university of groningen / faculty of science and engineering

Creating a reduced order Digital Twin through balanced truncation

Bachelor Integration Project

Industrial Engineering & Management

Authors:
J.J.G.B. Giesen, s3154831

First supervisors:
prof. dr. ir. J.M.A. Scherpen
MSc. Arijit Sarkar

Second supervisor:
dr. ir. M. Taheri

June 17th, 2022

„Handeln ist leicht, denken schwer;
nach dem Gedanken handeln unbequem.“

JOHANN WOLFGANG VON GOETHE

Acknowledgements

For all the continuous helpful support, I want to thank MSc. Arijit Sarkar.

Abstract

To mitigate increasingly complex systems in the field of engineering the Digital Twin is created. A one-one virtual representation of a physical system that is fed with past or real-time data. Digital Twins are used to provide real-time monitoring, predictive maintenance, and many more applications that are still being researched.

The dynamics of the physical system are described by the differential equations put in matrix form that is called the state-space model. Large-scale systems are computationally expensive, therefore model order reduction is performed. The number of states in the model is reduced which enables faster simulations and reduces the complexity of the system. This should be done in such a way that it still approximates the original model accurately enough.

To perform model reduction, two balanced truncation methods are considered in the scope of this research: generalized balancing and extended balancing. This research provides an comparative overview of the two balanced truncation methods and uses them to perform model reduction on a mass-spring-damper system with the aim of structure preservation.

A literature study is performed to create the theoretical framework that provides the requirements for the MATLAB model. This model is used to compare the outputs of the two balancing methods in the time and frequency domain and to see if structure preservation is possible for the mass-spring-damper system.

Generalized and extended balancing proved to be very similar in their outputs for all sizes of the reduced-order models. Furthermore, extended balancing does provide more benefits, such as a less conservative error bound and more degrees of freedom. Preservation of the Port-Hamiltonian structure proved possible in the literature, but preserving the structure of the physical system is still an open problem. In this research, the Port-Hamiltonian and the physical structure of the mass-spring-damper system could not be preserved using the created MATLAB model.

This research provides an in-depth comparison of generalized and extended balancing and sets up model requirements for creating a MATLAB model that aims for structure preservation.

Contents

Acronyms

CTLTI continuous-time linear time-invariant. 15

DMD Dynamic Mode Decomposition. 4, 8, 10

EB Extended Balancing. 4, 12, 14, 15, 17–21, 23–25, 27–31

GB Generalized Balancing. 4, 11, 14, 15, 17, 19–21, 23–25, 30, 31

LMI Linear Matrix Inequality. 9, 12, 17, 18, 30

MOR Model Order Reduction. 4, 8–11, 14, 24, 28

NPRA Norwegian Public Roads Administration. 7

PH Port-Hamiltonian. 15, 28–31

RB Reduced Basis. 4, 10

ROM Reduced Order Model. 8, 9, 11, 14, 15, 17, 20–29, 31

SVD Singular Value Decompostion. 10, 12

1 Introduction

In the last decades, advances in computing, communication, and technology have led to increasingly complex systems [Grieves, 2019]. A method created to mitigate this complexity is a Digital Twin model, a one-on-one virtual representation of the physical system it represents, called its Physical Twin [Grieves, 2019]. The virtual representation allows the user to simulate changes that could be made to the physical system before actually implementing them or see changes that occur in the physical system in real-time. This helps the user to quickly identify failures or improvements in large and complex systems, like bridges and turbines.

Two types of Digital Twins can be defined: the Digital Twin prototype, which is used in the design and manufacturing of products by using simulations, and the Digital Twin Instance, which is used to gather information about the product during its life cycle [Grieves, 2019]. They are theoretical models that link to the physical system by using measurement data flows from the physical to the virtual system, and information and process flows from the virtual to the physical system [Jones et al., 2020]. Due to the versatile character of the technology, there are many different applications throughout the life cycle of systems. Figure 1.1 shows different applications already in use in different phases in the life cycle.

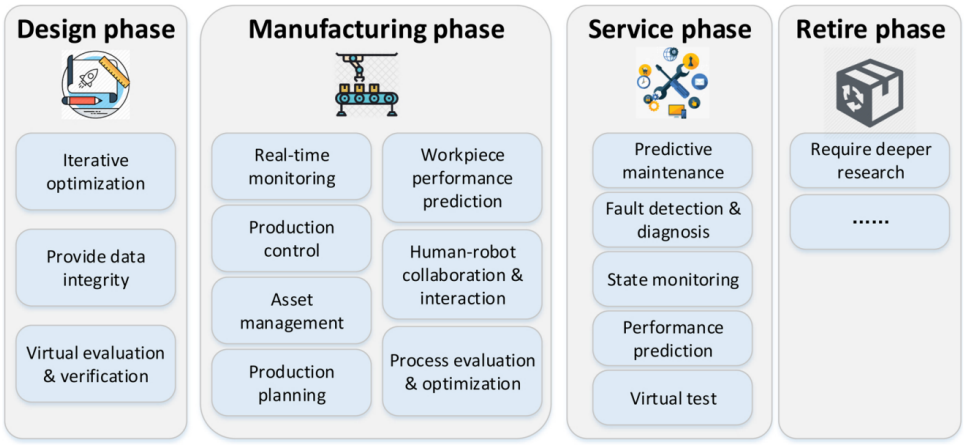


Figure 1.1: Possible applications of a digital twin for industry usage [Bilberg and Malik, 2019]

An example of the Digital Twin application can be found in Norway. The Norwegian Public Roads Administration (NPRA) has created Digital Twins for many of the country's 5800 bridges. With just a couple of sensors that are tactically placed on the bridge, the NPRA can create a complete real-time dynamical model showing all the forces and stresses acting upon the bridge. The models help the NPRA detect possible failures at an early stage by constantly monitoring the data sent to the cloud so that they can close the bridge before failure occurs. Furthermore, it reduces the costs of repairs and allows for the implementation of preventive maintenance as even small changes to the dynamics of the bridge are monitored.

A Digital Twin model requires quick simulations and iterations to enable real-time modeling

and iterative optimization. Therefore, techniques are required to transform and reduce large state-space systems, which are computationally heavy and therefore have long simulation times, into smaller ones. The leading technique to perform this job is Model Order Reduction (MOR) [Hartmann et al., 2020]. MOR reduces the system's complexity, reduces simulation times, and enables digital services by reducing the number of states in the system, thus the number of dynamical equations to be solved. Unfortunately, MOR decreases the accuracy of the Reduced Order Model (ROM) compared to the original system [Hartmann et al., 2020]. Consequentially, techniques exist, such as Dynamic Mode Decomposition DMD and Balanced Truncation [Benner et al., 2015], to analyze the system such that the reduced-order model represents the original input/output behavior as accurately as possible.

This research analyses and compares two techniques: generalized balanced truncation and extended balanced truncation [Borja et al., 2021]. After that, extended balanced truncation is used to create a Digital Twin prototype model of a physical system. A mass-spring-damper system is used for model reduction, which is widely used as a basis to represent other mechanical systems [Yamanaka et al., 2018].

1.1 Problem Statement

Model Order Reduction using balanced truncation cannot always ensure the preservation of the original structure of a physical system.

1.2 Research Objective

A clear and SMART research objective is defined from the problem statement, and it embeds the goal of the research and the final deliverables.

This research aims to compare generalized and extended balanced truncation through a literature study and with the use of a MATLAB model, to create and validate a Digital Twin model that can preserve the original structure of a physical system in MATLAB.

1.3 Research Questions

The problem analysis and research objective are translated into the following research question:

What is required to create a Digital Twin model of a physical system in MATLAB by using balanced truncation that preserves the original structure of the system?

To answer this central question, three subquestions are defined:

SQ1: What are the differences between generalized balanced truncation and extended balanced truncation?

SQ2: Which model requirements facilitate the design of a reduced order Digital Twin model in MATLAB?

SQ3: What is required to preserve the physical structure of the system using balanced truncation?

1.4 System Analysis

A simplified system of creating a Digital Twin of a physical system is shown in figure 1.2. The boxes with solid lines follow the scope of this research. This research focuses on creating a state-space system from a mass-spring-damper system and uses the two mentioned balancing techniques, whereafter MOR can be applied to derive a ROM. This ROM can then be used to create a Digital Twin of a physical system that will be defined during the research. Two other MOR techniques will be highlighted shortly to give an overview of existing techniques, but these will not be used for further research.

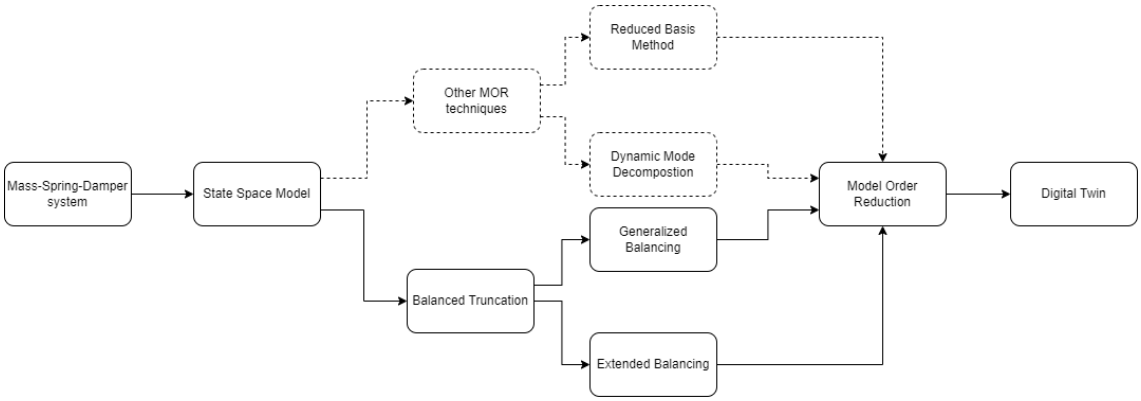


Figure 1.2: System description of the creation of a Digital Twin

1.5 Methods

1.5.1 Literature review

To gain more knowledge on generalized and extended balancing a literature study is performed. The literature study is used as the basis for the comparison between the two balancing methods and to find the requirements to design a Digital Twin of the mass-spring-damper system in MATLAB. After that, a list of requirements and steps to be taken is created that enables the creation of a MATLAB model to validate the findings.

