EECS 16B Designing Information Devices and Systems II
Fall 2020 UC Berkeley

Note 19

1 Overview

In the previous note, we developed the **Singular Value Decomposition** of a matrix A. We saw that it was possible to take any matrix A and write it in its **compact** form as a sum of rank 1 matrices. Alternatively we could write the SVD of A in its **full** form as product of three matrices U, Σ, V^T. We keep both of these forms in mind as each form will have its own useful application.

Moving onto the focus of this note, we will first look at the compact form and how the SVD can be used to compress images. Then we will look at the full SVD through a geometric interpretation and develop a concept called the **pseudoinverse** of a matrix. The pseudoinverse possess a special minimum norm property and we will see how it can be used to control a system with "minimum energy."

2 Truncated SVD

The compact SVD of a matrix is a sum of k rank 1 matrices.

$$A = \sigma_1 \vec{u}_1 \vec{v}_1^T + \ldots + \sigma_k \vec{u}_k \vec{v}_k^T \tag{1}$$

Each term $\vec{u}_i \vec{v}_i^T$ is a matrix with rank 1 and is referred to as an **outer-product**. More importantly, each of these rank 1 matrices are weighted by σ_i which are ordered from largest to smallest.

This means we can give a rank $r < k$ approximation of the matrix A by "truncating" all terms past r.

$$A \approx \sigma_1 \vec{u}_1 \vec{v}_1^T + \ldots + \sigma_r \vec{u}_r \vec{v}_r^T \tag{2}$$

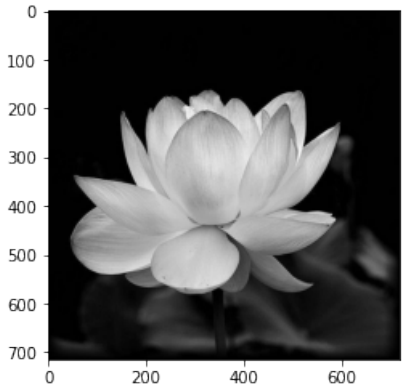
To demonstrate why this truncation is helpful let us think about the minimum number of entries required to express this matrix. If we were to look at an arbitrary $m \times n$ matrix A, one way to represent this matrix would be through its mn individual elements.

The SVD however, lets us compress this matrix by keeping the most important directions. To represent the same matrix A, we can keep track of three quantities: $\sigma_i, \vec{u}_i, \vec{v}_i$. For each rank, this would incur a total of $m+n+1$ entries meaning a rank r SVD approximation can be represented using around $(m+n+1) \times r$ entries. This will result in a significant amount of compression if $r \ll \min(m,n)$.

