Linear Algebra

Singular Value Decomposition

Automotive Intelligence Lab.

HANYANG UNIVERSITY

AI LAB Automotive Intelligence

Contents

Definition of Singular Value Decomposition

Singular Value Decomposition

- **For a set of orthogonal vectors,**
  - Orthogonal set whose $\boxed{\text{size}}$ changes after a linear transformation but still $\boxed{\text{orthogonal}}$.

- **Call singular value decomposition as SVD.**

SVD: One of the Matrix Decomposition Methods

- **SVD allows to decompose random $\boldsymbol{m \times n}$ matrix $\boldsymbol{A}$ as:**

$$\boldsymbol{A = U\Sigma V^T}$$

$\boldsymbol{A}: m \times n$ rectangular matrix

$\boldsymbol{U}: m \times m$ orthogonal matrix

$\boldsymbol{\Sigma}: m \times n$ diagonal matrix

$\boldsymbol{V}: n \times n$ orthogonal matrix

Four matrix's size and properties

Supplementary Explanation about Previous Page

- **Property of orthogonal matrix U.**
  - $U^T U = U U^T = \boxed{I}$.
  - $U^T = \boxed{U^{-1}}$.

- **Property of diagonal matrix Σ.**
  - Form of a matrix of size $m \times n$.

$$\begin{pmatrix} p_1 & 0 \\ 0 & p_2 \end{pmatrix}$$

$m = n = 2$

$$\begin{pmatrix} p_1 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & p_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

$m > n$

$$\begin{pmatrix} p_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & p_2 & \cdots & 0 & 0 & \cdots & 0 \\ & & \ddots & & & & \\ 0 & 0 & \cdots & p_m & 0 & \cdots & 0 \end{pmatrix}$$

$m < n$

Geometric Implications of SVD

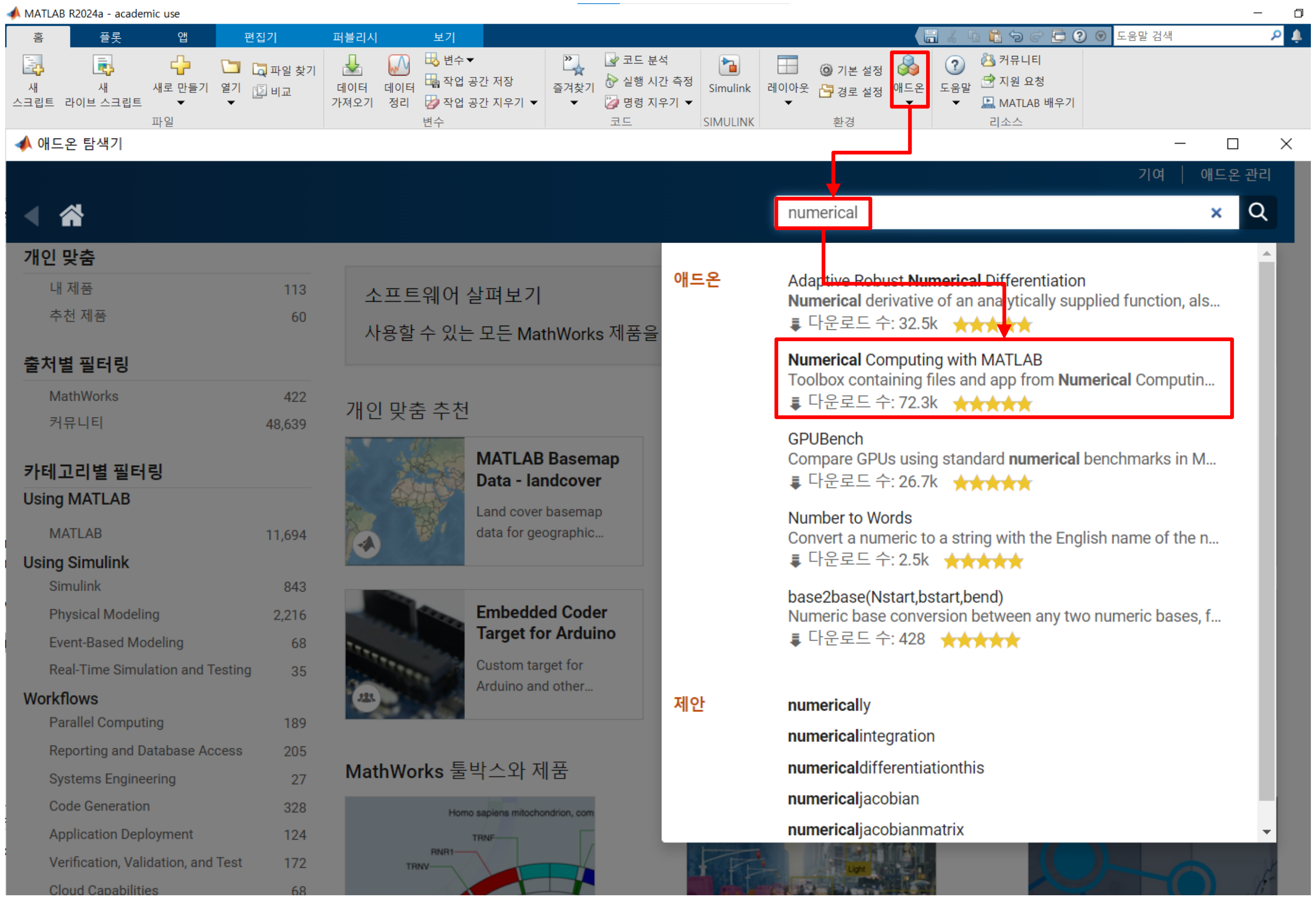
Example In a Two-Dimensional Vector Space

- **Can always find $\boxed{\text{orthogonal vector}}$ to a given vector.**
  - In case of given vector is in a two-dimensional real vector space.
  - Formal method
    - Using $\boxed{\text{Gram- schmidt process.}}$.

- **If same linear transformation is taken for two orthogonal vectors,**
  - Those two vectors are not guaranteed to be orthogonal.
  - Example of this is on the next page.

Preparation for 'Numerical Computing toolbox'

- You need 'Numerical Computing Toolbox' to run the code in this lecture.
- Follow the procedure to install the toolbox
- Also add the given files to current directory.



애드온

Adaptive Robust Numerical Differentiation
Numerical derivative of an analytically supplied function, als...
다운로드 수: 32.5k ★★★★★

Numerical Computing with MATLAB
Toolbox containing files and app from Numerical Computin...
다운로드 수: 72.3k ★★★★★

GPUBench
Compare GPUs using standard numerical benchmarks in M...
다운로드 수: 26.7k ★★★★★

Number to Words
Convert a numeric to a string with the English name of the n...
다운로드 수: 2.5k ★★★★★

base2base(Nstart,bstart,bend)
Numeric base conversion between any two numeric bases, f...
다운로드 수: 428 ★★★★★

제안

numerically
numericalintegration
numericaldifferentiationthis
numericaljacobian
numericaljacobianmatrix

Code Exercise of Visualization of Orthogonality After Linear Transformation

Code Exercise (14_01)

- Visualize the orthogonality of matrix after linear transformation.
- Run the code and type anything in command window, then the figure will appear.

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% REQUIREMENT
% You need 'Numerical Computing with MATLAB toolbox' to use 'eigshow()'

