EIGENVALUE REALISATION ALGOIRTHM (ERA) ON A MISO (MULTIPLE INPUT SINGLE OUTPUT) SYSTEM

North South University

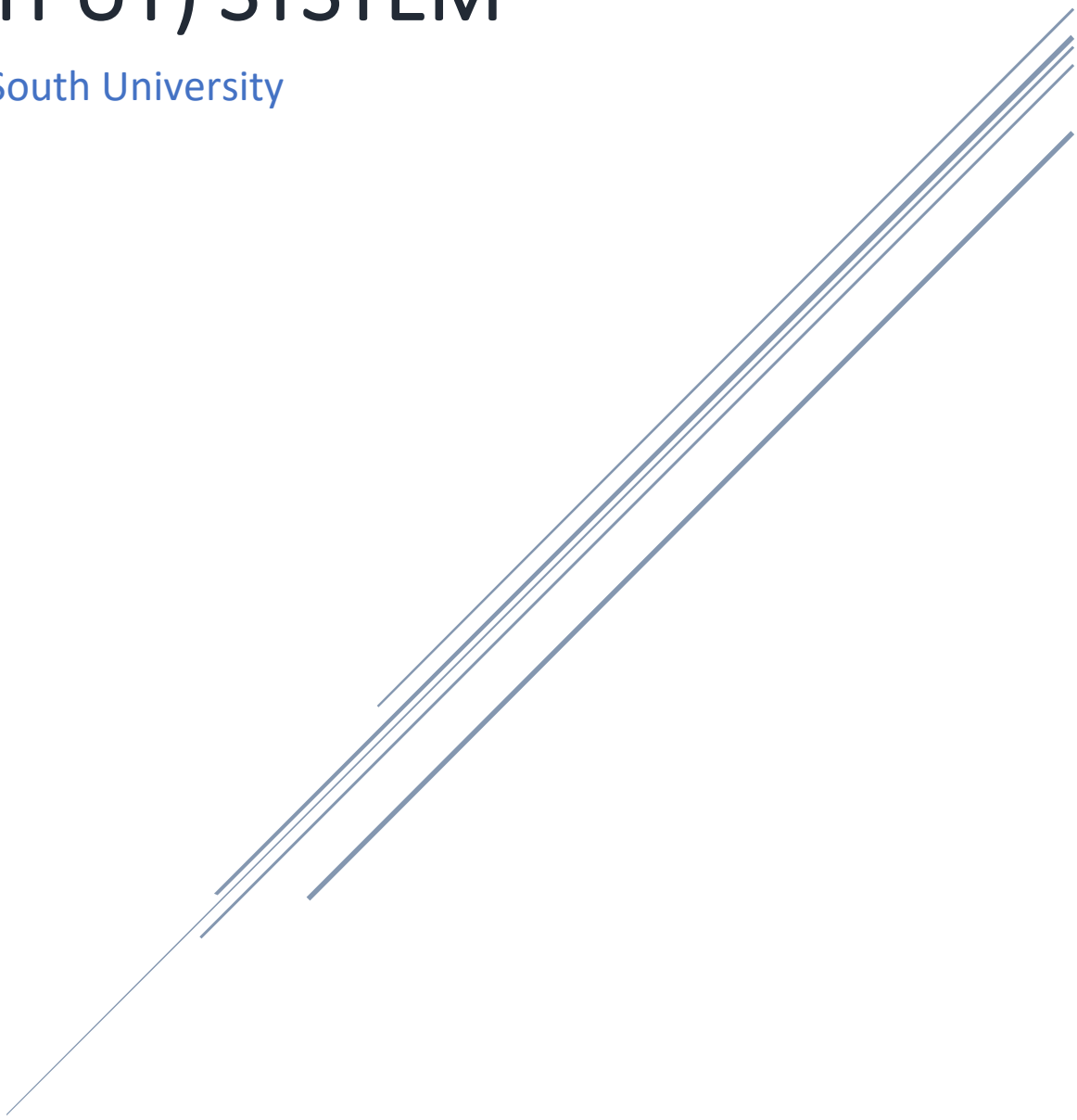

Table of Contents

1 INTRODUCTION

Eigensystem Realization Algorithm (ERA) is a tool that can produce a reduced order model (ROM) from just input-output data of a given system. ERA creates the ROM while keeping the number of internal states to a minimum level. This was first implemented by Juang and Pappa (1984) to analyze the vibration of aerospace structures from impulse response. We reviewed ERA and tested it on single input single output (SISO) system as well as on multiple input single output (MISO) system. ERA prediction agreed with the actual data. Unlike other model reduction techniques (Balanced truncation, balanced proper orthogonal decomposition), ERA works just as fine without the need of the adjoint system, that makes ERA a promising, completely data-driven, thrifty model reduction method. Currently, our aim is to update this algorithm by a preprocessor step to handle huge amount of data for a given system (e.g. multiple input multiple output system) and evaluate ERA's performance for an arbitrary input.

Historically several device recognition techniques [150, 178] have been constructed with few measurements for high-structures. ERA has been developed for example to model vibrations in aerospace systems such as the Hubble Space Telescope and the International Space Station [151]. Ho and Kalman's ERA is based on the minimal realization theory [132. In contrast, DMD was developed for nonlinear systems with full-state measurements of the high dimensions. Nonetheless, the ERA and DMD algorithms are very similar, and we can demonstrate that DMD is a special case of ERA [182, 290] in some conditions. Thanks to its relative simplicity of execution, ERA is commonly used in modeling and control. Recently,[184] it has also been shown that ERA yields models equal to BPOD[299, 233] which is useful for achieving approximate balanced truncation[201] on high-dimensional systems.