1.5.2 MATLAB modelling

MATLAB is used to create a balanced ROM that allows for simulations with changing variables. Therefore, it is possible to compare the ROM's to each other and the original model for different sizes of the state-space system. In MATLAB the solver "SeDuMi" is used to solve the Linear Matrix Inequality's (LMI) which will be further elaborated on in section 3.2.3.

2 Theoretical Framework

In the field of MOR many different techniques are used. As too many exist to fit into the scope of this research, next to balanced truncation, two other techniques will be highlighted in this section. Thereafter, an explanation is given on why balanced truncation is the technique that is used in this research.

2.1 Model reduction methods

2.1.1 Dynamic Mode Decomposition

DMD is a data-driven technique that does not require any knowledge of the equations of motion of the system. This method only requires measurement data that is acquired from numerical simulations or laboratory experiments [Schmid, 2010]. DMD attempts find the most important dynamical characteristics of the system which are: resonance, spectral properties, and unstable growth modes [Proctor et al., 2016]. This is done by performing the following steps:

1. Data is collected of the state of the system by taking a snapshot of the state at each point in time during a certain time span. This results in a large number of data points per time unit.
2. The gathered data points are then organized into two matrices X and X'. Each column in the X matrix represents a snapshot of the measurement where every entry is a data point. The X' matrix is almost identical to the X matrix except for the fact the first column is not the point x_1 but x_2.
3. To find the matrix A that we represents the underlying system the following equation is used:

$$A = X'X \tag{2.1}$$

   Since A is usually too big and computationally heavy to compute, A is approximated by taking the Singular Value Decompostion (SVD) of X:

$$X = U\Sigma V^T \tag{2.2}$$

   A can then be approximated in the following way:

$$A \approx X'V\Sigma^{-1}U^T \tag{2.3}$$

The dimensions of the matrices can then be reduced by eliminating states with low singular values.

2.1.2 Reduced Basis method

The Reduced Basis (RB) approach has two main goals: real-time modelling for purposes such as estimation and control, and simulation and statiscal analysis for design [Boyaval et al., 2010]. Fundamentally RB is a method of discretization that aims to approximate an accurate output. There are two main steps in the RB method:

1. **Offline step**: This is the most computationally heavy step of the process. A large sample of parameters is used to learn what number of N parameter values can be used in the next step. The algorithm approximates once for all how many parameters are required to make an accurate approximation of the original system in the next step.
2. **Online step**: In this step iterative approximations are made for the values of the parameters. Since the computationally heavy part is done in the offline step, this step can be iterated many times at high speeds.

This method uses a posteriori error bounds which means that the error bound is computed with the use of the computed solution and not the exact one. This provides an error between the reduced and the original system which is used to check the accuracy of the solution [Haasdonk and Ohlberger, 2008].

2.1.3 Balanced truncation

Balanced truncation is widely chosen as a method for model reduction as it preserves the model's stability and has a priori error bounds [Gugercin and Antoulas, 2004] which means that the bound depends on the exact solution and is already set beforehand. Consequently, this allows for a wide choice of the state space dimension and makes it suitable for large-scale systems [Mehrmann and Stykel, 2005]. It is based on the observability and controllability Gramians of the system, which are matrices that contain information about the controllability and observability of each state [Sandberg, 2008]. The Gramians are basic in standard balancing and get more extensive in generalized and extended balancing. The more extensive Gramians allow for more degrees of freedom that can be used to improve the error bound and possibly impose a certain structure to the ROM [Borja et al., 2021] as will be further elaborated on in chapters 4 and 6, respectively. Thus, balanced truncation is used as the method for model reduction as it preserves the system's stability and has a priori error bounds.

This paper will focus on generalized and extended balancing, thus to get a better understanding of the two balancing techniques a literature review is performed. In the following sections the steps taken in generalized balancing, extended balancing and MOR are explained. The following state-space system is considered:

$$G : \begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

2.2 Generalized Balancing (GB)

This technique aims to give a weighting value to each state according to their controllability and observability. The highest value means that the corresponding state captures the most input-output information [Brunton and Kutz, 2019]. This can be achieved by equalizing the controllability (W_c) and observability (W_o) Gramians [Scherpen, 2005]. The Gramians are defined as follows:

$$W_c = \int_0^\infty e^{At} B B^T e^{A^T t} dt \tag{2.4}$$

$$W_o = \int_0^\infty e^{A^T t} C^T C e^{At} dt \tag{2.5}$$

To best way to calculate the Gramians is by solving the Lyapunov inequalities [Scherpen and Fujimoto, 2018]:

$$AW_c + W_cA^T + BB^T \leq 0 \tag{2.6}$$

$$A^TW_o + W_oA + C^TC \leq 0 \tag{2.7}$$

These Gramians should be equalized such that the balancing coordinate transformation T makes them equal, diagonal, and with eigenvalues σ^2 on its diagonal [Brunton and Kutz, 2019] :

$$T^{-1}W_cW_oT = \Sigma^2 \tag{2.8}$$

$$\hat{W}_c = \hat{W}_o = \Sigma \tag{2.9}$$

The matrix Σ^2 can be found by taking the SVD of the matrix multiplication W_oW_c. The values on the diagonal of Σ are called the Hankel singular values, which express the importance of a state in capturing input to output information [Scherpen, 2005]. High values correspond to states that capture much information, and low values to states that do not and thus can be truncated. After that, the matrix T can be found by solving equation 2.8.

$\hat{W}_c$ and $\hat{W}_o$ are the balanced Gramians and are defined as follows:

$$\hat{W}_c = T^{-1}W_cT^{-T} \tag{2.10}$$

$$\hat{W}_o = T^TW_oT \tag{2.11}$$

This information can then be used to start truncating states which will be further explained in section 2.4

2.3 Extended Balancing (EB)

EB uses the generalized balanced Gramians and extends them with two main advantages: the error bound can be reduced and it results in extra degrees of freedom that allows to impose a specific structure on the [Borja et al., 2021]. The extended controllability and observability LMI can respectively be defined as follows:

$$\begin{pmatrix} -W_oA - A^TW_o - C^TC & W_o - \alpha S - A^TS \\ W_o - \alpha S^T - S^TA & S + S^T \end{pmatrix} \geq 0 \tag{2.12}$$

$$\begin{pmatrix} -PA - A^TP & -P + \beta R + A^TR & -2PB \\ -P + \beta R^T + R^TA & R + R^T & 2R^TB \\ -2B^TP & 2B^TR & 4I \end{pmatrix} \geq 0 \tag{2.13}$$

Where the extended observability Gramian consists of W_o, S, and α and the extended controllability Gramian of P, R, and β. P is the inverse of the generalized controllability Gramian ($P = W_c^{-1}$). As these LMI's are difficult to solve, we try to solve the following simpler LMI's:

$$2\alpha W_o + 2\Gamma_o - \Theta_o \geq 0 \tag{2.14}$$

$$2\beta W_c + 2\Gamma_c - BB^T - \Theta_c \geq 0 \tag{2.15}$$

where α and $\beta > 0$ are constants such that $\beta W_c + \Gamma_c > 0$ and $\alpha W_o + \Gamma_o > 0$. Furthermore, Θ_o and Θ_c are defined as follows:

$$\Theta_o = (\Gamma_o - W_o A) X_o^{-1} (\Gamma_o - A^T W_o) \tag{2.16}$$

$$\Theta_c = (-\Gamma_c P + A + BB^T P) X_c^{-1} (-P\Gamma_c + A^T + PBB^T) \tag{2.17}$$

with

$$X_o = -W_o A - A^T W_o - C^T C \tag{2.18}$$

$$X_c = -PA - A^T P - PBB^T P \tag{2.19}$$

Γ_c and Γ_o are symmetric matrices defined as follows:

$$\Gamma_o = \epsilon_o W_o \tag{2.20}$$

$$\Gamma_c = -\epsilon_c W_c \tag{2.21}$$

The matrices Γ_o and Γ_c can be chosen as desired by changing the values of ϵ_o and ϵ_c, where $\epsilon_o > 0$ and $0 < \epsilon_c < \alpha$. The selection of these matrices is important as this can improve the error bound or help preserve the system's physical structure after model reduction. With more degrees of freedom (more variables and matrices) in the extended Gramians, the chance and options are greater to find the right structure [Sandberg, 2010]. How a change of the Γ matrices affects the system will be further tested and discussed in chapter 3.

