m\times m}$ with linear independent eigenvectors, i.e we can express $A = S\Lambda S^{-1}$ with the columns of S being the eigenvectors of A and Λ is a diagonal matrix with the eigenvalues of A as entries.

Defining $b, x \in \mathbb{C}^m$ satisfying $b = Ax$, we can write:

$$[b]_{s^{-1}} = S^{-1}b \text{ and } [x]_{s^{-1}} = S^{-1}x$$

Where I'm denoting s^{-1} as the base expressed by the columns of S^{-1}, then the new expanded expression is:

$$b = Ax \Leftrightarrow S^{-1}b = S^{-1}Ax = S^{-1}S\Lambda S^{-1}x \Leftrightarrow S^{-1}b = \Lambda S^{-1}x$$
$$[b]_{s^{-1}} = \Lambda[x]_{s^{-1}}$$

2.6. Matrix Properties with SVD

For the next properties, let $A \in \mathbb{C}^{m\times n}$ and $r \leq \min(m, n)$ be the number of non-zero singular values

Theorem 2.6.1: $\operatorname{rank}(A) = r$

Proof: The rank of a diagonal matrix is the number of non-zero entries, well, if $A = U\Sigma V^*$, we know U and V are full-rank, then the rank of A must be the same as Σ, that is, r □

Theorem 2.6.2: $\operatorname{range}(A) = \operatorname{span}\{u_1, ..., u_r\}$, $\operatorname{range}(A^*) = \operatorname{span}\{v_1, ...v_r\}$, $\operatorname{null}(A) = \operatorname{span}\{v_{r+1}, ..., v_n\}$, $\operatorname{null}(A^*) = \operatorname{span}\{u_{r+1}, ..., u_m\}$

Proof: Let's remember how each matrix is structured:

$$A = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_m \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & 0 \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} -v_1^*- \\ \vdots \\ -v_n^*- \end{pmatrix}$$

It's easy to see why $\operatorname{range}(A) = \operatorname{span}\{u_1, ..., u_r\}$, because the entries of Σ makes only possible to span the first r columns of U.

About $\operatorname{null}(A) = \operatorname{span}\{v_{r+1}, ..., v_n\}$, notice how, if we do Av_j $r + 1 \leq j \leq n$, the first r lines will turn to 0 (All v_k are orthonormal between them) and, because the diagonal entries after the r-th one are 0, then we have U times the 0 matrix

To see the A^*'s properties, let's transpose A

$$A^* = \begin{pmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & 0 \\ & & & & \ddots \end{pmatrix} \begin{pmatrix} -u_1^*- \\ \vdots \\ -u_m^*- \end{pmatrix}$$

So, again, is easy to see $\operatorname{range}(A^*) = \operatorname{span}\{v_1, ...v_r\}$ and, using the same argument shown before, null $(A^*) = \operatorname{span}\{u_{r+1}, ..., u_m\}$ □

Theorem 2.6.3: $\|A\|_2 = \sigma_1$ and $\|A\|_F = \sqrt{\sigma_1^2 + \ldots + \sigma_r^2}$

Proof:
1. $\|A\|_2 = \|U\Sigma V\|_2 = \|\Sigma\|_2$, as we denoted before, from all entries, σ_1 is the greatest, that means $\|A\|_2 = \|\Sigma\|_2 = \sigma_1$
2. We know that $\|A\|_F = \sqrt{\operatorname{tr}(A^* A)} = \sqrt{\operatorname{tr}(V\Sigma^* U^* U\Sigma V^*)} = \sqrt{\operatorname{tr}(V\Sigma^*\Sigma V^*)}$. We also know that $\operatorname{tr}(A) = \lambda_1 + \ldots + \lambda_n$ with λ_j being the eigenvalues of A, and w can clearly see that the eigenvalues of $V\Sigma^*\Sigma V^*$ are σ_j^2, therefore $\|A\|_F = \sqrt{\sigma_1^2 + \ldots + \sigma_r^2}$

□

Theorem 2.6.4: $\sigma_j = \sqrt{\lambda_j}$ with σ_j being the singular values of A and λ_j the eigenvalues of $A^* A$

Proof: $A^* A = V\Sigma^* U^* U\Sigma V^* = V\Sigma^*\Sigma V^*$ □

Theorem 2.6.5: if $A = A^*$, then the singular values of A are the absolute values of A's eigenvalues