2 METHODOLOGY

ERA is an algorithm that is able to derive a state space system without even thinking about the states in the system. Which is to say that a person does not need to find the states or transfer function in order to design a control system. It works by considering the input and output of the system. In this regard, it is similar to machine learning. It can, hence, be called a data driven modelling technique. Nowadays, data is what is making the world spin. Hence this algorithm is crucial in leveraging the data available.

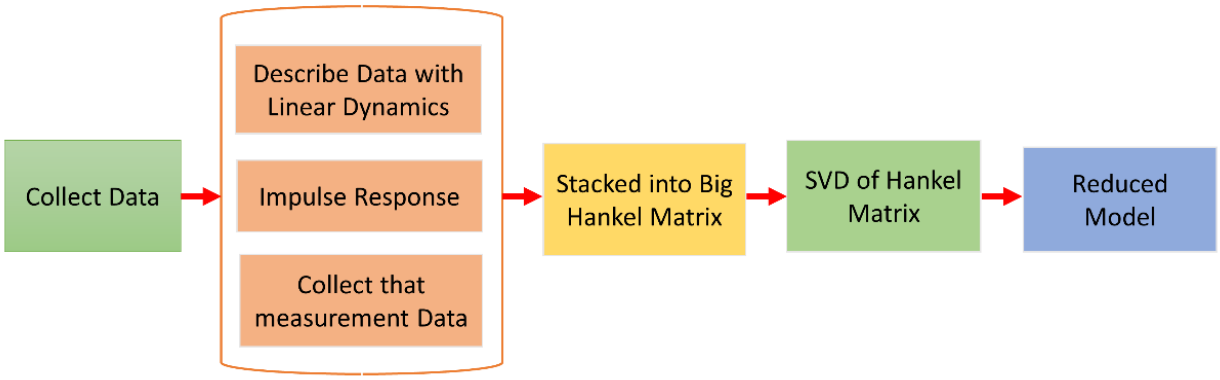

Figure 1 : Overall System Block Diagram

In **Figure 1**, we can see the summary of the whole system. We first have to collect data from the system. The data collected has to be impulse response for each of the inputs. The outputs will be, then, stacked into Hankel Matrices. The Hankel Matrices will have the information for us to extract a model. We will than perform Singular Value Decomposition (SVD) on the Hankel Matrices. The decomposition will allow us to extract the system properties. These system properties will let us create a state space model which is often in a reduced form.

In this section, we will describe in detail the process as shown in **Figure 1.** The modelling of the system and how the algorithm in general is derived will be explained in detail.

2.1 CONTROL SYSTEM MODEL

When modelling a linear system with state space representation, we can write the invoke the following relations:

$$x_{k+1} = Ax_k + Bu_k$$

$$y_k = Cx_k + Du_k$$

where,

x is the state vector,

y is the output vector,

u is the input vector,

A is the state matrix,

B is the input matrix,

C is the output matrix,

D is the feedthrough matrix.

In most cases the D matrix is often assumed to be zero.

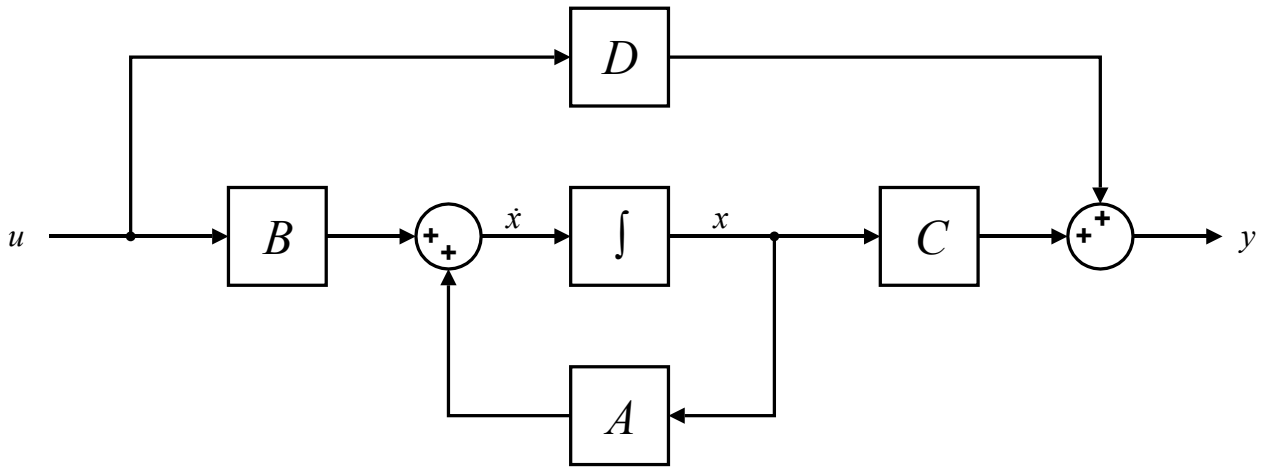

Figure 2: Block Diagram Representation of State Space Representation

Figure 2, the modelling is shown as a block diagram to visualize the system in work.

