



# REDUCTION OF LARGE-SCALE ELECTRICAL MODELS

Bachelor's Project Thesis

P.H.W. Hogendoorn, s2940884, P.H.W.Hogendoorn@student.rug.nl  
Supervisors: Dr. L.P. Borja , Dr. A.J. Bosch , & Prof.Dr.Ir. J.M.A. Scherpen

**Abstract:** The theoretical framework for the application of a model order reduction method named extended balanced truncation, developed by Borja, Scherpen and Fujimoto [5] allows one to reduce a continuous time linear time invariant model while preserving its physical interpretation. The method in question can potentially be applied in the context of large-scale linear electrical networks. This project aims to validate the theoretical framework for applying the reduction method in the mentioned context. This is done by first generating mathematical models of large-scale linear electrical networks using Matlab. Consequently extended balanced truncation is applied using the framework described in [5]. To do so again a Matlab script was written that was able to apply this framework, to persistently apply the framework eliminating deviation due to human error. In order to, validate if the method was successful, the reduced-order models are reconstructed in Simulink and compared to compare with another reduction method, namely the generalized balanced truncation approach and study its capabilities.

## 1 Introduction

In the technological world of today, most processes or systems are described by mathematical models, and simulations are used to predict the behaviour of a system [2]. Unfortunately, the simulation of a whole system is sometimes not feasible due to their often large dimensions [3]. As a result “Model order reduction is critical for engineers and scientists” [12], and offers a solution, by reducing the dimensions of the original model enabling simulations be performed while requiring a smaller amount of computing time.

Since the work of Moore [15] various techniques have been developed [3]. One example of a method for reducing a model is balanced truncation (BT) [17]. This method like many other reduction methods often makes use of a state-space representation for creating a reduced-order model. BT orders the components of a model based on their influence on the outcome and truncates the parts that have little to no influence. Another more complex method called generalized balanced truncation (GBT) uses generalized gramians. These gramians are the solutions to Lyapunov inequalities and are used to reduce the error bound of these reduced-order models [11][6]. Unfortunately, these reduction methods often result in a model without a physical interpretation. Which makes it difficult to interpret the reduced model. Sandberg developed an extension on this model reduction method applicable for discrete-time systems to tackle this problem. By using a combination of GBT and port-hamiltonian (PH) systems to develop a new reduction method called extended balanced truncation (EBT) [16]. One of the key benefits of this reduction method is not only its ability to reduce the dimensions of the original model, but it is also able to preserve a particular structure. Meaning the reduced-order model of, for example, an electrical circuit could again be represented in the form of an electrical circuit similar to its original model. The PH framework generally encodes more structural information about the physical system with respect to other mathematical modes, such as general input-affine representations [19]. Moreover, the PH system modelling can be regarded to bridge the gap between passive system models and explicit physical network realizations[19]. Scherpen and Fujimoto further developed this method for a possible application on continuous time linear time invariant (CTLTI) systems [18]. Borja, Scherpen and Fujimoto continued building on to this research to develop a framework for applying this reduction method of EBT on CTLTI [5].

A relevant context with the need for model reduction is that of large-scale electrical networks (LSEs). Electricity grids exhibit several typical features of complex networks[21] and are one example of these LSEs with often large dimensions where simulation of the whole network takes significant time. As a result, model order reduction is a relevant topic in this context. The Framework developed in [5] could further improve the capabilities of modeling LSEs.

This project aims to investigate the method for creating a reduced-order model as described in [18] and its possibilities for the application to LSEs by applying the theoretical framework described in [5]. One of the main objectives of this project is to validate the theoretical framework for applying the method in question in the relevant context of LSEs through simulations using Simulink. As a result, the simulations should prove the method guarantees an acceptable error ratio while preserving the physical interpretation of the reduced-order model. The acceptable error ratio dependent on the size of the truncated part. However, for practical purpose, we consider that such an error must be less than 5%.

## 2 Notation

$\dot{X}$  represents the time-derivative of  $X$   
 $C$  represents the capacitance of a capacitor  
 $L$  represents the inductance of an inductor  
 $R$  represents the resistance of a resistor  
 $I_x$  represents the current at a component  $x$   
 $V_x$  represents the voltage over a component  $x$   
 $U$  represents the voltage as an input from a voltage source  
 $I$  represents the identity matrix  
 $Matrix A \geq 0$  notes a positive-semi-definite matrix  
 $\sigma$  represents a singular value

## 3 Preliminaries

This research uses a verity of principles and originated from linear algebra. For understanding how the methods used in this paper some a brief description of these principles are given.

**Preliminary 1** One of these principles for linear algebra is a positive-definite matrix. A matrix is said to be positive-definite if its eigenvalues are strictly greater than zero. When a matrix has its eigenvalues greater or equal to zero, this matrix is said to be positive-semi-definite. Lastly, a matrix is said to be negative-definite or negative-semi-definite when all its eigenvalues are lower than zero or lower or equal to zero respectively [4].