A =[1 3;4 2]/4;
n_steps = 100;
step_mtx = eye(2);
[x, y] = ndgrid(-1:0.15:1);
xy_min = min(min(A*[x(:), y(:)]'))*1.5;
xy_max = max(max(A*[x(:), y(:)]'))*1.5;

dot_colors = jet(length(x(:)));

xlim([xy_min, xy_max]);
ylim([xy_min, xy_max]);
pause;
for i_steps = 1:n_steps
   step_mtx = (A-eye(2))/n_steps*i_steps;

   new_xy = (eye(2)+step_mtx)*[x(:), y(:)]';
   scatter(new_xy(1,:), new_xy(2,:),30,dot_colors,'filled')
   grid on;
   xlim([xy_min, xy_max]); ylim([xy_min, xy_max]);
   pause(0.01);
end
```

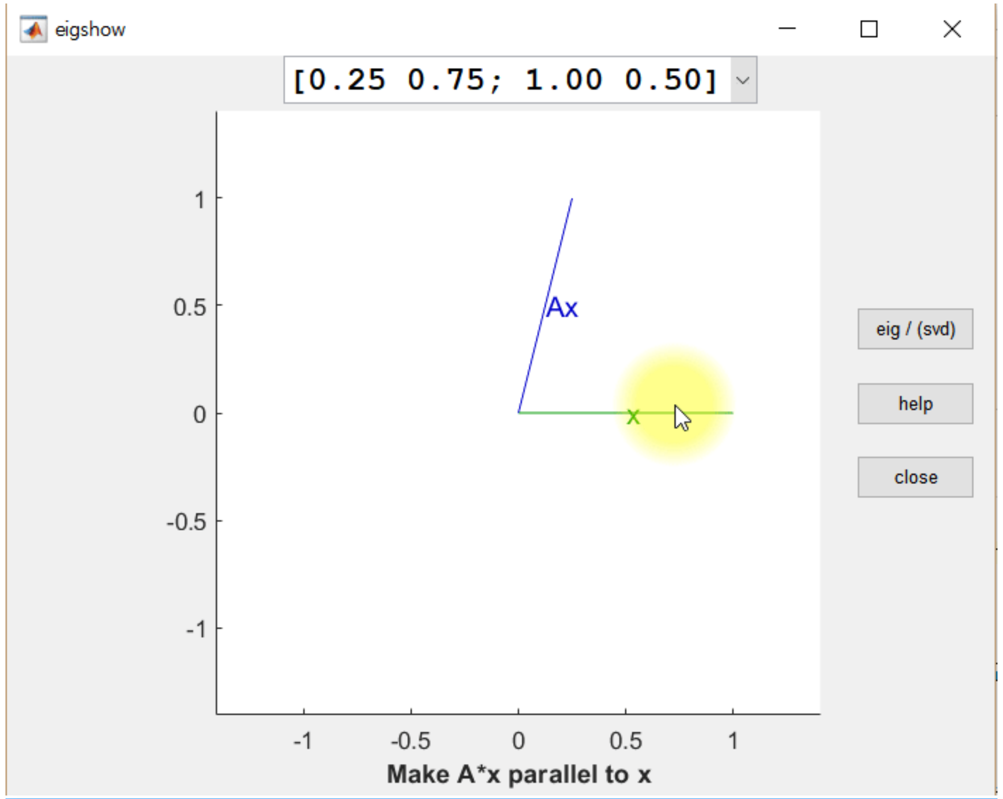
```
% Animation with circle

t=linspace(0,2*pi,100);
x=cos(t);
y=sin(t);
plot(x,y);
[temp] = A*[x;y];
plot(temp(1,:),temp(2,:))
XLIM=[xy_min, xy_max];
YLIM=[xy_min, xy_max];

% Animation
figure;
plot(x,y);
grid on;
xlim(XLIM);
ylim(YLIM);
pause;

for i_steps = 1:n_steps
   step_mtx = (A-eye(2))/n_steps*i_steps;
   new_xy = (eye(2)*step_mtx)*[x;y];
   plot(new_xy(1,:), new_xy(2,:));
   grid on;
   xlim(XLIM);
   ylim(YLIM);
   pause(0.01);
end