After computing the values for all these different variables it is then possible to compute the extended Gramians with equations 2.22 and 2.23

$$S = Q(\alpha Q + \Gamma_o)^{-1} Q \tag{2.22}$$

$$W = (\beta P + \Gamma_c)^{-1} \tag{2.23}$$

Contrary to equation 2.9, a system is extended balanced if the extended observability Gramian is equal to the inverse of the extended controllability Gramian [Borja et al., 2021]. These are then both equal to the diagonal matrix Λ:

$$S = W^{-1} = \Lambda \tag{2.24}$$

Again, a transformation matrix W_e can be found such that its inverse balances the state and equals both Gramians:

$$W_e^{-1} W^{-1} S W_e = \Lambda^2 \tag{2.25}$$

This transformation matrix can be found with the use of MATLAB or another computing algorithm. The values on the diagonal of the matrix Λ are again ranked in order from highest to lowest.

2.4 Model Order Reduction

After using either GB or EB it is possible to reduce the number of states in the system. This can be done by eliminating the states that are ranked the lowest in the Σ and Λ matrix which corresponds to the lower rows. We can define the balanced model as follows [Borja et al., 2021]:

$$\begin{aligned} \dot{\overline{x}} &= \overline{A}\overline{x} + \overline{B}u \\ \overline{y} &= \overline{C}\overline{x} \end{aligned} \tag{2.26}$$

Where $\overline{A}, \overline{B}, \overline{C}$ are the new transformed matrices which are defined in equation 2.27 and $\overline{x}$ represents the transformed states given by $\overline{x} = W_e^{-1}x$. Below T is used as the transformation matrix but for EB this would be the transformation W_e.

$$\overline{A} = T^{-1}AT, \quad \overline{B} = T^{-1}B, \quad C = CT \tag{2.27}$$

These can then be split up as follows:

$$\overline{A} = \begin{bmatrix} \overline{A}_{11} & \overline{A}_{12} \\ \overline{A}_{21} & \overline{A}_{22} \end{bmatrix} \quad \overline{B} = \begin{bmatrix} \overline{B}_1 \\ \overline{B}_2 \end{bmatrix} \quad \overline{C} = \begin{bmatrix} \overline{C}_1 \overline{C}_2 \end{bmatrix} \quad \overline{x} = \begin{bmatrix} \overline{x}_1 \\ \overline{x}_2 \end{bmatrix} \tag{2.28}$$

As mentioned before only the most controllable and observable states will remain in the ROM. $\overline{x}_1$ represents the number of k states that are to be preserved in the model. This results in the final reduced order model:

$$\hat{G} : \begin{cases} \dot{\hat{x}} = \hat{A}\hat{x} + \hat{B}u \\ \hat{y} = \hat{C}x \end{cases}$$

where

$$\hat{x} = \overline{x}_1, \quad \hat{A} = \overline{A}_{11}, \quad \hat{B} = \overline{B}_1, \quad \hat{C} = \overline{C}_1 \tag{2.29}$$

2.4.1 Error bound

One of the benefits of balanced truncation is that it has an *a priori* error bound [Gugercin and Antoulas, 2004]. This is defined in section 2.1.3 as an error bound that depends on the exact solution an is already set beforehand. Equation 2.30 displays that the error bound depends on the size of the Hankel singular values of the states that are not included in the ROM. Therefore, the error bound can be determined if the Hankel singular values and the desired size of the ROM are known. If the states that are truncated from the system correspond to small Hankel singular values, the error bound will be small. Thus it can be concluded that the ROM is a good approximation of the original model [Scherpen, 2011]. Furthermore, the error bound only provides the upper bound for the error. Its exact value is determined by calculating the H_∞-norm over that space. The error bound is defined below:

$$||G - \hat{G}||_\infty \leq 2 \sum_{j=k+1}^{n} \sigma_j \tag{2.30}$$

Where in the left half of the equation $||G - \hat{G}||_\infty$ is the H_∞-norm that represents the maximum singular value over that space [Wang et al., 1999]. In the right half of the equation, the bound is represented by the sum of the truncated Hankel singular values [Borja et al., 2021].

As EB will provide a less conservative error bound and extra degrees of freedom, this technique is the best to use for structure preservation.

3 Model Description

With the use of the theoretical framework in chapter 2, a MATLAB model is created. This model is used to gather more data about balanced truncation, compare GB and EB, and test different values and sizes of the ROM. The state-space model that is used is in the Port-Hamiltonian (PH) framework. First, an explanation will be given of this framework, and after that, the state-space model will be explained.

3.1 Port-Hamiltonian structure

The data set that will later be used does not just provide the A,B,C, and D matrices to the system. The A matrix still has to be compiled from the F and H matrices in the PH framework. The Hamiltonian, in equation 3.1, is just as the well known Lagrangian, in equation 3.2, a framework to represent the total energy in the system. It is comprised of the kinetic energy (T) and the potential energy (V), where the Hamiltonian is the sum and the Lagrangian the difference of these energy's.

$$H = T + V \tag{3.1}$$

$$L = T - V \tag{3.2}$$

The F matrix, the first matrix after the equal sign in equation 3.3, exists out of a zeros block, two identity blocks (I) and one block with the damping coefficients (D). The H matrix, right of the F matrix, exists out of two zeros blocks, one block with the spring constants K, and one block with the inversed masses (M). The A matrix can be found by multiplying the F and H matrices. The B matrix is all zeros except for the 101st entry where G represents a vector with all zeros and one 1 entry.

$$\begin{bmatrix} \dot{q} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & I_{100} \\ -I_{100} & -D \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & M^{-1} \end{bmatrix} \begin{bmatrix} q \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ G \end{bmatrix} u \tag{3.3}$$

3.2 Derive a state space model

3.2.1 Deriving equations

As input to the model, a state-space system is used, which first needs to be created. In this research we study a continuous-time linear time-invariant (CTLTI) system. If a system is linear time-invariant, the output of the system will stay the same if the input is applied T seconds later. The only difference then will be that there is a time delay in the system of T seconds. To start of, the equations of motion are derived from a 4 mass-spring-damper system, shown in figure 3.1, that results in a state-space system with 8 state variables. This system provides the basis for the larger state-space system that is defined in section 3.2.2. The system, the forces, and the equations of motion can be found in figure 3.2. It is assumed that no friction or other forces is acting on the system.

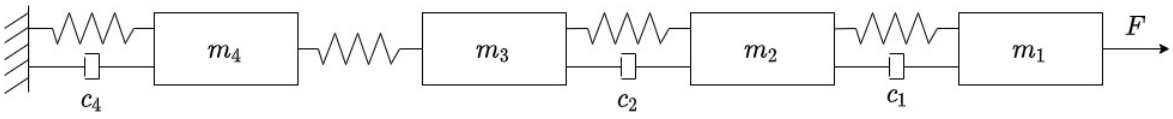


Figure 3.1: Mass-Spring-Damper system with 4 masses

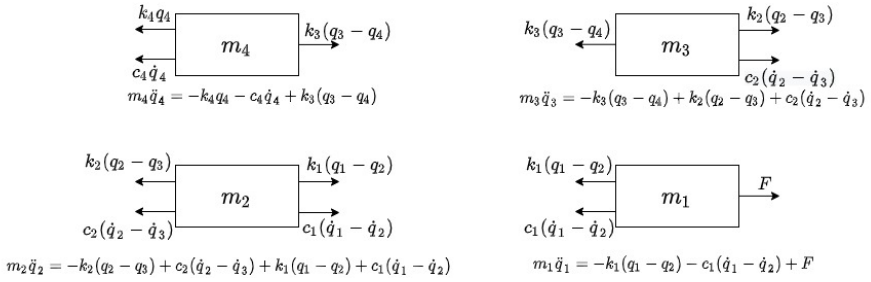


Figure 3.2: Equations of motion of the 4 Mass-Spring-Damper system

The equations of motion can then be rewritten into state space form. The state space matrices can be found in equations 3.4 and 3.5

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1' & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{-k_1}{m_1} & \frac{k_1}{m_1} & 0 & 0 & \frac{-c_1}{m_1} & \frac{c_1}{m_1} & 0 & 0 \\ \frac{k_1}{m_1} & \frac{-k_2-k_1}{m_2} & \frac{k_2}{m_2} & 0 & \frac{c_1}{m_2} & \frac{-c_2-c_1}{m_2} & \frac{c_2}{m_2} & 0 \\ 0 & \frac{k_2}{m_3} & \frac{-k_2-k_2}{m_3} & \frac{k_3}{m_3} & 0 & \frac{c_2}{m_3} & \frac{-c_2}{m_3} & 0 \\ 0 & 0 & \frac{k_3}{m_4} & \frac{-k_4-k_3}{m_4} & 0 & 0 & 0 & \frac{-c_4}{m_4} \end{bmatrix} \tag{3.4}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad D = 0 \tag{3.5}$$

3.2.2 Large Mass-Spring-Damper system

For this small system, with only 8 states, it is impossible to truncate any states as they are all required to describe the system's dynamics. Therefore, a larger data set is loaded into MATLAB to simulate a higher dimensional system. This system is described by figure 3.3, and the state-space model is derived similarly to the smaller system above. This system consists of 200 masses, resulting in 400 states. As this system is very large and expensive to simulate in MATLAB, the system will be reduced to 100 masses and 200 states. All further simulations

are based upon this reduced-order system, called the original system, in the continuation of this report. The stability of the system is assessed by evaluating the eigenvalues. If all the eigenvalues have negative real parts, the system is stable, which can easily be checked in MATLAB with the command "eig". Consequently, it is concluded that this system is stable. As balanced truncation preserves the system's stability, which is presented in section 2.1.3, it follows that the ROM is also stable.