To construct the ERA model, we need to first extract the impulse response. To get the impulse response we need to set these initial conditions:

Assume an impulse force, at t = 0, and 0 Initial Conditions,

$$U_k: \begin{bmatrix} I, & k = 0 \\ 0, & k > 0 \end{bmatrix},$$

$$and\ x_0 = 0$$

By using the state space model and the initial conditions, we will be able to derive the values of x and y in terms of A, B, C and D.

for x:

$$\underline{at\ k = 0, u_0 = 1,}$$

$$x_1 = Ax_0 + Bu_0 = B\ ,$$

$$\underline{at\ k = 1, u_1 = 0,}$$

$$x_2 = Ax_1 + Bu_1 = AB,$$

$$\underline{at\ k = 2, u_2 = 0,}$$

$$x_3 = Ax_1 + Bu_1 = A^2B$$ ..... and so on.

for y:

$$\underline{at\ k = 0,}$$

$$y_0 = Cx_0 + Du_0 = D$$

$$\underline{at\ k = 1,}$$

$$y_1 = Cx_1 + Du_1 = CB$$

$$\underline{at\ k = 2,}$$

$$y_2 = Cx_2 + Du_2 = CAB$$

$$\underline{at\ k = 3,}$$

$$y_3 = Cx_2 + Du_2 = CA^2B$$ ... ... and so on.

So y can be summarized as:

$$y: \begin{bmatrix} D, & k=0 \\ CA^{k-1}B, & k>0 \end{bmatrix}$$

2.2 HANKEL MATRIX FORMATION

In linear algebra, a Hankel matrix , named after Hermann Hankel, is a square matrix in which each ascending skew-diagonal from left to right is constant. These Hankel matrices are always symmetrical in nature. In ERA, the impulse responses are arranged into Hankel matrices. The initial condition is not taken into account as it is used only to get D matrix. Two Hankel matrices are formed where H_1 and H_2 are just shifted from each other by just one timeframe as shown below:

$$H_1 = \begin{bmatrix} y_1 & y_2 & y_3 \\ y_2 & y_2 & y_3 \\ y_3 & y_4 & y_5 \end{bmatrix} = \begin{bmatrix} CB & CAB & CA^2B \\ CAB & CA^2B & CA^3B \\ CA^2B & CA^3B & CA^4B \end{bmatrix}$$

$$H_2 = \begin{bmatrix} y_2 & y_3 & y_4 \\ y_3 & y_4 & y_5 \\ y_4 & y_5 & y_6 \end{bmatrix} = \begin{bmatrix} CAB & CA^2B & CA^3B \\ CA^2B & CA^3B & CA^4B \\ CA^3B & CA^4B & CA^5B \end{bmatrix}$$

When the data is reconfigured in this fashion, we will be able to use linear algebra to extract system properties (or Markov Parameters).

2.3 CONTROLLABILITY AND OBSERVABILITY MATRIX

Controllability and observability are dual aspects of the same problem. Roughly, the concept of controllability denotes the ability to move a system around in its entire configuration space using only certain admissible manipulations. Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs.

$$H_1 = \begin{bmatrix} y_1 & y_2 & y_3 \\ y_2 & y_2 & y_3 \\ y_3 & y_4 & y_5 \end{bmatrix} = \begin{bmatrix} CB & CAB & CA^2B \\ CAB & CA^2B & CA^3B \\ CA^2B & CA^3B & CA^4B \end{bmatrix} = \bar{O}\bar{C}$$

$$H_2 = \begin{bmatrix} y_2 & y_3 & y_4 \\ y_3 & y_4 & y_5 \\ y_4 & y_5 & y_6 \end{bmatrix} = \begin{bmatrix} CAB & CA^2B & CA^3B \\ CA^2B & CA^3B & CA^4B \\ CA^3B & CA^4B & CA^5B \end{bmatrix} = \bar{O}\,\boldsymbol{A}\,\bar{C}$$

The $\bar{O}$ and $\bar{C}$ matrices represent observability and controllability respectively. We can also see that the H_2 matrix can simply be obtained from H_1 if we can calculate $\bar{O}$ and $\bar{C}$ matrices. But the inverse is also true as $\bar{O}$ and $\bar{C}$ matrices along with H_1 and H_2 matrices will allow us to

calculate A, B and C. To do so, we need to find the singular value decomposition of the Hankel matrices.

2.4 SINGULAR VALUE DECOMPOSITION

The SVD of H_1 matrix can be shown as:

$$H_1 => U\Sigma V^T$$

$$H_1 => UT^2V^T \text{ where, } \Sigma = T^2$$

$$\bar{O}\bar{C} = UT^2V^T$$

Where,

$$\bar{O} = UT$$

$$\bar{C} = TV^T$$

Usually, SVD of a matrix gives Σ matrix but for ease of calculation and decomposition we substituted it with T^2. By doing this, we will be able to find the $\bar{O}$ and $\bar{C}$ matrix easily and hence find the Markov parameters.

2.5 FINDING REDUCTION MODEL

After the SVD of the H_1 matrix, we will be able to obtain the Markov parameters in the following way:

$$\bar{O}\ \hat{A}\ \bar{C} = H_2$$

$$UT\hat{A}TV^T = H_2$$

$$\hat{A} = U^{-1}T^{-1}\ H_2(V^T)^{-1}T^{-1}$$

$$= U^T T^{-1} H_2 V T^{-1}$$

$$\hat{C} = \begin{bmatrix} I_q & 0 \\ 0 & 0 \end{bmatrix} UT$$

$$\hat{B} = TV^T \begin{bmatrix} I_P & 0 \\ 0 & 0 \end{bmatrix}$$

The U and V matrices are orthogonal matrices. This means that their inverse can be calculated as transpose. This is useful as calculating transpose is less computationally expensive then computing inverse. The Markov parameters extracted can be used to create a reduced order model. Hence, we have now calculated a state space system without calculating the state of the system.