% Eigshow
figure;
eigshow(A)
```

MATLAB code of visualize the orthogonality

Visualization of Orthogonality After Linear Transformation

- **Figure below shows results of linear transformation $A\vec{x}$.**
  - Matrix $A = \begin{pmatrix} 0.25 & 0.75 \\ 1 & 0.5 \end{pmatrix}$, random vector $\vec{x}$.

$$A \underbrace{\begin{bmatrix} \vec{x} & \vec{y} \end{bmatrix}}_{\text{orthogonal}} = \underbrace{\begin{bmatrix} A\vec{x} & A\vec{y} \end{bmatrix}}_{\text{orthogonal}}$$



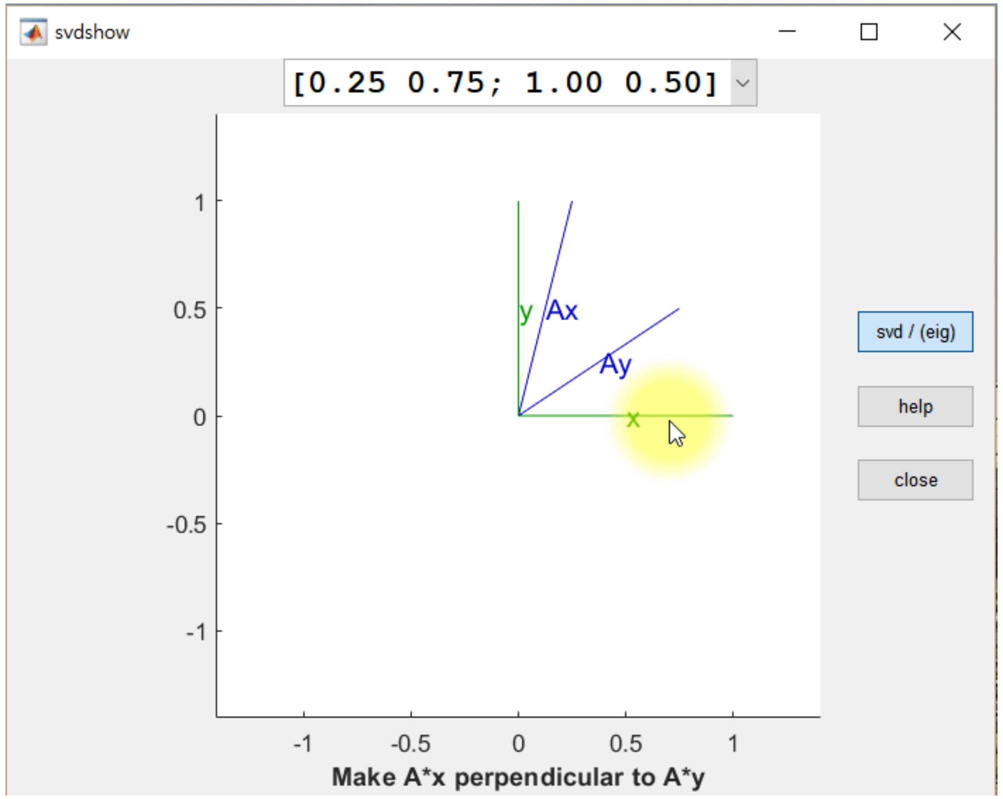
[0.25 0.75; 1.00 0.50]

Make A*x parallel to x

Result of linear transformation $A\vec{x}$

Two Orthogonal Vectors Remain Orthogonal After Linear Transformation

- **Figure below shows results of linear transformation $A\vec{x}$, $A\vec{y}$.**
  - $\vec{x}$ and $\vec{y}$ are two orthogonal vectors.



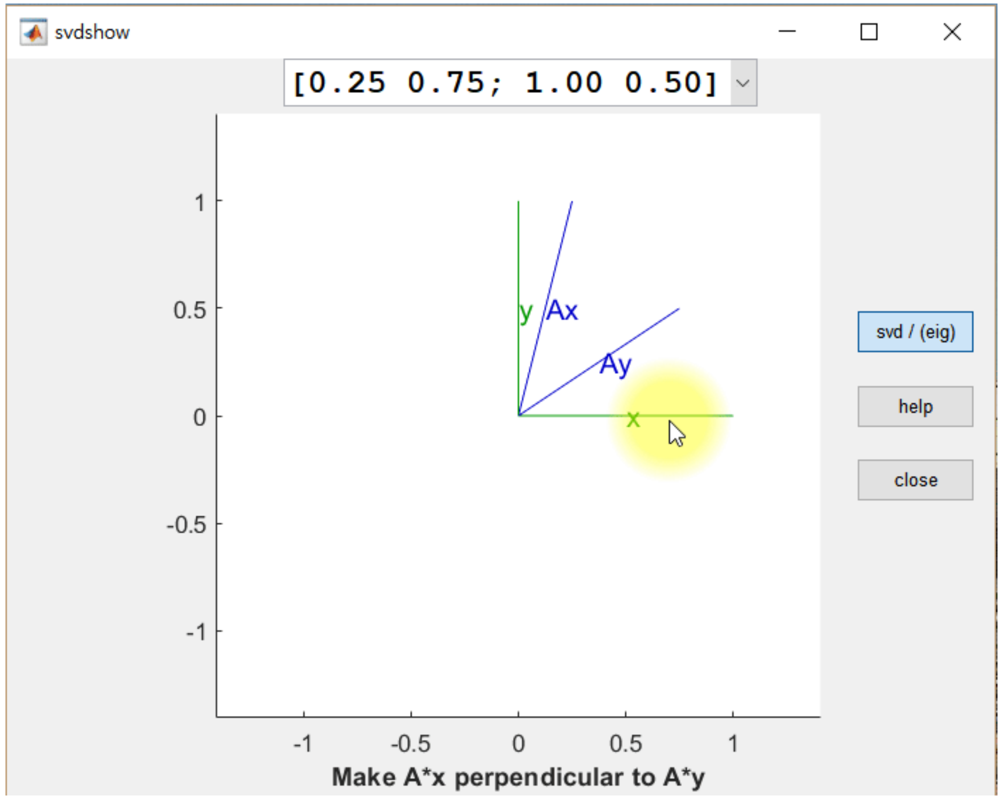
Random vector x and vector y orthogonal to x, and Ax and Ay

Two Main Things to Note In The Figure

- **Not just one case where are orthogonal.**

- **Length changed slightly.**
  - After $\vec{x}$ and $\vec{y}$ are converted through a matrix A.
  - These changed length value are **scaling factor**.
    - Generally called $\boxed{\text{Singular value}}$.
    - Start from the largest values: $\sigma_1, \sigma_2, \ldots$

- **Let's go back to SVD.**



Random vector x and vector y orthogonal to x, and Ax and Ay

Definition of SVD

- $A = U\Sigma V^T \Rightarrow AV = U\Sigma$
  - V: Matrix of Orthogonal vectors before linear transformation.
  - Σ: Diagonal matrix consisting of singular values.
  - U: Matrix of orthogonal vectors after. linear transformation.
    - Each vectors are normalized to 1.

$$\boldsymbol{V} = [\vec{x} \quad \vec{y}] \qquad \boldsymbol{U} = [\overrightarrow{u_1} \quad \overrightarrow{u_2}] \qquad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

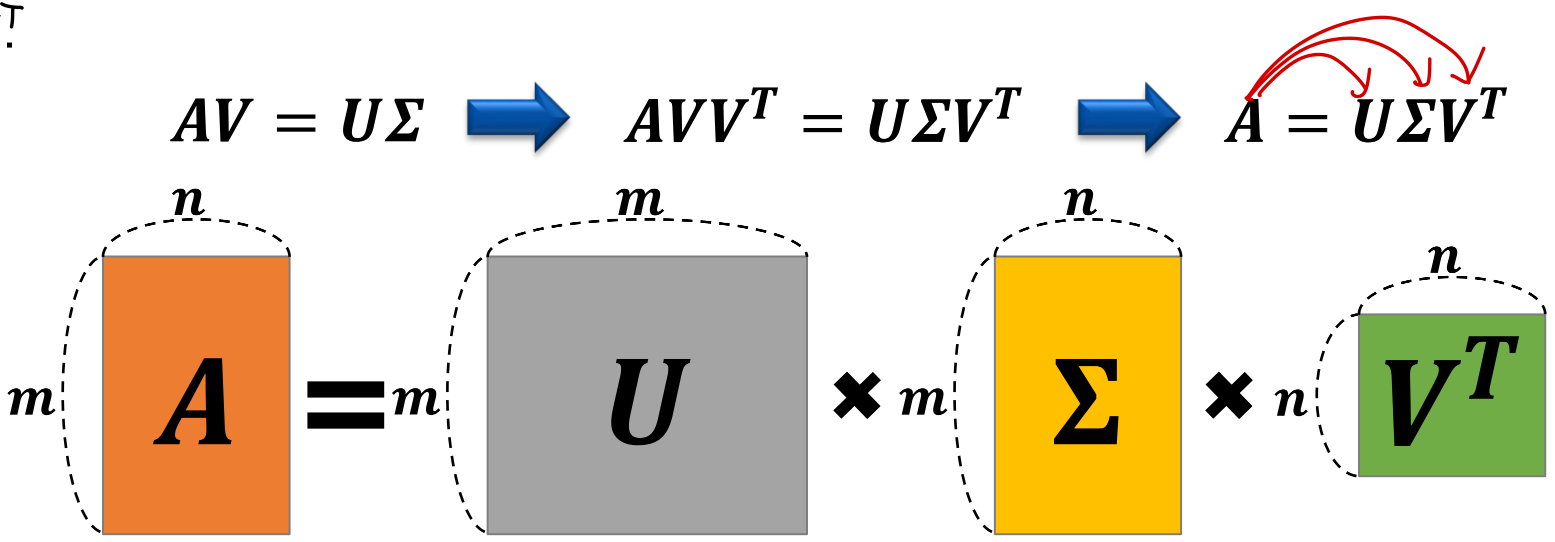
Element matrices of SVD

Relationships between 4 matrices (A, V, U, Σ)

- **From a linear transformation perspective: $AV = U\Sigma$.**
  - It gives you a question.
  - Is it possible to find matrix U ?
    - After linearly transforming column vector in matrix V through matrix A.
    - Size of matrix V is changed by **singular value σ_1, σ_2**.
    - But column vectors in U are still **orthogonal**.

- **V is orthogonal matrix**
  - $V^{-1} = V^T$.

$$AV = U\Sigma \Rightarrow AVV^T = U\Sigma V^T \Rightarrow A = U\Sigma V^T$$



Visualization of the results of SVD of an arbitrary matrix A.

SVD with Different Input and Output Dimensions

좀만 더 하자

파이팅♡

Decomposition of Matrix A In Case of Non-Square Matrix

- **Matrix A is $m \times n$ dimensions.**
  - If A is 2×3 matrix,
    - Matrix A lowers the dimension from 3D to $\boxed{2D}$.
  - Question about what SVD requires:
    - Linearly transform 3 vectors that were orthogonal in a 3-dimensional space.
    - Convert them to 2 dimensions.
    - Is it possible to make two vectors orthogonal in a 2-dimensional space?

$$\begin{tikzcd}
{[2\times 1]} & = & {[2\times 3]} & {[3\times 1]} \\
2D & & & 3D \arrow[lll, bend left]
\end{tikzcd}$$

Code Exercise of Visualization of Linear Transformation by Unsquared Matrix A

■ Code Exercise (14_02)

- ▶ Visualize the linear transformation by unsquared matrix.
- ▶ Run the code and type anything in command window, then the figure will appear.

```matlab
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Define A matrix and vectors
vector1 = [-1,2,1]';
vector2 = [1,1,1]';
A   =   [vector1/norm(vector1)   vector2/norm(vector2)]*[vector1/norm(vector1)
vector2/norm(vector2)]';

% Anination with dots
[X,Y,Z] = ndgrid(-1:0.3:1);
n_steps = 100;
step_mtx = eye(3);
newXYZ = A*[X(:), Y(:), Z(:)]';
xyz_min = min(min(min([newXYZ(:), newXYZ(:), newXYZ(:)]')))*1.5;
xyz_max = max(max(max([newXYZ(:), newXYZ(:), newXYZ(:)]')))*1.5;
LIMS = [xyz_min, xyz_max];

dot_colors = jet(length(X(:)));
figure(2)
scatter3(X(:), Y(:), Z(:),30, dot_colors,'filled');
xlim(LIMS); ylim(LIMS); zlim(LIMS);

grid on;
hold on;
line([xyz_min, xyz_max], [0,0], [0,0], 'linewidth', 3)
line([0,0], [xyz_min, xyz_max], [0,0], 'linewidth', 3)
line([0,0], [0,0], [xyz_min, xyz_max], 'linewidth', 3)
xlabel('x'); ylabel('y'); zlabel('z')
hold on;
pause;
```

```matlab
for i_steps = 1:n_steps
   step_mtx = (A-eye(3))/n_steps*i_steps;

   new_xyz = (eye(3)+step_mtx)*[X(:), Y(:), Z(:)]';
   scatter3(new_xyz(1,  :),  new_xyz(2,  :),  new_xyz(3,  :),  30,  dot_colors,
'filled');
   grid on;
   hold on;
   line([xyz_min, xyz_max], [0,0], [0,0], 'linewidth', 3)
   line([0,0], [xyz_min, xyz_max], [0,0], 'linewidth', 3)
   line([0,0], [0,0], [xyz_min, xyz_max], 'linewidth', 3)
   hold off;
   xlim(LIMS); ylim(LIMS); zlim(LIMS);
   xlabel('x'); ylabel('y'); zlabel('z');
   pause(0.01);
end