Proof: By the Spectral Theorem, we know A has an eigenvalue decomposition

$$A = Q\Lambda Q^*$$

We can rewrite it as

$$A = Q|\Lambda|\operatorname{sign}(\Lambda)Q^*$$

Where the entries of $|\Lambda|$ is $|\lambda_j|$ and the entries of $\operatorname{sign}(\Lambda)$ are $\operatorname{sign}(\lambda_j)$. We can show that, if Q is unitary, $\operatorname{sign}(\Lambda)Q$ is unitary, that means $Q|\Lambda|\operatorname{sign}(\Lambda)Q^*$ is a SVD of A □

Theorem 2.6.6: For $A \in \mathbb{C}^{m\times m}$, $|\det(A)| = \prod_{i=1}^{m} \sigma_i$

Proof: $|\det(A)| = |\det(U\Sigma V^*)| = |\det(U)\det(\Sigma)\det(V)| = |\det(\Sigma)|$ □

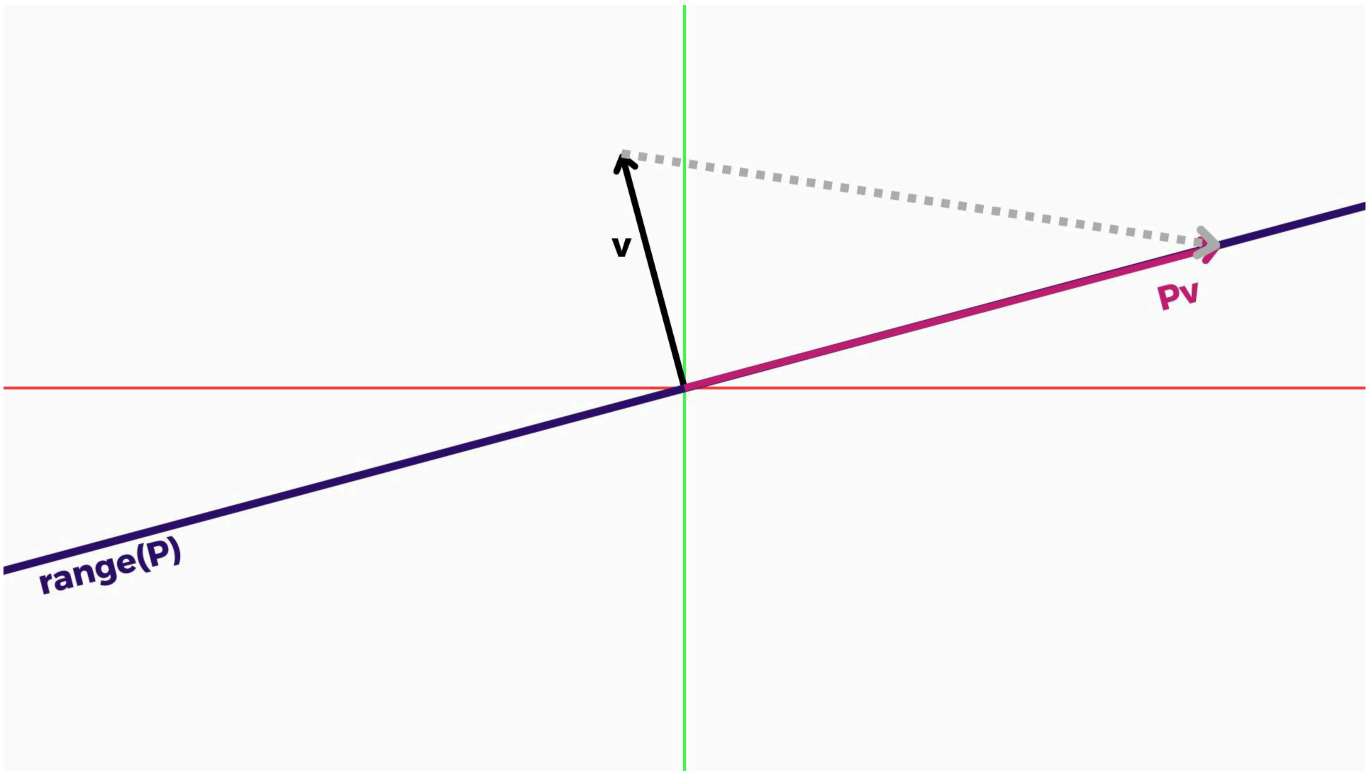
3. Lecture 6 - Projectors

$P \in C^{m\times n}$ is said to be a **Projector** if

$$P^2 = P$$

also called *idempotent*. You might confuse thinking only on orthogonal projections, that ones we get the vector and project it in a way to make a 90 degree angle in the projected space! But we are talking about **EVERY** projection, including non-orthogonal ones

Imagine we put a light on that vector, it will give a shadow somewhere, but you agree with me we can get that shadow somehow right? Let's see a 2D example



