University of Groningen

Bachelor Integration Project

Identification of LTI system with auxiliary data from a similar system

Author:
Aleksander Peszko *(s3696871)*

Lecturer:
prof. dr. Nima Monshizadeh

January 20, 2023

rijksuniversiteit groningen

Contents

Abstract

This report describes the dynamical system identification with limited access to the data. The goal of the research is to prove that leveraging data from a similar system can reduce the identification error by complementing the original system's data with additional data from the auxiliary system. The identification will be performed using the regression analysis method of weighted least squares. Once the error is identified, a detailed analysis of the influencing factors will be provided. It includes altering various parameters such as process noise, number of experiments and weight parameter in order to observe the behaviour of the error and ideally reduce it. The experimental setup includes two linear time-invariant (LTI) systems, namely target and source systems that will be subjected to the identification, and based on the obtained results, I will draw conclusions and validate the theoretical assumptions.

1 Introduction

Dynamical system identification is used in various fields of science as a method of constructing dynamical systems from measured data [1]. Sometimes the system is complex and cannot be described with physical principles. System identification allows deriving a mathematical model basing upon the collected data samples and the relationship between parameters. The data samples are gathered through experiments and they dictate the dynamics of identified system. The larger the number of experiments, the more accurate dynamics can be modelled [1]. This is called a *multiple trajectory setup* and is often used in system identification [1]. In this particular setup, multiple independent experiments are performed in order to collect as much data as possible. In each trajectory, the system is run from beginning to the end giving opportunity for more unbiased measurements [1].

The experimental analysis enables to determine the dynamics of the system with only two kinds of data, namely input and output. This is in fact useful in modelling the complex systems because the behaviour is reduced to just input-output relationship. However, in many cases, the internal state of the system is sought to be established [2]. Therefore, the input-state relationship is crucial there. By knowing this relationship, the internal state at next time step can be foreseen. A mathematical model can be then derived by subjecting the measured input-state signals to some identification method such as the *least squares method* (LS) [2].

In every real world systems there are disturbances (noise) attached. They distort the measured data affecting the whole system and resulting in the dynamics not being accurate anymore. The noise is a main reason for causing *identification error*. Identification error is the deviation between the actual system and its mathematical model [2]. As the parameters can no longer be described by a relation, it is difficult to predict the system's future behaviour [3]. In order to reduce the error, the number of experiments shall be increased because more trials generate more reliable measurements [2]. However, in some cases, acquiring data from an actual system is not feasible as some obstacles can arise [4]. With limited data, the identified system does not illustrate the behaviour of the actual system. Therefore, one seeks help from an auxiliary system, which is a twin system to the actual system. The reason is that it shares similar dynamics and it is abundant in data [5].

In this research, the primary focus lies on identifying the dynamics of a certain *linear time-invariant* (LTI) system by leveraging data from an auxiliary system on top of the actual system's data. The data is generated in a multiple trajectories setup with each experiment having a constant data length [1][6]. Furthermore, the identification method that is used throughout the process is weighted least squares method (WLS).

System identification will be achieved in steps according to Xin, L., et al. 2022 so that the transitions between key phases are supported with mathematical correlations. Firstly, the influence of noise on the system identification will be discussed. It includes investigating how changing the parameters such as the energy of the noise and/or number of experiments affects the identification error. Second step is to show how adding the source system's data will affect the identified system's dynamics and consequently the identification error. Next, the specific weight will be assigned for auxiliary data in order to capture the change in behaviour. Lastly, the simulations of the identification error will be shown and discussed in detail in terms of increasing amount of actual system's samples, auxiliary system's samples and the weight parameter assigned to these samples. The goal at the end of the research is to find such a combination of variables that reduce the identification error between the actual and identified systems.

2 Mathematical Terminology

$||\cdot||$: Spectral norm of a matrix i.e. the largest singular value of a matrix,
$||\cdot||_F$: Frobenius norm of a matrix,
$u \sim \mathcal{N}(\mu, \Sigma)$: Gaussian distributed random vector, where μ is the mean and Σ is the covariance matrix [1],
I_n : Identity matrix of dimensions $n \times n$,
$diag(q)$: Identity matrix with q in diagonal.

3 Theory

System identification firstly requires establishing the system. Xin, L., et al. 2022 and Fattahi, S., and Sojoudi, S., 2018 describe the target system with a discrete-time LTI system equation

$$\bar{x}_{k+1} = \bar{A}\bar{x}_k + \bar{B}\bar{u}_k + \bar{w}_k \tag{1}$$

where
$\bar{x}_k \in R^n$ is target system state,
$\bar{u}_k \in R^m$ is target system input,
$\bar{w}_k \in R^n$ is target system noise,
$\bar{A} \in R^{n\times n}$ and $\bar{B} \in R^{n\times m}$ are target system matrices.

The input and process noise are assumed to be i.i.d Gaussian, with $\bar{u}_k \sim \mathcal{N}(0,\, \sigma_u^2 I_m)$ and $\bar{w}_k \sim \mathcal{N}(0,\, \sigma_w^2 I_n)$ [1]. Note that σ^2 is the energy of the noise which dictates its deviation. Gaussian noise creates random samples whose actual distribution is unknown [7]. It is commonly used in system identification due to the fact that the data is normally distributed in the array.

Assuming the multiple trajectory setup, N_r independent experiments are conducted with new data being generated at each trajectory. Every experiment starts at an initial state $\bar{x}_0 \sim \mathcal{N}(0,\, \sigma_x^2 I_n)$, and has data length T. During these experiments, the system measures state and input at each time step put them into state-input pairs. These samples are called a *rollout* and can be denoted as $\{(\bar{x}_k^i, \bar{u}_k^i) : 1 \le i \le N_r\ ,\ 0 \le k \le T\}$, where i is the rollout index and k is the time index [1][6].

Note that since $\bar{x}_k$ and $\bar{u}_k$ can be measured by the system, then let $\bar{z}_k^i = \begin{bmatrix} \bar{x}_k^i \\ \bar{u}_k^i \end{bmatrix} \in R^{n+m}$ be a state-input parameter.

Next step is to establish data matrices so that with each experiment the system is able to collect data. For each rollout i, define $\bar{X}^i = [\bar{x}_T^i\ \cdots\ \bar{x}_1^i] \in R^{n\times T}$, $\bar{Z}^i = [\bar{z}_{T-1}^i\ \cdots\ \bar{z}_0^i] \in R^{(n+m)\times T}$, $\bar{W}^i = [\bar{w}_{T-1}^i\ \cdots\ \bar{w}_0^i] \in R^{n\times T}$.

Then, combine experiments together into single matrix by defining the batch matrices $\bar{X} = [\bar{X}^1\ \cdots\ \bar{X}^{N_r}] \in R^{n\times N_r T}$, $\bar{Z} = [\bar{Z}^1\ \cdots\ \bar{Z}^{N_r}] \in R^{(n+m)\times N_r T}$, $\bar{W} = [\bar{W}^1\ \cdots\ \bar{W}^{N_r}] \in R^{n\times N_r T}$.

Selecting $\theta = [\bar{A}\ \bar{B}]$ results in Equation 1 becoming

$$\bar{X} = \theta\bar{Z} + \bar{W} \tag{2}$$

where θ is responsible for the dynamics of the system [1].

Using least squares method theorem (from Chen, L., et al. 2021), one should solve

$$\min_{\tilde{\theta}\in R^{n\times(n+m)}} \|\bar{X} - \tilde{\theta}\bar{Z}\|_F^2$$

to obtain the estimated value of $\theta_{LS} = [\bar{A}_{LS}\ \bar{B}_{LS}]$.

Analytically calculated θ_{LS} from Frobenius norm has a following formula (from Xin, L., et al. 2022)

$$\theta_{LS} = \bar{X}\bar{Z}^\top(\bar{Z}\bar{Z}^\top)^{-1} \tag{3}$$

Knowing that not many experiments can be conducted for target system i.e. N_r is small, the estimation error due to noise cannot be reduced. Nevertheless, there exists a possibility of accessing data from a source system. Assuming similarity of these systems, source system's samples complement target system's samples. Both systems share similar dynamics, therefore the source system can be analogically described by following equation

$$\hat{x}_{k+1} = \hat{A}\hat{x}_k + \hat{B}\hat{u}_k + \hat{w}_k \tag{4}$$

where
$\hat{x}_k \in R^n$ is source system state,
$\hat{u}_k \in R^m$ is source system input,
$\hat{w}_k \in R^n$ is source system noise,
$\hat{A} \in R^{n\times n}$ and $\hat{B} \in R^{n\times m}$ are source system matrices.

Similarly to Equation (1), the input and process noise are assumed to be i.i.d Gaussian, with $\hat{u}_k \sim \mathcal{N}(0, \sigma_u^2 I_m)$ and $\hat{w}_k \sim \mathcal{N}(0,\ \sigma_w^2 I_n)$ [1].

Now, let us replace $\hat{A}$ and $\hat{B}$ with $\bar{A} + \delta_A$ and $\bar{B} + \delta_B$ in Equation (4), respectively, where $\delta_A = \hat{A} - \bar{A}$ and $\delta_B = \hat{B} - \bar{B}$ are the differences in systems' matrices, and obtain

$$\hat{x}_{k+1} = (\bar{A} + \delta_A)\hat{x}_k + (\bar{B} + \delta_B)\hat{u}_k + \hat{w}_k \tag{5}$$

Again, assuming the multiple trajectory setup, however in this case N_p independent experiments are conducted. Every experiment starts at an initial state $\hat{x}_0 \sim \mathcal{N}(0,\ \sigma_x^2 I_n)$, and has data length T. During these experiments, the system measures state and input at each time step put them into state-input pairs. These samples are called a *rollout* and can be denoted as $\{(\hat{x}_k^i, \hat{u}_k^i) : 1 \leq i \leq N_p\ ,\ 0 \leq k \leq T\}$, where i is the rollout index and k is the time index [1][6].

Note that since $\hat{x}_k$ and $\hat{u}_k$ can be measured by the system, then let $\hat{z}_k^i = \begin{bmatrix} \hat{x}_k^i \\ \hat{u}_k^i \end{bmatrix} \in R^{n+m}$ be a state-input parameter.