% SVD
[U,S,V] = svd(A);
hold on;
for i = 1:3
   mArrow3([0,0,0],  [U(1,  i)*S(i,  i),  U(2,  i)*S(i,  i),  U(3,  i)*S(i,i)],
'color', 'b');
   hold on;
end

for i = 1:3
   mArrow3([0,0,0], [V(1, i), V(2, i), V(3, i)], 'color', 'g');
end
```

MATLAB code of visualize linear transformation by unsquared matrix

Visualization of Linear Transformation by Non-Square Matrix A

- **Figure below shows transition from 3D vector space to 2D vector space.**
  - Matrix $\boldsymbol{A}$ is 2×3 matrix.
  - Figure 1.
    - Transformation that projects 3 dimensional vectors onto a plane by matrix A.
  - Figure 2, 3.
    - Apply SVD to matrix A, visualize vector orthogonal before and after linear transformation.
    - **Green** arrows: orthogonal vectors before linear transformation.
    - **Blue** arrows: orthogonal vectors after linear transformation.

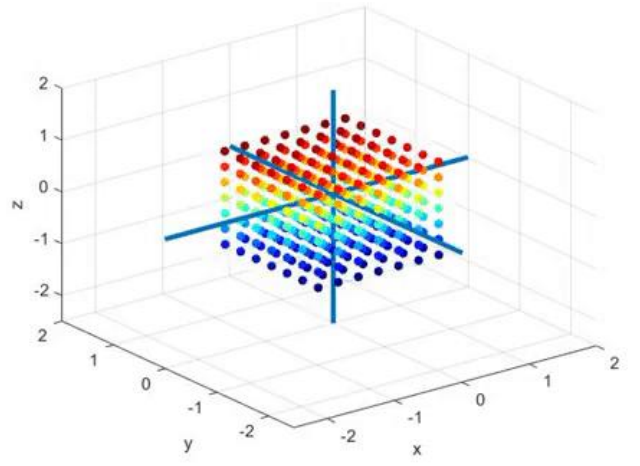


Figure 1.

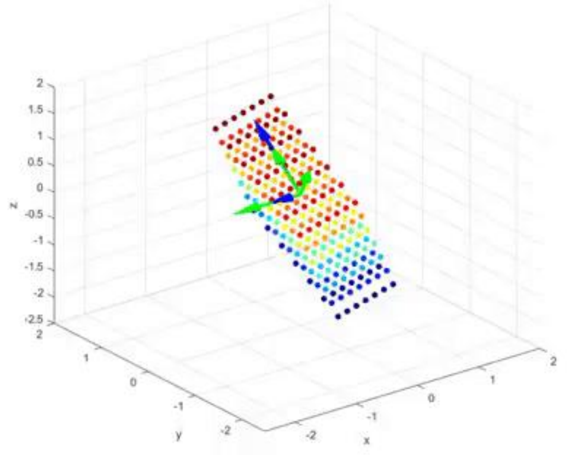


Figure 2.

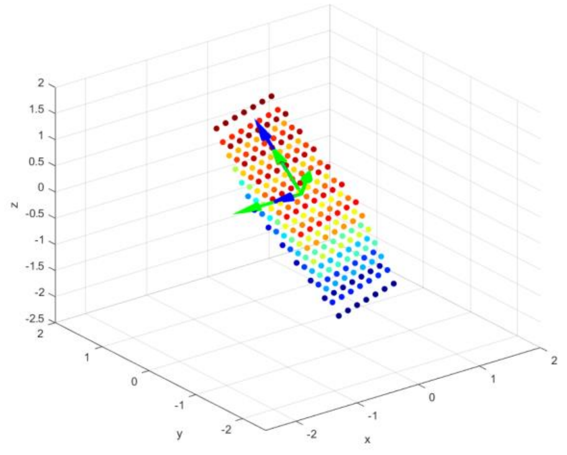


Figure 3.

Calculation Method of SVD

Example of SVD Calculation: Calculate U

- $A = U\Sigma V^T$ $\rightarrow A^T = (U\Sigma V^T)^T = V\Sigma^T U^T$
- $A^T = V\Sigma^T U^T$

$VV^T = I$

- $AA^T = U\Sigma \cancel{V^T V}\Sigma^T U^T = U\Sigma\Sigma^T U^T = U\Sigma^2 U^T = U\Sigma^2 U^{-1}$
- $A^T A = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T\Sigma V^T = V\Sigma^2 V^T = V\Sigma^2 V^{-1}$

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

Eigen decomposition

Example of SVD Calculation: Calculate U

- $\boldsymbol{A = U\Sigma V^T}$
  - Calculate U (left singular vector).
    - Calculate AA^T.

$$\boldsymbol{A} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \qquad \boldsymbol{AA^T} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix}$$

    - Calculate **eigenvalues** and **eigenvectors** of $\boldsymbol{AA^T}$.
      - Arrange eigenvectors in order of largest eigenvalue.

$$\begin{aligned} \lambda_1 &= 10 \rightarrow \overrightarrow{\boldsymbol{v_1}} = \begin{bmatrix} 1 & -1 \end{bmatrix} \\ \lambda_2 &= 12 \rightarrow \overrightarrow{\boldsymbol{v_2}} = \begin{bmatrix} 1 & 1 \end{bmatrix} \end{aligned} \quad \rightarrow \quad \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

    - **Normalize** each eigenvector.
      - Then, you can get $\boldsymbol{U}$.

$$\boldsymbol{U} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

Process to calculate U

Example of SVD Calculation: Calculate V^T

- $\boldsymbol{A} = \boldsymbol{U\Sigma V^T}$
  - Calculate V(right singular vector).
    - Calculate $\boldsymbol{A^T A}$.

$$\boldsymbol{A} = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} \quad \boldsymbol{A^T A} = \begin{bmatrix} 3 & -1 \\ 1 & 3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix}$$

    - Calculate **eigenvalues** and **eigenvectors** of $\boldsymbol{A^T A}$.
      - Arrange eigenvectors in order of largest eigenvalue.

$$\begin{aligned} \lambda_1 &= 12 \rightarrow \overrightarrow{\boldsymbol{v_1}} = [1 \quad 2 \quad 1] \\ \lambda_2 &= 10 \rightarrow \overrightarrow{\boldsymbol{v_2}} = [2 \quad -1 \quad 0] \\ \lambda_3 &= 0 \rightarrow \overrightarrow{\boldsymbol{v_3}} = [1 \quad 2 \quad -5] \end{aligned} \rightarrow \begin{bmatrix} 1 & 2 & 1 \\ 2 & -1 & 2 \\ 1 & 0 & -5 \end{bmatrix}$$

    - **Normalize** each eigenvector.
      - Then you can get $\boldsymbol{V}$.

$$\mathbf{V} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{30}} \\ \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{5}} & \frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{6}} & 0 & -\frac{5}{\sqrt{30}} \end{bmatrix} \rightarrow \boldsymbol{V^T} = \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & -\frac{5}{\sqrt{30}} \end{bmatrix}$$

Process to calculate V

Example of SVD Calculation: Calculate Σ

- $A = U\Sigma V^T$
  - Calculate Σ.
    - Σ is $m \times n$ rectangular diagonal matrix.
      - Same size of matrix A.
    - It's diagonal elements:
      - Square root of eigenvalues obtained through eigenvalue decomposition of matrices A^TA or AA^T.
