**CMDA 3605 Jung H Choi**

# Term Project Part II

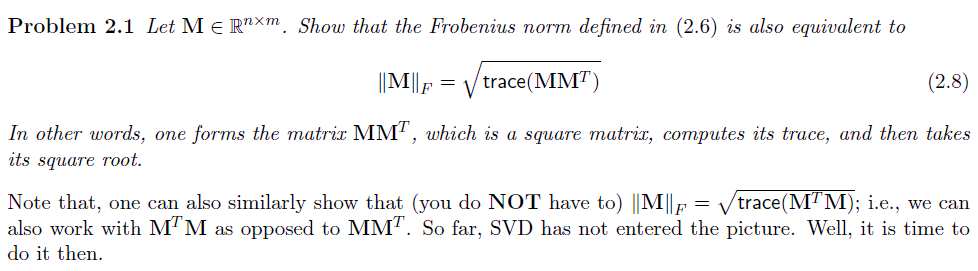
SVD and Its Application to Dynamical Systems

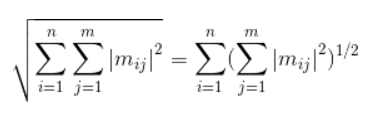
## Abstract

The goal of this project is to employ your Difference and Differential Equations, Linear Algebra and Matlab skills for applications in Dynamical Systems. At the end, these skills will be combined to construct dynamical systems directly from data (without access to original mathematical equations) and to reduce the number of equations in a dynamical system. The main tool in achieving this goal would be an important matrix decomposition, called the Singular Value Decomposition (SVD).

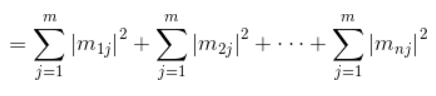
# PART II: The SVD and Learning Discrete-time Dynamical Systems from Data

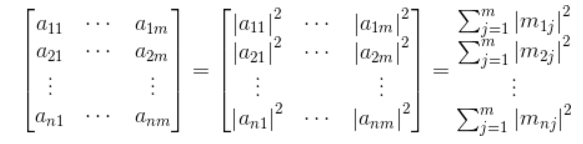
In Part I of this term project, we have studied SVD and its power in producing optimal low-rank approximations. Given a matrix **A** (representing an image, or a database of images), we used its SVD to compute its low-rank approximation. In this section, we will use SVD to learn a discrete-time dynamical system from its time-domain history. Thus, we will not be given the **A** matrix; rather we will be given time-domain simulation data from a dynamical system and use SVD to construct the underlying transition matrix **A**.





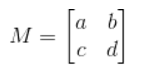
Let’s look at the double sum inside the square root and simplify them into less complex form.

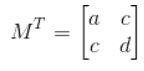




Each element is squared and took summation of the row by row. As you can see, Frobenius norms are very similar to the usual Euclidean 2-norms of the matrix.

Let’s assume following matrix **M** ∈ \mathbb{R}n x mare,

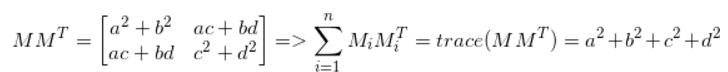




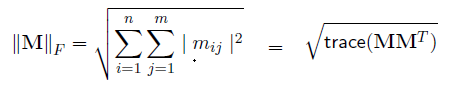
When we apply the matrix M, we get the following answers.



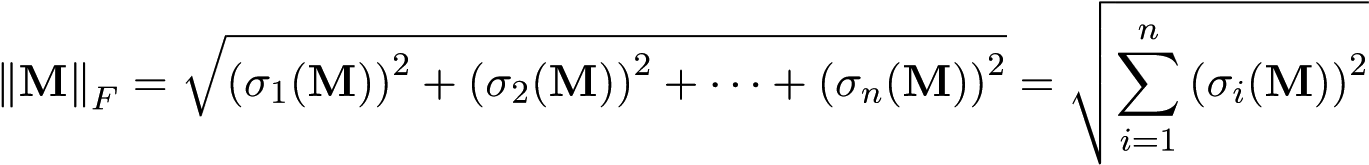
Whereas **M \* MT**and **trace(M \* MT)** by applying the trace equation in (2.7), they are both equal to each other.