Rearranging terms in the Equation (5) gives

$$\hat{x}_{k+1} = \bar{A}\hat{x}_k + \bar{B}\hat{u}_k + \delta_A\hat{x}_k + \delta_B\hat{u}_k + \hat{w}_k \tag{6}$$

which can by further written as

$$\hat{x}_{k+1} = [\bar{A}\ \bar{B}]\hat{z}_k + [\delta_A\ \delta_B]\hat{z}_k + \hat{w}_k \tag{7}$$

Repeating the steps between Equations (1) and (3) for source system's parameters will output data matrices for each rollout i, such that $\hat{X}^i = [\hat{x}_T^i\ \cdots\ \hat{x}_1^i] \in R^{n\times T}$, $\hat{Z}^i = [\hat{z}_{T-1}^i\ \cdots\ \hat{z}_0^i] \in R^{(n+m)\times T}$, $\hat{W}^i = [\hat{w}_{T-1}^i\ \cdots\ \hat{w}_0^i] \in R^{n\times T}$.

Then, combine experiments together into single matrix by defining the batch matrices $\hat{X} = [\hat{X}^1\ \cdots\ \hat{X}^{N_p}] \in R^{n\times N_pT}$, $\hat{Z} = [\hat{Z}^1\ \cdots\ \hat{Z}^{N_p}] \in R^{(n+m)\times N_pT}$, $\hat{W} = [\hat{W}^1\ \cdots\ \hat{W}^{N_p}] \in R^{n\times N_pT}$.

Additionally, introduce new variables to connect target and source systems, namely $X = [\bar{X}\ \hat{X}] \in R^{n\times(N_r+N_p)T}$, $Z = [\bar{Z}\ \hat{Z}] \in R^{(n+m)\times(N_r+N_p)T}$, $W = [\bar{W}\ \hat{W}] \in R^{n\times(N_r+N_p)T}$ and $\delta = [\delta_A\ \delta_B] \in R^{n\times(n+m)}$.

Next, for each experiment include the differences in systems' matrices in terms of source system's samples $\Delta^i = [\delta\hat{z}_{T-1}\ \cdots\ \delta\hat{z}_0] \in R^{n\times T}$ so that it can be later implemented in final equation [1]. Combining Δ^i for all $i \in \{1, \ldots, N_p\}$ results in $\Delta = \begin{bmatrix} 0 & \cdots & 0 & \Delta^1 & \cdots & \Delta^{N_p} \end{bmatrix} \in R^{n\times(N_r+N_p)T}$.

Once previous steps are followed, the obtained relationship of the target and source systems is

$$X = \theta Z + W + \Delta \tag{8}$$

At this point, the combined relationship of both systems is expressed with the dynamics of the target system (i.e. $\theta = [\bar{A}\ \bar{B}]$). In this case, the target and source systems' samples are equally weighted. However, by assigning weight to source system's samples, one can influence the behaviour seeking more accurate identification [3].

Designing a parameter $q \in R_{\geq 0}$ allows to assign the relative weight to the samples from source system (4). In order to match the weight with data length T and number of experiments N_p, one should define $Q = diag(q) \in R^{T\times T}$ and $\hat{Q} = diag(Q, \cdots, Q) \in R^{N_pT\times N_pT}$. Furthermore, define $Q = diag(I_{N_rT}, \hat{Q}) \in R^{(N_r+N_p)T\times(N_r+N_p)T}$ so that only the source system's samples are affected in Equation (8). Using weighted least squares method theorem (from Chen, L., et al. 2021), one should solve

$$\min_{\tilde{\theta} \in R^{n\times(n+m)}} \|XQ^{\frac{1}{2}} - \tilde{\theta}ZQ^{\frac{1}{2}}\|_F^2$$

to obtain the estimated value of $\theta_{WLS} = [\bar{A}_{WLS}\ \bar{B}_{WLS}]$.

Repeating (3) for θ_{WLS} outputs

$$\theta_{WLS} = XQZ^\top(ZQZ^\top)^{-1} \tag{9}$$

Combining it with (8), the estimation error can be calculated as

$$\theta_{WLS} - \theta = WQZ^\top(ZQZ^\top)^{-1} + \Delta QZ^\top(ZQZ^\top)^{-1} \tag{10}$$

The theoretical knowledge described above will be used while constructing a Matlab algorithm in later sections of the research. Every aspect of the algebra behind the code has been covered and delivered in the form of equations.

4 Algorithm

As stated in the introduction, the system identification consists of number of stages. Each stage has its own Matlab code which is provided in the Appendix. In Stage 1, the code is designed to solve the identification error without noise (i.e. $\bar{X} = \theta\bar{Z}$). Stage 2 includes steps from previous stage with addition of noise to the system Equation (2) to observe its influence on the system. Stage 3 will tackle how altering certain parameters (such as σ^2 and N_r) changes the identification error. During Stage 4, both target and source systems will be used for system identification (Equation (8)). Lastly, in Stage 5, the specific weight will be assigned for source system's samples and WLS will be performed, i.e. $\theta_{WLS} = XQZ^\top(ZQZ^\top)^{-1}$.

In the end, the algorithm should include following (from Xin, L., et al. 2022):

1. Gather N_r length T rollouts of samples generated from the target system 1, where $\bar{x}_0^i \sim \mathcal{N}(0,\, \sigma_x^2 I_n)$ for all $1 \leq i \leq N_r$.
2. Gather N_p length T rollouts of samples generated from the source system 4, where $\hat{x}_0^i \sim \mathcal{N}(0,\, \sigma_x^2 I_n)$ for all $1 \leq i \leq N_p$.
3. Construct the matrices X, Q, Z and compute $\theta_{WLS} = XQZ^\top(ZQZ^\top)^{-1}$.
4. Return the first n columns of θ_{WLS} as an estimated $\bar{A}$, and the remaining columns of θ_{WLS} as an estimated $\bar{B}$.

4.1 Stage 1: System Identification without noise

During this stage, the noise is excluded from the system (1), i.e. $\bar{x}_{k+1} = \bar{A}\bar{x}_k + \bar{B}\bar{u}_k$. First step is to fill empty matrices with data created by the system. In order to achieve that two data creating `for loop` Matlab functions are introduced, the outer one for N_r and the inner for T (see Appendix A).

The second step is to use this data to identify the system dynamics, i.e. $\theta_{LS} = [\bar{A}_{LS}\ \bar{B}_{LS}]$, and consequently the identification error by solving $\theta_{LS} = \bar{X}\bar{Z}^\top(\bar{Z}\bar{Z}^\top)^{-1}$.

4.2 Stage 2: System Identification with noise

Repeat the previous stage of the code, but this time include noise. Instead of $\bar{x}_{k+1} = \bar{A}\bar{x}_k + \bar{B}\bar{u}_k$, use $\bar{x}_{k+1} = \bar{A}\bar{x}_k + \bar{B}\bar{u}_k + \bar{w}_k$ and the output will include the disturbances caused by noise.

4.3 Stage 3: Plotting identification error as function of σ_w^2 and N_r

Two plots will be generated at the end of this stage both showing the behaviour of identification error in terms of increasing parameters. Taking σ_w^2 as the increasing parameter requires changing the noise deviation to an increasing array, for instance

$$\sigma_w^2 = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$$

.

Since the number of entries is established, the code requires solving the identification error for each entry. It is achieved by creating a `for loop` function (see Appendix A).

For the second parameter, namely N_r, the new code needs to be adjusted for increasing number of experiments. Similarly to σ^2, change the number of experiments from a single value to an increasing array, for instance

$$N_r = \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$$

.

Again, introduce a `for loop` Matlab command and solve the identification error for each entry.

4.4 Stage 4: LS for target and source systems

During this stage of the code, the dynamics of combined target and sources system is identified. Firstly, pre-define matrices for both systems and create a second `for loop` Matlab function for source system data. As the loop is executed, introduce variables that connect target and source system data, i.e. X, Z, W and solve $\theta_{LS} = XZ^\top(ZZ^\top)^{-1}$ to obtain the identification error (see second step of Stage 2).

4.5 Stage 5: WLS for target and source systems

Repeat the steps from Stage 4. However, before obtaining the identification error, add a weight specifying parameter q and a `for loop` Matlab function for weighted samples (see Appendix A). Once the data is created, one should solve $\theta_{WLS} = XQZ^\top(ZQZ^\top)^{-1}$ to obtain the identified system. Last activity in this code is to identify the error by computing $||\theta_{WLS} - \theta|| = WQZ^\top(ZQZ^\top)^{-1} + \Delta QZ^\top(ZQZ^\top)^{-1}$.

5 Experimental Setup

Two examples of system identification will be provided in order to test the practical execution of theoretical knowledge. One example represents a virtual LTI system while the other example concerns a real physical LTI system. Both systems will be subjected to system identification and the results will be discussed in detail.

5.1 Virtual System

The following system is a virtual system described by Xin, L., et al. 2022. It does not represent any "real-world" system, nonetheless it can provide insights on the accuracy of system identification.

Key important assumptions are listed below:

1. Dimensions are predefined, i.e. $n = 3$ and $m = 2$,
2. Target and Source system's matrices are predefined:

$$\bar{A} = \begin{bmatrix} 0.6 & 0.5 & 0.4 \\ 0 & 0.4 & 0.3 \\ 0 & 0 & 0.3 \end{bmatrix}, \bar{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \\ 0.5 & 0.5 \end{bmatrix},$$

$$\hat{A} = \begin{bmatrix} 0.7 & 0.5 & 0.4 \\ 0 & 0.4 & 0.3 \\ 0 & 0 & 0.3 \end{bmatrix}, \hat{B} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1 \\ 0.5 & 0.5 \end{bmatrix},$$

3. Number of experiments and data length are predefined, i.e. $N_r = 10$, $N_p = 10$, $T = 100$,
4. The input, process noise and initial state are Gaussian $\sim \mathcal{N}(0, \sigma^2 I)$ with σ^2 being the energy (deviation). Knowing that, let us use Matlab build-in function `randn` to generate normally distributed vectors.

5.2 Physical System

The physical system that is being analyzed is the batch reactor process described by Rosenbrock [8]. The system describes the behavior of certain fluids when mixing with materials under various conditions of temperature and pressure [8]. Let this reactor be our target system. It is of dimensions $n = 4$ and $m = 2$ and its matrices are following:

$$\bar{A} = \begin{bmatrix} 1.38 & -0.2077 & 6.715 & -5.676 \\ -0.5814 & -4.29 & 0 & 0.675 \\ 1.067 & 4.273 & -6.654 & 5.893 \\ 0.048 & 4.273 & 1.343 & -2.104 \end{bmatrix}, \bar{B} = \begin{bmatrix} 0 & 0 \\ 5.679 & 0 \\ 1.136 & -3.146 \\ 1.136 & 0 \end{bmatrix}.$$

Due to the dangerous environment, the batch reaction cannot be precisely measured generating very little data samples [8]. Nevertheless, there exists a similar batch reactor that is less dangerous to access and to gather data. The basis for its process is based upon Rosenbrock's batch reactor, therefore it can be classified as a source system [9]. The system matrices are as follow:

$$\hat{A} = \begin{bmatrix} 1.178 & -0.161 & 4.511 & -4.403 \\ -0.851 & -2.661 & -0.011 & 0.261 \\ 1.076 & 4.335 & -7.560 & 4.382 \\ 0 & 4.335 & 1.089 & -1.849 \end{bmatrix}, \hat{B} = \begin{bmatrix} 0.004 & -0.087 \\ 6.467 & 0.001 \\ 1.213 & -3.235 \\ 2.213 & -0.016 \end{bmatrix}.$$

and the number of experiments and data length are predefined, i.e. $N_r = 10$, $N_p = 10$, $T = 2$.