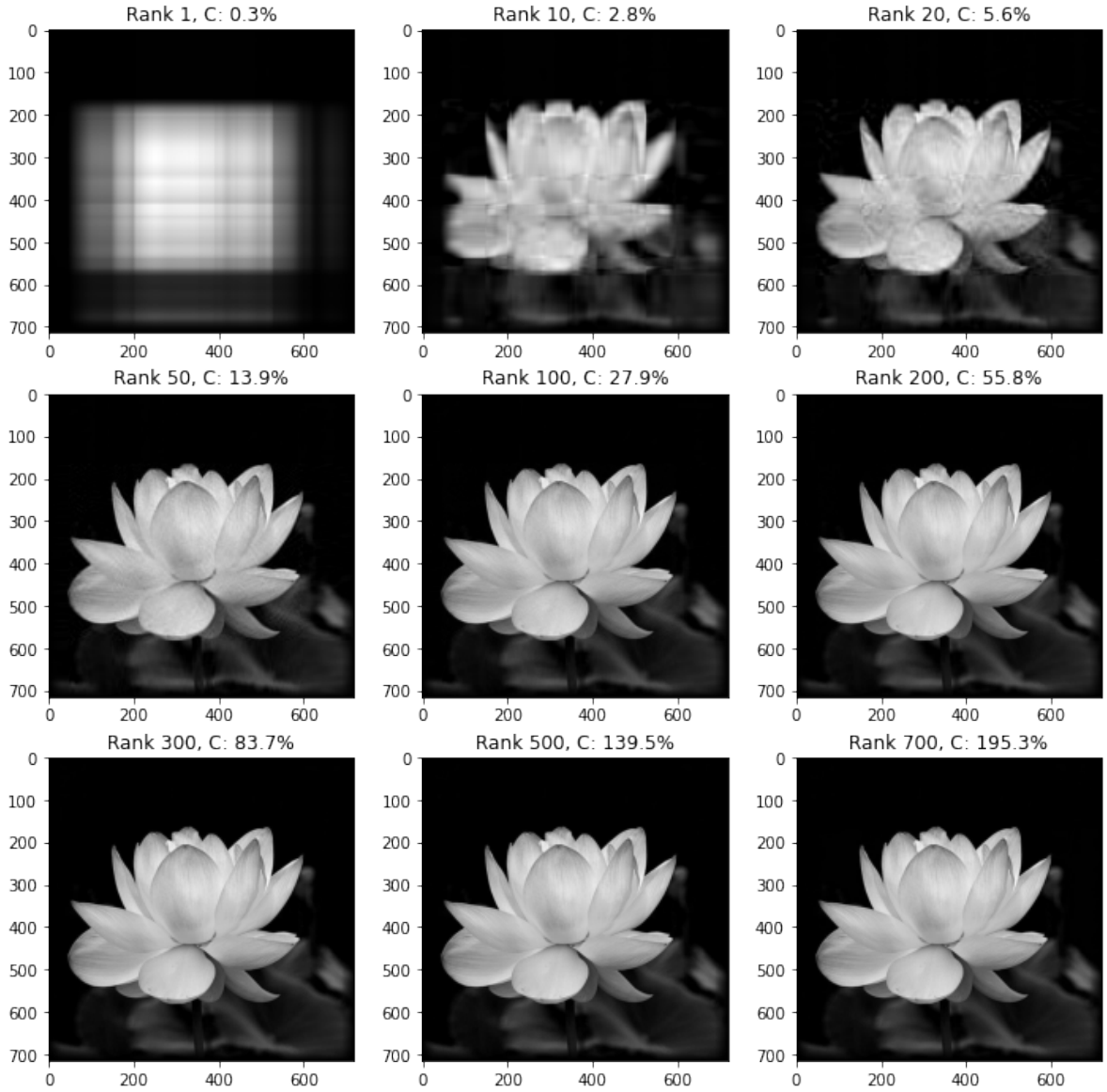
3 Image Compression

Using the truncated SVD defined in the previous section, let's try using it to compress an image. An image can be numerically expressed as a matrix whose entries represent the color of a specific pixel.

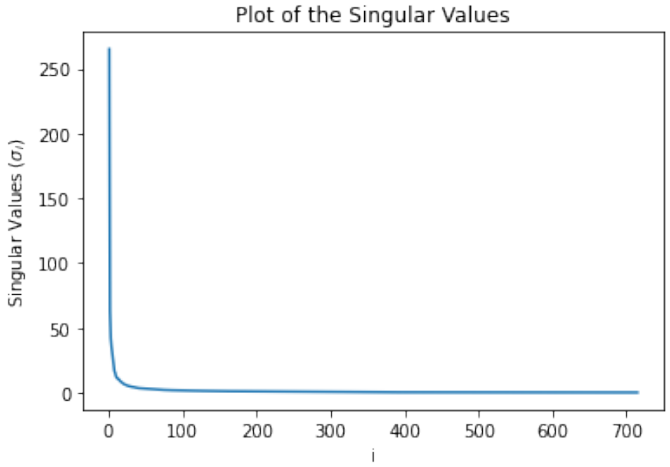
We will start off simple and take a look at a 720×720 grayscale image of a flower. This image is represented as a 720×720 matrix with values ranging between 0 and 255.



We can use the truncated SVD to create a rank r approximation of the original image. For each image, we also keep track of the compression ratio which we define as $C = \frac{(m+n+1)\cdot r}{m\cdot n}$.



At around $r = 100$, we can notice that the compressed image looks near identical to the original image. To provide a more technical analysis, let's take a look at a plot of the singular values of the matrix.



Remember that the singular values σ_i are ordered from largest to smallest. The plot shows the magnitude of σ_i on the y-axis and the index i on the x-axis. Upon analyzing the plot, we can notice that the values of σ_i drop off rather quickly and are near zero when $r = 100$.