**Preliminary 2** singular value decomposition (SVD) is a method used for separating a matrix into its key features [7]. The SVD of a matrix consist of three matrices, A orthogonal matrix  $U$ ,  $\Lambda$  and orthogonal matrix  $V^T$ , where  $\Lambda$  contains all singular values of the corresponding matrix, ordered in its diagonal [7] (see equation 3.1).

$$svd(A) = U_A \Lambda_A V_A^T \quad (3.1)$$

Matlab is easily able to obtain the SVD of a matrix using the method as described by [1]. The singular values of a matrix are equivalent to the absolute values of the eigenvalues. If a matrix is a strictly diagonal and positive-definite then all its eigenvalues are stored in the diagonal of this matrix. In addition the eigenvalues and singular-values are the same. Moreover, when a matrix is diagonal  $U_A = V_A$ .

**Preliminary 3** The method of EBT requires similarity an congruence transformations. In linear algebra when a matrix ( $A$ ) is diagonal it is possible to state the following:

$A \in R^{2nx2n}$ , symplectically similar to  $B \in R^{2nx2n}$  if there exists a symplectic matrix  $C \in R^{2nx2n}$  such that  $C^1 A C = B$  [10]

$A \in R^{2nx2n}$ , symplectically congruent to  $B \in R^{2nx2n}$  if there exists a symplectic matrix  $C \in R^{2nx2n}$  such that  $C^T A C = B$  [10]

**Preliminary 4** Some more generally used definition in linear algebra are obtained from Horn and Johnson [13]. The following two definitions found in [13]. These definitions are only valid for any matrix  $A$  for which there exists a matrix  $B$  which is equivalent the inverse of matrix  $A$ . This definition is essential for applying the frame work of [5] and is defined as flow;

$$A * A^{-1} = I \quad (3.2)$$

$$A * I = A^{-1} \quad (3.3)$$

## 4 Modeling of electrical networks

Before it is possible to apply the theoretical framework in question in the relevant context, models of Large-scale electrical networks need to be obtained. To do so a Matlab script is written that can generating

random models representing LSEs. The theory used for creating these models is based on the research of Castaños, Jayawardhana, Ortega and García-Canseco [8] and Jeltsema [14]. The models are made using a combination of Kirchhoff's circuit laws for linear (Equation 4.1 and 4.2) and parallel circuits (Equation 4.3 and 4.4) and Ohm's law for current over an capacitors and voltage over an indicators (Equation 4.5 and 4.6)

$$\sum_{i=1}^n V_i = 0 \quad (4.1)$$

$$I_1 = I_2 = \dots = I_n \quad (4.2)$$

$$V_1 = V_2 = \dots = V_n \quad (4.3)$$

$$\sum_{i=1}^n I_i = 0 \quad (4.4)$$

$$V_l = \dot{I}_l L \quad (4.5)$$

$$I_c = \dot{V}_c C \quad (4.6)$$

By applying these laws to several standard forms of electrical circuits a mathematical representation of these electrical circuits can be obtained.

**Assumption 1** In this study, we assume that if a circuit has a voltage source there is always one resister directly in series with this voltage source.

The method of EBT uses a state-space representation in a port-Hamiltonian form (see 4.7). In a port-hamiltonian representation, the hamiltonian,  $H(x)$  contains the total energy of the system [20], with  $H = H > 0$ ; and  $R = R^T 0, J = J^T$  [5]. In other words, The matrix  $H$  contains all information regarding the energy storing elements. In the case of A RLC circuit, this means all components of  $L$  (inductor) and  $C$  (capacitors). The matrix  $R$  consists of information regarding the resisting elements.

$$\sum_H : \begin{cases} \dot{x} &= (J - R)Hx + Bu \\ y &= B^T Hx \\ \mathcal{H}(x) &= \frac{1}{2}x^T Hx \end{cases} \quad (4.7)$$

in order to easily obtain the matrices  $J, R, H$  and  $B$  the model are generated by expressing  $\dot{V}_{ci} C$  and  $\dot{I}_{li} L$  in term of  $V_c, I_l, R$  and *input*  $U$  or  $I_0$ . this will allow an expression in the following form to be obtained:

$$\begin{pmatrix} L & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} \dot{I}_l \\ \dot{V}_c \end{pmatrix} = \begin{pmatrix} R_l & J_1 \\ -J_1^T & R_c \end{pmatrix} \begin{pmatrix} I_l \\ V_c \end{pmatrix} + \begin{pmatrix} B \end{pmatrix} U \quad (4.8)$$

where:

$$\begin{pmatrix} L & 0 \\ 0 & C \end{pmatrix} = H \quad (4.9)$$

$$\begin{pmatrix} I_l \\ V_c \end{pmatrix} = x \quad (4.10)$$

$$\begin{pmatrix} R_l & J_1 \\ -J_1^T & R_c \end{pmatrix} = J - R \quad (4.11)$$

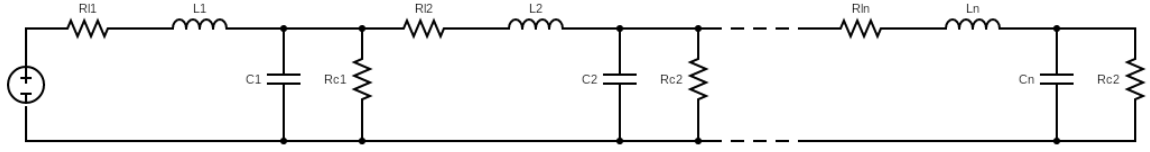
#### 4.1 Model electrical circuit type 1

The first standard electrical network is represented in figure 4.1. Here an inductor is always connected in-series to a resistor and, a capacitor in parallel with a resistor.