    - Arrange the values diagonally starting from the largest value.

$$AA^T = \begin{bmatrix} 11 & 1 \\ 1 & 11 \end{bmatrix} \Rightarrow \begin{matrix} \lambda_1 = \boxed{10} \\ \lambda_2 = \boxed{12} \end{matrix}$$

$$A^TA = \begin{bmatrix} 10 & 0 & 2 \\ 0 & 10 & 4 \\ 2 & 4 & 2 \end{bmatrix} \Rightarrow \begin{matrix} \lambda_1 = \boxed{12} \\ \lambda_2 = \boxed{10} \\ \lambda_3 = 0 \end{matrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & \sqrt{10} & 0 \end{bmatrix}$$

Process to calculate Σ

Reduced SVD

Full SVD

- **Decomposing $m \times n$ matrix A into SVD as shown Fig 1..**
  - Only in case of $m > n$.

- **In reality…,**
  - It is rare to perform full SVD.
  - It is common to perform reduced SVD.

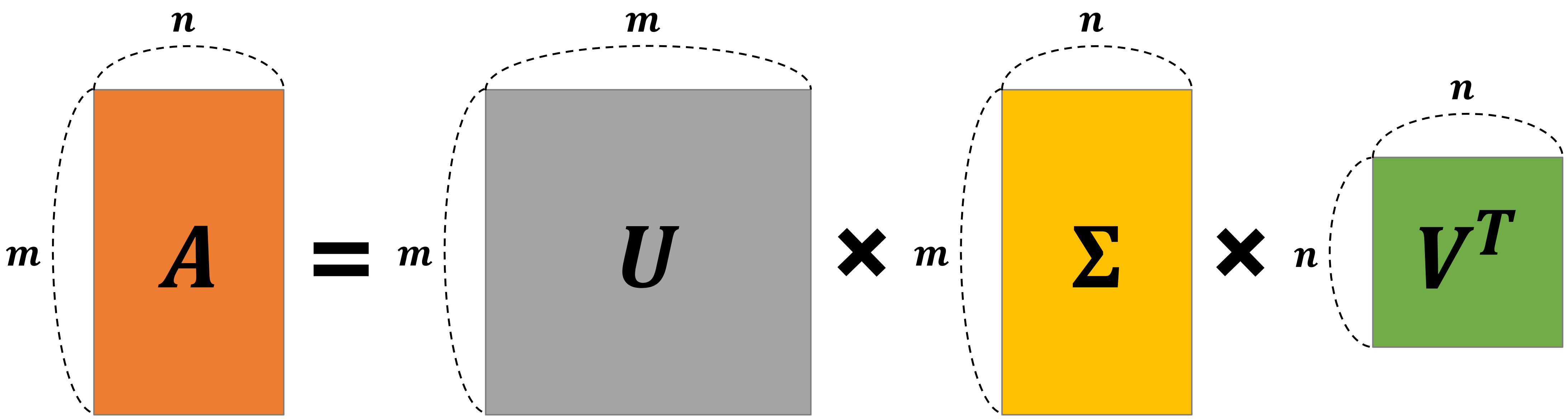
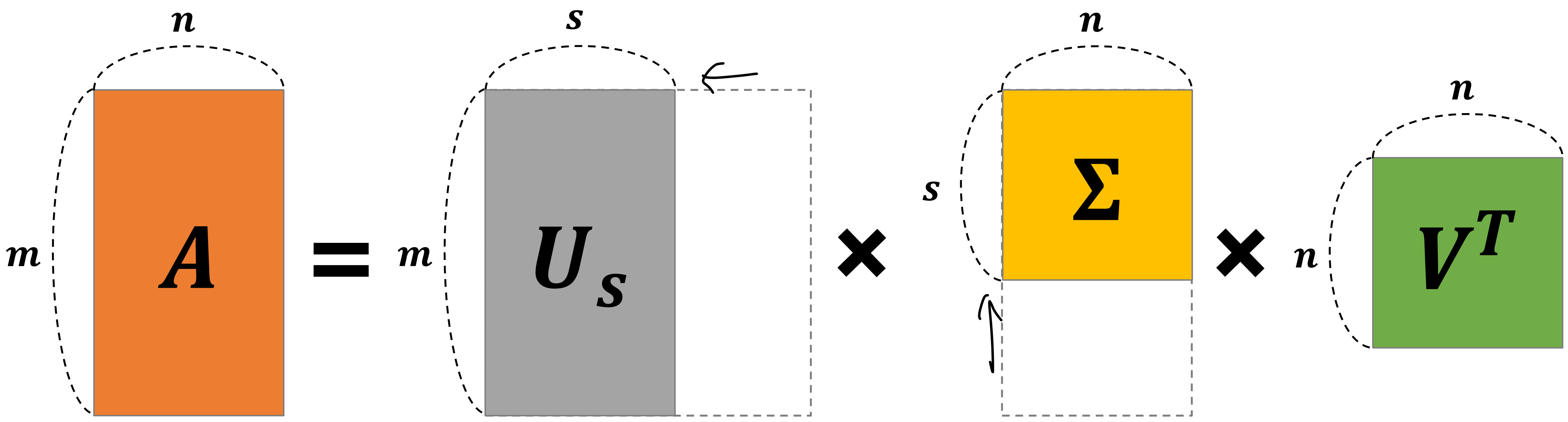


Fig 1. Full SVD

Reduced SVD: Thin SVD

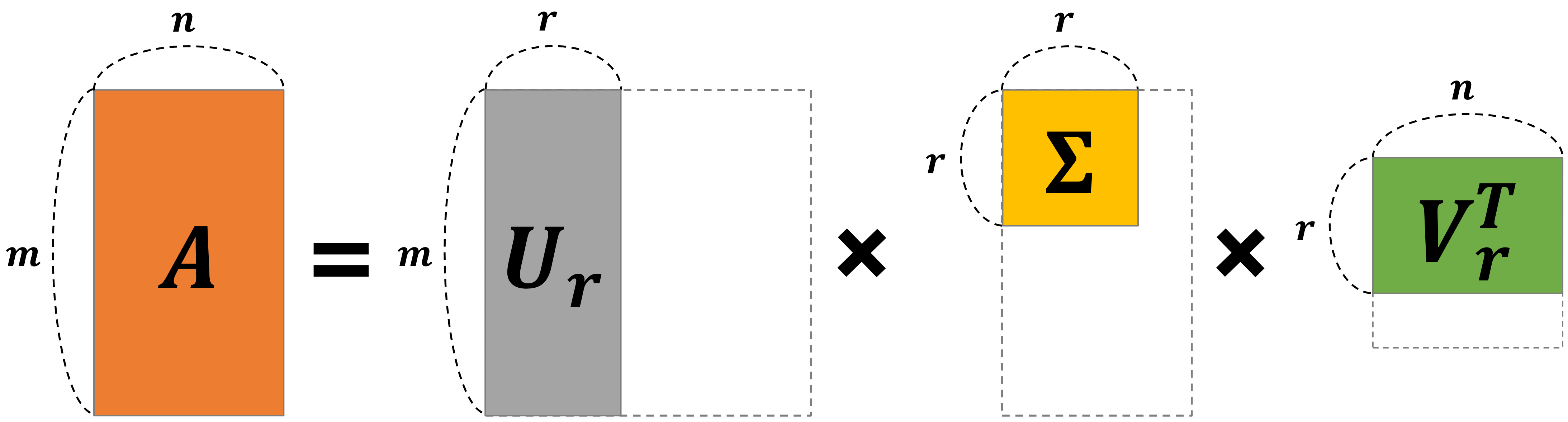
- **Assume**
  - $s = n$
- **Form**
  - In $\boldsymbol{\Sigma}$
    - Non-diagonal part consisting of 0 is removed.
  - In $\boldsymbol{U}$
    - Corresponding column vectors are removed.



Thin SVD

Reduced SVD: Compact SVD

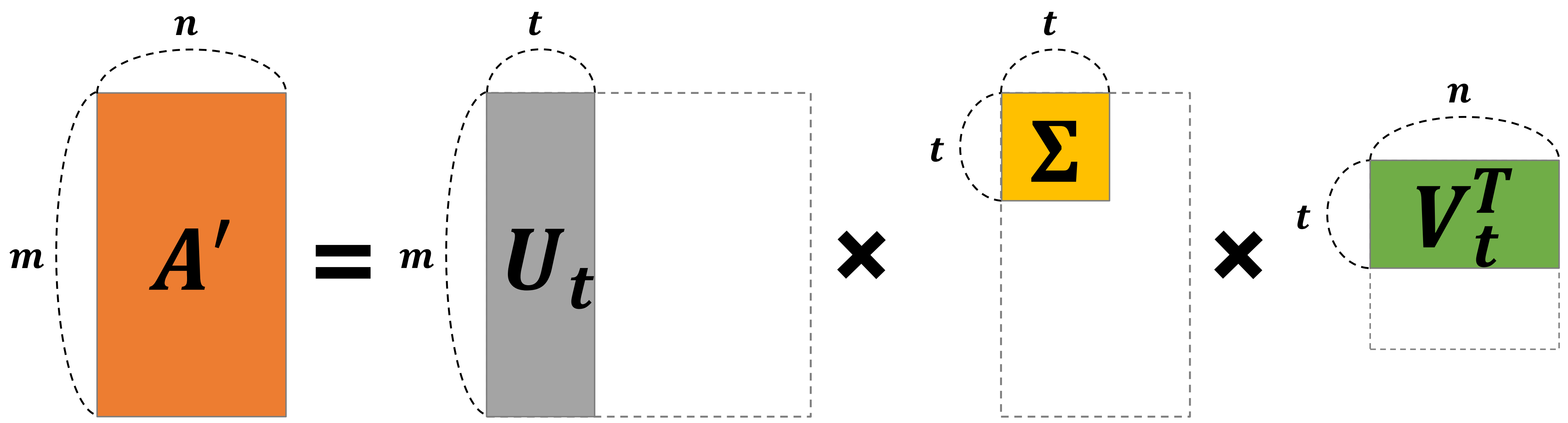
- **Assume**
  - r is number of non-zero values, among $s = n$ singular values.
- **Form**
  - In $\boldsymbol{\Sigma}$
    - Not only off-diagonal elements but also singular values of 0 are removed.
- **It can be easily confirmed.**
  - Calculated $\boldsymbol{A}$ is same matrix as original $\boldsymbol{A}$.



Compact SVD

Reduced SVD: Truncated SVD

- **Assume**
  - $t < r$
    - r is number of non-zero values, among $s = n$ singular values.
- **Form**
  - In $\mathbf{\Sigma}$
    - Even singular values other than 0 are removed.
- **Approximation matrix A' for A is produced.**



Truncated SVD

Reduced SVD: Truncated SVD

Rewrite formula for SVD below.

- $\overrightarrow{u_1}\,\overrightarrow{v_1}^T$: $m \times n$ matrix.
  - Component values in matrix $\overrightarrow{u_1}\,\overrightarrow{v_1}$ have values between $\boxed{\text{-1 and 1}}$.
    - Because $\overrightarrow{u_1}$ and $\overrightarrow{v_1}$ are normalized vector.
- $\sigma_1\overrightarrow{u_1}\,\overrightarrow{v_1}^T$: size of this matrix is determined by σ_1.

Decompose a random matrix A into several matrices with same size of A matrix.

- By using **SVD**!
- Size of element value of each decomposed matrix is determined by $\boxed{\sigma}$.

$$\boldsymbol{A = U\Sigma V^T}$$

$$= \begin{bmatrix} | & | & & | \\ \overrightarrow{\boldsymbol{u_1}} & \overrightarrow{\boldsymbol{u_2}} & \cdots & \overrightarrow{\boldsymbol{u_m}} \\ | & | & & | \end{bmatrix} \begin{bmatrix} \sigma_1 & & & & 0 \\ & \sigma_2 & & & 0 \\ & & \ddots & & 0 \\ & & & \sigma_m & 0 \end{bmatrix} \begin{bmatrix} - & \overrightarrow{\boldsymbol{v_1}}^T & - \\ - & \overrightarrow{\boldsymbol{v_2}}^T & - \\ - & \vdots & - \\ - & \overrightarrow{\boldsymbol{v_n}}^T & - \end{bmatrix}$$

$$= \sigma_1\overrightarrow{\boldsymbol{u_1}\boldsymbol{v_1}}^T + \sigma_2\overrightarrow{\boldsymbol{u_2}\boldsymbol{v_2}}^T + \cdots \sigma_m\overrightarrow{\boldsymbol{u_m}\boldsymbol{v_m}}^T$$

Formula of SVD

Matrix A'

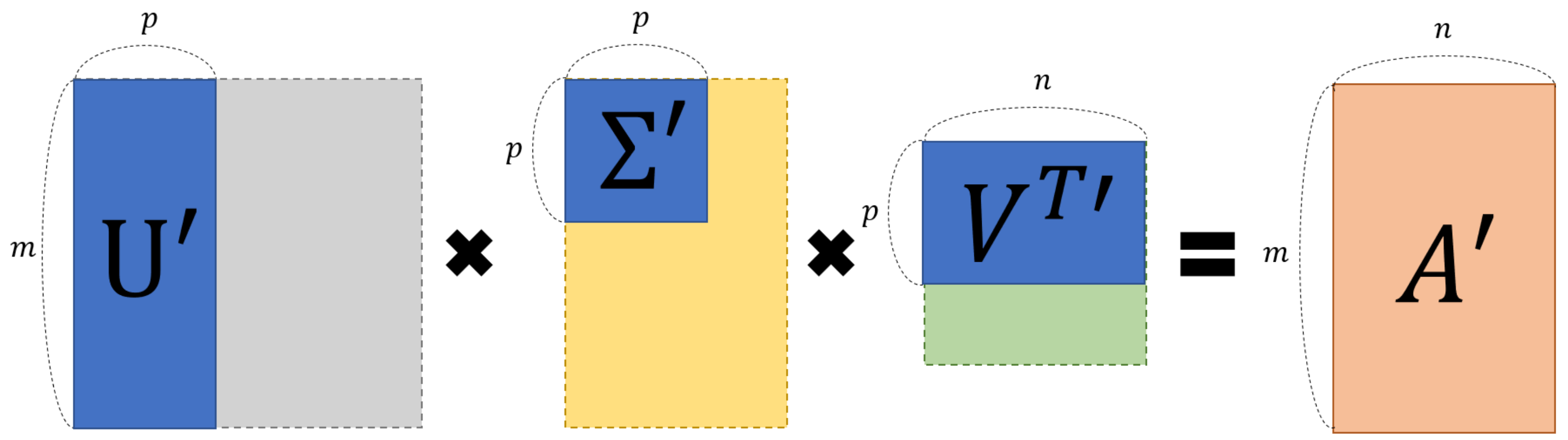
- **Matrix approximated by truncated SVD.**
- **Rank t matrix**
  - Minimize matrix norm $||A - A'||$
- **Application**
  - compression
  - removal

Application of SVD

Use of SVD

- **SVD shines its applicability in the process of** recombining **decomposed matrix.**
  - Rather than in the decomposition process.

- **Decomposed matrix A can be partially restored.**
  - Using only p singular values.
  - Amount of information A is determined by Size of singular value.
    - Sufficient useful information can be maintained even with several large **singular values**.



Process of partially restoring appropriate A' using only part of the U.

Code Exercise of Partial Restoration Process Through Photos

Code Exercise (14_03)

- Run the code with given image 'lena_std.tif'.