As you can see, the dashed vector tells us the direction where the light is projecting the shadow of v onto P. We can express this direction as $Pv - v$. Is important to remember that, if you're laying on the ground, you won't have a shadow right? Or even better, shadows doesn't have shadows! Translating this on our contexts:

Theorem 3.1: If $v \in \text{range}(P)$, then $Pv = v$

Proof: Every $v \in \text{range}(P)$ can be expressed as $v = Px$ for some x, that means $Pv = P^2x = Px = v$ □

Notice that, if we apply the projection onto the direction we had early

$$P(Pv - v) = P^2v - Pv = Pv - Pv = 0$$

That means $Pv - v \in \text{null}(P)$. Also notice we can rewrite the direction as

$$Pv - v = (P - I)v = -(I - P)v$$

Look what's even stranger!

$$(I - P)^2 = I - 2P + P^2 = I - P$$

That mens $I - P$ is also a projector! A projector that projects into the direction of projection of P

3.1. Complementary Projectors

If P is a projector, $I - P$ is its complementary projector

Theorem 3.1.1: $I - P$ projects onto $\text{null}(P)$ and P projects onto $\text{null}(I - P)$

Proof:

1. $\text{range}(I - P) \subseteq \text{null}(P)$ because $v - Pv \in \text{null}(P)$ and $\text{range}(I - P) \supseteq \text{null}(P)$ because, if $Pv = 0$, we can rewrite it as $(I - P)v = v$, that means $\text{null}(P) = \text{range}(I - P)$
2. If we rewrite the expression as $P = I - (I - P)$, then, using the same argument as before, we have $\text{range}(P) = \text{null}(I - P)$

□

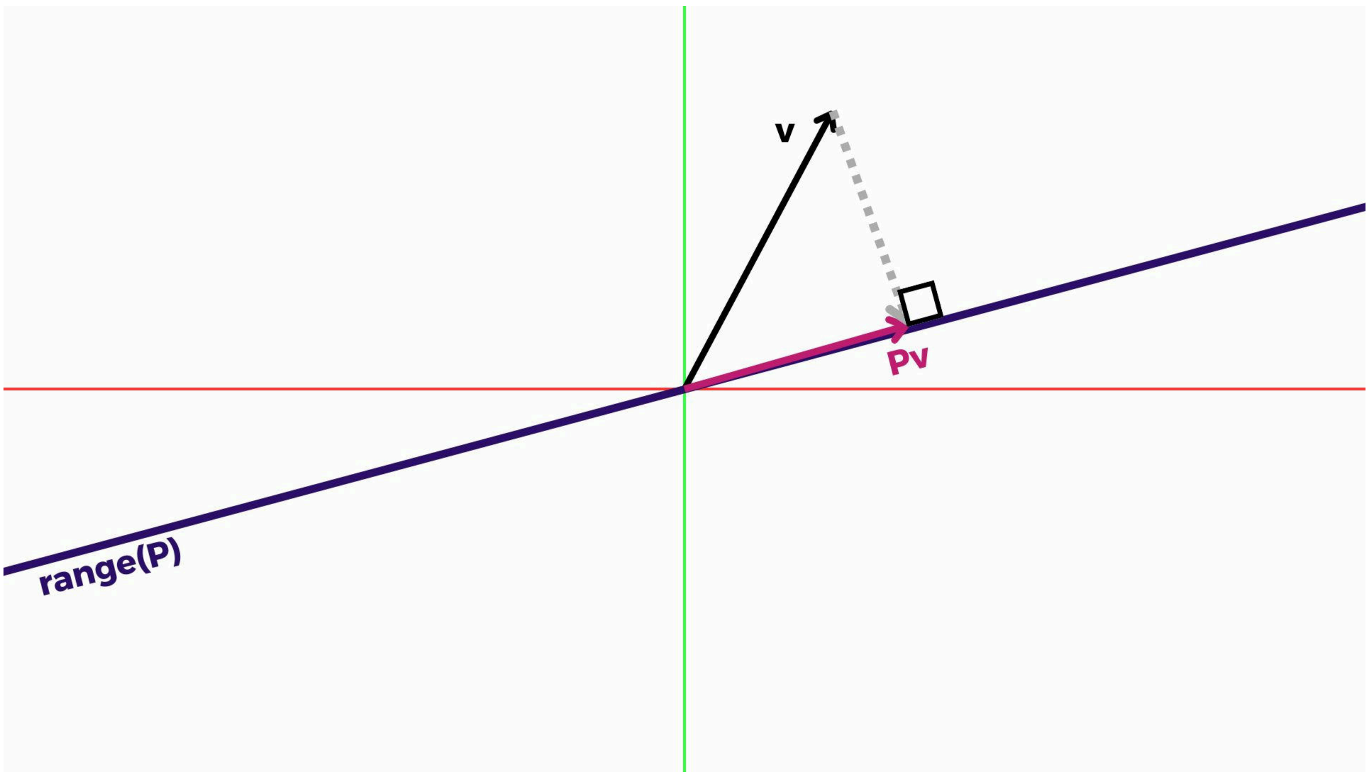
Theorem 3.1.2: $\text{null}(I - P) \cap \text{null}(P) = \{0\}$