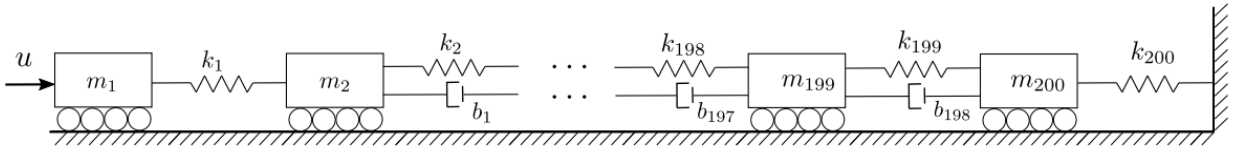


Figure 3.3: Mass-Spring-Damper system with 200 masses [Borja et al., 2021]

3.3 MATLAB code

3.3.1 Generalized Balancing

The MATLAB code is constructed using the theoretical framework as presented in Chapter 2. There are six main steps taken in the code:

1. Importing and creating state space matrices
2. Solving the Lyapunov equations for the controllability and observability Gramians
3. Finding the transformation matrix T
4. Creating the balanced state space system
5. Create ROM
6. Find the error bound for GB

The complete MATLAB code for GB can be found in Appendix A. The code until line 77 is for GB.

3.3.2 Extended Balancing

As stated in chapter 2, EB is an extension on the generalized Gramians which are used to calculate the extended Gramians. Therefore, the GB part of the code is required for EB. The code is extended with the following steps:

1. New constants and matrices are defined
2. The variables α and β are found by solving the LMI's
3. Solve to find the extended Gramians
4. Finding the transformation matrix W_e
5. Creating the balanced state space system
6. Create ROM
7. Find the error bound for EB

The complete MATLAB code can be found in Appendix A. The additional code for EB starts at line 78.

3.3.3 SeDuMi Solver

A solver is required to solve the LMI's to find the generalized Gramians and the values for α and β. Therefore, the YALMIP toolbox is installed as an add-on to MATLAB. This toolbox includes multiple programs and solvers that can solve and model optimization problems [Lofberg, 2004]. SeDuMi is the solver used from the toolbox in this research [Sturm, 1999], a semidefinite programming solver. This solver is used as it can solve for semidefiniteness constraints and large scale optimization problems. The solver is used in the MATLAB code in the following way:

1. Define the solver, SeDuMi, to be used
2. Define number of "sdpvar" decision variables
3. Define the constraints
   (a) The matrix/variable should be positive semidefinite
   (b) The LMI to be solved
4. Optimize the constraints and, for the matrices, the values on the diagonal of the matrix
5. Return the numerical value of the decision variables

4 Generalized vs Extended Balancing

In this section GB and EB are compared by looking at their Hankel singular values and the error bound. Before this is done, changes in the values of Γ_o and Γ_c are studied since these matrices influence the outcome of the system.

4.1 Optimal values for Γ_o and Γ_c

The decision for Γ_o and Γ_c influences the value of α and β as can be seen in equations 2.20 and 2.21. Consequently, these values change both the extended Gramians, as seen in equations 2.22 and 2.23. The possibility to tweak the values of these matrices and variables provides extra degrees of freedom that can be used to impose a specific structure on the system [Borja et al., 2021]. To see the influence of the Γ matrices on the error bound, several runs are done with different values for ϵ_o and ϵ_c, as these decide the values of the Γ matrices, as can be seen in equations 2.20 and 2.21. For these simulations, the system is reduced from 200 to 100 states.

Table 4.1 shows the error bound for some of the values of ϵ_o and ϵ_c that were used. Firstly, it can be seen that for values between the order of 1e-4 and 8e3, there is no solution available. This is because the resulting observability Gramian is not positive definite, which means it is not larger than or equal to zero. Thereafter, the Cholesky factorization cannot be performed in line 127 of the MATLAB code in Appendix A, which results in an error. Secondly, the error bound seems to be the smallest for the values of ϵ_o and ϵ_c that is the smallest order with a feasible solution. The error bound increases and seems to stabilize with really high orders of magnitude, like 1e10, and the low orders, like 1e-10. This idea is confirmed by looking at increasingly higher and lower orders, where the value of the error bound is 2.4688e-04 at values 1e20 and 1e-20 for ϵ_o and ϵ_c . Finally, the lowest error bound is found for a value of 9000 for ϵ_o and ϵ_c.

ϵ_o	ϵ_c	**Error bound EB**
1e-10	1e-10	2.4680e-04
1e-6	1e-6	2.3727e-04
1e-5	1e-5	1.2236e-04
1e-4-8e3	1e-4-8e3	NaN
9e3	9e-3	3.3702e-05
1e4	1e4	6.5844e-05
1e6	1e6	2.4461e-04
1e10	1e10	2.4683e-04

Table 4.1: Error bound for different values of ϵ_o and ϵ_c

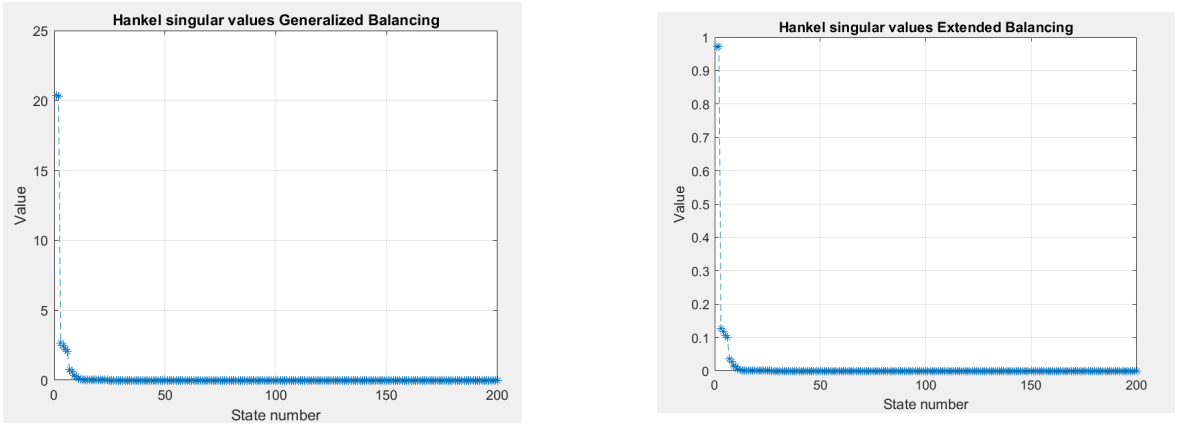
After determining the order of magnitude that provides the lowest error bound, some more values around 9e3 are tested. As a result, it is found that a value of 9000 for ϵ_o and 9300 for ϵ_c gives the lowest error bound of 1.1805e-05, resulting in a value of 9346 for α and β. Consequently, these values are used for EB in the continuation of this report.

4.2 Comparison of GB and EB

To see the differences between GB and EB, the Hankel singular values and error bound with different numbers of truncated states are compared. This gives an indication of the difference in the accuracy of the approximation of the ROM's. As mentioned in chapter 3, the system is run for 100 masses which results in 200 states of the system. For the rest of the simulations in this report the input to the system will be $u = 2sin(2t)$ from time 0 to 50 seconds. The sinusoidal equation is chosen as input to the system because of the oscillations that it infers. These oscillations are the natural behaviour a mass-spring-damper system will have when a force acts upon the system.

4.2.1 Hankel Singular Values

In figure 4.1 the Hankel singular values of GB and EB can be found. The values that correspond to the first couple of states are around 10 to 20 times smaller for EB than for GB. From equation 2.30, it is seen that the error bound is smaller if the Hankel singular values are smaller. Consequently, it follows that the error bound will be smaller for EB as will be shown in the following section.



(a) Hankel singular values of generalized balancing with 200 states

(b) Hankel singular values of extended balancing with 200 states

Figure 4.1: Hankel singular values of generalized and extended balancing with 200 states

4.2.2 Error bound

The theoretical framework states that the error bound for EB is less conservative than that of GB. This does not imply that the actual error will be smaller for EB but just that it can be predicted less conservatively than for GB. To validate this for the mass-spring-damper system, multiple simulations are done with different sizes of the ROM.

Size ROM	Error bound GB	Error bound EB
150	1.1599e-04	5.5484e-06
100	2.4677e-04	1.1805e-05
50	3.7882e-04	1.8122e-05
30	4.3236e-04	2.0683e-05
15	0.0077	3.7045e-04
10	0.4421	0.0211
5	8.4509	0.4043

Table 4.2: Error bound of the ROM's of GB and EB

The results in table 4.2 show that it is indeed true in this case that the error bound is smaller for EB than GB for all sizes of the ROM.

5 Original vs Reduced Order Models

To evaluate if the ROM's approximate the original model well, their outputs will be compared. Furthermore, the bode plots will be analyzed to see and compare the changes in phase and gain.

5.1 Output comparison

5.1.1 Output plots

In figure 5.1 the overall output of the three systems can be seen. This represents the response of the systems to the sinusoidal input. Only one of the lines is visible in this figure as their outputs are very close to each other. This was to be expected as the error bound that was found was already of a small order of magnitude because of the small Hankel singular values. To visualize the differences in the outputs, close-ups are made of the output plots, which are shown in figure 5.2 for ROM's of 150, 100, 50, 25, 10, and 5 states.