```
% Clear workspace, command window, and close all figures
clc; clear; close all;

% Use image
img = double(rgb2gray(imread('lena_std.tif')));

[U, S, V] = svd(img);
figure;

for i = 1:size(S,1)
   imagesc(U(:, 1:i)*S(1:i, 1:i)*V(:, 1:i)');
   colormap('gray')
   name = ['layers added upto ', num2str(i)];
   title(name);
   if i<30
      pause(0.5)
   elseif 3<=i<100
      pause(0.1)
   else
      pause(0.01)
   end
end
```

MATLAB code of partial restoration process

Partial Restoration Process Through Photos

- **Use only important information**
  - ▶ Size of the photo will be reduced.
  - ▶ But still be able to **preserve** the content that the photo wants to show.

Can apply SVD to partially restore the photo

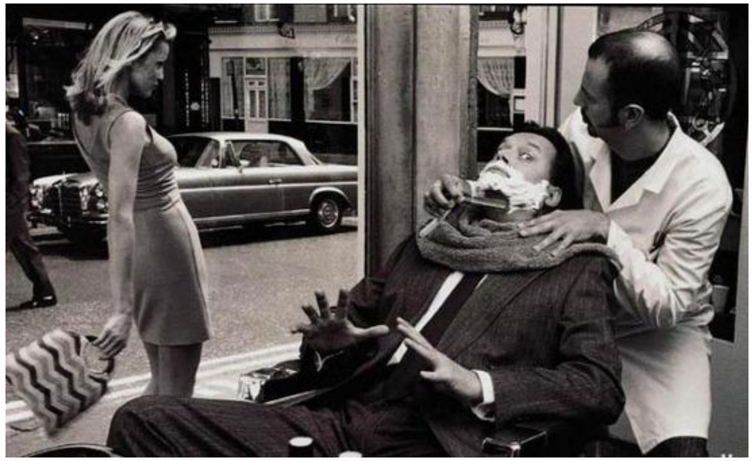
Example of Data Compression

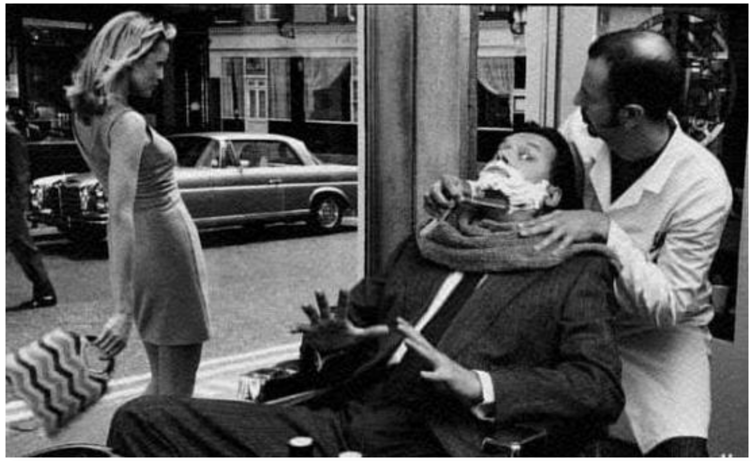
- **Let's compress 600×367 image with SVD.**
  - 1. Take 600×367 matrix $\boldsymbol{A}$ with pixel values of this image as element values.
  - 2. Perform truncated SVD.
    - Rotate original image by 90 degrees.
      - To create typical $m > n$ form
    - Apply SVD.
  - 3. Obtain approximation matrix $\boldsymbol{A}'$.
  - 4. Display it as image again.

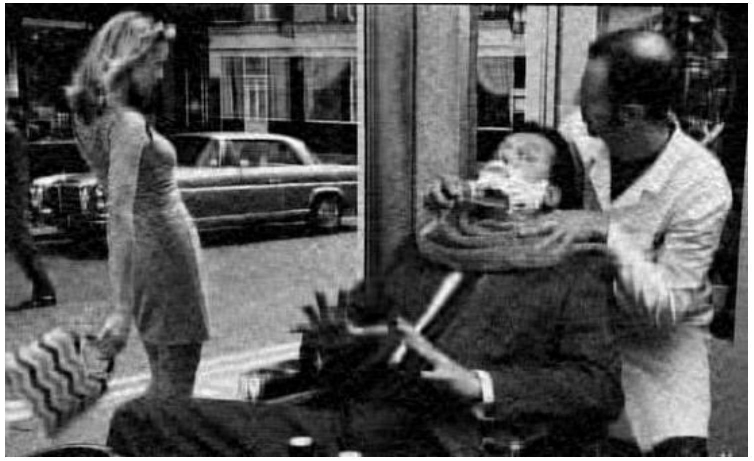
One of Wallis' "dressed to kill" advertising images

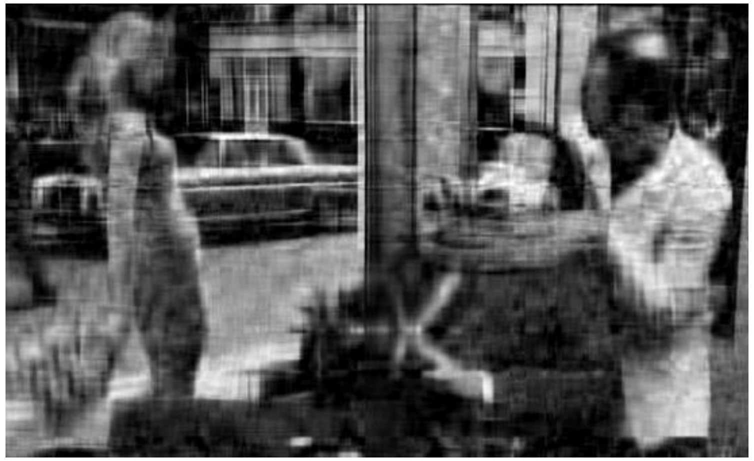
Display Result of Data Compression as Image

Approximation with t singular values


Original image


Approximation with 100 singular values


Approximation with 50 singular values

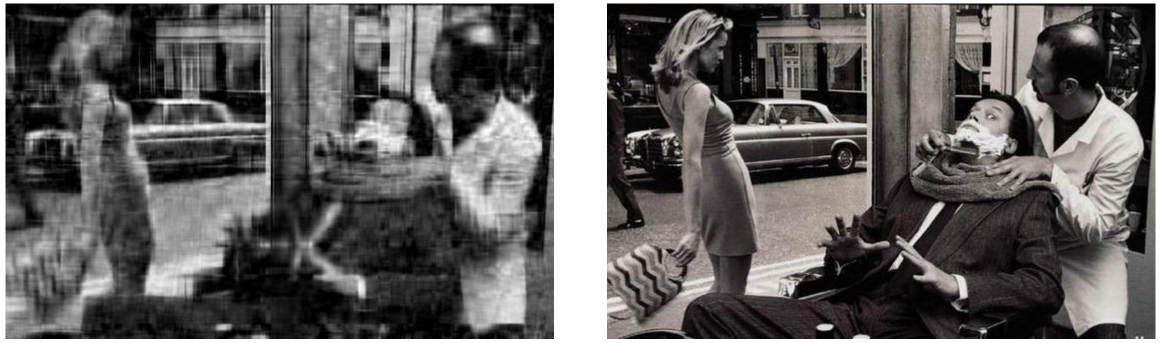

Approximation with 20 singular values

Numerical Result of Data Compression

- **Memory**
  - In original image
    - $220{,}200$
      - $367 * 600$
  - In case of $t = 20$
    - $19{,}360$
      - $600 * 20(\boldsymbol{U}) + 20(\boldsymbol{\Sigma}) + 20 * 367(\boldsymbol{V})$

- **Data compression ratio**
  - 8.8%
    - $19{,}360/220{,}200 * 100$

- **Looking at image quality**
  - It is not good compression method.


Data compression with 8.8% can represent original image well

- **But…,**
  - Data approximation through truncated SVD captures core of original data well.

- **We will practice data compression with SVD in next week!**

SVD and Pseudo Inverse

Linear System $Ax = b$

- **If inverse matrix of A exists**
  - Solution to this system can be easily found as $\boxed{x = A^{-1}b}$.
  - However, in most real problem
    - There are very few cases where **inverse matrix exists**.
  - In such cases
    - $\boxed{\text{pseudo inverse}}$ can be used!

- **If inverse matrix of A doesn't exist**
  - Solution to this system can be calculated as $\boldsymbol{x} = \boldsymbol{A}^{+}\boldsymbol{b}$.
    - $\boldsymbol{A}^{+}$ is $\boxed{\text{pseudo inverse.}}$ of $\boldsymbol{A}$.
  - $\boldsymbol{x}$ becomes solution.
    - Minimizes $||\boldsymbol{Ax} - \boldsymbol{b}||$.

- **Finding solution using pseudo inverse has same meaning.**
  - As the least square method.

Pseudo Inverse

- **It can be defined for random $m \times n$ matrices.**
  - Originally, inverse matrix was defined only for square matrices.

$$
\begin{aligned}
A^+ &= (A^{\overset{T}{*}}A)^{-1}A^{\overset{T}{*}} \\
&= (V\Sigma \cancel{U^{\overset{T}{*}}U}\Sigma V^{\overset{T}{*}})^{-1}V\Sigma U^{\overset{T}{*}} \\
&= (V\Sigma^2 V^{\overset{T}{*}})^{-1}V\Sigma U^{\overset{T}{*}} \\
&= (V^*)^{-1}\Sigma^{-2}V^{-1}V\Sigma U^* \\
&= V\Sigma^{-2}\Sigma U^* \\
&= V\Sigma^{-1}U^*
\end{aligned}
$$

- **SVD**
  - One of the most powerful (and stable) methods
    - To compute **pseudo inverse**.

- **If SVD of matrix A is Eq 1.**
  - Pseudo inverse of A is Eq 2.
    - Σ^+ is matrix obtained by taking **reciprocal** of $\boxed{\text{non-zero singular}}$ values in original Σ and then **transposing** it.

$$\boxed{A = U\Sigma V^T} \checkmark$$

Eq 1. SVD of matrix A

$$A^+ = V\Sigma^+U^T$$

Eq 2. Pseudo inverse of matrix A

Note about Pseudo Inverse

- **Order of U and V changes.**