Comparing to the virtual system, it can be observed that the target and source matrices of the physical system have slightly different values. Using both systems allows us to better understand the behavior of the identification error under various conditions.

6 Analysis of LS identification error

When running the codes for Stages 1-4 for both examples, even though the matrices vary, the results show commonalities. For the virtual system, the identification error excluding noise equals $||\theta_{LS} - \theta|| = 2.0958 * 10^{-15}$ whereas for the physical system the value is $||\theta_{LS} - \theta|| = 0.85522 * 10^{-15}$.

As it can be observed, in both examples its value is negligible, therefore the identification is very precise which means that the identified θ_{LS} is very close to real θ. However, when noise is added to the system, the precise identification becomes very difficult to obtain. Moreover, the results should change drastically. Let us prove this assumption.

For $\sigma_w^2 = 1$ and $N_r = 10$, the identification error equals

1. $||\theta_{LS} - \theta|| = 0.0879$ for the virtual system,
2. $||\theta_{LS} - \theta|| = 0.4441$ for the physical system.

Comparing to the previous case where noise was excluded, the error has now a crucial impact on the dynamics of the identified system. The reason is that noise is not scaled by θ thus cannot be controlled within the system [10]. Unfortunately, external disturbances are present in every system [3]. This code is designed to control them by means of other parameters which will be shown further.

Assuming one has access to a greater number of experiments and/or can scale the energy of the noise, then it should have an influence on the behaviour of the identification error. Running the code of Stage 3 for both systems will result in following plots (see Figures 1 and 2).

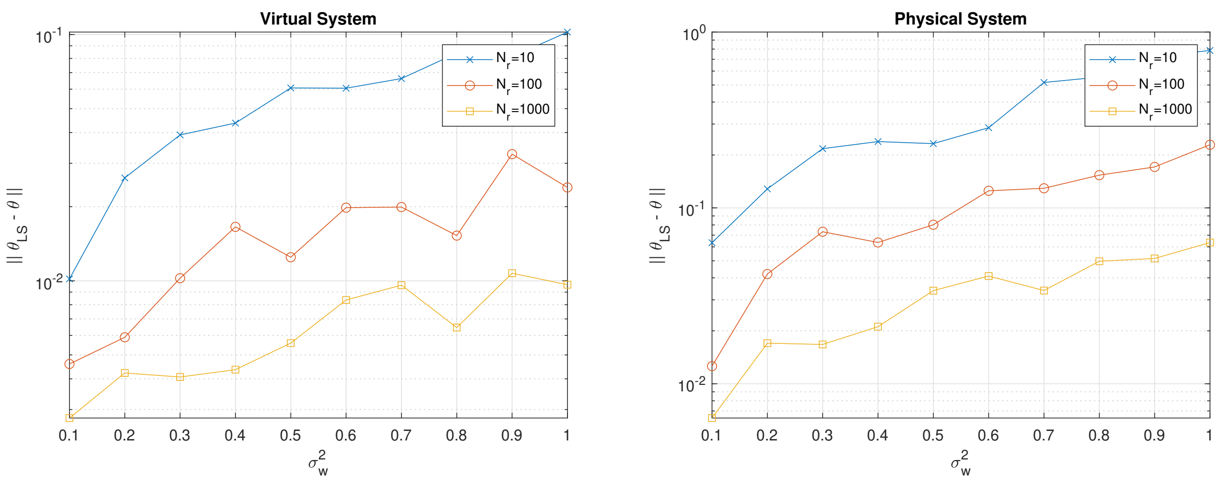

Figure 1: Identification error for the increasing energy of the noise

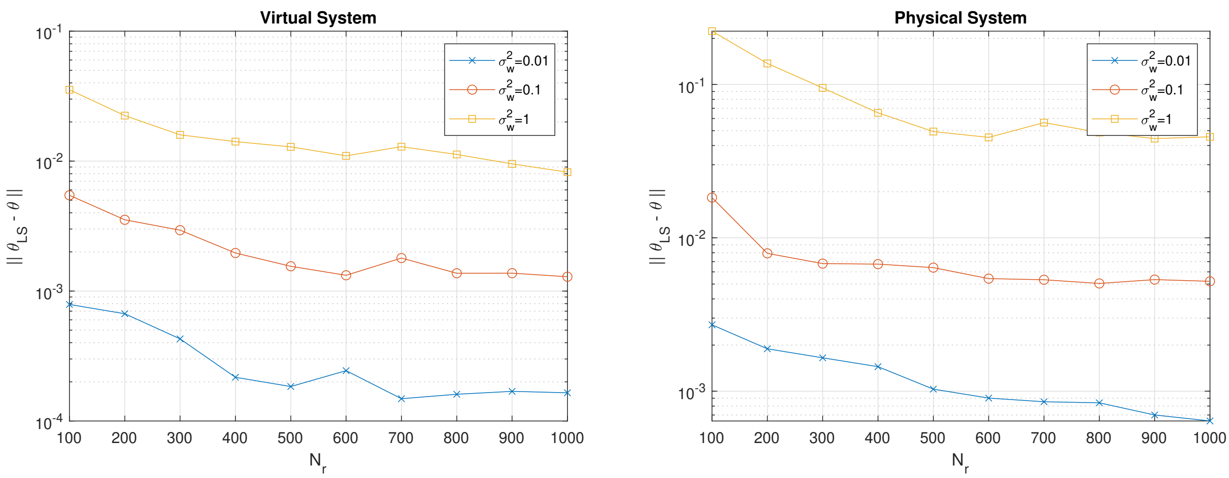

Figure 2: Identification error for the increasing number of experiments

Depending on the value of the energy of the noise, the identification error changes. The relationship is that as the energy increases so does the error (see Figure 1). It is based on the fact that higher disturbances to the system are more difficult to control resulting in greater error [3]. On the other hand, with increasing number of experiments, the identification error decreases (Figure 2). The reason is that more data samples are generated, which in consequence gives more accurate measurements [11].
Note that due to the fact that data is generated randomly (i.i.d Gaussian) for each experiment, the increment of curves in the graph is not ideal, however the increasing/decreasing tendencies are correct.

Now, it is established that the more experiments are withdrawn from the system, the more accurate is the identification. However, often acquiring a lot of actual system's samples is infeasible (as in 5.2). With a limited number of experiments, the accurate identification is difficult to obtain. Therefore, one ought to seek help from a source system which shares similar dynamics. Let us investigate the behavior of the error when 10 complementary experiments are added from the source system.

For $N_r = 10$ and $N_p = 10$, the identification error equals

1. $||\theta_{LS} - \theta|| = 0.0659$ for the virtual system,
2. $||\theta_{LS} - \theta|| = 1.3827$ for the physical system.

While for the virtual system with almost identical target and source matrices the additional data samples decreased the error, for the physical system the result was opposite. Instead of giving more reliable data samples, the divergence between the target and source matrices δ results in auxiliary samples being less informative consequently impeding an accurate identification.

According to the Theory Section 3, the influence of source system can be reduced by introducing a weigh parameter q. Assigning a specific weight to the source system's samples allows for including a higher number of auxiliary data without distorting the system identification [7]. In the next section, the system identification will be performed using *weighted least squares* (WLS) approach to observe the impact of the weight assignment on the identification error.

7 Analysis of WLS identification error

Let us repeat the previous code, however this time assigning a specific weight factor to the source system's samples.

For $N_r = 10$, $N_p = 10$ and $q = 0.3$, the identification error equals

1. $||\theta_{WLS} - \theta|| = 0.0465$ for the virtual system,
2. $||\theta_{WLS} - \theta|| = 0.8599$ for the physical system.

Assigning the weight factor to the source system's samples reduced the error in both cases. Therefore, in order to minimise the identification error, one should access more data samples with less emphasis on the source system. Furthermore, the objective is to find the most convenient combination of the number of experiments and the value of weight parameter.

To fully investigate the influence of the weight assignment, a detailed analysis of the identification error will be provided. It includes various simulations of the identification error in terms of increasing amount of actual system's samples, auxiliary system's samples and the weight parameter assigned to these samples. The aim of the analysis is to identify the impact of certain parameters on the behavior of the error so that the identification is as much precise as possible.

Since the identification error consists of three terms, i.e. $||WQZ^\top||$, $||\Delta QZ^\top||$, $||(ZQZ^\top)^{-1}||$ (see Equation (10) in Theory Section 3), each term influences the error separately.

The first term corresponds to the error due to noise from both target and source systems. Unfortunately, the noise cannot be directly controlled by the system. The variable that is controllable is the weight parameter q.

The second term corresponds to the error due to differences in the target and source systems' matrices. Having two perspectives of both virtual and physical systems will illustrate how the divergence between the target and source systems' matrices influences the identification error.

In general, the most optimal scenario is to have at least one conditions satisfied, i.e. small σ_w^2 (less noisy source system) and/or small δ (negligible difference between the target and source systems), and/or large N_p (numerous samples from the source system). In such cases, the auxiliary samples become more informative [1].

Let us assume two possible scenarios.

7.1 Scenario 1: Both N_r and N_p are increasing

In the *Scenario 1* it is assumed that the number of experiments from both target and source systems is increasing. However, due to the fact that acquiring data samples from the target system is difficult, the number of rollouts from the auxiliary system is set to $N_p = 3N_r$. In other words, each rollout collected from the target system is complemented by three rollouts from the source system.

Executing the code from the last section of Appendix A outputs two graphs - one for the virtual system and one for the physical system - that are illustrated in Figure 3.