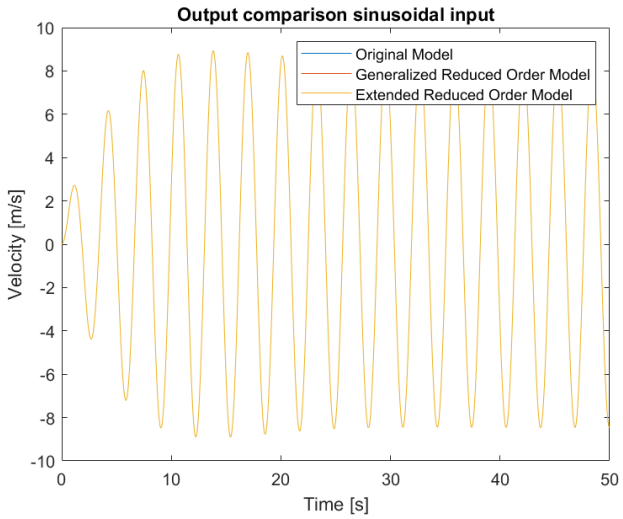
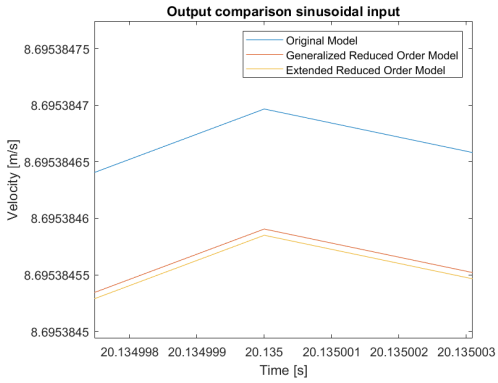
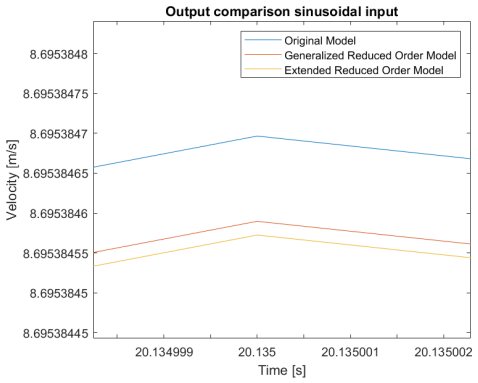


Figure 5.1: Plot of the output of the original system and both ROM's with 100 states

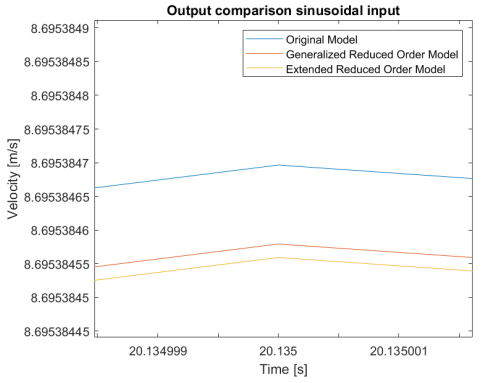
When looking at figure 5.2, multiple observations can be made. Firstly, both ROM's approximate the original system very accurately. This was expected as both error bounds proved to be small, as seen in table 4.2. On the contrary, how smaller the ROM gets, the larger the deviation is from the original model. Since there are fewer states to describe the model, there are fewer dynamics to describe the system. Thus, the accuracy of the approximation decreases. Furthermore, what is remarkable is that a small system, with only 5 of the original 200 states left, can still provide such a close approximation. This shows that balanced truncation allows for the creation of accurate ROM's.



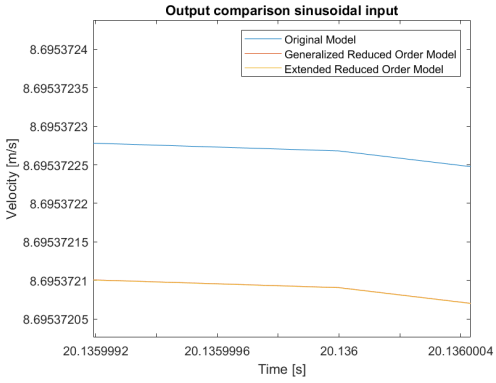
(a) ROM of 150 states



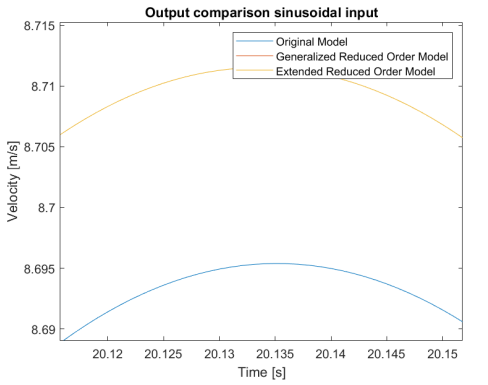
(b) ROM of 100 states



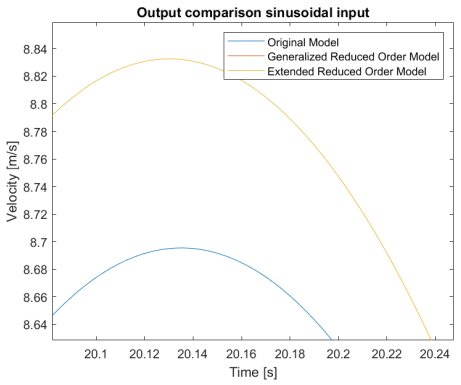
(c) ROM of 50 states



(d) ROM of 25 states



(e) ROM of 10 states



(f) ROM of 5 states

Figure 5.2: Output comparison between the original, generalized reduced and extended reduced system

Secondly, what stands out when comparing the GB and EB outputs, is that the output of the generalized ROM is closer to the original system than that of the extended ROM. In the previous chapter is was shown that the error bound for EB is significantly smaller than that of GB. Figure 5.2 shows that this does not directly imply that the actual error will then also be smaller. The error bound for EB is said to be less conservative than that of GB. This means that the range wherein the actual error will be is smaller.

Finally, the output line of both ROM's are close to identical for all the different numbers of states. Therefore, it is observed that the generalized and extended ROM will provide an evenly accurate approximation of the original model. Furthermore, the smaller the ROM's get, the closer the two output lines come together.

5.1.2 Simulation times

To visualize the influence of MOR the simulation times of the sections that produce the individual output graphs are compared. This is done by creating separate sections in the MATLAB code that produce the same graph as figure 5.1 but then for each of the systems individually. The function "tic-toc" is used in MATLAB to measure the simulation time of each section. This can be found from line 197 to 235 in Appendix A. As the simulation time is different each run, the average time over 10 runs is taken as the final value. The original model has a simulation time of 0.439 seconds on average, the values for GB and EB can be found in table 5.1.

Size ROM	**Simulation time GB [s]**	**Simulation time EB [s]**
150	0.278	0.264
100	0.116	0.115
50	0.0808	0.0722
25	0.0711	0.0724
10	0.0705	0.0624

Table 5.1: Simulation times for the output graph of the ROM's in seconds

Figure 5.3 displays the trend that can be seen in the progression of the running times. The purple trend line shows seemingly exponential behavior. Therefore, removing the first 100 states has a bigger influence on running time than the states after that. This finding can influence the choice for the size of the ROM to be used. As the reduction from 50 to 25 states has a relatively small impact on the running time, a further reduction was not deemed useful. Furthermore, a system described by 50 states captures more information than 25 states. Moreover, the reduction of simulation time with a reduction of the number of states underlines the importance of MOR as a tool to create Digital Twins (e.g. for real-time monitoring).

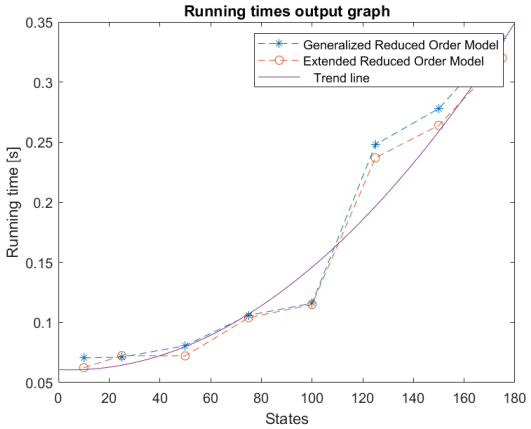


Figure 5.3: Running times for the output graph of the ROM's for multiple states

5.1.3 H_∞-norm

The error of the system is given by the H_∞-norm that can be found in equation 2.30. This norm can be calculated with the add-on function "*hinfnorm*" in MATLAB. The H_∞-norm for GB and EB for 4 sizes of the ROM can be found in table 5.2.

Size ROM	**H_∞-norm GB**	**H_∞-norm EB**
100	7.691894764580693e-07	7.500652422388620e-07
50	7.868965942337915e-07	7.844768447115839e-07
25	9.429077738116799e-06	9.429146955797937e-06
10	0.191472902205917	0.191472901928204

Table 5.2: H_∞-norm for GB and EB

Table 5.2 shows that the H_∞-norm is almost identical for GB and EB. This result is in line with the results found in section 5.1, as a small error results in the outputs being almost similar. All the values that are found for the H_∞-norm are well within the error bounds that are displayed in table 4.2. The error is slightly smaller for EB for the ROM sizes 100, 50, and 10 and for GB for 25. Thus, overall EB approximates the original system slightly better.