Proof: $\text{null}(A) \cap \text{range}(A) = \{0\} \Rightarrow \text{null}(P) \cap \text{range}(P) = \{0\} \Leftrightarrow \text{null}(P) \cap \text{null}(I - P) = \{0\}$ □

That means, if we have a projector P in $\mathbb{C}^{m \times m}$, this projector separates $\mathbb{C}^m$ in to spaces $S_1 \land S_2$, in a way that $S_1 \cap S_2 = \{0\}$ and $S_1 + S_2 = \mathbb{C}^m$

3.2. Orthogonal Projectors

Finally! The projectors we hear all the time! They project a vector in a space making the direction form a 90 degree with the projection



That means $(Pv)^*(v - Pv) = 0$

Theorem 3.2.1: P is a orthogonal projector $\Leftrightarrow P = P^*$

Proof:

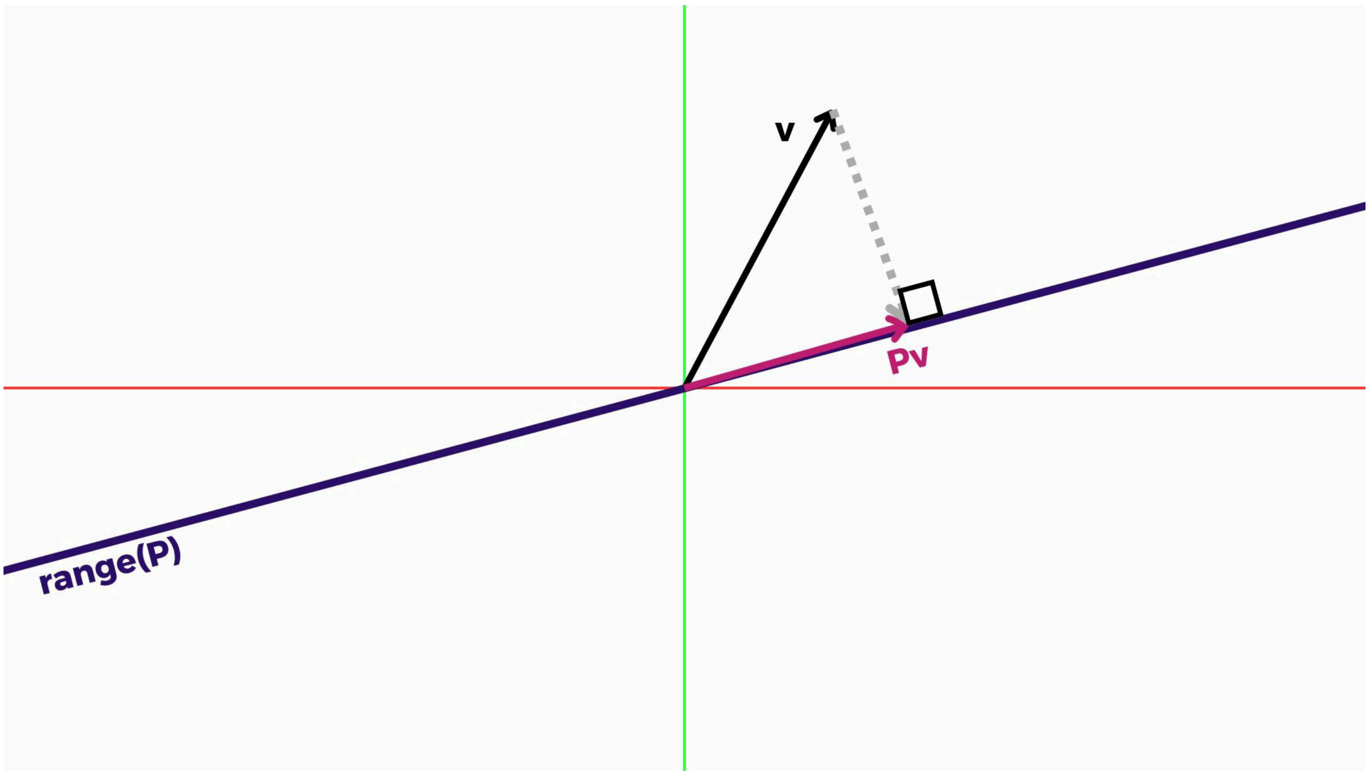
1. $\Leftarrow$) Given $x, y \in \mathbb{C}^m$, then $x^* P^* (I - P) y = x^* (P^* - P^* P) y = x^* (P - P^2) y = x^* (P - P) y = 0$
2. $\Rightarrow$) Let $\{q_1, …, q_m\}$ be a orthonormal base of $\mathbb{C}^m$ where $\{q_1, …, q_n\}$ is base of S_1 and $\{q_{n+1}, …, q_m\}$ is base of S_2. For $j \leq n$ we have $Pq_j = q_j$ and for $j > n$ we have $Pq_j = 0$, let Q be the matrix with columns $\{q_1, …, q_m\}$ we have: $Q = \begin{pmatrix} | & & | \\ q_1 & \dots & q_m \\ | & & | \end{pmatrix} \Leftrightarrow PQ = \begin{pmatrix} | & & | & | & \\ q_1 & \dots & q_n & 0 & \dots \\ | & & | & | & \end{pmatrix} \Leftrightarrow Q^* PQ = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & & 0 & \\ & & & & & \ddots \end{pmatrix}$, wich means we found a SVD decomposition for P:

$$P = Q\Sigma Q^* \Leftrightarrow P^* = Q\Sigma^* Q^* = Q\Sigma Q^* = P$$

□

3.3. Orthogonal projection onto a vector

Let's use the same example used previously



Let q be the vector that spans P, we know that $Pv = \alpha q$

$$(v - Pv)^* q = 0 = (v - \alpha q)^* q = 0$$

Now we look for the α that makes this equation valid

$$v^* q - \alpha q^* q = 0 \Leftrightarrow v^* q = \alpha q^* q \Leftrightarrow \alpha = \frac{v^* q}{q^* q}$$

$$Pv = \alpha q \Leftrightarrow Pv = \frac{v^* q}{q^* q} q \Leftrightarrow Pv = \frac{q^* v}{q^* q} q \Leftrightarrow Pv = q \frac{q^* v}{q^* q} \Leftrightarrow Pv = \frac{q q^*}{q^* q} v \Rightarrow P = \frac{q q^*}{q^* q}$$

3.4. Projection with orthonormal basis

We saw in Theorem 3.2.1's proof that some singular values of P are 0, so we could remove those lines of Σ and reduce it to I, also removing the columns and lines of Q, getting:

$$P = \hat{Q} \hat{Q}^*$$

Let $\{q_1, …, q_n\}$ be any set of orthonormal vectors in $\mathbb{C}^m$ and let them be the columns of $\hat{Q}$, we know that, for any vector $v \in \mathbb{C}^m$:

$$v = r + \sum_{i=1}^{n} q_i q_i^* v$$

What? when we saw that? Calm down, let me recap for you:

Theorem 3.4.1: Let $\{q_1, ..., q_n\}$ be any set of orthonormal vectors in $\mathbb{C}^m$, then any $v \in \mathbb{C}^m$ can be expressed as

$$v = r + \sum_{i=1}^{n} q_i q_i^* v$$

With r being another vector in C^m orthogonal to $\{q_1, ..., q_n\}$ and, $\rightarrow n = m \Rightarrow r = 0$ and the set of vectors chosen is a base for $\mathbb{C}^m$

Proof: You know that, given a base of $\mathbb{C}^m$, any vector can be expressed as a linear combination of those vectors. Imagine the cannon base (With some rotations, this logic can be expanded for other orthonormal bases), you can imagine that, if you project the vector you have onto any vector of the cannon base, you'll obtain a vector that, if you sum with another vector r, you'll obtain your original vector again! And we can continue this process until we do it with n vectors of the canon base, obtaining the original r that, if we sum all our projections, we get the original vector again, that is:

$$v = r + \sum_{i=1}^{n} q_i q_i^* v$$

□

Ok, knowing a vector can be expressed like this, we can see that the sum part is the same as doing:

$$\hat{Q}\hat{Q}^* v$$

That is, $\sum_{i=1}^{n} q_i q_i^* v$ is a projector onto $\text{range}\left(\hat{Q}\right)$

Theorem 3.4.2: The complement of an orthogonal projector is also an orthogonal projector