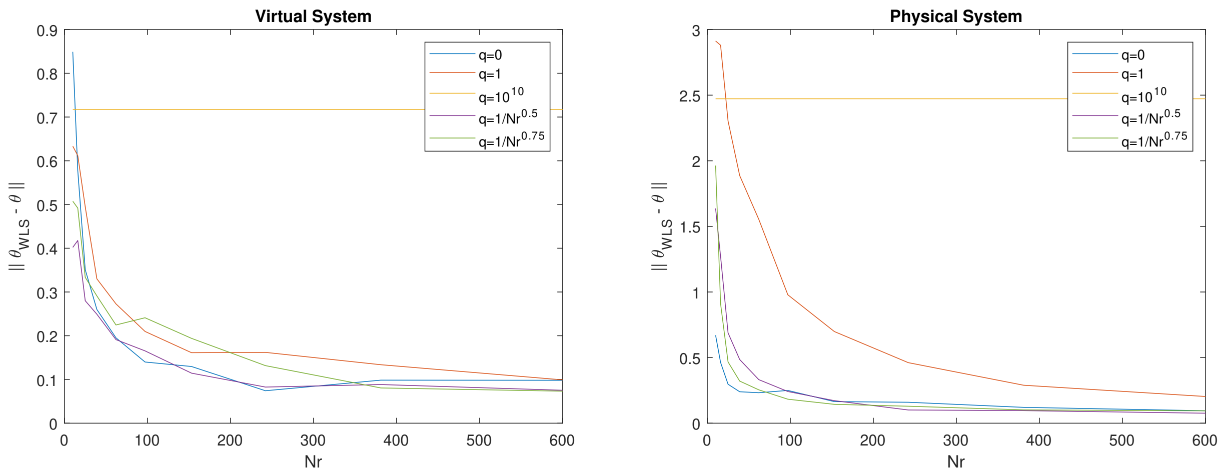

Figure 3: *Scenario 1*: Both N_r and N_p are increasing

As expected, in both cases the error tends to decrease over the increased number of experiments. Nevertheless, numerous differences can be distinguished when comparing these systems. First major difference is the magnitude of the error. For the virtual system the value varies within the limits of 0.9 to 0.1 whereas for the physical system the value even reaches 3. The explanation is rooted in the Equation (10). Since, the physical system has more differences in target and source matrices, i.e. larger δ, the error due to term $||\Delta Q Z^\top||$ will be higher than for the virtual system.

In the case of the virtual system, when N_r is small, the identification error for $q = 0$ (data leveraged only from the target system) has the greatest value. It is due to the fact that there is not enough data to support the precise identification [7]. As the N_r increases, the error rapidly decreases because the system starts to gather sufficient data. In order to decrease the error at the early stage, one should increase the value of q. Setting $q > 0$ incorporates the samples from the auxiliary system to the equations giving more precise identification. However, the emphasis on the source system samples cannot be excessive, otherwise the output is adverse. In contrast, when $q = 10^{10}$, the system considers almost exclusively the samples from the source system. It results in a constant error throughout the graph.

The Figure 3 also shows that the most optimal weight factor q includes N_r in its formula. As the N_r increases, the q should decrease exponentially in order to avoid the auxiliary data becoming dominant. For both $q = \frac{1}{\sqrt{N_r}}$ and $q = \frac{1}{\sqrt[\frac{4}{3}]{N_r}}$ the identification error is exponentially decreasing for all N_r.

Considering the physical system, the selection of the weight parameter q is even more crucial then for the virtual system because of the differences in the target and source system matrices (δ). For $q = 1$ and $q = 10^{10}$ the system incurs more error than not using the the auxiliary data ($q = 0$). Furthermore, the emphasis on the auxiliary data should be reduced. Similarly to the virtual system, the most optimal curves include $q = \frac{1}{\sqrt{N_r}}$ and $q = \frac{1}{\sqrt[\frac{4}{3}]{N_r}}$ because the specific weight q decreases as N_r increases, consequently outputting the desired behavior.

The conclusions of the *Scenario 1* are following. When N_p and N_r are both increasing linearly ($N_p = 3N_r$), using a specific weight q helps to reduce the system identification error. As N_r is small, the system leverages data from the auxiliary system and over increasing N_r the system starts to reduce excessive bias from the auxiliary system [1].

7.2 Scenario 2: N_p is fixed and N_r is increasing

Scenario 2 assumes that the number of experiments from the source systems is fixed at $N_p = 2400$ and only the number of experiments from the target system N_r is increasing. We would like to investigate the behavior of the identification error when the main source of data comes from the auxiliary system.

Executing the code from the last section of Appendix A outputs two graphs - one for the virtual system and one for the physical system - that are illustrated in Figure 4.

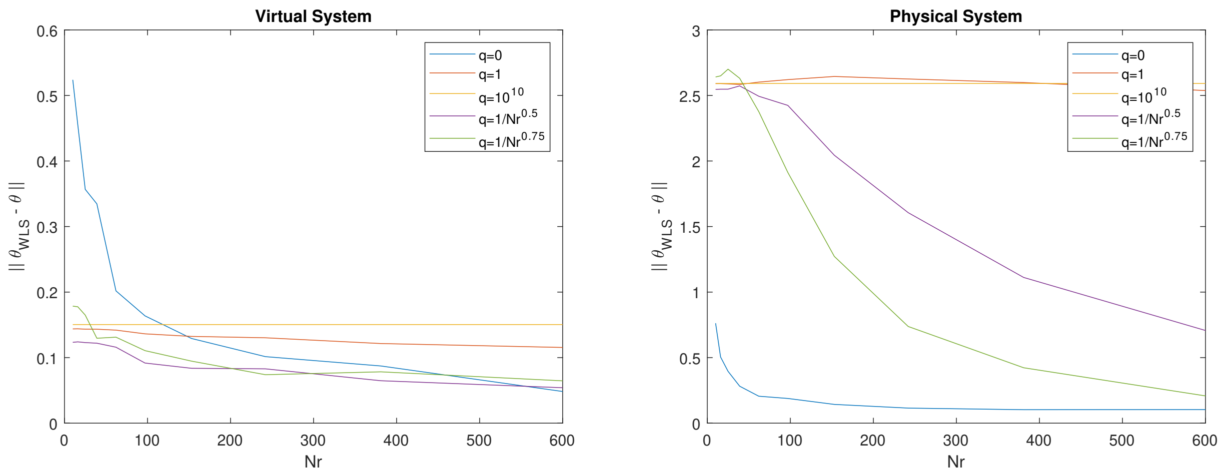

Figure 4: *Scenario 2*: N_p is fixed, N_r is increasing

Similarly to the *Scenario 1* (7.1), the error tends to decrease as the number of experiments increases. As established in Section 6, having access to more experiments provides more accurate identification. Another similarity is the difference in the error's magnitude between the systems. Once again, this can be explained by the Equation (10), namely that the physical system has more differences in target and source matrices, therefore the error due to term $||\Delta Q Z^\top||$ will be higher than for the virtual system.

When looking at the virtual system (Figure 4), one can observe that when there is little data from the target system, i.e. N_r is small, identifying the system by leveraging data only from the target system ($q = 0$) leads to a high identification error [7]. As the N_r increases, the error starts to slowly flatten towards a smaller value, however the deceleration rate is lower than for the *Scenario 1.* It emerges from the auxiliary samples that are superior to the target system's samples ($N_p = 2400$).

In order to decrease the error at the early stage, one should increase the value of q. Setting $q > 0$ outputs a smaller identification error (see Figure 4). During the initial phase (small N_r) using the auxiliary samples as the main source of data helps reduce the error because of the marginal δ. Even for the extreme value of $q = 10^{10}$, the identification at the early stage ($N_r \leq 150$) is more accurate than for $q = 0$. However, the weight parameter q cannot be too large, otherwise the error will remain constant over the increasing N_r. Thus, establishing a connection between the parameters q and N_r happens to be the most convenient. As the N_r increases, the q should decrease exponentially so that the influence of the auxiliary data decreases, too. For both $q = \frac{1}{\sqrt{N_r}}$ and $q = \frac{1}{\sqrt[4]{N_r}}$ the identification error has the smallest values among all curves.

While this scenario can be beneficial for the virtual system, it cannot be incorporated when the difference between the target and source system matrices δ is large. In contrast, when the physical system is subjected to identification, the error is hardly controllable by any value of q (see Figure 4) because it is dominated by Δ in $||\Delta Q Z^\top||$.

The outcome of *Scenario 2* considers the situation when N_p is large and δ is small. At the initial phase, i.e. when N_r is small. Since there is not enough target system's data to support the accurate identification, setting q to high number is the solution to the issue. Emphasising the auxiliary samples reduces the identification error until the the target system becomes more informative. As N_r is becoming larger, the need for complimentary data decreases. The system has enough data to perform identification without auxiliary samples. Therefore, the weight parameter q shall be lowered so that the identification error due to the term $||\Delta Q Z^\top||$ is reduced.

8 Conclusion

In this work, I have performed an identification of a certain LTI system with limited access to data by complementing it with additional data from an auxiliary system. The research has proven that leveraging data from a similar system can reduce the identification error in certain situations (scenarios). Due to the fact that the error consist of several terms, it can be controlled within the parameters that dictates its behaviour. We have learned that by assigning a specific weight to the auxiliary samples one can control the precision of identification especially when lacking in the number of experiments from the target system. As more experiments become accessible, the contribution of auxiliary samples is less significant. Moreover, finding a right proportion between the weight parameter and the number of experiments reduces the identification error, in consequence providing an accurate identification.

References

[1] Lei Xin, Lintao Ye, George Chiu, and Shreyas Sundaram. Identifying the dynamics of a system by leveraging data from similar systems. *2022 American Control Conference (ACC)*, pages 818–824, 6 2022.

[2] Rolf Isermann and Marco Münchhof. *Introduction.* Springer Berlin Heidelberg, 2011.

[3] Torsten Söderström. Errors-in-variables methods in system identification. *Automatica*, 43:939–958, 6 2007.

[4] Ivan Y. Tyukin, Alexander N. Gorban, Konstantin I. Sofeykov, and Ilya Romanenko. Knowledge transfer between artificial intelligence systems. *Frontiers in Neurorobotics*, 12, 8 2018.

[5] Honggui Han, Hongxu Liu, Cuili Yang, and Junfei Qiao. Transfer learning algorithm with knowledge division level. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2022.

[6] Lei Xin, George Chiu, and Shreyas Sundaram. Learning the dynamics of autonomous linear systems from multiple trajectories. *2022 American Control Conference (ACC)*, pages 3955–3960, 6 2022.

[7] Salar Fattahi and Somayeh Sojoudi. Data-driven sparse system identification. *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 462–469, 10 2018.

[8] Michael Green and David JN Limebeer. *Linear robust control.* Courier Corporation, 2012.