- **Σ also changes from $m \times n$ to $\boxed{n \times m.}$ matrix.**
- **If 0 is included among singular values**
  - Only non-zero singular values are reciprocated.
  - Original 0 is left as 0 in Σ^+.
    - Reciprocal is only for non-zero values.

$$\overset{\Sigma\ m\times n}{} \xrightarrow{\quad} \overset{\Sigma^{+}\ n\times m.}{}$$

$$A = U \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_s & \\ & & & 0 \end{pmatrix} V^T \rightarrow A^+ = V \begin{pmatrix} 1/\sigma_1 & & & \\ & \ddots & & \\ & & 1/\sigma_s & 0 \end{pmatrix} U^T$$

SVD of matrix $\boldsymbol{A}$ to pseudo inverse of matrix $\boldsymbol{A}$

How to Find Identity Matrix

If all singular values are positive

- $m \geq n$
  - $\boldsymbol{A}^+\boldsymbol{A}$ becomes $n \times n$ identity matrix.
- $m \leq n$
  - $\boldsymbol{A}\boldsymbol{A}^+$ becomes $m \times m$ identity matrix.

If singular values of 0 is included

- No matter what order you multiply it, **you will not get □ matrix**.

$$\boldsymbol{A}^+\boldsymbol{A} = (\boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T)(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = \boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{\Sigma}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{E}_n \ \ (m \geq n)$$

$$\boldsymbol{A}\boldsymbol{A}^+ = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T)(\boldsymbol{V}\boldsymbol{\Sigma}^+\boldsymbol{U}^T) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^+\boldsymbol{U}^T = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{E}_m \ \ (m \leq n)$$

How to find identity matrix when all singular values are positive

Apply Pseudo Inverse to System of Linear Equation

- **$m \geq n$ (usually)**
  - Number of equation (data) is greater than number of unknowns..
  - Multiply front of both sides of $Ax = b$ by A^+.
  - Find x in form $A^+Ax = A^+b \Rightarrow x = A^+b$.

- **$m < n$**
  - Number of unknowns is greater than number of equations (data).
  - Solution is not uniquely determined.
  - This is like problem of determining a plane using two points.
  - Pseudo inverse can be obtained.
    - But since it must be used in form of multiplication after A as in Eq 1..
    - It is not valid for problems of form $Ax = b$.

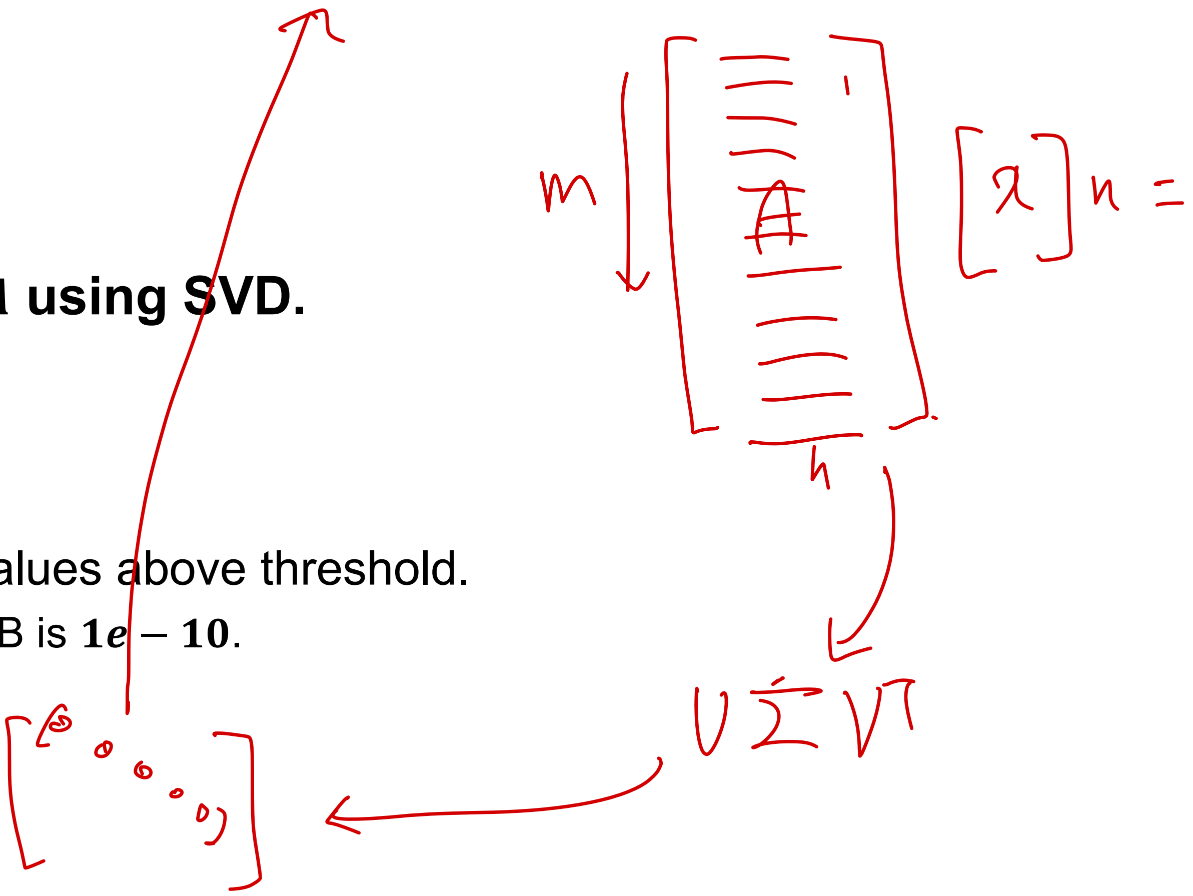
$$AA^+ = (U\Sigma V^T)(V\Sigma^+U^T) = U\Sigma\Sigma^+U^T = UU^T = E_m \ \ (m < n)$$

How to find identity matrix when m<n

When Singular Value is Close to 0

- **Even if singular value is not 0, if it is very close to 0.**
  - It is common to treat it as $\boxed{\text{noise}}$.
    - Change it to 0.
    - Then obtain **pseudo inverse**.

- **Find pseudo inverse of matrix A using SVD.**
  - Find singular values.
  - Change singular values to 0.
    - Very close to 0
  - Take reciprocal of only singular values above threshold.
    - Default threshold used in MATLAB is $\mathbf{1}\boldsymbol{e}-\mathbf{10}$.
  - Find pseudo inverse.



Threshold of Singular Value

- **It is called $\boxed{\text{Tolerance}}$ of SVD.**
- **It is closely related to truncated SVD.**
- **Resulting pseudo inverse and linear system solutions may be changed.**
  - How much of value is considered noise.
  - How tolerance value is given.

Summary

Summary

- SVD can be decomposed into two orthogonal matrices and one diagonal matrix.
  - $A = U\Sigma V^T$
- SVD can be decomposed even if it is not a square matrix.
- SVD is widely used in data compression and noise removal process.
- SVD is the most powerful way to calculate pseudo inverse.

Code Exercises

Properties of a Symmetric Matrix

- You learned that for a symmetric matrix, the singular values and the eigenvalues are the same. How about the singular vectors and eigenvectors? Use MATLAB to answer this question using a random 5×5 $A^T A$ matrix. Next, try it again using the additive method for creating a symmetric matrix $(A^T + A)$. Pay attention to the signs of the eigenvalues of $A^T + A$.

REF

0 1 4 5

0 0 P 2 3 4

RREF