3 IMPLEMENTATION ON A MISO SYSTEM

In this section, we will be implementing ERA on a MISO (Multiple Input Single Output) system. We will first take impulse response for all the inputs. This impulse response will then be used by ERA to extract reduced order state space model. The original and the reduced model will then be fed different inputs. Their outputs will be compared and analyzed.

3.1 EXTRACTING REDUCED ORDER MODEL USING ERA

To implement ERA on a MISO system, we first need to find the matrices that represents the state space model of the MISO system. For a linear system with a general equation:

$$x_{k+1} = Ax_k + Bu_k,$$

$$y_k = Cx_k + Du_k$$

$$\boldsymbol{A} = \begin{bmatrix} -0.5572 & -0.7814 \\ 0.7814 & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} -0.5572 & -0.7814 \\ 0.7814 & 0 \end{bmatrix},$$

$$\boldsymbol{C} = \begin{bmatrix} 1.9691 & 6.4493 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

This is a system with two inputs and one output, hence a MISO system. By examining the matrices, we can see that the system has 2 states.

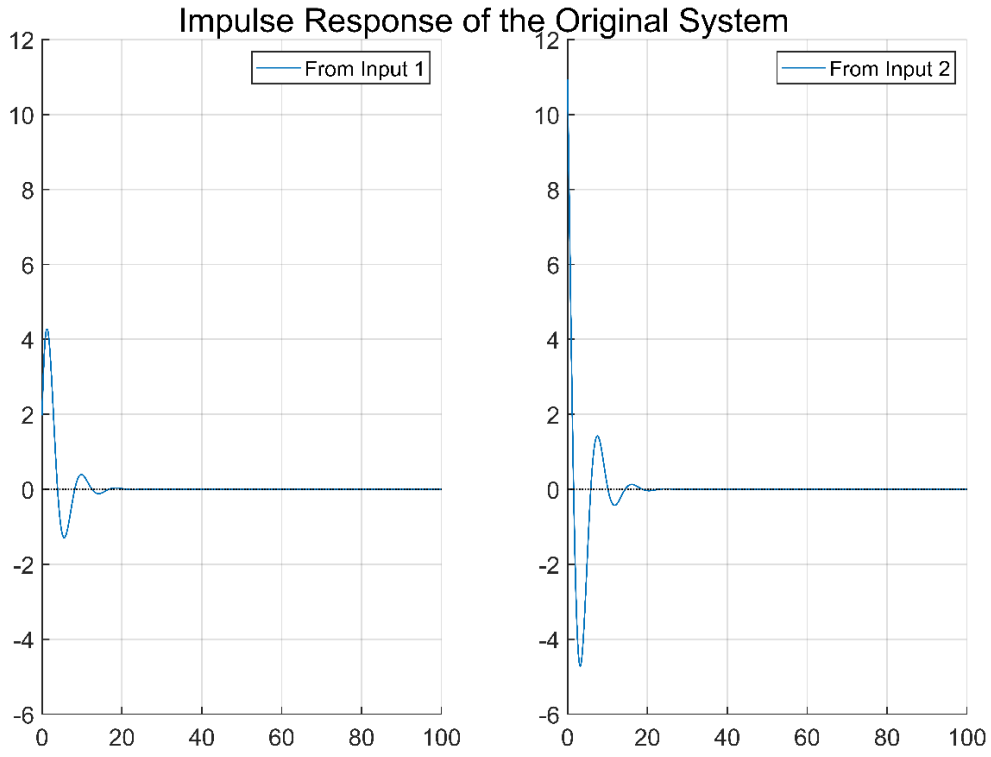

Figure 3 : Impulse Response of the Original System

When we simulate the impulse response for each of the input of the system (as shown in **Figure 3**), we get two sets of data. These data are stacked together to create an Output matrix that will be used by ERA to extract a new state space model without thinking about the states in the system. The new model extracted by ERA has different matrices:

$$\boldsymbol{A} = \begin{bmatrix} 0.9985 & 0.0137 \\ -0.0040 & 0.9959 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} -1.7916 & -1.9913 \\ -0.5389 & 2.1738 \end{bmatrix},$$

$$\boldsymbol{C} = \begin{bmatrix} -2.0568 & 3.1113 \end{bmatrix}, \quad \boldsymbol{D} = \begin{bmatrix} 1.9691 & 10.9295 \end{bmatrix}$$

Despite having different state space model, when fed same input, both the original and new model will have the same output.

3.2 IMPULSE

In this part, we will set both the input as impulse. We will then compare the output generated by both the model.