[9] Claudio De Persis and Pietro Tesi. Formulas for data-driven control: Stabilization, optimality, and robustness. *IEEE Transactions on Automatic Control*, 65(3):909–924, 2019.

[10] Lingyu Chen, Hang Yin, Linjie Wang, and Guomin Ji. A disguising method for linear vector data based on least square method. *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 2245–2249, 3 2021.

[11] Hailin Liu, Fangqing Gu, and Zixian Lin. Auto-sharing parameters for transfer learning based on multi-objective optimization. *Integrated Computer-Aided Engineering*, 28:295–307, 6 2021.

[12] Xiao-Jun Zeng and M.G. Singh. Fuzzy bounded least squares method for systems identification. *Proceedings of 28th Southeastern Symposium on System Theory*, pages 61–65, 1996.

[13] V. Karanko and M. Honkala. Least squares solution of nearly square overdetermined sparse linear systems. *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, pages IV–830–IV–833, 2002.

[14] Max Simchowitz, Ross Boczar, and Benjamin Recht. Learning linear dynamical systems with semi-parametric least squares. pages 2714–2802, 2019.

A Appendix: Matlab Codes

A.1 Stage 1: System Identification without noise

```
clear; clc;

%% first step: create data
%  assume you know the system (A,B)
%  use this (A,B) to creat data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

Ad = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bd = [1 0.5; 0.5 1; 0.5 0.5];


Nr = 10;                 % no of experiments
T  = 100;                % data length
uu = randn(dimM*Nr,T);   % pre-define the input sequence
xx = zeros(dimN*Nr,T+1);     % empty state array
zz = zeros((dimM+dimN)*Nr,T); % empty state-input array
xx(:,1) = randn(dimN*Nr,1);  % random state input
X = zeros(dimN,Nr*T);        % empty batch matrix for state samples
Z = zeros(dimN+dimM,Nr*T);   % empty batch matrix for state-input samples

% data creating loop
for ii = 1:Nr          % outer loop for each expriment
    for ij = 1:T       % inner loop for each data length

        xr1 = (ii-1)*dimN+1;        % upper boundary for state data
        xr2 = ii*dimN;              % lower boundary for state data
        ur1 = (ii-1)*dimM+1;        % upper boundary for input data
        ur2 = ii*dimM;              % lower boundary for input data
        zr1 = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
        zr2 = ii*(dimN+dimM);       % upper boundary for state-input data

        xx(xr1:xr2,ij+1) = Ad*xx(xr1:xr2,ij) + Bd*uu(ur1:ur2,ij);
        zz(zr1:zr2,ij) = [xx(xr1:xr2,ij)' uu(ur1:ur2,ij)']';

        Xi = flip(xx(xr1:xr2,2:T+1),2); % combined state data length for single experiment
        Zi = flip(zz(zr1:zr2,1:T),2);   % combined state-input data length for single experiment
    end

    xxr1 = (ii-1)*T+1; % left boundary
    xxr2 = ii*T;       % right boundary

    X(:,xxr1:xxr2) = Xi; % combined state data length for all experiments
    Z(:,xxr1:xxr2) = Zi; % combined state-input data length for all experiments

end

Dif = X - [Ad Bd]*Z;


%% second step: use data to identify the system (A,B)
%  X1 = Ad*X0 + Bd*U0
%  X1 = [Ad Bd]*[X0' U0']'

% ABidy is the identified system
ABidy =  X*Z'*inv(Z*Z'); % analytical solution for least square method

% divide columns to obtain matrices A,B
Aidy = ABidy(:,1:dimN);
Bidy = ABidy(:,dimN+1:dimN+dimM);

% compare the difference between [Aidy Bidy] and [Ad Bd]
ABdif =  [Aidy-Ad Bidy-Bd];

error = max(svd(ABdif))
```

A.2 Stage 2: System Identification with noise

```
clear; clc;

%% first step: create data
%  assume you know the system (A,B)
%  use this (A,B) to creat data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

Ad = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bd = [1 0.5; 0.5 1; 0.5 0.5];

ed = 1;               % energy of the noise
```

```
Nr = 10;                  % no of experiments
T  = 100;                 % data length
uu = randn(dimM*Nr,T);    % pre-define the input sequence
ww = randn(dimN*Nr,T)*ed; % pre-define the noise sequence
xx = zeros(dimN*Nr,T+1);      % empty state array
zz = zeros((dimM+dimN)*Nr,T); % empty state-input array
xx(:,1) = randn(dimN*Nr,1);   % random state input
X = zeros(dimN,Nr*T);         % empty batch matrix for state samples
W = zeros(dimN,Nr*T);         % empty batch matrix for noise samples
Z = zeros(dimN+dimM,Nr*T);    % empty batch matrix for state-input samples

% data creating loop
for ii = 1:Nr           % outer loop for each expriment
    for ij = 1:T        % inner loop for each data length

        xr1 = (ii-1)*dimN+1;        % upper boundary for state data
        xr2 = ii*dimN;              % lower boundary for state data
        ur1 = (ii-1)*dimM+1;        % upper boundary for input data
        ur2 = ii*dimM;              % lower boundary for input data
        zr1 = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
        zr2 = ii*(dimN+dimM);       % upper boundary for state-input data

        xx(xr1:xr2,ij+1) = Ad*xx(xr1:xr2,ij) + Bd*uu(ur1:ur2,ij) + ww(xr1:xr2,ij);
        zz(zr1:zr2,ij) = [xx(xr1:xr2,ij)' uu(ur1:ur2,ij)']';

        Xi = flip(xx(xr1:xr2,2:T+1),2); % combined state data length for single experiment
        Wi = flip(ww(xr1:xr2,1:T),2);   % combined noise data length for single experiment
        Zi = flip(zz(zr1:zr2,1:T),2);   % combined state-input data length for single experiment
    end

    xxr1 = (ii-1)*T+1; % left boundary
    xxr2 = ii*T;       % right boundary

    X(:,xxr1:xxr2) = Xi; % combined state data length for all experiments
    W(:,xxr1:xxr2) = Wi; % combined noise data length for all experiments
    Z(:,xxr1:xxr2) = Zi; % combined state-input data length for all experiments

end

Dif = X - [Ad Bd]*Z - W;


%% second step: use data to identify the system (A,B)
%  X1 = Ad*X0 + Bd*U0
%  X1 = [Ad Bd]*[X0' U0']'

% ABidy is the identified system
ABidy =  X*Z'*inv(Z*Z'); % analytical solution for least square method

% divide columns to obtain matrices A,B
Aidy = ABidy(:,1:dimN);
Bidy = ABidy(:,dimN+1:dimN+dimM);

% compare the difference between [Aidy Bidy] and [Ad Bd]
ABdif =  [Aidy-Ad Bidy-Bd];

error = max(svd(ABdif))
```

A.3 Stage 3: Plotting identification error as function of σ_w^2 and N_r

```
clear; clc;

%% first step: create data
%  assume you know the system (A,B)
%  use this (A,B) to creat data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

Ad = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bd = [1 0.5; 0.5 1; 0.5 0.5];


xEd = 0.1:0.1:1;                % energy of the noise
lenEd = length(xEd);
yDif  = zeros(1,length(xEd));
Nr = 10;                  % no of experiments
T  = 100;                 % data length
uu = randn(dimM*Nr,T);    % pre-define the input sequence
xx = zeros(dimN*Nr,T+1);      % empty state array
zz = zeros((dimM+dimN)*Nr,T); % empty state-input array
xx(:,1) = randn(dimN*Nr,1);   % random state input
X = zeros(dimN,Nr*T);         % empty batch matrix for state samples
W = zeros(dimN,Nr*T);         % empty batch matrix for noise samples
Z = zeros(dimN+dimM,Nr*T);    % empty batch matrix for state-input samples

% data creating loop
for in = 1:lenEd
```

```
    for ii = 1:Nr                % outer loop for each expriment
        for ij = 1:T             % inner loop for each data length
            xr1 = (ii-1)*dimN+1;        % upper boundary for state data
            xr2 = ii*dimN;              % lower boundary for state data
            ur1 = (ii-1)*dimM+1;        % upper boundary for input data
            ur2 = ii*dimM;              % lower boundary for input data
            zr1 = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
            zr2 = ii*(dimN+dimM);       % upper boundary for state-input data

            ww = randn(dimN*Nr,T)*xEd(in); % pre-define the noise sequence`
            xx(xr1:xr2,ij+1) = Ad*xx(xr1:xr2,ij) + Bd*uu(ur1:ur2,ij) + ww(xr1:xr2,ij);
            zz(zr1:zr2,ij) = [xx(xr1:xr2,ij)' uu(ur1:ur2,ij)']';

            Xi = flip(xx(xr1:xr2,2:T+1),2); % combined state data length for single experiment
            Wi = flip(ww(xr1:xr2,1:T),2);   % combined noise data length for single experiment
            Zi = flip(zz(zr1:zr2,1:T),2);   % combined state-input data length for single experiment
        end

        xxr1 = (ii-1)*T+1; % left boundary
        xxr2 = ii*T;       % right boundary

        X(:,xxr1:xxr2) = Xi; % combined state data length for all experiments
        W(:,xxr1:xxr2) = Wi; % combined noise data length for all experiments
        Z(:,xxr1:xxr2) = Zi; % combined state-input data length for all experiments
    end

    ABidy =  X*Z'*inv(Z*Z'); % analytical solution for least square method

    Aidy = ABidy(:,1:dimN);           % obtain matrices A,B
    Bidy = ABidy(:,dimN+1:dimN+dimM);

    ABdif =  [Aidy-Ad Bidy-Bd];  % compare the difference between [Aidy Bidy] and [Ad Bd]
    yDif(in) = max(svd(ABdif));
end

Dif = X - [Ad Bd]*Z - W;

plot(xEd, yDif);


%% plot

load 'Nr10.mat'
semilogy(xEd, yDif,'-','markersize',6); hold on;

load 'Nr100.mat'
semilogy(xEd, yDif,'-','markersize',6); hold on;

load 'Nr1000.mat'
semilogy(xEd, yDif,'-','markersize',6); hold on;

title('Identification error for different no of experiments');
xlabel('ed');
ylabel('|| \theta_L_S - \theta ||');
legend('Nr=10','Nr=100','Nr=1000');

grid on;
```

```
clear; clc;

%% first step: create data
%  assume you know the system (A,B)
%  use this (A,B) to creat data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

Ad = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bd = [1 0.5; 0.5 1; 0.5 0.5];


Ed = 1;                % energy of the noise
xNr   = 100:100:1000;
lenNr = length(xNr);
yDif  = zeros(1,length(xNr));
%Nr = 10;                  % no of experiments
T  = 100;                 % data length
uu = randn(dimM*xNr(lenNr),T);    % pre-define the input sequence
ww = randn(dimN*xNr(lenNr),T)*Ed; % pre-define the noise sequence
xx = zeros(dimN*xNr(lenNr),T+1);      % empty state array
zz = zeros((dimM+dimN)*xNr(lenNr),T); % empty state-input array
xx(:,1) = randn(dimN*xNr(lenNr),1);   % random state input
X = zeros(dimN,xNr(lenNr)*T);         % empty batch matrix for state samples
W = zeros(dimN,xNr(lenNr)*T);         % empty batch matrix for noise samples
Z = zeros(dimN+dimM,xNr(lenNr)*T);    % empty batch matrix for state-input samples

% data creating loop
for in = 1:lenNr
    for ii = 1:xNr(in)           % outer loop for each expriment
        for ij = 1:T             % inner loop for each data length
```

```
            xr1 = (ii-1)*dimN+1;        % upper boundary for state data
            xr2 = ii*dimN;              % lower boundary for state data
            ur1 = (ii-1)*dimM+1;        % upper boundary for input data
            ur2 = ii*dimM;              % lower boundary for input data
            zr1 = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
            zr2 = ii*(dimN+dimM);       % upper boundary for state-input data

            xx(xr1:xr2,ij+1) = Ad*xx(xr1:xr2,ij) + Bd*uu(ur1:ur2,ij) + ww(xr1:xr2,ij);
            zz(zr1:zr2,ij) = [xx(xr1:xr2,ij)' uu(ur1:ur2,ij)']';

            Xi = flip(xx(xr1:xr2,2:T+1),2); % combined state data length for single experiment
            Wi = flip(ww(xr1:xr2,1:T),2);   % combined noise data length for single experiment
            Zi = flip(zz(zr1:zr2,1:T),2);   % combined state-input data length for single experiment
        end

        xxr1 = (ii-1)*T+1; % left boundary
        xxr2 = ii*T;       % right boundary

        X(:,xxr1:xxr2) = Xi; % combined state data length for all experiments
        W(:,xxr1:xxr2) = Wi; % combined noise data length for all experiments
        Z(:,xxr1:xxr2) = Zi; % combined state-input data length for all experiments
    end

    ABidy =  X*Z'*inv(Z*Z'); % analytical solution for least square method

    Aidy = ABidy(:,1:dimN);           % obtain matrices A,B
    Bidy = ABidy(:,dimN+1:dimN+dimM);

    ABdif =  [Aidy-Ad Bidy-Bd];  % compare the difference between [Aidy Bidy] and [Ad Bd]
    yDif(in) = max(svd(ABdif));
end

Dif = X - [Ad Bd]*Z - W;

plot(xNr, yDif);

%% plot

load 'Ed0,01.mat'
semilogy(xNr, yDif,'-','markersize',6); hold on;

load 'Ed0,1.mat'
semilogy(xNr, yDif,'-','markersize',6); hold on;

load 'Ed1.mat'
semilogy(xNr, yDif,'-','markersize',6); hold on;

title('Identification error for different noise energies');
xlabel('Nr');
ylabel('|| \theta_L_S - \theta ||');
legend('ed=0.01','ed=0.1','ed=1');

grid on;
```