Which means the **Frobenius norm** formula in (2.6) can also be defined into the formula in (2.8)

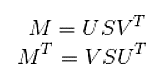


**Problem 2.2** *Let* **M** ∈ R*n*×*m. For simplicity assume that n* ≤ *m (this is not necessary, we can do it for n > m as well). Show that the Frobenius norm formula in* (2.8) *is equivalent to*

*,* (2.9)

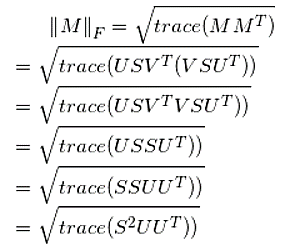
*where σi*(**M**) *is the i*th *singular value of the matrix* **M***.*

**Hint:** *Let* **M** = **USV***T be the SVD of* **M** *and plug this into* (2.8)*. You may also use the fact that* trace(**BC**) = trace(**CB**) *where* **B** *and* **C** *are two matrices of appropriate sizes.*

Using the **M** given, we get the following **Left singular vectors** which are eigenvectors of **MMT**. With entries *σi,* which implies that UUT = UTU = I. This gives:



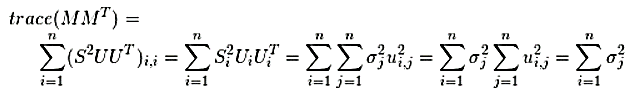
Note: S = ST since the singular values are on the diagonal



Let’s ignore the square root for now. By Cyclic property of trace, we have the following Frobenius norm:

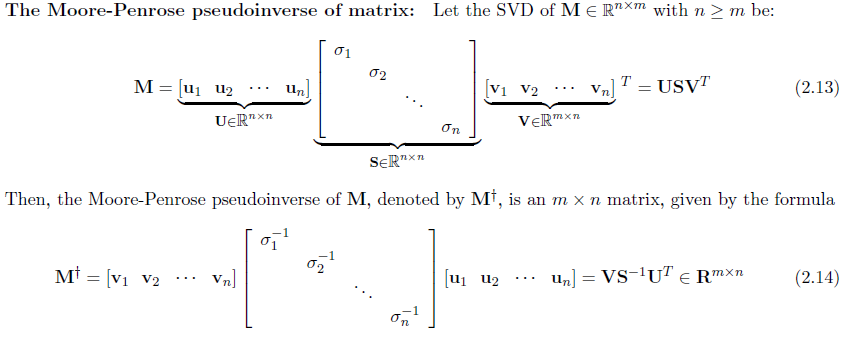


We could also write it as,



Since *σi*(**M**) is the ith singular value of the matrix **M**, Sum of all the nonzero singular values are added. Then Frobenius norm takes sqrt of the whole as following:

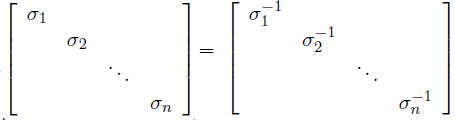




Note that in this definition, we have assumed that all the singular values *σi* are non-zero. The definition changes if some of the singular values are zero; only the nonzero ones are inverted. This will be further investigated in Part 3 of the Term Project.

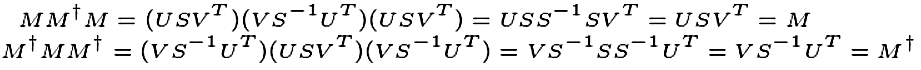
**Problem 2.3** *Show that if* **M** *is an invertible (and thus a square matrix), the pseduoinverse* **M**† *becomes the regular matrix inverse, i.e.,* **M**† = **M**−1 *if* **M** *is an invertible matrix. For more details on the MoorePenrose pseduoinverse, see [1].*