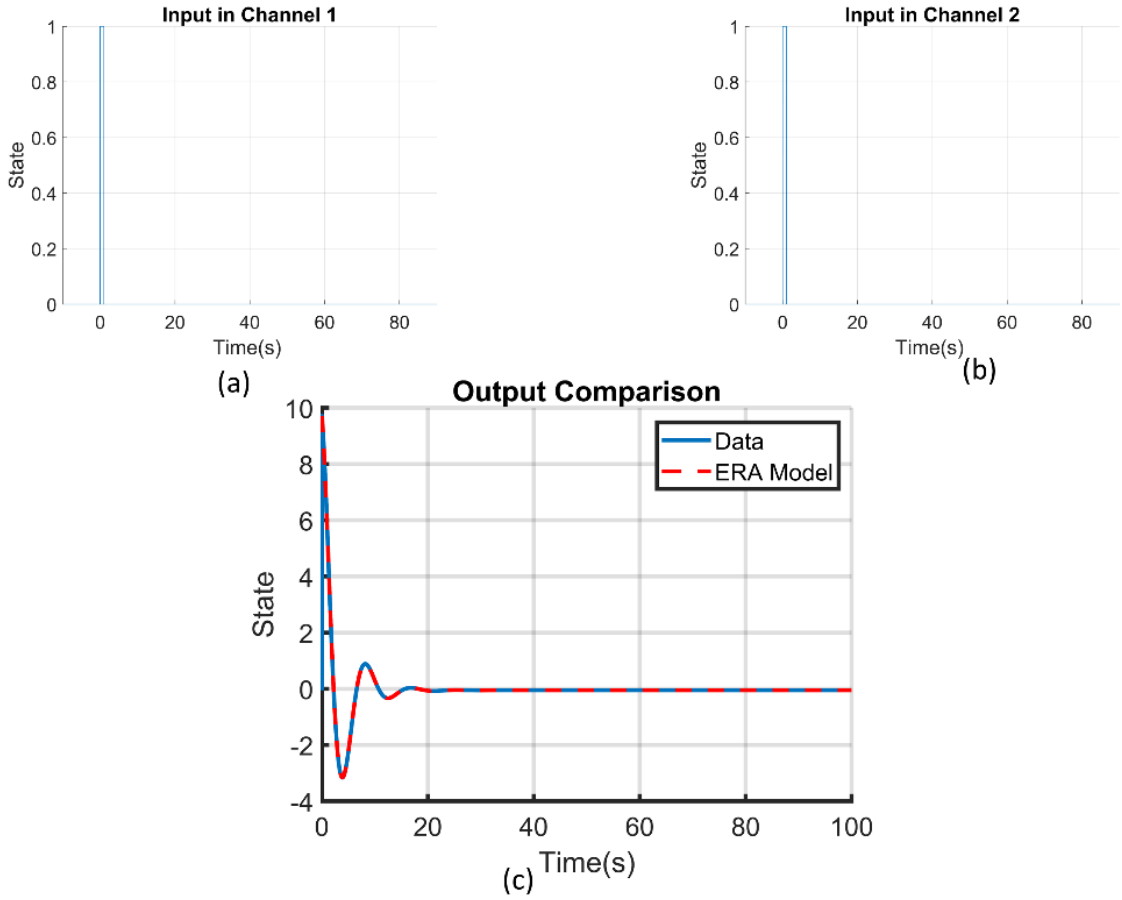

Figure 4: For impulse input (a) Input in Channel 1, (b) Input in Channel 2, (c) Comparison of the response from two models.

3.3 UNIT STEP

In this part, we will set both the input as unit step. We will then compare the output generated by both the model.