A.4 Stage 4: LS for target and source systems

```
clear; clc;

%% first step: create data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

% target system matrices
Adash = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bdash = [1 0.5; 0.5 1; 0.5 0.5];

% source system matrices
Ahat = [0.7 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bhat = [1.1 0.5; 0.5 1; 0.5 0.5];

deltaA = Ahat - Adash;
deltaB = Bhat - Bdash;


% target system dataset
ed = 1;                  % energy of the noise
Nr = 10;                    % no of experiments
T  = 100;                   % data length
uudash = randn(dimM*Nr,T);    % pre-define the input sequence
wwdash = randn(dimN*Nr,T)*ed; % pre-define the noise sequence
xxdash = zeros(dimN*Nr,T+1);      % empty state array
zzdash = zeros((dimM+dimN)*Nr,T); % empty state-input array
xxdash(:,1) = randn(dimN*Nr,1);   % random state input
Xdash = zeros(dimN,Nr*T);         % empty batch matrix for state samples
Wdash = zeros(dimN,Nr*T);         % empty batch matrix for noise samples
Zdash = zeros(dimN+dimM,Nr*T);    % empty batch matrix for state-input samples

```

```
% source system dataset
Np = 10;                   % no of experiments
uuhat = randn(dimM*Np,T);    % pre-define the input sequence
wwhat = randn(dimN*Np,T)*ed; % pre-define the noise sequence
xxhat = zeros(dimN*Np,T+1);      % empty state array
zzhat = zeros((dimM+dimN)*Np,T); % empty state-input array
xxhat(:,1) = randn(dimN*Np,1);   % random state input
Xhat = zeros(dimN,Np*T);         % empty batch matrix for state samples
What = zeros(dimN,Np*T);         % empty batch matrix for noise samples
Zhat = zeros(dimN+dimM,Np*T);    % empty batch matrix for state-input samples
Delta = zeros(dimN,(Nr+Np)*T);   % empty batch matrix for Delta samples

% data creating loop for target system
for ii = 1:Nr           % outer loop for each expriment
    for ij = 1:T        % inner loop for each data length

        xr1dash = (ii-1)*dimN+1;        % upper boundary for state data
        xr2dash = ii*dimN;              % lower boundary for state data
        ur1dash = (ii-1)*dimM+1;        % upper boundary for input data
        ur2dash = ii*dimM;              % lower boundary for input data
        zr1dash = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
        zr2dash = ii*(dimN+dimM);       % upper boundary for state-input data

        xxdash(xr1dash:xr2dash,ij+1) = Adash*xxdash(xr1dash:xr2dash,ij) + Bdash*uudash(ur1dash:ur2dash,ij) + wwdash(xr1dash:xr2dash,ij);
        zzdash(zr1dash:zr2dash,ij) = [xxdash(xr1dash:xr2dash,ij)' uudash(ur1dash:ur2dash,ij)']';

        Xidash = flip(xxdash(xr1dash:xr2dash,2:T+1),2); % combined state data length for single experiment
        Widash = flip(wwdash(xr1dash:xr2dash,1:T),2);   % combined noise data length for single experiment
        Zidash = flip(zzdash(zr1dash:zr2dash,1:T),2);   % combined state-input data length for single experiment
    end

    xxr1dash = (ii-1)*T+1; % left boundary
    xxr2dash = ii*T;       % right boundary

    Xdash(:,xxr1dash:xxr2dash) = Xidash; % combined state data length for all experiments
    Wdash(:,xxr1dash:xxr2dash) = Widash; % combined noise data length for all experiments
    Zdash(:,xxr1dash:xxr2dash) = Zidash; % combined state-input data length for all experiments

end

% data creating loop for source system
for jj = 1:Np           % outer loop for each expriment
    for ji = 1:T        % inner loop for each data length

        xr1hat = (jj-1)*dimN+1;        % upper boundary for state data
        xr2hat = jj*dimN;              % lower boundary for state data
        ur1hat = (jj-1)*dimM+1;        % upper boundary for input data
        ur2hat = jj*dimM;              % lower boundary for input data
        zr1hat = (jj-1)*(dimN+dimM)+1; % upper boundary for state-input data
        zr2hat = jj*(dimN+dimM);       % upper boundary for state-input data

        xxhat(xr1hat:xr2hat,ji+1) = (Adash+deltaA)*xxhat(xr1hat:xr2hat,ji) + (Bdash+deltaB)*uuhat(ur1hat:ur2hat,ji) + wwhat(xr1hat:xr2hat,ji);
        zzhat(zr1hat:zr2hat,ji) = [xxhat(xr1hat:xr2hat,ji)' uuhat(ur1hat:ur2hat,ji)']';

        delta = [deltaA deltaB];

        Xihat = flip(xxhat(xr1hat:xr2hat,2:T+1),2); % combined state data length for single experiment
        Wihat = flip(wwhat(xr1hat:xr2hat,1:T),2);   % combined noise data length for single experiment
        Zihat = flip(zzhat(zr1hat:zr2hat,1:T),2);   % combined state-input data length for single experiment
        Deltai = delta*Zihat;                       % combined Delta for single experiment
    end

    xxr1hat = (jj-1)*T+1; % left boundary
    xxr2hat = jj*T;       % right boundary

    Xhat(:,xxr1hat:xxr2hat) = Xihat;    % combined state data length for all experiments
    What(:,xxr1hat:xxr2hat) = Wihat;    % combined noise data length for all experiments
    Zhat(:,xxr1hat:xxr2hat) = Zihat;    % combined state-input data length for all experiments
    Delta(:,Nr*T+xxr1hat:Nr*T+xxr2hat) = Deltai;  % combined Delta for all experiments

end

X = [Xdash Xhat];
Z = [Zdash Zhat];
W = [Wdash What];


Dif = X - [Adash Bdash]*Z - W - Delta;


%% second step: use data to identify the system (A,B)
%  X1 = Ad*X0 + Bd*U0
%  X1 = [Ad Bd]*[X0' U0']'

% ABidy is the identified system
ABidy =  X*Z'*inv(Z*Z'); % analytical solution for least square method

% divide columns to obtain matrices A,B
Aidy = ABidy(:,1:dimN);
```

```
Bidy = ABidy(:,dimN+1:dimN+dimM);

% compare the difference between identified and target system matrices
ABdif =  [Aidy-Adash Bidy-Bdash];

error = max(svd(ABdif))
```