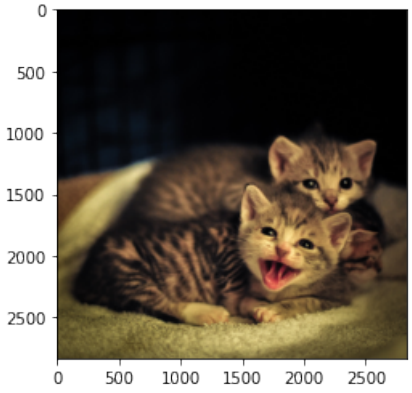
The compression error can be defined as $\varepsilon = A - A_r$ where A_r is a rank r approximation of the original image A. From the SVD, we can see that the matrix ε is

$$\varepsilon = A - A_r = \sum_{i=r+1}^{n} \sigma_i \vec{u}_i \vec{v}_i^T$$

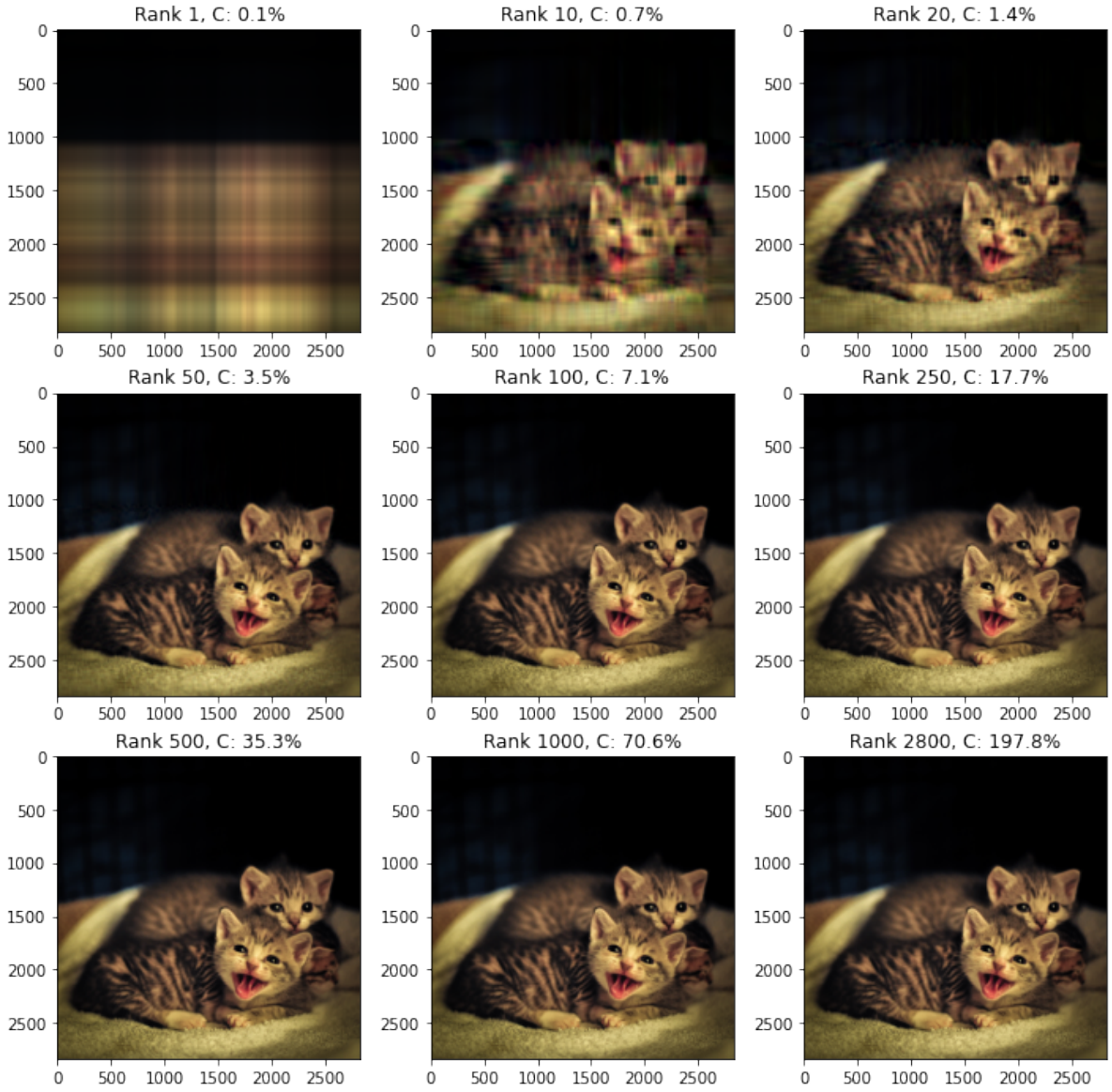
Since the σ_i are near zero, the ε will be close to zero as well.