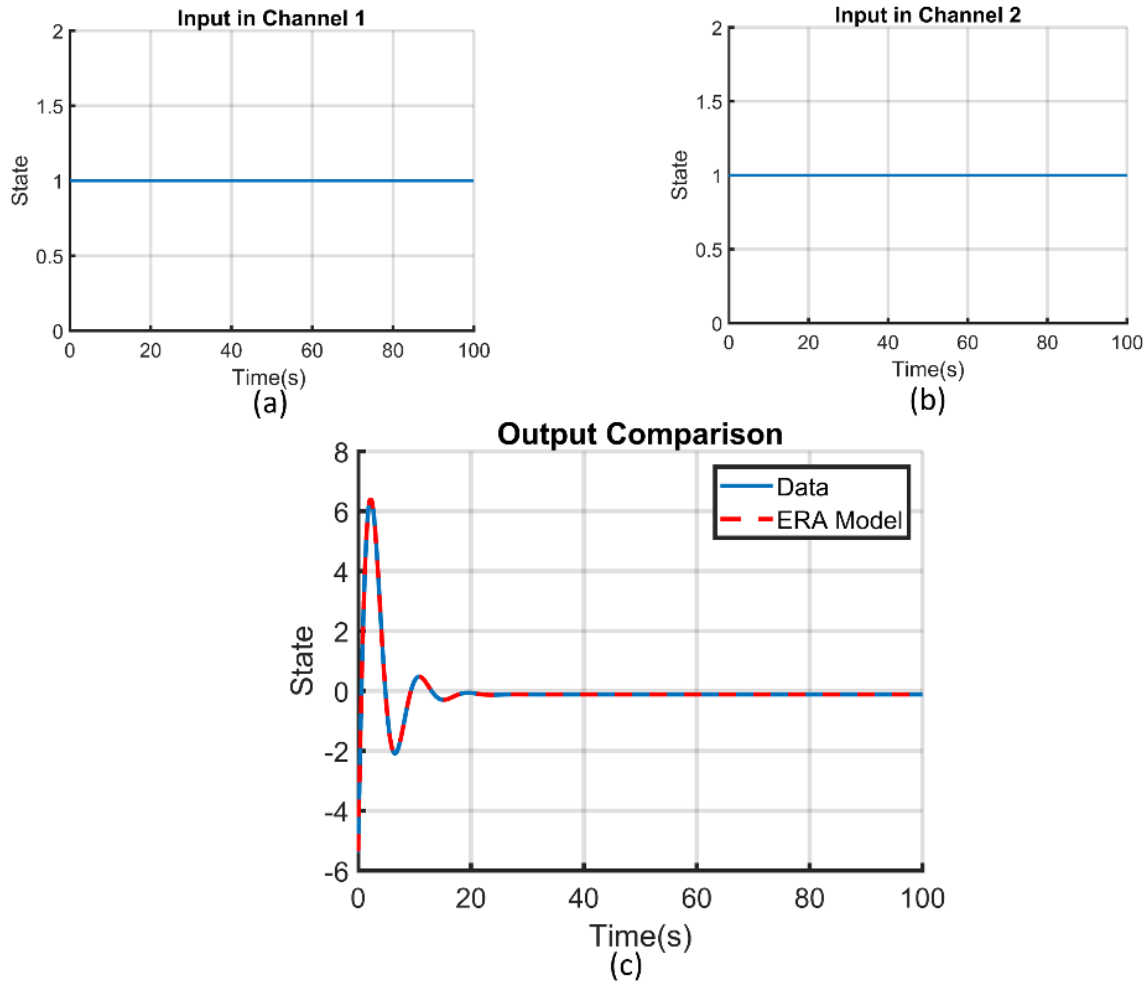

Figure 5: For unit step input (a) Input in Channel 1, (b) Input in Channel 2, (c) Comparison of the response from two models.

3.4 RAMP

In this part, we will set both the input as a ramp signal We will then compare the output generated by both the model.

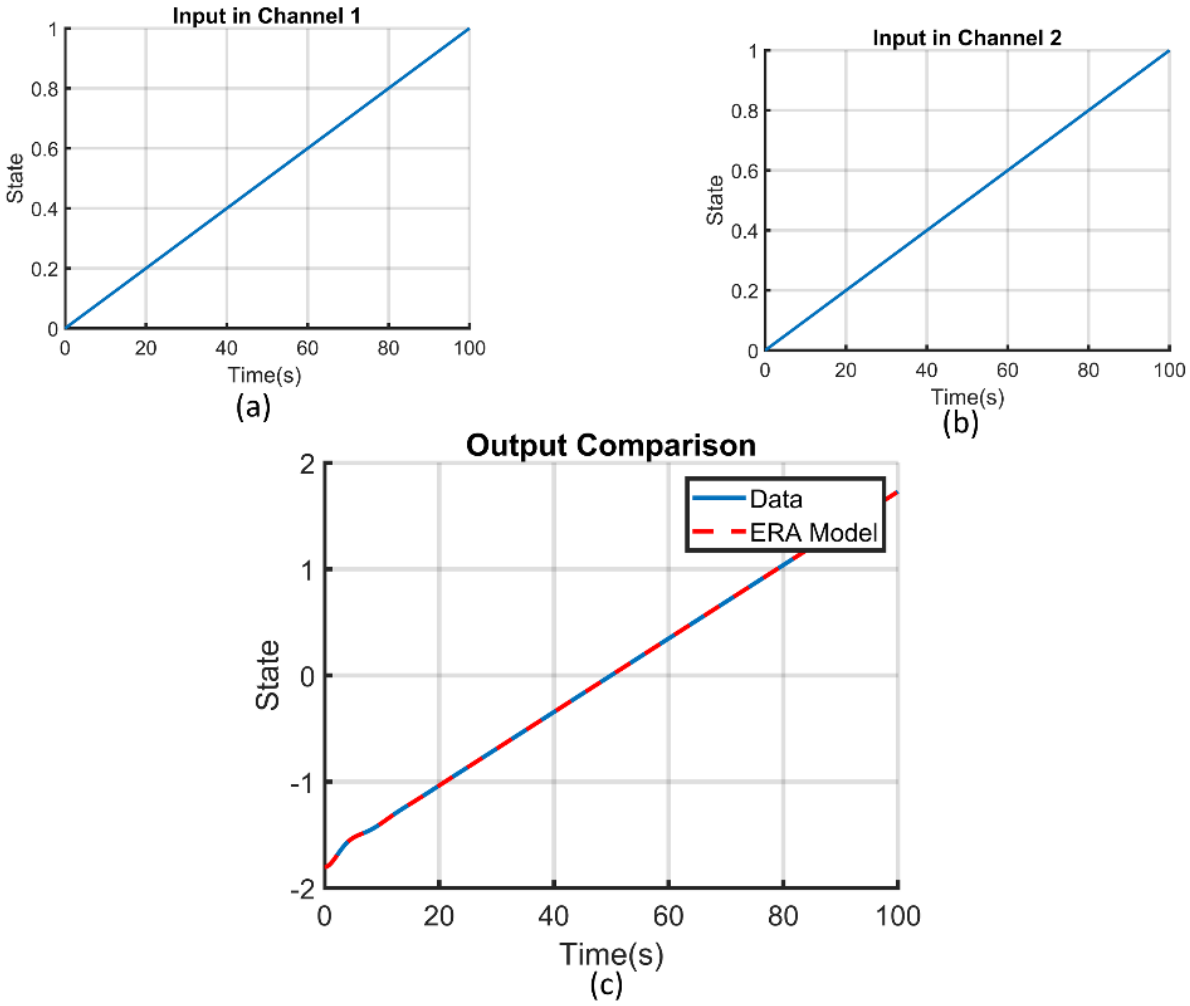

Figure 6: For ramp input (a) Input in Channel 1, (b) Input in Channel 2, (c) Comparison of the response from two models.