Proof:
1. $\left(I - \hat{Q}\hat{Q}^*\right)^2 = I - 2\hat{Q}\hat{Q}^* + \left(\hat{Q}\hat{Q}^*\right)^2 = I - 2\hat{Q}\hat{Q}^* + \hat{Q}\hat{Q}^* = I - \hat{Q}\hat{Q}^*$
2. $\left(I - \hat{Q}\hat{Q}^*\right)^* = I - \left(\hat{Q}\hat{Q}^*\right)^* = I - \hat{Q}\hat{Q}^*$

□

A special case is the rank-one orthogonal projector, this gets the vector and get the component of a single direction q, wich can be written:

$$P_q = qq^*$$

And its complement is the $(m - 1)$ rank matrix

$$P_{\perp q} = I - qq^*$$

This concept is valid for non-unitary vectors too:

$$P_a = \frac{aa^*}{a^* a}$$

$$P_{\perp a} = I - \frac{aa^*}{a^* a}$$

Just to clear the things. If we project a vector v onto a vector a, we're restraining v in the direction of the projection, so, if we project onto the complement of a, is like, we can express v as a linear combination of a and some other vectors, and then remove the part of a in this linear combination, having only the other vectors expressing a new vector

3.5. Projection on arbitrary basis

Given an arbitrary base $\{a_j\}$, we let the vectors of this base be the columns of A. Given v with $Pv = y \in$ range (A), that means $y - v \perp$ range(A), that means $a_j^*(y - v) = 0\ \forall j$. We know $y \in$ range(A), so let's write it as $Ax = y$, then we can rewrite $a_j^*(y - v) = 0\ \forall j$ as:

$$A^*(Ax - v) = 0 \Leftrightarrow A^*Ax - A^*v = 0 \Leftrightarrow A^*Ax = A^*v \Leftrightarrow x = (A^*A)^{-1}A^*v$$

$$Ax = A(A^*A)^{-1}A^*v \Leftrightarrow y = A(A^*A)^{-1}A^*v$$

$$\Rightarrow P = A(A^*A)^{-1}A^*$$

4. Lecture 7- QR Factorization

The scary one, the part where anyone knows anything! Let's calm down and see everything with patiently.

How does this factorization works? We want to express A as:

$$A = QR$$

With Q being an orthogonal matrix and R an upper triangular matrix. But why would I want to do such a thing? The main reason is resolving linear systems! Let's get the system $b = Ax$, we rewrite it as

$$b = QRx \Leftrightarrow Q^*b = Rx \Leftrightarrow c = Rx$$

We have a equivalent system, and this one is a **triangular** system, i.e, a trivial system for us and for a computer to solve!

4.1. The idea of the reduced factorization

Let $\{a_j\}$ denote the columns of $A \in \mathbb{C}^{m\times n},\ m \geq n$. In some applications, we are interested in the columns *spaces* of A, i.e, the sequential spaces spanned by the columns of A:

$$\text{span}\{a_1\} \subseteq \text{span}\{a_1, a_2\} \subseteq \text{span}\{a_1, a_2, a_3\} \subseteq \ldots$$

For now, we'll assume A has full-rank n. First of all, we want to obtain a set of orthonormal vectors with the following property:

$$\text{span}\{a_1, \ldots, a_j\} = \text{span}\{q_1, \ldots, q_j\} \text{ with } j = 1, \ldots, n$$

Well, I think a good idea for doing this, is vectors such that we could express a_j as a linear combination of $\{q_1, \ldots, q_j\}$. So that means:

$$a_1 = r_{11}q_1$$

$$a_2 = r_{12}q_1 + r_{22}q_2$$

$$\vdots$$

$$a_n = r_{1n}q_1 + r_{2n}q_2 + \ldots + r_{nn}q_n$$

We could express this equations as a matrix product!

$$\begin{pmatrix} | & | & & | \\ a_1 & a_2 & \ldots & a_n \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ q_1 & q_2 & \ldots & q_n \\ | & | & & | \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix}$$

So we have $A = \hat{Q}\hat{R}$, where $Q \in \mathbb{C}^{m\times n}$ and $R \in \mathbb{C}^{n\times n}$

4.2. Full QR Factorization

Goes a bit further. We know $\{q_1, \ldots, q_n\}$ is a set of orthonormal vectors of $\mathbb{C}^m$, this means that we have $m - n$ more vectors orthonormal to the ones we had before, so we can create a base for $\mathbb{C}^m$, adding these vectors as columns of $\hat{Q}$, we have a orthogonal matrix Q. But what do we do for A stay the same? We can simply add lines of 0 bellow $\hat{R}$, creating $R \in \mathbb{C}^{m \times n}$ $(m \geq n)$, obtaining

$$A = QR$$

4.3. Gram-Schidt Orthogonalization

Oh no… the scary one… Let's go very calmly. We saw previously a way to calculate all q_j, let's remember it:

$$a_1 = r_{11} q_1$$

$$a_2 = r_{12} q_1 + r_{22} q_2$$

$$\vdots$$

$$a_n = r_{1n} q_1 + r_{2n} q_2 + \dots + r_{nn} q_n$$

Well, this suggests an algorithm for calculating the next q_j, let's think, we have all a_j, and each q_j needs the vectors $\{q_1, \dots, q_{j-1}\}$. Well, we can have some freedom here! Let's see what happens when we try to calculate q_j:

$$a_j = r_{1j} q_1 + \dots + r_{jj} q_j$$

Let's isolate q_j:

$$q_j = \frac{a_j - r_{1j} q_1 - r_{2j} q_2 - \dots - r_{2(j-1)} q_{j-1}}{r_{jj}}$$

Well, that suggest us that r_{jj} is the norm of the vector $a_j - \sum_{k=1}^{j-1} r_{ij} q_i$, but what is r_{ij}? Remember the decomposition on orthogonal factors? Yes, that one, $v = r + \sum_{i=1}^{n} q_i q_i^* v$. If we change v for a_j, we have almost the same thing we defined previously!

$$a_j - \sum_{k=1}^{j-1} r_{ij} q_i, \;\; v - \sum_{i=1}^{k} q_i q_i^* v$$

And you remember that r is orthogonal to span$\{q_1, \dots, q_k\}$? That's exactly what q_j is! All this things I just said, suggest I can define r_{ij} as $q_i^* a_j$ $(i \neq j)$. And our algorithm is done! Let's recap it all here:

$$q_1 = \frac{a_1}{\|a_1\|_2}$$

$$q_2 = \frac{a_2 - q_1 q_1^* a_2}{\|a_2 - q_1 q_1^* a_2\|_2}$$

$$q_3 = \frac{a_3 - q_1 q_1^* a_3 - q_2 q_2^* a_3}{\|a_3 - q_1 q_1^* a_3 - q_2 q_2^* a_3\|_2}$$

$$\vdots$$

$$q_n = \frac{a_n - \sum_{i=1}^{n-1} q_i q_i^* a_n}{\|a_n - \sum_{i=1}^{n-1} q_i q_i^* a_n\|_2}$$

Writing in an algorithm form:

```
for $j = 1$ to $n$
    $v_j = a_j$
    for $i = 1$ to $j - 1$
        $r_{ij} = q_i^* a_j$
        $v_j = v_j - r_{ij} q_i$
    $r_{jj} = \|v_j\|_2$
    $q_j = \frac{v_j}{r_{jj}}$
```

4.4. Existence and Uniqueness

Theorem 4.4.1: Every $A \in \mathbb{C}^{m \times n}, \ (m \geq n)$ has a full QR factorization, hence also a reduced QR factorization

Proof: If $\text{rank}(A) = n$, we can build the reduced factorization using Gram-Schmidt as we did before. The only problem here is if, at some point, $v_j = a_j - \sum_{k=1}^{j-1} q_k q_k^* a_j = 0$ thus cannot be normalized. If this happens, that means A hasn't full-rank, that means I can choose whatever orthogonal vector I want to continue the process. □

Theorem 4.4.2: Each $A \in \mathbb{C}^{m \times n} \ (m \geq n))$ of full rank has a unique reduced QR factorization $A = \hat{Q}\hat{R}$ with $r_{jj} > 0$

Proof: We know that, if A is full rank $\Rightarrow r_{jj} \neq 0$ and thus at each successive step j the formulas shown previously determine r_{ij} and q_j fully, the only problem is the signal of r_{jj}, once we say $r_{jj} > 0$, this problem is solved □

5. Lecture 8 - Gram-Schmidt Orthogonalization

We can describe the Gram-Schmidt algorithm using projectors, but why would we want this? Actually this is an introduction for another algorithm we'll see later. When we talk about algorithms, we want them to be stable, in a sense that if we put an input into the computer, it will return me an answer next to the right one (Computers won't solve continuous problems exactly), and the Gram-Schmidt process isn't stable (We'll talk about it on next lectures)

Remember I told you that, if you have a vector v and decompose it as

$$v = r + \sum_{k=1}^{n} q_k q_k^* v$$

The part $\sum_{k=1}^{n} q_k q_k^*$ is a projector that project onto the matrix $\hat{Q}\hat{Q}^*$ ($\hat{Q}$ has columns $\{q_1, ..., q_n\}$)? Well, it turns out that we can express the Gram-Schmidt algorithm steps the same way! Let's remember. At the j-th step, we have:

$$q_j = \frac{a_j - \sum_{i=1}^{j-1} q_i q_i^* a_j}{\|a_j - \sum_{i=1}^{j-1} q_i q_i^* a_j\|_2}$$