5.2 Bode plots

After evaluating the outputs in the time domain, it is also relevant to evaluate the response of the three systems in the frequency domain. Analysis in the frequency domain shows signal characteristics, such as magnitude and phase, over a spectrum of frequencies, thus providing more information about it than the behavior over a time period. Therefore, bode plots are used, which consist of two graphs: the magnitude plot and the phase plot. The magnitude plot, which is on a logarithmic scale in decibels, displays the amplitude of the system at a range of frequencies. The phase plot, on a linear scale in degrees, displays the phase/argument at a range of frequencies. This is interpreted as the angle between the real and imaginary axis in the complex plane.

Figures 5.4, 5.5, and 5.6 show the Bode plots of the original, generalized reduced, and extended reduced system respectively. It is immediately possible to conclude that all systems are stable as none of them cross the phase angle below -180 degrees.

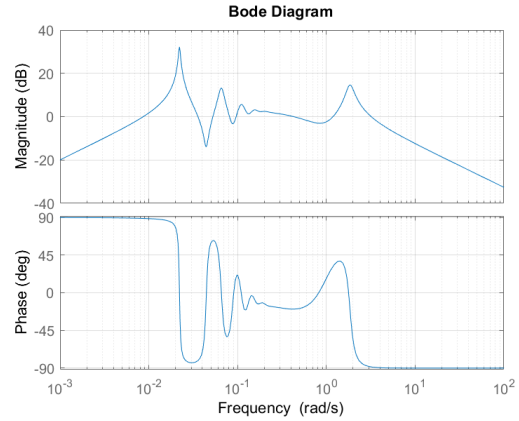


Figure 5.4: Bode plot of the original system with 200 states

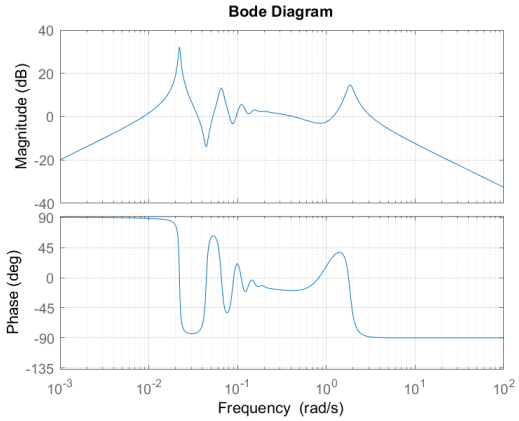


Figure 5.5: Bode plot of the generalized ROM with 50 states

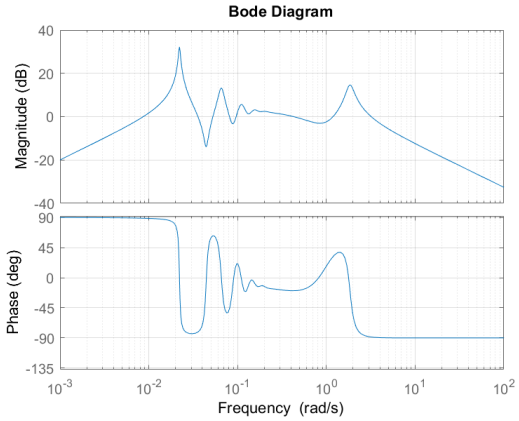
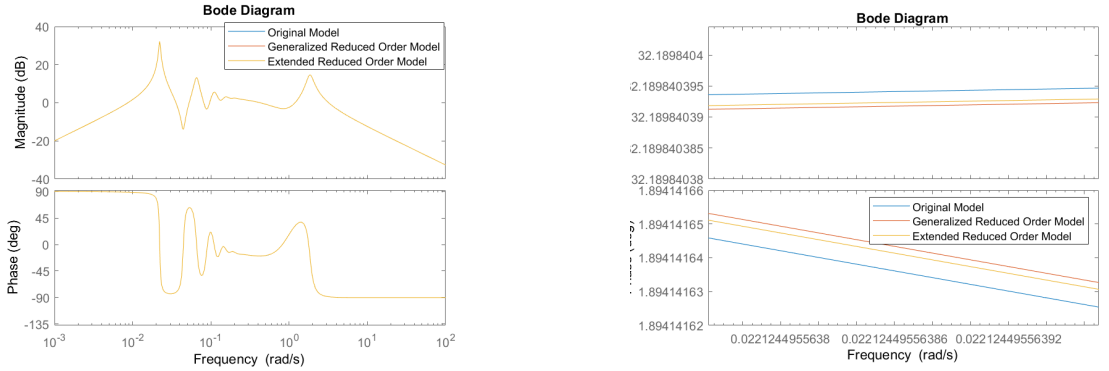


Figure 5.6: Bode plot of the extended ROM with 50 states

Comparing the three bode plots, it can be seen that they are all close to similar. Therefore, in figure 5.7 the three bode plots are plotted in one figure. In figure 5.7a, it is still not possible to make a clear distinction between the three systems. Therefore, a close-up is made of the plot in figure 5.7b. This plot shows that the line of the extended ROM is just slightly closer to that of the original model than the generalized ROM. Thus, the EB response in the frequency domain is slightly more accurate.



(a) Bode plot of the original, generalized reduced and extended reduced system

(b) Close-up of the bode plot of the original, generalized reduced and extended reduced system

Figure 5.7: Bode plot and close-up of the original, generalized reduced and extended reduced system

6 Structure Preservation

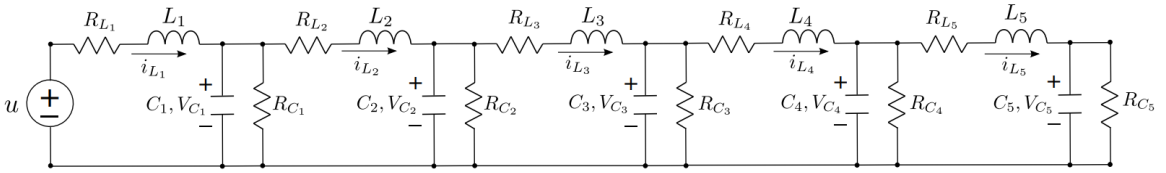
One of the open problems in the field of MOR is structure preservation. In this research, two types of structure preservation are relevant: preservation of the PH/Lagrangian structure and of the physical interpretation of the structure (e.g. mass-spring-damper system).

Preservation of the PH or Lagrangian structure has been a topic in the field of MOR since the start [Werner, 2021] and Borja et al. mention one of the requirements to preserve the PH structure. If the transformation matrix is block diagonal, the Hamiltonian matrix, which is a diagonal matrix, will remain diagonal after balanced truncation [Borja et al., 2021]. The Hamiltonian, in equation 3.1, and Lagrangian, in equation 3.2, represent the internal energy of the system in terms of its kinetic (T) and potential (V) energy [Chaturantabut et al., 2016].

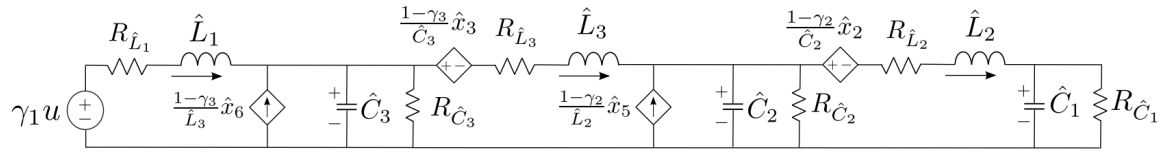
The preservation of the system's physical structure is a newer and more open field of research. When the physical structure is preserved, the ROM can be interpreted in the same physical form as the original model. Thus, if the original model is a mass-spring-damper system with 200 states and 100 masses, the ROM with 100 states can then still be interpreted as a mass-spring-damper system but with 50 masses. This form of structure preservation can be used for various systems like electrical networks, communication networks, and mechanical systems. In section 6.1, one case is presented where the system's physical structure was preserved for an electrical network. After that, in section 6.2, the mass-spring-damper system used in this research will be analyzed to see if the structure can be preserved as well.

6.1 Preservation of an electrical network

Borja et al. managed to preserve the structure of an RLC circuit using EB. The electrical network is reduced from 10 states, in figure 6.1a, to 6 states in figure 6.1b.



(a) Original RLC circuit



(b) Reduced RLC circuit

Figure 6.1: Structure preservation of an RLC circuit [Borja et al., 2021]

Three observations were made in preserving the RLC circuit structure [Borja et al., 2021]:

1. The transformation matrix should be a block diagonal matrix
2. States are not truncated individually but in pairs. For every truncated inductor there is also a capacitor truncated.
3. Setting the Γ matrices as nonzero implies that the entries of the balanced matrix Σ are different. Therefore, the smallest entries of Σ can be truncated

6.2 Preservation of the Mass-Spring-Damper structure

The requirements for structure preservation defined in section 6.1, can then be used to evaluate the ROM of the mass-spring-damper system.

Firstly, the transformation matrix Te is checked if it is a block diagonal matrix. The 200 x 200 matrix is too large to display on paper but in the MATLAB code it can be seen that the transformation matrix Te is not diagonal nor block diagonal. Consequentially, from the matrix structure in equation 3.3 it can be derived that the A matrix of the system will have the following structure:

$$\begin{bmatrix} 0 & M^{-1} \\ -K & -DM^{-1} \end{bmatrix} \tag{6.1}$$

The A matrix after the balanced transformation does not show this block structure and every entry in the matrix has a value. So through the use of EB the physical as well as the PH structure is not preserved for the mass-spring-damper system.