3.5 RANDOM

In this part, we will set both the input as some randomly generated signal. The random number is generated such that the values are ranged from -1 to 1 and are normally distributed. We will then compare the output generated by both the model.

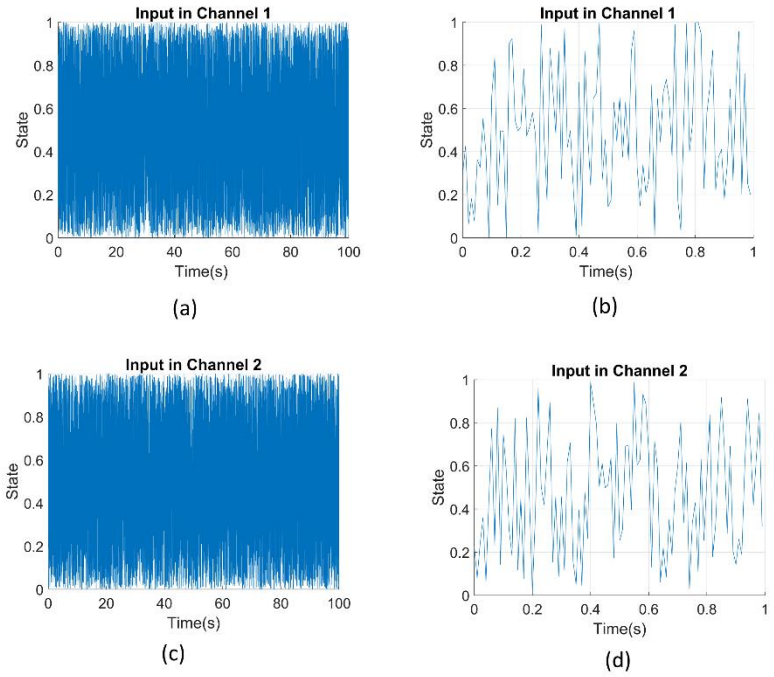

Figure 7: For a randomly generated signal as input (a) Input in channel 1, (b) Detailed section of (a), (c) Input in channel 2, (d) Detailed section of (c)

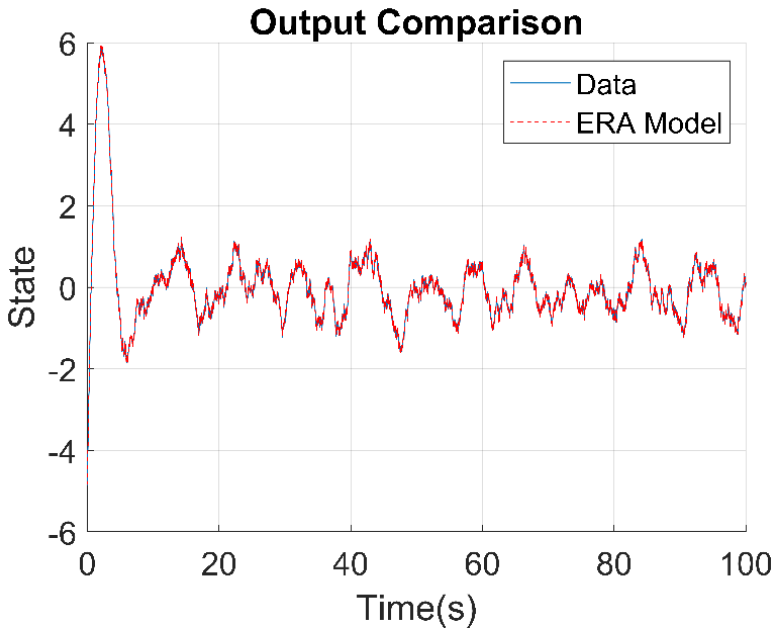

Figure 8:Comparison of the response from two models.

In **Figure 7,** we can see two randomly generated signals. To observe the amount of noise in the signal, we have plotted the signal over a period of 1 second. This allows us to see that the data is very chaotic. Even so, the model was able to capture the output as shown in **Figure 8**.

4 RESULT

In the previous section, we saw how we extracted a new model just from impulse response. We then fed them different inputs and compared the outputs visually. In this section, we will compare the output performance using some performance criteria.

Table 1: Comparing performance of the new model with original model by feeding different inputs

Input	MAE	RMSE	R^2
Impulse	2.6 x 10^-3	0.098	0.99
Unit Step	1.5 x 10^-3	5.1 x 10^-3	1.00
Ramp	4.1 x 10^-5	1.1 x 10^-4	1.00
Random Number	0.039	0.045	0.99

Table 1 shows the different inputs and the performance for it. In the table we have some evaluation criteria. They are:

- MAE: Absolute Error is the amount of predicted error. The Mean Absolute Error is the mean of all absolute errors.

$$\text{MAE} = \frac{1}{n}\sum_n \left|X_p - X\right|$$

- RMSE: Root Mean Squared Error (RMSE) is the standard deviation of the residuals (prediction error). Residuals are a measure of how far away the data points are from the regression line; RMSE is a measure of how these residuals are spread out.

$$\text{RMSE} = \sqrt{\frac{\sum\left|X_p - X\right|^2}{n}}$$

- R^2: R^2 is the co-efficient of determination. It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

Here, Xp is the predicted data while the ground truth data is X and n is the number of samples.