That means we can rewrite this as

$$q_j = \frac{\left(I - \hat{Q}_{j-1}\hat{Q}_{j-1}^*\right)a_j}{\|\left(I - \hat{Q}_{j-1}\hat{Q}_{j-1}^*\right)a_j\|_2}$$

Where $\hat{Q}_{j-1} = \begin{pmatrix} | & | \\ q_1 & q_{j-1} \\ | & | \end{pmatrix}$. Let's define, for simplification, the projector P_j as:

$$P_j = I - \hat{Q}_{j-1}\hat{Q}_{j-1}^*$$

5.1. Modified Gram-Schmidt Algorithm

Using the definitions before, let's rewrite the Gram-Schmidt Algorithm.

For each value of j, the original Gram-Schmidt algorithm computes a single orthogonal projection of rank $m - (j-1)$. I'm just translating to language using projectors, it does this:

$$v_j = P_j a_j = \left(I - \hat{Q}_{j-1}\hat{Q}_{j-1}^*\right)a_j$$

If you go back to what I said early, you get the original formula, I'm just changing that bunch of sums and vectors for a matrix product. The original algorithm makes this computation using a single projector, but the one we'll see does this by a sequence of $j-1$ projectors of rank $m-1$. By the definition of P_j we can state that:

Theorem 5.1.1:

$$P_j = P_{\perp q_{j-1}} ... P_{\perp q_2} P_{\perp q_1}$$

Proof: Remember that $P_{\perp q_k} = I - q_k q_k^*$, and what does it do? It projects a vector v onto the subspace $\{q_1, ..., q_{k-1}\}$, removing the components $\{q_k, ...\}$ of v. The projector P_j does the exact same thing right? So, you can think that, if I project onto the complement of q_1, then onto the complement of q_2 and so on, at the j-th step, I'll have a vector that is orthogonal to the previous ones, that means, I remove all the previous components, leaving the projected vector as a linear combination of $\{q_k, ...\}$ □

Okay! If we define $P_1 = I$ we can rewrite $v_j = P_j a_j$ as:

$$v_j = P_{\perp q_{j-1}} ... P_{\perp q_2} P_{\perp q_1} a_j$$

The new modified algorithm is based on this new equation. We can get the same result stated on the previous version of the algorithm as:

$$v_j^{(1)} = a_j$$

$$v_j^{(2)} = P_{\perp q_1} v_j^{(1)}$$

$$v_j^{(3)} = P_{\perp q_2} v_j^{(2)}$$

$$v_j = v_j^{(j)} = P_{\perp q_{j-1}} v_j^{(j-1)}$$

We can rewrite it in form of pseudo-code:

```
for $i = 1$ to $n$
    $v_i = a_i$
for $i = 1$ to $n$
    $r_{ii} = \|v_i\|$
    $q_i = \frac{v_i}{r_{ii}}$
    for $j = i + 1$ to $n$
        $r_{ij} = q_i^* v_j$
        $v_j = v_j - r_{ij} q_i$
```

5.2. Gram-Schmidt as Triangular Orthogonalization

We can interpret each step of the Gram-Schmidt algorithm as a right-multiplication by a square upper-triangular matrix. Wait, what? Why? Get the R matrix:

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & & \vdots \\ & & \ddots & \vdots \\ & & & r_{nn} \end{pmatrix}$$

You can separate it as:

$$\begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & 1 & & \vdots \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ & r_{22} & & \vdots \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix} \cdots$$

So, we can easily see that, for the j-th matrix, the inverse of it is:

$$\begin{pmatrix} \ddots & & & & \\ & 1 & & & \\ & & r_{jj} & r_{j(j+1)} & \cdots \\ & & & \ddots & \end{pmatrix}^{-1} = \begin{pmatrix} \ddots & & & & \\ & 1 & & & \\ & & \frac{1}{r_{jj}} & -\frac{r_{j(j+1)}}{r_{jj}} & \cdots \\ & & & \ddots & \end{pmatrix}$$

That means we can understand the Gram-Schmidt algorithm as a orthogonalization by triangular matrices

$$AR_1 R_2 \ldots R_n = \hat{Q}$$

$$R_1 R_2 \ldots R_n = R^{-1}$$

6. Lecture 10 - Householder Triangularization

7. Lecture 11 - Least Squares problems

8. Lecture 12 - Conditioning and Condition Numbers

9. Lecture 13 - Floating Point Arithmetic

10. Lecture 14 and 15 - Stability

11. Lecture 16 - Stability of Householder Triangularization