3.1 Color Images

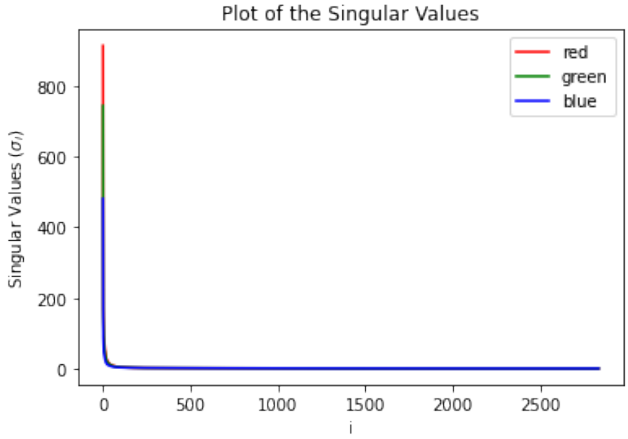
Now let's look at an example of a 2800×2800 color image. This image follows the RGB color scheme so the setup will be near identical except we will have a separate matrix for each color.



We can use the SVD to compress each of the red, green, and blue matrices separately and reconstruct the image by concatenating the three matrices. We will also keep track of the image compression ratio as defined in the previous section.



By $r = 50$ the reconstructed image looks near identical to the original and the compression is all the way down to around 3.5%. We plot the singular values of the matrix once more to illustrate this effect.



4 Pseudoinverse of a Matrix

In this section, we will take a look at the geometric interpretation of the SVD and how it can be used to solve a system of equations. The SVD naturally gives rise to a pseudoinverse of a matrix A.

Given a system of equations $A\vec{x} = \vec{y}$, we have developed multiple ways to solve this problem

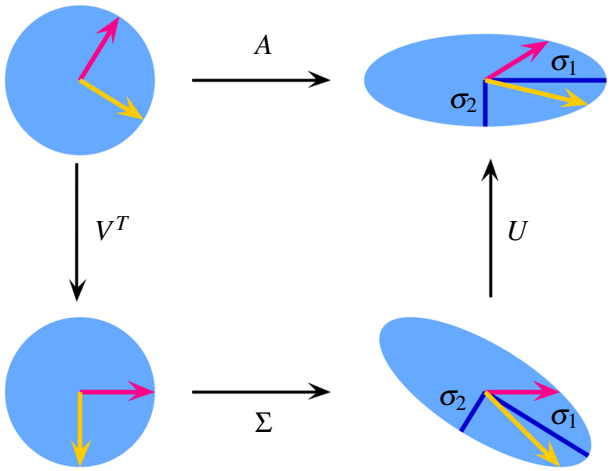
- If A is square and invertible, we can solve $\vec{x} = A^{-1}\vec{y}$
- If A is tall and full rank, we can use Least-Squares to say $\vec{x}^* = (A^TA)^{-1}A^T\vec{y}$.
- If A is wide and has a spare solution, we can use Orthogonal Matching Pursuit to estimate $\vec{x}$.

We will now develop a solution for $A\vec{x} = \vec{y}$ making no assumptions about the sparsity of the solution. Similar to how we performed matrix inversion, we will define a matrix $A^\dagger$ called the **pseudoinverse**.