5 Discussion

In **Section 3**, we showed the implementation of ERA on a MISO system. After extracting the Markov parameters and creating a model, we tested the model with different inputs. We compared these outputs with outputs from the original model. The error between these outputs were minimal as shown in **Table 1**. Even when the input was a randomly generated signal, the model was able to capture the characteristics of the signal such that it was able to replicate what the original model would have done.

Furthermore, the Markov parameters of the two systems are different but they produce the same results. This means that ERA did not magically find the original parameters but instead found the parameters from the data.

6 Future Work

The data used in this work was generated by using a predefined state space system. Real life data and system are, however, more chaotic in nature. We aim to work with these data so that we can apply this knowledge to practical fields. The data we are hoping to work with are related to aerodynamics, aviation and thermodynamics.

7 MATLAB CODE

7.1 LOAD

```
Clear all; close all; clc;
tic

a = [-0.5572 -0.7814;0.7814  0];
b = [1 -1;0 2];
c = [1.9691  6.4493];
sys1 = ss(a,b,c,0);

t = linspace(0,100,10000);
f = impulse(sys1,t); % creating initial impulse response
dt = t(2)-t(1);
```

7.2 HANKEL MATRIX

```
H1 =[ f(1:end-2);f(2:end-1)];


H2 =[ f(2:end-1); f(3:end)];
```

7.3 ERA

```
[U,S,V]=svd(H1,'econ');
r = rank(S);

% Reconfiguring Data for the ERA function
YY(1,1,:) = f(:,1,1);
YY(1,2,:) = f(:,1,2);

% Applying ERA to get reduced order model
[Ar,Br,Cr,Dr,HSVs] = ERA(YY,200,200,2,1,r);

% Creating state-space model from ERA
sys = ss(Ar,Br,Cr,Dr,dt);
```

7.4 DESIGNING INPUTS

```
temp = zeros(length(t),1);


unitstep = t>=0; % unit step input
ramp = t.*unitstep/100; % ramp input
impu = temp; impu(1,1) = 1; % impulse input
rand1 = rand(length(t),1);
rand2 = rand(length(t),1);
```

7.5 TESTING MODEL

```
u = zeros(length(t),2);

u(:,1) = impu;

u(:,2) = impu;

y = lsim(sys,u,t);      %response from new system
f_org = lsim(sys1,u,t); %response from original system
```

7.6 RMSE

```
yfit = normalize(y);

Y_test = normalize(f_org);

error = yfit-Y_test;

error_squared = error.^2;

SSE = sum(error_squared); % sum of squared error
MAE = sum(abs(error)/length(error));
MSE = SSE/length(error);
RMSE = sqrt(MSE);
R2 = 1 - (MSE)/(sum((Y_test-mean(Y_test)).^2)/length(error));
```

7.7 PLOTS

```
figure()

hold on
grid on
plot(t,normalize(f_org))
plot(t,normalize(y),'--r')
xlabel('Time(s)'), ylabel('State')
legend('Data ','ERA Model')
title('Output Comparison')
figure()
hold on
grid on
plot(t,u(:,1))
xlabel('Time(s) '), ylabel('State')
title('Input in Channel 1')

figure()
hold on
grid on
plot(t,u(:,2))
xlabel('Time(s)'), ylabel('State')
title('Input in Channel 2')
```

7.8 States

```
fprintf('The size of A is %d X %d \n',size(Ar,1),size(Ar,2)) fprintf('The size of B is %d X %d
\n',size(Br,1),size(Br,2)) fprintf('The size of C is %d X %d \n',size(Cr,1),size(Cr,2))
fprintf('The size of D is %d X %d \n',size(Dr,1),size(Dr,2))
```

7.9 Done

```
toc

disp('DONE!')
```

7.10 User Defined Function - ERA

```
function [Ar,Br,Cr,Dr,HSVs] = ERA(YY,m,n,nin ,nout,r)

for i=1:nout
    for j=1:nin
        Dr(i,j) = YY(i,j,1);
        Y(i,j,:) = YY(i,j,2:end);
    end
end

% Yss = Y(1,1,end);
% Y = Y-Yss;
% Y(i,j,k)::
% i refers to ith output
% j refers to jth input
% k refers to kth time step
% nin,nout number of inputs and outputs
% m,n dimensions of Hankel matrix
% r, dimensions of reduced model

assert(length(Y(:,1,1))==nout);
assert(length(Y(1,:,1))==nin);
assert(length(Y(1,1,:))>=m+n);

% how do you determine the dimensions of hankel matrix
for i=1:m
    for j=1:n
        for Q=1:nout
            for P=1:nin
                H(nout*i-nout+Q,nin*j-nin+P) = Y(Q,P,i+j-1);
                H2(nout*i-nout +Q,nin*j-nin+P) = Y(Q,P,i+j);
            end
        end
    end
end
```

```
[U,S,V] = svd(H,'econ');
Sigma = S(1:r,1:r);
Ur = U(:,1:r);
Vr = V(:,1:r);
Ar = Sigma^(-.5)*Ur'*H2*Vr*Sigma^(-.5);
Br = Sigma^(-.5)*Ur'*H(:,1:nin);
Cr = H(1:nout ,:)*Vr*Sigma^(-.5);
HSVs = diag (S);
```

Published with MATLAB® R2019b