The extra degrees of freedom gained from the Γ matrices can be altered to try and adjust the structure of the balanced models. Furthermore, additional constraints could be added to impose a more specific structure on the transformation matrix.

7 Discussion

This research focused on providing a comparative overview of GB and EB. This was done by creating an extensive theoretical framework that was used to create a MATLAB model with the aim of simulating an accurate approximation and possible structure preservation.

For the creation of the state-space model from the data set, the first 100 x 100 entries were taken from the D, K, and M matrices resulting in a model with 100 masses. As this data set describes the dynamics of a system with 200 masses, just taking the first 100 entries will already influence the system's structure. Mass 100 in the system is connected to mass 101, which is not considered in the system used in this research.

The complete system, with 200 masses, from the data set, could not be used because the SeDuMi solver would have an unexplained crash. The reason for this crash is unknown, and this problem could not be solved. Using another solver or optimizing the code could be a possible solution to this problem.

Furthermore, the complete MATLAB code has a long simulation time of around 2 days. This is primarily due to solving the LMI's for the generalized Gramians. Altering the solver settings or the code might improve the simulation time. For example, the solver could be stopped some iterations earlier if the solution is already within a certain predetermined bound.

As the PH and physical structure of the system could not be preserved, additional constraints could be added to impose a specific structure on the transformation matrix, thus the balanced state-space system. The requirements for structure preservation, as mentioned in chapter 6, could be used for this.

8 Conclusion

This research aimed to provide an overview of the differences between GB and EB and the requirements to create a MATLAB model that can preserve the system's structure. A literature study was performed, and a MATLAB code was created to obtain this goal.

The differences between GB and EB proved small in this research as the outputs in both the time and the frequency domain were almost similar. Furthermore, their H_∞-norm was also almost identical. Therefore, it can be concluded that for this mass-spring-damper system, there is not much difference in the output of GB and EB. On the other hand, EB is a more adjustable technique that provides a smaller error bound and extra degrees of freedom. These features make EB a technique that is better suited for structure preservation and flexibility.

As found in the literature and with the use of the MATLAB code, the extra degrees of freedom can be exploited by tweaking the. constants ϵ_o and ϵ_c. The values of ϵ_o and ϵ_c influence the Γ matrices and the values of α and β, such that the desired balanced system is created. A value of 9000 for ϵ_o and 9300 ϵ_c, resulting in a value of 9346 for α and β, were chosen as these provided the lowest singular values, thus the lowest error bound.

From the literature, specific requirements were found for the preservation of the PH and the physical structure of a system. Moreover, these requirements are used to check if, in the current state, the created MATLAB model preserves these structures of the mass-spring-damper system. The transformation matrix is not block diagonal, as required, and the A matrix of the ROM does not have the same structure as the original A matrix. Therefore, it can be concluded that the PH and the mass-spring-damper structure of the system could not be preserved in this research.

Overall, it can be concluded that GB and EB provide very similar outputs, but EB provides extra benefits as a reduced error bound and extra degrees of freedom. Furthermore, the requirements for structure preservation were presented, and it was shown that structure preservation is possible for an electrical network. For the mass-spring-damper system in this research, the PH and mass-spring-damper structure could not be preserved, but possible solutions were presented.

Bibliography

[Benner et al., 2015] Benner, P., Gugercin, S., and Willcox, K. (2015). A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM review*, 57(4):483–531.

[Bilberg and Malik, 2019] Bilberg, A. and Malik, A. A. (2019). Digital twin driven human–robot collaborative assembly. *CIRP annals*, 68(1):499–502.

[Borja et al., 2021] Borja, P., Scherpen, J. M., and Fujimoto, K. (2021). Extended balancing of continuous lti systems: a structure-preserving approach. *IEEE Transactions on Automatic Control*.

[Boyaval et al., 2010] Boyaval, S., Le Bris, C., Lelievre, T., Maday, Y., Nguyen, N. C., and Patera, A. T. (2010). Reduced basis techniques for stochastic problems. *Archives of Computational methods in Engineering*, 17(4):435–454.

[Brunton and Kutz, 2019] Brunton, S. L. and Kutz, J. N. (2019). *Balanced Models for Control*, page 321–344. Cambridge University Press.

[Chaturantabut et al., 2016] Chaturantabut, S., Beattie, C., and Gugercin, S. (2016). Structure-preserving model reduction for nonlinear port-hamiltonian systems. *SIAM Journal on Scientific Computing*, 38(5):B837–B865.

[Grieves, 2019] Grieves, M. W. (2019). Virtually intelligent product systems: digital and physical twins.

[Gugercin and Antoulas, 2004] Gugercin, S. and Antoulas, A. C. (2004). A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766.

[Haasdonk and Ohlberger, 2008] Haasdonk, B. and Ohlberger, M. (2008). Reduced basis method for finite volume approximations of parametrized linear evolution equations. *ESAIM: Mathematical Modelling and Numerical Analysis*, 42(2):277–302.

[Hartmann et al., 2020] Hartmann, D., Herz, M., Paffrath, M., Rommes, J., Tamarozzi, T., Van der Auweraer, H., and Wever, U. (2020). 12 model order reduction and digital twins. *Citation for published version (APA): Benner, P., Grivet-Talocia, S., Quarteroni, A., Rozza, G., Schilders, WHA, & Silveira, LM (Eds.)(2020). Applications.(Model Order Reduction; Vol. 3). Walter de Gruyter GmbH. https://doi. org/10.1515/9783110499001*, page 379.

[Jones et al., 2020] Jones, D., Snider, C., Nassehi, A., Yon, J., and Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29:36–52.

[Lofberg, 2004] Lofberg, J. (2004). Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE.

[Mehrmann and Stykel, 2005] Mehrmann, V. and Stykel, T. (2005). Balanced truncation model reduction for large-scale systems in descriptor form. In *Dimension Reduction of Large-Scale Systems*, pages 83–115. Springer.

[Proctor et al., 2016] Proctor, J. L., Brunton, S. L., and Kutz, J. N. (2016). Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1):142–161.

[Sandberg, 2008] Sandberg, H. (2008). Model reduction of linear systems using extended balanced truncation. In *2008 American Control Conference*, pages 4654–4659. IEEE.

[Sandberg, 2010] Sandberg, H. (2010). An extension to balanced truncation with application to structured model reduction. *IEEE Transactions on Automatic Control*, 55(4):1038–1043.

[Scherpen, 2005] Scherpen, J. (2005). Model reduction for nonlinear control systems. *DISC model reduction coure notes*.

[Scherpen, 2011] Scherpen, J. M. (2011). Balanced realizations, model order reduction, and the hankel operator. In *The Control Handbook. Control System Advanced Methods*, pages 4–1.

[Scherpen and Fujimoto, 2018] Scherpen, J. M. and Fujimoto, K. (2018). Extended balanced truncation for continuous time lti systems. In *2018 European Control Conference (ECC)*, pages 2611–2615. IEEE.

[Schmid, 2010] Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28.

[Sturm, 1999] Sturm, J. F. (1999). Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653.

[Wang et al., 1999] Wang, G., Sreeram, V., and Liu, W. (1999). A new frequency-weighted balanced truncation method and an error bound. *IEEE Transactions on Automatic Control*, 44(9):1734–1737.

[Werner, 2021] Werner, S. W. (2021). *Structure-Preserving Model Reduction for Mechanical Systems*. PhD thesis, Otto-von-Guericke Universität Magdeburg.

[Yamanaka et al., 2018] Yamanaka, Y., Yaguchi, T., Nakajima, K., and Hauser, H. (2018). Mass-spring damper array as a mechanical medium for computation. In *International Conference on Artificial Neural Networks*, pages 781–794. Springer.