4.1 SVD as Rotations

To derive the pseudoinverse, we will decompose the matrix A into a series of rotations and scaling. Since U and V are orthonormal matrices, they do not change the norm of a vector and will only rotate it.

$$\begin{aligned}
\vec{y} = A\vec{x} = U\Sigma V^T\vec{x} &\qquad A \text{ sends } \vec{x} \in \mathbb{R}^n \text{ to } \vec{y} \in \mathbb{R}^m.\\
\vec{z} = V^T\vec{x} &\qquad V^T \text{ rotates the vector } \vec{x} \text{ to a new vector } \vec{z}.\\
\vec{w} = \Sigma\vec{z} &\qquad \Sigma \text{ scales } \vec{z} \text{ and sends it to a vector } \vec{w} \in \mathbb{R}^m.\\
\vec{y} = U\vec{w} &\qquad U \text{ rotates the vector } \vec{w} \text{ to a new vector } \vec{y}.
\end{aligned}$$



4.2 Undoing the Matrices

In order to undo the effect of the matrix A, the plan is to undo each rotation and scaling one by one.

$$\begin{aligned}
\vec{y} = A\vec{x} = U\Sigma V^T\vec{x} &\quad A \text{ sends } \vec{x} \in \mathbb{R}^n \text{ to } \vec{y} \in \mathbb{R}^m.\\
U^T\vec{y} = \Sigma V^T\vec{x} &\quad \text{Undoing the rotation } U.\\
\Sigma^\dagger U^T\vec{y} = V^T\vec{x} &\quad \text{“Unscaling” the matrix } \Sigma.\\
\vec{x} = V\Sigma^\dagger U^T\vec{y} &\quad \text{Undoing the rotation } V^T
\end{aligned}$$

The matrices U and V^T are invertible but the matrix Σ is not even square.

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_m & \end{bmatrix}$$

If $\Sigma\vec{z} = \vec{w}$, the best we can do is divide by the singular values where division is possible. Therefore, if A is a wide matrix of rank k, we can undo σ_i for $i = 1, \ldots, k$.

$$\sigma_i z_i = w_i \implies z_i = \frac{w_i}{\sigma_i}$$

For the remaining singular values $i > k+1$, we will fill in the matrix with zeros. As a result, we define $\Sigma^\dagger$ as the following

$$\Sigma^\dagger = \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \ldots & 0 \\ \vdots & \ddots & \ldots & 0 \\ 0 & \ldots & \frac{1}{\sigma_k} & \vdots \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix} \tag{3}$$

To summarize, the pseudoinverse of a matrix A can be written as

$$\boxed{A^\dagger = V\Sigma^\dagger U^T} \tag{4}$$

4.3 Matrix Form of the Compact SVD

Sometimes, we like to write out the **compact SVD** in its matrix form noting that $\sigma_i = 0$ for $i > k$

$$A = U_c \Sigma_c V_c^T = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T \tag{5}$$

$$U_c = \begin{bmatrix} \vec{u}_1 & \ldots & \vec{u}_k \end{bmatrix} \qquad \Sigma_c = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \qquad V_c = \begin{bmatrix} \vec{v}_1 & \ldots & \vec{v}_k \end{bmatrix} \tag{6}$$

Here Σ_c is a $k \times k$ diagonal matrix with the singular values on its diagonal.

In fact, we can write the pseudoinverse of a matrix $A^\dagger$ using the compact SVD.

$$\boxed{A^\dagger = V_c \Sigma_c^{-1} U_c^T} \tag{7}$$

4.4 Minimum Norm Property

If A is an $m \times n$ matrix that that is wide $(n > m)$ with full row-rank, the system of equations $A\vec{x} = \vec{y}$ has infinite solutions. We claim the pseudoinverse gives the solution with minimum norm

$$\vec{x} = A^{\dagger}\vec{y} = V_c \Sigma_c^{-1} U_c^T \vec{y} \tag{8}$$

As an optimization problem, we can phrase this as the following[1]

$$\min_{\vec{x} \in \mathbb{R}^n} \|\vec{x}\|^2 \quad \text{subject to } A\vec{x} = \vec{y} \tag{9}$$

We will now prove that the pseudoinverse gives the solution to this optimization problem.

4.5 Proof:

The compact SVD helps us understand the matrix A in terms of its fundamental subspaces. We can write out the matrix A in terms of its SVD but also break it down into its compact and null-space terms.

$$A = U\Sigma V^T = \begin{bmatrix} U_c \end{bmatrix} \begin{bmatrix} \Sigma_c & 0 \end{bmatrix} \begin{bmatrix} V_c^T \\ V_n^T \end{bmatrix} = U_c \Sigma_c V_c^T \tag{10}$$

Here $U = U_c$ since the matrix A has rank m.