A.5 Stage 5: WLS for target and source systems

```
clear; clc;

%% first step: create data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

% target system matrices
Adash = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bdash = [1 0.5; 0.5 1; 0.5 0.5];

% source system matrices
Ahat = [0.7 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bhat = [1.1 0.5; 0.5 1; 0.5 0.5];

deltaA = Ahat - Adash;
deltaB = Bhat - Bdash;


% target system dataset
ed = 1;                % energy of the noise
Nr = 10;                  % no of experiments
T  = 100;                 % data length
uudash = randn(dimM*Nr,T);    % pre-define the input sequence
wwdash = randn(dimN*Nr,T)*ed; % pre-define the noise sequence
xxdash = zeros(dimN*Nr,T+1);      % empty state array
zzdash = zeros((dimM+dimN)*Nr,T); % empty state-input array
xxdash(:,1) = randn(dimN*Nr,1);   % random state input
Xdash = zeros(dimN,Nr*T);         % empty batch matrix for state samples
Wdash = zeros(dimN,Nr*T);         % empty batch matrix for noise samples
Zdash = zeros(dimN+dimM,Nr*T);    % empty batch matrix for state-input samples

% source system dataset
Np = 10;                  % no of experiments
uuhat = randn(dimM*Np,T);    % pre-define the input sequence
wwhat = randn(dimN*Np,T)*ed; % pre-define the noise sequence
xxhat = zeros(dimN*Np,T+1);      % empty state array
zzhat = zeros((dimM+dimN)*Np,T); % empty state-input array
xxhat(:,1) = randn(dimN*Np,1);   % random state input
Xhat = zeros(dimN,Np*T);         % empty batch matrix for state samples
What = zeros(dimN,Np*T);         % empty batch matrix for noise samples
Zhat = zeros(dimN+dimM,Np*T);    % empty batch matrix for state-input samples

Delta = zeros(dimN,(Nr+Np)*T);   % empty batch matrix for Delta samples

% data creating loop for target system
for ii = 1:Nr           % outer loop for each expriment
    for ij = 1:T        % inner loop for each data length

        xr1dash = (ii-1)*dimN+1;        % upper boundary for state data
        xr2dash = ii*dimN;              % lower boundary for state data
        ur1dash = (ii-1)*dimM+1;        % upper boundary for input data
        ur2dash = ii*dimM;              % lower boundary for input data
        zr1dash = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
        zr2dash = ii*(dimN+dimM);       % upper boundary for state-input data

        xxdash(xr1dash:xr2dash,ij+1) = Adash*xxdash(xr1dash:xr2dash,ij) + Bdash*uudash(ur1dash:ur2dash,ij) + wwdash(xr1dash:xr2dash,ij);
        zzdash(zr1dash:zr2dash,ij) = [xxdash(xr1dash:xr2dash,ij)' uudash(ur1dash:ur2dash,ij)']';

        Xidash = flip(xxdash(xr1dash:xr2dash,2:T+1),2); % combined state data length for single experiment
        Widash = flip(wwdash(xr1dash:xr2dash,1:T),2);   % combined noise data length for single experiment
        Zidash = flip(zzdash(zr1dash:zr2dash,1:T),2);   % combined state-input data length for single experiment
    end

    xxr1dash = (ii-1)*T+1; % left boundary
    xxr2dash = ii*T;       % right boundary

    Xdash(:,xxr1dash:xxr2dash) = Xidash; % combined state data length for all experiments
    Wdash(:,xxr1dash:xxr2dash) = Widash; % combined noise data length for all experiments
    Zdash(:,xxr1dash:xxr2dash) = Zidash; % combined state-input data length for all experiments

end

% data creating loop for source system
for jj = 1:Np           % outer loop for each expriment
    for ji = 1:T        % inner loop for each data length

        xr1hat = (jj-1)*dimN+1;        % upper boundary for state data
```

```
        xr2hat = jj*dimN;              % lower boundary for state data
        ur1hat = (jj-1)*dimM+1;        % upper boundary for input data
        ur2hat = jj*dimM;              % lower boundary for input data
        zr1hat = (jj-1)*(dimN+dimM)+1; % upper boundary for state-input data
        zr2hat = jj*(dimN+dimM);       % upper boundary for state-input data

        xxhat(xr1hat:xr2hat,ji+1) = (Adash+deltaA)*xxhat(xr1hat:xr2hat,ji) + (Bdash+deltaB)*uuhat(ur1hat:ur2hat,ji) + wwhat(xr1hat:xr2hat,ji);
        zzhat(zr1hat:zr2hat,ji) = [xxhat(xr1hat:xr2hat,ji)' uuhat(ur1hat:ur2hat,ji)']';

        delta = [deltaA deltaB];

        Xihat = flip(xxhat(xr1hat:xr2hat,2:T+1),2); % combined state data length for single experiment
        Wihat = flip(wwhat(xr1hat:xr2hat,1:T),2);   % combined noise data length for single experiment
        Zihat = flip(zzhat(zr1hat:zr2hat,1:T),2);   % combined state-input data length for single experiment

        Deltai = delta*Zihat;                       % combined Delta for single experiment
    end

    xxr1hat = (jj-1)*T+1; % left boundary
    xxr2hat = jj*T;       % right boundary

    Xhat(:,xxr1hat:xxr2hat) = Xihat;    % combined state data length for all experiments
    What(:,xxr1hat:xxr2hat) = Wihat;    % combined noise data length for all experiments
    Zhat(:,xxr1hat:xxr2hat) = Zihat;    % combined state-input data length for all experiments

    Delta(:,Nr*T+xxr1hat:Nr*T+xxr2hat) = Deltai;  % combined Delta for all experiments

end

X = [Xdash Xhat];
Z = [Zdash Zhat];
W = [Wdash What];



%% weighted least square

q  = 0.3;         % assigning weight
Qi = eye(T)*q;
Qhat = zeros(Np*T,Np*T);

for kk = 1:Np

    qq1 = (kk-1)*T+1;
    qq2 = kk*T;

    Qhat(qq1:qq2,qq1:qq2) = Qi; % Np times

end

Q = blkdiag(eye(Nr*T),Qhat);

Dif = X - [Adash Bdash]*Z - W - Delta;


%% second step: use data to identify the system (A,B)

% ABidy is the identified system
ABidy =  X*Q*Z'*inv(Z*Q*Z'); % analytical solution for least square method

% divide columns to obtain matrices A,B
Aidy = ABidy(:,1:dimN);
Bidy = ABidy(:,dimN+1:dimN+dimM);

% compare the difference between identified and target system matrices
ABdif =  [Aidy-Adash Bidy-Bdash];

error = max(svd(ABdif))

ABdif1 = W*Q*Z'*inv(Z*Q*Z')+Delta*Q*Z'*inv(Z*Q*Z');

diff = ABdif-ABdif1;
```