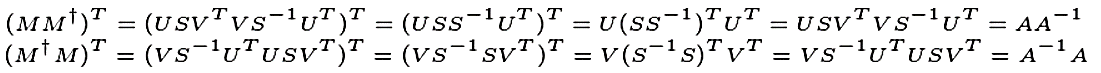
By the definition of inverse, matrix M for an n x n square matrix A is invertible for which AA-1 = *I* = A-1A. If we have r = n = m; the matrix M has full rank then S† = S-1:



The product of S and S† is a square matrix whose first r diagonal entries are 1 and whose other entries are 0.[1]

Now Let’s check the conditions that stratifies the *Moore-Penrose pseudo-Inverse theorem*.by plugging in the formulas given in (2.13) and (2.14).



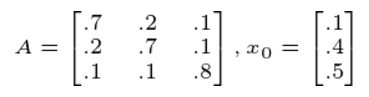


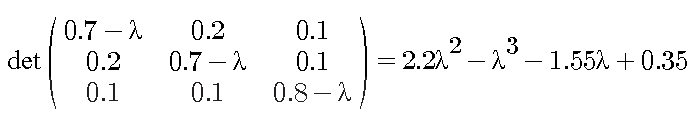
Thus, checking the inverse condition we can then determine invertibility of the matrix M by checking the proceeding discussion.

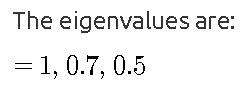
**Problem 2.4** *We will apply Algorithm 2.1 to the Voting Preference Model of Problem 3 from Homework 4. Of course, in this case we know the underlying transition matrix* **A***. We will use this to check our results. In practice one would not have access to* **A***; this will come in Part 3 of Term Project.*

1. *We will first construct the data in* (2.10)*. Note that our algorithm will only use this data and not the* **A** *matrix. Simulate the dynamical system in Problem 3 of Homework 4 up to t* = 20 *years using the initial condition* **x**(0) = [0*.*1 0*.*4 0*.*5]*T to construct measurement data* (2.10)*.*

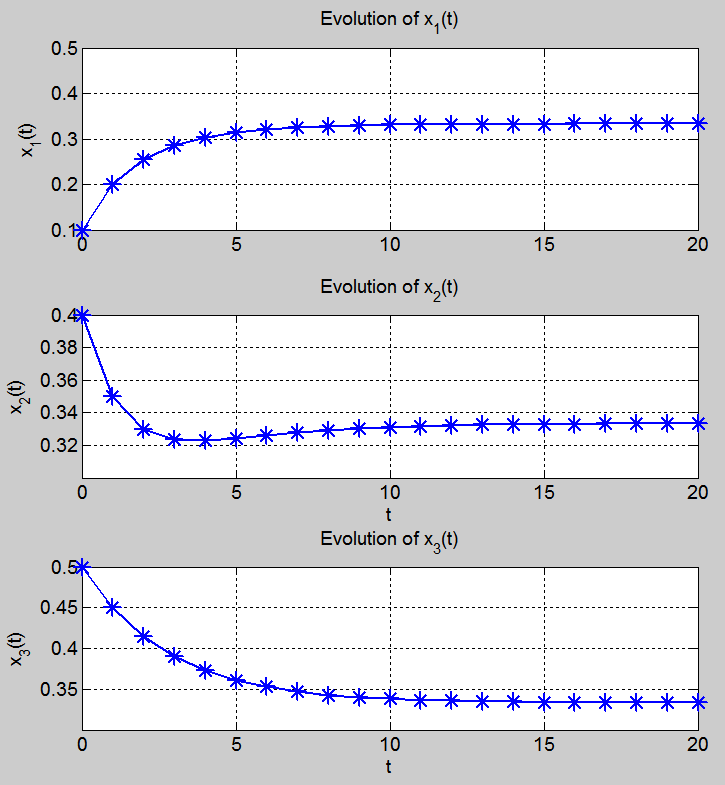
We have the following given:



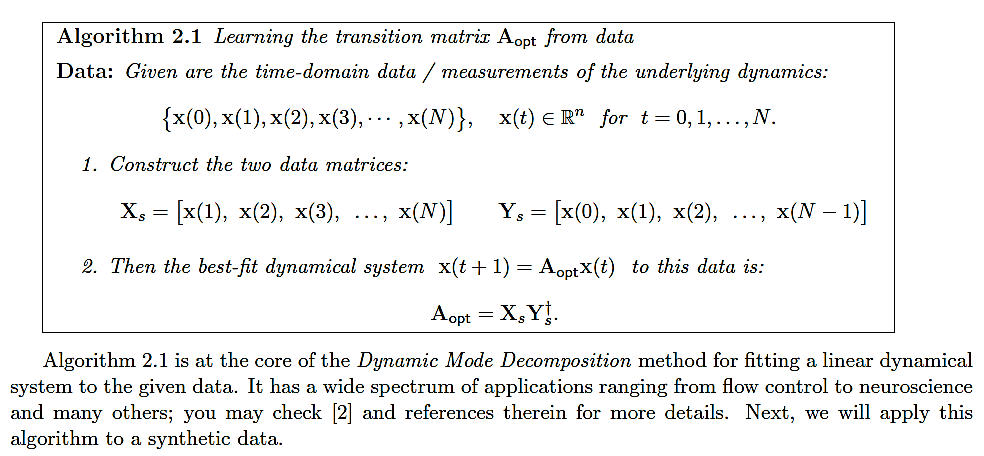




Using MATLAB code(see Appendix B), we can approximate the system for 20 years with initial condition given **x**(0) = [0*.*1 0*.*4 0*.*5]*T*.



1. *Apply Algorithm 2.1 to this data. Compare* **A***opt you obtained to the true* **A***. Did you exactly recover the true dynamics? You should be able to for this problem. Consider what you achieved here. Somebody handed you the yearly voting preferences, and just from this data, you have constructed the underlying dynamics.*



**Appendix B**

% illustrate the stability of x(t+1) = A x(t)

% Modified version of Dr. Gugercin's check\_stability.m

set(0, 'defaultaxesfontsize',14,'defaultaxeslinewidth',1.0,...

'defaultlinelinewidth',2.0,'defaultpatchlinewidth',1.0,...

'defaulttextfontsize',18,'DefaultLineMarkerSize',14)

x0 = [.1;.4;0.5];

l1 = 1; v1 = [1 1 1]'; % first eigenpair

l2 = 0.7; v2 = [-1 -1 2]'; % second eigenpair

l3 = 0.5; v3= [-1 1 0]';

V = [v1 v2 v3]; % eigenvector matrix

L = diag([l1 l2 l3]); % eigenvalue matrix

% A = V\*(L/V);

A = [0.7, .2, .1; 0.2, 0.7, .1; .1, .1, .8];

% b = [3; 9];

% Simulate x(t+1) = A x(t), x(0) = x0 and store

x = x0;

for i=1:20

x(:,i+1) = A\*x(:,i);

end

% Plot the solution

subplot(3,1,1)

plot(0:i,x(1,:),'-\*')

ylabel('x\_1(t)')

title('Evolution of x\_1(t)')

grid on

subplot(3,1,2)

plot(0:i,x(2,:),'-\*')

ylabel('x\_2(t)')

xlabel('t')

title('Evolution of x\_2(t)')

grid on

subplot(3,1,3)

plot(0:i,x(3,:),'-\*')

ylabel('x\_3(t)')

xlabel('t')

title('Evolution of x\_3(t)')

grid on

**References**

1. G. Golub and C. van Loan, *Matrix Comptutations*, 3rd Edition, JHU Press, 2012.
2. https://ocw.mit.edu/courses/mathematics/18-06sc-linear-algebra-fall-2011/positive-definite-matrices-and-applications/left-and-right-inverses-pseudoinverse/MIT18\_06SCF11\_Ses3.8sum.pdf
3. http://mathworld.wolfram.com/InvertibleMatrixTheorem.html
4. CMDA3605\_TermProject\_Part2.pdf; Serkan Gugercin, 2017