The constraint of our optimization problem can we rewritten by expanding out the compact SVD of A.

$$A\vec{x} = \vec{y} \implies U_c \Sigma_c V_c^T \vec{x} = \vec{y} \tag{11}$$

4.5.1 Change of Coordinates

The vectors of the V matrix form an orthonormal basis for $\mathbb{R}^n$. Recall that the last k columns of V form a basis for $\text{Nul}(A)$. Now let $\vec{z}$ be the coordinates of $\vec{x}$ using the basis V. We can break up these coordinates into two components: $\vec{z}_c$ representing the first k coordinates and $\vec{z}_n$ representing the last $n-k$.

$$\vec{x} = V\vec{z} = \begin{bmatrix} V_c & V_n \end{bmatrix} \begin{bmatrix} \vec{z}_c \\ \vec{z}_n \end{bmatrix} = V_c\vec{z}_c + V_n\vec{z}_n \tag{12}$$

Plugging our coordinate representation $\vec{z}$, we see that

$$A\vec{x} = U_c \Sigma_c V_c^T \vec{x} = U_c \Sigma_c V_c^T V_c \vec{z}_c + U_c \Sigma_c V_c^T V_n \vec{z}_n = U_c \Sigma_c \vec{z}_c = \vec{y} \tag{13}$$

The last equality comes from the fact that the columns V_c are orthonormal to V_n.

The objective $\|\vec{x}\|^2$ can be rewritten as follows using the coordinates $\vec{z}$.

$$\|\vec{x}\|^2 = \|V_c\vec{z}_c + V_n\vec{z}_n\|^2 = \|\vec{z}_c\|^2 + \|\vec{z}_n\|^2 \tag{14}$$

Since the constraint $A\vec{x} = \vec{y}$ does not depend on $\vec{z}_n$, we can pick $\vec{z}_n = \vec{0}$. The optimization problem can then

[1] Minimizing the norm of $\vec{z}$ is equivalent to minimizing the squared norm since norms are positive definite and $f(x) = x^2$ is a monotonic transform.

be rewritten as

$$\min_{\vec{z}_c \in \mathbb{R}^k} \|\vec{z}_c\|^2 \quad \text{subject to } U_c \Sigma_c \vec{z}_c = \vec{y} \tag{15}$$

Since U_c and Σ_c are invertible matrices, there must be a unique solution $\vec{z}_c = \Sigma_c^{-1} U_c^T \vec{y}$. Lastly, converting back to standard basis coordinates, it follows that

$$\vec{x} = V\vec{z} = \begin{bmatrix} V_c & V_n \end{bmatrix} \begin{bmatrix} \Sigma_c^{-1} U_c^T \vec{y} \\ 0 \end{bmatrix} = V_c \Sigma_c^{-1} U_c^T \vec{y} = A^\dagger \vec{y} \tag{16}$$

5 Optimal Control

Now that we have equipped ourselves with the pseudoinverse and have proved its minimum norm property, let us take a look at an application to control systems. Suppose we have a discrete-time system with the following dynamics

$$\vec{x}[t+1] = A\vec{x}[t] + B\vec{u}[t]$$

Let us assume that this system is controllable meaning we can reach any target state $\vec{t} \in \mathbb{R}^n$ in at most n time-steps. While it may be desirable to reach our target $\vec{t}$ as quickly as possible, we may be limited by the physical constraints of the system which prevents high valued inputs.

Therefore, we can try to increase the number of time-steps in order to relax our system's constraints. We will be using the squared norm as a measure of the amount of "energy" it takes to move our system to a target state. By relxaing our system to reach our target in $m > n$ states, we can write out $\vec{x}[m]$ as

$$\vec{x}[m] = A^n\vec{x}[0] + A^{m-1}B\vec{u}[0] + A^{n-2}B\vec{u}[1] + \ldots + AB\vec{u}[m-2] + B\vec{u}[m-1] \tag{17}$$

$$= A^n\vec{x}[0] + \begin{bmatrix} B & AB & \cdots & A^{m-1}B \end{bmatrix} \begin{bmatrix} \vec{u}[m-1] \\ \vec{u}[m-2] \\ \vdots \\ \vec{u}[0] \end{bmatrix} \tag{18}$$

$$= A^n\vec{x}[0] + H\vec{w} \tag{19}$$

Reaching $\vec{t}$ in m time-steps with minimum energy can be phrased as the following optimization problem

$$\min_{\vec{w} \in \mathbb{R}^{m \cdot p}} \|\vec{w}\|^2 \quad \text{subject to } H\vec{w} = \vec{t} - A^n\vec{x}[0]$$