A MATLAB code

```
clear
clc

%% Import data set
Dataset = importdata('mechanical_example_data.mat');
Dt = Dataset.D;      % Import Damping costants matrix
Kt = Dataset.K;      % Import Spring constants matrix
Mt = Dataset.M;      % Import Mass matrix
Bt = Dataset.B;      % Import B matrix
Ct = Dataset.C;      % Import C matrix
N = 50;             % Number of masses to be used

Damp = Dt(1:N,1:N);  % Select a part of the D matrix
K = Kt(1:N,1:N);  % Select a part of the K matrix
M = Mt(1:N,1:N);  % Select a part of the M matrix

F = [zeros(N) eye(N); -eye(N) -Damp]; % Creates the F matrix
H = [K zeros(N); zeros(N) inv(M)]; % Creates the H matrix
A = F*H; % Calculates the A matrix
B = [zeros([N 1]); Bt(201:200+N)]; % Creates the B matrix
C = [zeros([1 N]) Ct(201:200+N)]; % Creates the C matrix
D = 0; % Sets D to zero
original_system = ss(A,B,C,D); % Creates a state space system

n = 100;              % Define number of states to be used

%% Solve Lyapunov equation for controllability Gramian
sdpsettings('solver','sedumi'); % Selects the sedumi solver
P_1 = sdpvar(n,n);  % n by n matrix with decision variables
F = [P_1 ≥ 10^(-10)*eye(n), (A*P_1)+(P_1*A')+(B*B') ≤ 0]; % Constraints
optimize(F,trace(P_1)); % Optimizes the constraints and sum of the ...
    diagonal of P
P = value(P_1); % Presents the value of the variables in P

%% Solve Lyapunov equation for observability Gramian
sdpsettings('solver','sedumi'); % Selects the sedumi solver
Q_1 = sdpvar(n,n);  % n by n matrix with decision variables
L = [Q_1 ≥ 10^(-10)*eye(n), A'*Q_1+Q_1*A+C'*C ≤ 0]; % Constraints
optimize(L,trace(Q_1)); % Optimizes the constraints and sum of the ...
    diagonal of Q
Q = value(Q_1); % Presents the value of the variables in Q

%% Find transformation matrix T for generalized balancing
PhP=chol(P);    % Finds a matrix R such that P = R'*R by cholesky ...
    factorization
[UPQ,Q2PQ]=svd(PhP*Q*PhP'); % Finds the singular value decomposition
QPQ=sqrt(Q2PQ); % Sigma
Tg=PhP'*UPQ*sqrt(inv(QPQ)); % Finds the transformation matrix T


%% Put the state space in generalized balanced form
```

```
Ag = value(inv(Tg)*A*Tg);          % Balances the original A matrix
Bg = value(inv(Tg)*B);             % Balances the original B matrix
Cg = value(C*Tg);                  % Balances the original C matrix
Dg = 0;                            % Balances the original D matrix

generalized_systems = ss(Ag,Bg,Cg,Dg);    % Creates a state-space system

%% Find the reduced order system
keep = 10; % Defines the number of states in the ROM

Agr = Ag(1:keep,1:keep); % Reduces the A matrix
Bgr = Bg(1:keep); % Reduces the B matrix
Cgr = Cg(1:keep); % Reduces the C matrix
Dgr = 0; % D is still zero

generalized_reduced = ss(Agr,Bgr,Cgr,Dgr); % Creates the generalized ROM

%% Find error bound of generalized balancing
q = diag(QPQ);  % Creates vector d of all the singular values of Sigma
General_TRUNC = q(keep+1:n);    % Creates vector of all truncated singular ...
    values
general_error_bound = 2*sum(General_TRUNC) % Provides the error bound

%% Visualize Hankel singular values of generalized balancing
plot(q,'--*') % Plots the hankel singular values
grid on
title('Hankel singular values Generalized Balancing')
xlabel('State number')
ylabel('Value')

%% Defining new constants and matrices
P_2 = inv(P); % Defines a new variable as the inverse of P

eo = 9000;  % Epsilon o 9000
ec = 9300; % Epsilon c < alpha 9300

Go = eo*Q; % Creates a new matrix Gamma o
Gc = -ec*P;  % Creates a new matrix Gamma c

Xo = -Q*A - A'*Q - C'*C; % Creates a new matrix Xo
Xc = -P_2*A - A'*P_2 - P_2*B*B'*P_2; % Creates a new matrix Xc

Thetao = (Go - Q*A)*(inv(Xo))*(Go - A'*Q); % Creates a new matrix Thetao
Thetac = (-Gc*P_2 + A + B*B'*P_2)*(inv(Xc))*(-P_2*Gc + A' + P_2*B*B'); % ...
    Creates a new matrix Thetac


%% Solve for beta
sdpsettings('solver','sedumi'); % Selects the sedumi solver
beta_1 = sdpvar(1);     % Decision variable for beta
M = [beta_1 ≥ 10^(-10)*eye(n), 2*beta_1*P + 2*Gc - B*B' - Thetac ≥ 0]; % ...
    Constraints
optimize(M,[]);  % Optimizes the constraints
beta = value(beta_1);   % Presents the value of beta

%% Solve for alpha
sdpsettings('solver','sedumi'); % Selects the sedumi solver
```

```
alpha_1 = sdpvar(1);    % Decision variable for beta
U = [alpha_1 ≥ 10^(-10)*eye(n), 2*alpha_1*Q + 2*Go - Thetao ≥ 0];  % ...
    Constraints
optimize(U,[]); % Optimizes the constraints
alpha = value(alpha_1); % Presents the value of alpha

zeta = max(alpha,beta); % Maximum value of alpha or beta

%% Find the Extended controllability and observability Gramians
W = (inv(zeta*P + Gc)); % Computes the extended controllability Gramian
S = Q*(inv(zeta*Q + Go))*Q; % Computes the extended observability Gramian


%% Find transformation matrix T
WhW=chol(inv(W));    % Finds a matrix R such that P = R'*R by cholesky ...
    factorization
[UWS,S2WS]=svd(WhW*S*WhW'); % Finds the singular value decomposition
SWS=sqrt(S2WS); % Sigma
Te=WhW'*UWS*sqrt(inv(SWS)); % Finds the transformation matrix T


%% Put the state space in extended balanced form
Ae = value(inv(Te)*A*Te);   % Balances the original A matrix
Be = value(inv(Te)*B);      % Balances the original B matrix
Ce = value(C*Te);           % Balances the original C matrix
De = 0;                     % Balances the original D matrix

extended_system = ss(Ae,Be,Ce,De); % Creates a state-space system

%% Create reduced order model
Aer = Ae(1:keep,1:keep); % Reduced A matrix
Ber = Be(1:keep);        % Reduced B matrix
Cer = Ce(1:keep);        % Reduced C matrix
Der = De;                % Reduced D matrix
Sigma_Re = SWS(1:keep);  % Reduced Sigma matrix

extended_reduced = ss(Aer,Ber,Cer,Der);  % Reduced state space system

%% Find error bound of extended balancing
d = diag(SWS);  % Creates vector d of all the singular values of Sigma
extended_TRUNC = d(keep+1:n);    % Creates vector of all truncated ...
    singular values
extended_error_bound = 2*sum(extended_TRUNC) % Provides the error bound

%% Visualize Hankel singular values of extended balancing
plot(d,'--*'); % Plots the hankel singular values
grid on
title('Hankel singular values Extended Balancing')
xlabel('State number')
ylabel('Value')

%% Plot Original vs Reduced
t = 0:0.001:50; % Defines the time span for the plot
u = 2*sin(2*t); % Input to the state space systems
y = lsim(original_system,u,t); % Creates a vector of the output of the ...
    original system
```

```
155 z = lsim(generalized_reduced,u,t); % Creates a vector of the output of the ...
        generalized reduced system
156 p = lsim(extended_reduced,u,t); % Creates a vector of the output of the ...
        extended reduced system
157
158 figure(1)
159 plot(t,y)   % Plots the output of the original system
160 hold on
161 plot(t,z)   % Plots the output of the generalized reduced system
162 hold on
163 plot(t,p)   % Plots the output of the extended reduced system
164 title('Output comparison sinusoidal input')
165 legend('Original Model','Generalized Reduced Order Model','Extended ...
        Reduced Order Model')
166 xlabel('Time [s]')
167 ylabel('Velocity [m/s]')
168
169 %% Bode plot Original system
170 bode(original_system) % Plots the bode diagram of the original system
171 grid on
172
173 %% Bode plot Generalized ROM
174 bode(generalized_reduced) % Plots the bode diagram of the generalized ...
        reduced system
175 grid on
176
177 %% Bode plot Extended ROM
178 bode(extended_reduced) % Plots the bode diagram of the extended reduced system
179 grid on
180
181 %% Combined Bode plots
182 figure(2)
183 bode(original_system) % Plots the bode diagram of the original system
184 hold on
185 bode(generalized_reduced) % Plots the bode diagram of the generalized ...
        reduced system
186 hold on
187 bode(extended_reduced) % Plots the bode diagram of the extended reduced system
188 legend('Original Model','Generalized Reduced Order Model','Extended ...
        Reduced Order Model')
189
190 %% H-infinity norm
191 Diff_gb = original_system - generalized_reduced; % Difference between ...
        original and generalized reduced system
192 Diff_eb = original_system - extended_reduced; % Difference between ...
        original and extended reduced system
193
194 hinfnorm(Diff_gb) % H-infinity norm of the generalized reduced system
195 hinfnorm(Diff_eb) % H-infinity norm of the extended reduced system
196
197 %% Running time output original system
198 for i = 1:10
199 tic
200 y = lsim(original_system,u,t); % Creates a vector of the output
201 plot(t,y);
202 title('Output comparison sinusoidal input')
203 legend('Original Model')
```

```
xlabel('Time [s]')
ylabel('Velocity [m/s]')
time_original(i) = toc; % Creates a vector of the results
end
average_original = mean(time_original) % Takes the average of the vector

%% Running time output generalized reduced system
for i = 1:10
tic
z = lsim(generalized_reduced,u,t); % Creates a vector of the output
plot(t,z);
title('Output comparison sinusoidal input')
legend('Generalized Reduced Order Model')
xlabel('Time [s]')
ylabel('Velocity [m/s]')
time_generalized(i) = toc; % Creates a vector of the results
end
average_generalized = mean(time_generalized) % Takes the average of the vector

%% Running time output extended reduced system
for i = 1:10
tic
p = lsim(extended_reduced,u,t); % Creates a vector of the output
plot(t,p);
title('Output comparison sinusoidal input')
legend('Extended Reduced Order Model')
xlabel('Time [s]')
ylabel('Velocity [m/s]')
time_extended(i) = toc; % Creates a vector of the results
end
average_extended = mean(time_extended) % Takes the average of the vector

%% Plot running times
states = [10 25 50 75 100 125 150 175]
time_gb = [0.0705 0.0711 0.0808 0.106 0.116 0.248 0.278 0.336]
time_eb = [0.0624 0.0724 0.0722 0.104 0.115 0.237 0.264 0.320]
plot(states,time_gb,'--*')
hold on
plot(states,time_eb,'--o')
title('Running times output graph')
legend('Generalized Reduced Order Model','Extended Reduced Order Model')
xlabel('States')
ylabel('Running time [s]')
```