```
% create a symmetric matrix
A = randn(5,5);
A = A' * A;

% eigendecomposition
[evecs, evals] = eig(A)
evals = diag(evals)

% sorting them helps the comparison!
[evals, order] = sort(evals, 'descend')
evecs = evecs(:, order)

% SVD
[U, S, V] = svd(A)
S = diag(S);

% compare the eigenvalues and singular values
disp('Eigenvalues and singular values:')
disp([evals, s])

% now compare the left and right singular vectors
disp('Left-Right singular vectors (should be zeros)')
disp(round(U - V, 10)) % remember to compare V not V^T!

% then compare singular vectors with eigenvectors
disp('Singular vectors - eigenvectors (should be zeros)')
disp(round(U - evecs, 10)) % subtract and
disp(' ')
disp(round(U + evecs, 10)) % add for sign indeterminancy
```

Code sample

Economy SVD

- MATLAB can optionally return the “economy” SVD, which means that the singular vectors matrices are truncated at the smaller of M or N. Consult the docstring to figure out how to do this. Confirm with tall and wide matrices. Note that you would typically want to return the full matrices, economy SVD is mainly used for really large matrices and/or really limited computational power.

[illegible]

```
% sizes (try tall and wide)
m = 10;
n = 4;

% random matrix and its economy (aka reduced) SVD
A = ;
[U, S, V] = ;

% print sizes
disp(['Size of A:  [', num2str(size(A, 1)), ', ', num2str(size(A, 2)), ']']);
disp(['Size of U:  [', num2str(size(U, 1)), ', ', num2str(size(U, 2)), ']']);
disp(['Size of V'': [', num2str(size(V, 1)), ', ', num2str(size(V, 2)), ']']);
```

Code sample

Properties of Orthogonal Matrix

- One of the important features of an orthogonal matrix (such as the left and right singular vectors matrices) is that they rotate, but do not scale, a vector. This means that the magnitude of a vector is preserved after multiplication by an orthogonal matrix. Prove that $\|Uw\| = \|w\|$. Then demonstrate this empirically in MATLAB by using a singular vectors matrix from the SVD of a random matrix and a random vector.

```
% The proof that |Uw| = |w| comes from expanding the vector magnitude to the dot product:
% |Uw|^2 = (Uw)'(Uw) = w'U'U'w = w'Iw = w'w = |w|^2

% random matrix size -> 5,5
% empirical demonstration:
[U, S, V] = ;
w = randn(5, 1);

% print out the norms
disp(norm(U * w));
disp(norm(w));
```

Code sample

THANK YOU FOR YOUR ATTENTION