If the matrix B is of size $n \times p$ then H will be a wide matrix of size $n \times mp$. The system $H\vec{w} = \vec{y}$ will have infinite solutions, but as we saw in the previous section, the minimum norm solution comes from the pseudoinverse.

$$\vec{w}^* = H^\dagger(\vec{t} - A^n\vec{x}[0])$$

5.1 Car Example

Let us take a look at the car model once more represented by following dynamics

$$\frac{d}{dt}\begin{bmatrix} p(t) \\ v(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p(t) \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M} \end{bmatrix} u(t)$$

Suppose $M = 1\,\text{kg}$ and we discretized our car model with rate $T = 0.1\,\text{s}$. The discretized model would be

$$\begin{bmatrix} p[t+1] \\ v[t+1] \end{bmatrix} = \begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p[t] \\ v[t] \end{bmatrix} + \begin{bmatrix} 0.005 \\ 0.05 \end{bmatrix}$$

Assuming the car starts at rest $\vec{x}[0] = \vec{0}$ let us try to move the car to $p = 10\,\text{m}$ with no velocity in two time-steps. [2]

$$\vec{x}[2] = \begin{bmatrix} 10 \\ 0 \end{bmatrix} = A^2\vec{x}[0] + ABu[0] + Bu[1] = \begin{bmatrix} B & AB \end{bmatrix} \begin{bmatrix} u[1] \\ u[0] \end{bmatrix}$$

The inputs $u[0]$ and $u[1]$ can be computed through a matrix inverse as follows

$$\vec{w} = \begin{bmatrix} u[1] \\ u[0] \end{bmatrix} = \mathscr{C}^{-1}\vec{x}[2] = \begin{bmatrix} -2000 \\ 2000 \end{bmatrix}$$

To reach our target in two time-steps, we would need to apply forces of $20\,\text{kN}$ which would break our car. Therefore, let's try to set up a minimum norm problem to reach our target in ten time-steps.

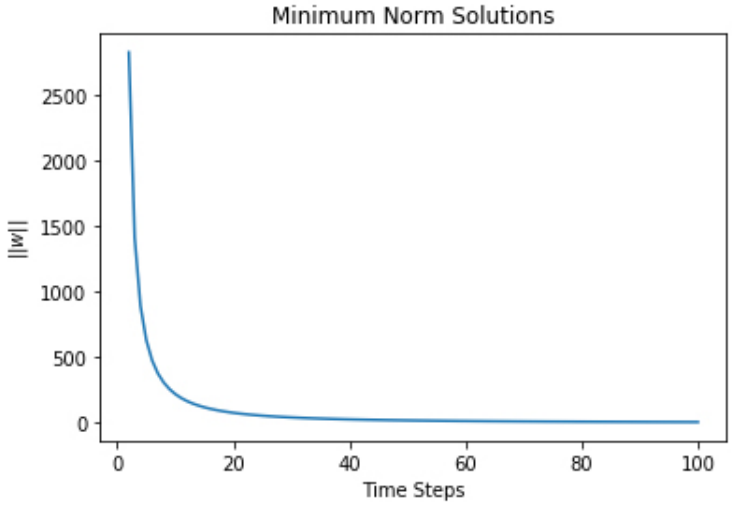
$$H = \begin{bmatrix} B & AB & \cdots & A^9B \end{bmatrix} \qquad \vec{w} = \begin{bmatrix} u[9] & u[8] & \cdots & u[0] \end{bmatrix}^T$$

$$\min_{\vec{w} \in \mathbb{R}^{10}} \|\vec{w}\|^2 \quad \text{subject to } H\vec{w} = \vec{t}$$

Again we compute the minimum norm solution by applying the pseudoinverse

$$\vec{w}^* = H^\dagger \vec{t} \qquad \|\vec{w}\| = 220$$

Notice how the norm drops significantly. As we add more time-steps, the norm $\|\vec{w}\|$ will continue to drop off. The results are plotted below.



Contributors:

- Taejin Hwang.

[2] Warning: Do not try this at home. Moving 10 m in 0.02 would be traveling at an average of 1100 miles per hour.