A.6 Analysis of WLS identification error

```
clear; clc;

%% first step: create data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

% target system matrices
Adash = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bdash = [1 0.5; 0.5 1; 0.5 0.5];

% source system matrices
```

```
Ahat = [0.7 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bhat = [1.1 0.5; 0.5 1; 0.5 0.5];

deltaA = Ahat - Adash;
deltaB = Bhat - Bdash;


% target system dataset
ed = 1;                 % energy of the noise
xNr = round(logspace(log10(10), log10(600), 10));      % no of experiments
lenNr = length(xNr);
yDif  = zeros(1,length(xNr));
%Nr = 10;               % no of experiments
T  = 2;                 % data length
uudash = randn(dimM*xNr(lenNr),T);        % pre-define the input sequence
wwdash = randn(dimN*xNr(lenNr),T)*ed;     % pre-define the noise sequence
xxdash = zeros(dimN*xNr(lenNr),T+1);      % empty state array
zzdash = zeros((dimM+dimN)*xNr(lenNr),T); % empty state-input array
xxdash(:,1) = randn(dimN*xNr(lenNr),1);   % random state input
Xdash = zeros(dimN,xNr(lenNr)*T);         % empty batch matrix for state samples
Wdash = zeros(dimN,xNr(lenNr)*T);         % empty batch matrix for noise samples
Zdash = zeros(dimN+dimM,xNr(lenNr)*T);    % empty batch matrix for state-input samples

% source system dataset
Np = 2400;                  % no of experiments
uuhat = randn(dimM*Np,T);        % pre-define the input sequence
wwhat = randn(dimN*Np,T)*ed;     % pre-define the noise sequence
xxhat = zeros(dimN*Np,T+1);      % empty state array
zzhat = zeros((dimM+dimN)*Np,T); % empty state-input array
xxhat(:,1) = randn(dimN*Np,1);   % random state input
Xhat = zeros(dimN,Np*T);         % empty batch matrix for state samples
What = zeros(dimN,Np*T);         % empty batch matrix for noise samples
Zhat = zeros(dimN+dimM,Np*T);    % empty batch matrix for state-input samples

Delta = zeros(dimN,(xNr(lenNr)+Np)*T);   % empty batch matrix for Delta samples

for in = 1:lenNr
    % data creating loop for target system
    for ii = 1:xNr(in)           % outer loop for each expriment
        for ij = 1:T        % inner loop for each data length

            xr1dash = (ii-1)*dimN+1;        % upper boundary for state data
            xr2dash = ii*dimN;              % lower boundary for state data
            ur1dash = (ii-1)*dimM+1;        % upper boundary for input data
            ur2dash = ii*dimM;              % lower boundary for input data
            zr1dash = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
            zr2dash = ii*(dimN+dimM);       % upper boundary for state-input data

            xxdash(xr1dash:xr2dash,ij+1) = Adash*xxdash(xr1dash:xr2dash,ij) + Bdash*uudash(ur1dash:ur2dash,ij) + wwdash(xr1dash:xr2dash,ij);
            zzdash(zr1dash:zr2dash,ij) = [xxdash(xr1dash:xr2dash,ij)' uudash(ur1dash:ur2dash,ij)']';

            Xidash = flip(xxdash(xr1dash:xr2dash,2:T+1),2); % combined state data length for single experiment
            Widash = flip(wwdash(xr1dash:xr2dash,1:T),2);   % combined noise data length for single experiment
            Zidash = flip(zzdash(zr1dash:zr2dash,1:T),2);   % combined state-input data length for single experiment
        end

        xxr1dash = (ii-1)*T+1; % left boundary
        xxr2dash = ii*T;       % right boundary

        Xdash(:,xxr1dash:xxr2dash) = Xidash; % combined state data length for all experiments
        Wdash(:,xxr1dash:xxr2dash) = Widash; % combined noise data length for all experiments
        Zdash(:,xxr1dash:xxr2dash) = Zidash; % combined state-input data length for all experiments

    end

    % data creating loop for source system
    for jj = 1:Np           % outer loop for each expriment
        for ji = 1:T        % inner loop for each data length

            xr1hat = (jj-1)*dimN+1;        % upper boundary for state data
            xr2hat = jj*dimN;              % lower boundary for state data
            ur1hat = (jj-1)*dimM+1;        % upper boundary for input data
            ur2hat = jj*dimM;              % lower boundary for input data
            zr1hat = (jj-1)*(dimN+dimM)+1; % upper boundary for state-input data
            zr2hat = jj*(dimN+dimM);       % upper boundary for state-input data

            xxhat(xr1hat:xr2hat,ji+1) = (Adash+deltaA)*xxhat(xr1hat:xr2hat,ji) + (Bdash+deltaB)*uuhat(ur1hat:ur2hat,ji) + wwhat(xr1hat:xr2hat,ji);
            zzhat(zr1hat:zr2hat,ji) = [xxhat(xr1hat:xr2hat,ji)' uuhat(ur1hat:ur2hat,ji)']';

            delta = [deltaA deltaB];

            Xihat = flip(xxhat(xr1hat:xr2hat,2:T+1),2); % combined state data length for single experiment
            Wihat = flip(wwhat(xr1hat:xr2hat,1:T),2);   % combined noise data length for single experiment
            Zihat = flip(zzhat(zr1hat:zr2hat,1:T),2);   % combined state-input data length for single experiment

            Deltai = delta*Zihat;                       % combined Delta for single experiment
        end

        xxr1hat = (jj-1)*T+1; % left boundary
```

```
        xxr2hat = jj*T;       % right boundary

        Xhat(:,xxr1hat:xxr2hat) = Xihat;    % combined state data length for all experiments
        What(:,xxr1hat:xxr2hat) = Wihat;    % combined noise data length for all experiments
        Zhat(:,xxr1hat:xxr2hat) = Zihat;    % combined state-input data length for all experiments

        Delta(:,xNr(lenNr)*T+xxr1hat:xNr(lenNr)*T+xxr2hat) = Deltai;  % combined Delta for all experiments

    end

    X = [Xdash Xhat];
    Z = [Zdash Zhat];
    W = [Wdash What];



    q  = 1/((xNr(in))^(0.75));         % assigning weight
    Qi = eye(T)*q;
    Qhat = zeros(Np*T,Np*T);

    for kk = 1:Np

        qq1 = (kk-1)*T+1;
        qq2 = kk*T;

        Qhat(qq1:qq2,qq1:qq2) = Qi; % Np times

    end

    Q = blkdiag(eye(xNr(lenNr)*T),Qhat);

    ABidy =  X*Q*Z'*inv(Z*Q*Z'); % analytical solution for least square method

    Aidy = ABidy(:,1:dimN);          % obtain matrices A,B
    Bidy = ABidy(:,dimN+1:dimN+dimM);

    ABdif =  [Aidy-Adash Bidy-Bdash]; % compare the difference between [Aidy Bidy] and [Adash Bdash]
    yDif(in) = max(svd(ABdif));
end

Dif = X - [Adash Bdash]*Z - W - Delta;

plot(xNr, yDif);


%% plot

load 'Nr_q_0.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Nr_q_1.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Nr_q_10^10.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Nr_q_Nr^0,5.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Nr_q_Nr^0,75.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

title('Identification error for different weight assignment');
xlabel('Nr');
ylabel('|| \theta_W_L_S - \theta ||');
legend('q=0','q=1','q=10^1^0','q=1/Nr^0^.^5','q=1/Nr^0^.^7^5');
```

```
clear; clc;

%% first step: create data

dimN  = 3;  % dimension of state x
dimM  = 2;  % dimension of input u

% target system matrices
Adash = [0.6 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bdash = [1 0.5; 0.5 1; 0.5 0.5];

% source system matrices
Ahat = [0.7 0.5 0.4; 0 0.4 0.3; 0 0 0.3];
Bhat = [1.1 0.5; 0.5 1; 0.5 0.5];

deltaA = Ahat - Adash;
deltaB = Bhat - Bdash;


% target system dataset
ed = 1;                 % energy of the noise
% xNr   = 100:100:600;
xNr = round(logspace(log10(10), log10(600), 10));
```

```
lenNr = length(xNr);
yDif  = zeros(1,length(xNr));
%Nr = 10;                  % no of experiments
T  = 2;                 % data length
uudash = randn(dimM*xNr(lenNr),T);    % pre-define the input sequence
wwdash = randn(dimN*xNr(lenNr),T)*ed; % pre-define the noise sequence
xxdash = zeros(dimN*xNr(lenNr),T+1);      % empty state array
zzdash = zeros((dimM+dimN)*xNr(lenNr),T); % empty state-input array
xxdash(:,1) = randn(dimN*xNr(lenNr),1);   % random state input
Xdash = zeros(dimN,xNr(lenNr)*T);         % empty batch matrix for state samples
Wdash = zeros(dimN,xNr(lenNr)*T);         % empty batch matrix for noise samples
Zdash = zeros(dimN+dimM,xNr(lenNr)*T);    % empty batch matrix for state-input samples

% source system dataset
xNp = 3*xNr;
lenNp = length(xNp);
%Np = 10;             % no of experiments
uuhat = randn(dimM*xNp(lenNp),T);    % pre-define the input sequence
wwhat = randn(dimN*xNp(lenNp),T)*ed; % pre-define the noise sequence
xxhat = zeros(dimN*xNp(lenNp),T+1);      % empty state array
zzhat = zeros((dimM+dimN)*xNp(lenNp),T); % empty state-input array
xxhat(:,1) = randn(dimN*xNp(lenNp),1);   % random state input
Xhat = zeros(dimN,xNp(lenNp)*T);         % empty batch matrix for state samples
What = zeros(dimN,xNp(lenNp)*T);         % empty batch matrix for noise samples
Zhat = zeros(dimN+dimM,xNp(lenNp)*T);    % empty batch matrix for state-input samples

Delta = zeros(dimN,(xNr(lenNr)+xNp(lenNp))*T);   % empty batch matrix for Delta samples

for in = 1:lenNr
% data creating loop for target system
    for ii = 1:xNr(in)           % outer loop for each expriment
        for ij = 1:T        % inner loop for each data length

            xr1dash = (ii-1)*dimN+1;        % upper boundary for state data
            xr2dash = ii*dimN;              % lower boundary for state data
            ur1dash = (ii-1)*dimM+1;        % upper boundary for input data
            ur2dash = ii*dimM;              % lower boundary for input data
            zr1dash = (ii-1)*(dimN+dimM)+1; % upper boundary for state-input data
            zr2dash = ii*(dimN+dimM);       % upper boundary for state-input data

            xxdash(xr1dash:xr2dash,ij+1) = Adash*xxdash(xr1dash:xr2dash,ij) + Bdash*uudash(ur1dash:ur2dash,ij) + wwdash(xr1dash:xr2dash,ij);
            zzdash(zr1dash:zr2dash,ij) = [xxdash(xr1dash:xr2dash,ij)' uudash(ur1dash:ur2dash,ij)']';

            Xidash = flip(xxdash(xr1dash:xr2dash,2:T+1),2); % combined state data length for single experiment
            Widash = flip(wwdash(xr1dash:xr2dash,1:T),2);   % combined noise data length for single experiment
            Zidash = flip(zzdash(zr1dash:zr2dash,1:T),2);   % combined state-input data length for single experiment
        end

        xxr1dash = (ii-1)*T+1; % left boundary
        xxr2dash = ii*T;       % right boundary

        Xdash(:,xxr1dash:xxr2dash) = Xidash; % combined state data length for all experiments
        Wdash(:,xxr1dash:xxr2dash) = Widash; % combined noise data length for all experiments
        Zdash(:,xxr1dash:xxr2dash) = Zidash; % combined state-input data length for all experiments

    end

% data creating loop for source system
    for jj = 1:xNp(in)           % outer loop for each expriment
        for ji = 1:T        % inner loop for each data length

            xr1hat = (jj-1)*dimN+1;        % upper boundary for state data
            xr2hat = jj*dimN;              % lower boundary for state data
            ur1hat = (jj-1)*dimM+1;        % upper boundary for input data
            ur2hat = jj*dimM;              % lower boundary for input data
            zr1hat = (jj-1)*(dimN+dimM)+1; % upper boundary for state-input data
            zr2hat = jj*(dimN+dimM);       % upper boundary for state-input data

            xxhat(xr1hat:xr2hat,ji+1) = (Adash+deltaA)*xxhat(xr1hat:xr2hat,ji) + (Bdash+deltaB)*uuhat(ur1hat:ur2hat,ji) + wwhat(xr1hat:xr2hat,ji);
            zzhat(zr1hat:zr2hat,ji) = [xxhat(xr1hat:xr2hat,ji)' uuhat(ur1hat:ur2hat,ji)']';

            delta = [deltaA deltaB];

            Xihat = flip(xxhat(xr1hat:xr2hat,2:T+1),2); % combined state data length for single experiment
            Wihat = flip(wwhat(xr1hat:xr2hat,1:T),2);   % combined noise data length for single experiment
            Zihat = flip(zzhat(zr1hat:zr2hat,1:T),2);   % combined state-input data length for single experiment

            Deltai = delta*Zihat;                       % combined Delta for single experiment
        end

        xxr1hat = (jj-1)*T+1; % left boundary
        xxr2hat = jj*T;       % right boundary

        Xhat(:,xxr1hat:xxr2hat) = Xihat;    % combined state data length for all experiments
        What(:,xxr1hat:xxr2hat) = Wihat;    % combined noise data length for all experiments
        Zhat(:,xxr1hat:xxr2hat) = Zihat;    % combined state-input data length for all experiments

        Delta(:,xNr(lenNr)*T+xxr1hat:xNr(lenNr)*T+xxr2hat) = Deltai;  % combined Delta for all experiments

```

```
    end


    X = [Xdash Xhat];
    Z = [Zdash Zhat];
    W = [Wdash What];


    q  = 1/((xNr(in))^(0.75));         % assigning weight
    Qi = eye(T)*q;
    Qhat = zeros(xNp(lenNp)*T,xNp(lenNp)*T);

    for kk = 1:lenNp

        qq1 = (kk-1)*T+1;
        qq2 = kk*T;

        Qhat(qq1:qq2,qq1:qq2) = Qi; % Np times

    end

    Q = blkdiag(eye(xNr(lenNr)*T),Qhat);


    ABidy =  X*Q*Z'*inv(Z*Q*Z'); % analytical solution for least square method

    Aidy = ABidy(:,1:dimN);          % obtain matrices A,B
    Bidy = ABidy(:,dimN+1:dimN+dimM);

    ABdif =  [Aidy-Adash Bidy-Bdash]; % compare the difference between [Aidy Bidy] and [Adash Bdash]
    yDif(in) = max(svd(ABdif));
end

Dif = X*Q - [Adash Bdash]*Z - W - Delta;

plot(xNr, yDif);


%% plot

load 'Np_3Nr_q_0.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Np_3Nr_q_1.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Np_3Nr_q_10^10.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Np_3Nr_q_Nr^0,5.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

load 'Np_3Nr_q_Nr^0,75.mat'
plot(xNr, yDif,'-','markersize',6); hold on;

title('Identification error for different weight assignment');
xlabel('Nr');
ylabel('|| \theta_W_L_S - \theta ||');
legend('q=0','q=1','q=10^1^0','q=1/Nr^0^.^5','q=1/Nr^0^.^7^5');
```