The smallest circuit of this form only requires  $L_1, C_1, R_{l1}, R_{c1}$  and a power supply  $U$ . Using Kirchhoff's laws (Equation 4.1 , 4.2 , 4.3 and 4.4) and Ohm's law (Equation 4.5 and 4.6) we obtain the following two equations to represent this model:

$$U = V_{c1} + L_1 \dot{I}_{l1} + I_{l1} R_{l1} \quad (4.12)$$

$$I_{l1} = C_1 \dot{V}_{c1} + \frac{V_{c1}}{R_{c1}} \quad (4.13)$$



**Figure 4.1: Standard circuit type 1**

Reordering these equations will allow us to easily create a state-space representation of this model (equations 4.14, 4.15 and 4.16).

$$L_1 \dot{I}_{l1} = U - V_{c1} - I_{l1} R_{l1} \quad (4.14)$$

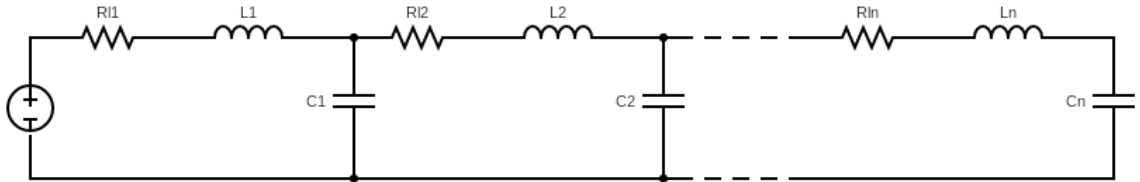
$$C_1 \dot{V}_{c1} = I_{l1} + \frac{V_{c1}}{R_{c1}} \quad (4.15)$$

$$\begin{pmatrix} L_1 \dot{I}_{l1} \\ C_1 \dot{V}_{c1} \end{pmatrix} = \begin{pmatrix} -R_{l1} & -1 \\ 1 & -\frac{1}{R_{c1}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ V_{c1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} U \quad (4.16)$$

In this standard model of an electrical circuit, addition a component containing  $L$ ,  $C$ ,  $R_l$ ,  $R_c$  will influence Equations 4.13 by replacing the term  $V_{ci}$  with the highest value for  $i$  with  $R_{l(i+1)} I_{l(i+1)} + L_{(i+1)} \dot{I}_{l(i+1)} + V_{c(i+1)}$ . A similar change occurs to Equation 4.12. Here a term  $I_{l(i+1)}$  is added per additional component to the original equation. Witch can later be replaced by  $V_{c(i+1)} / R_{c(i+1)} + C_{(i+1)} \dot{V}_{c(i+1)}$ . As a result, a circuit of model type 1 can be represented as equations B.1 and the corresponding state-space form B.2

#### 4.1.1 Model electrical circuit type 1.2

Electrical circuit type 1.2 is a variant of type 1. The main difference being the absence of the resistor over the capacitor (see figure 4.2).



**Figure 4.2: Standard circuit type 1.2**

In this case only a component  $L_1$ ,  $C_1$ ,  $R_{l1}$  and a power supply  $U$  are needed for the smallest circuit of this form. Again, using Kirchhoff's laws (Equation 4.1 , 4.2 , 4.3 and 4.4) and Ohm's laws (Equation 4.5 and 4.6) we are able to a mathematical representation of the circuit, which also can be represented in state-space form.

$$U = V_{c1} + L_1 \dot{I}_{l1} + I_{l1} R_{l1} \quad (4.17)$$

$$I_{l1} = C_1 \dot{V}_{c1} \quad (4.18)$$

$$\begin{pmatrix} L_1 \dot{I}_{l1} \\ C_1 \dot{V}_{c1} \end{pmatrix} = \begin{pmatrix} -R_{l1} & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} I_{l1} \\ V_{c1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} U \quad (4.19)$$

With electrical circuit 1.2, addition of a component containing  $L$ ,  $C$  and  $R_l$  requires an additional equation. Fortunately this equation is easily to ascertain. The equation in question is an alteration of equations 4.17 and is obtained by by taking out the term  $V_{ci}$  with the highest value for  $i$  and replacing

this with  $R_{l(i+1)} I_{l(i+1)} + L_{(i+1)} \dot{I}_{l(i+1)} + V_{c(i+1)}$ . with equation 4.18 a term  $I_{l(i+1)}$  is added to the right side of the equation. Moreover,  $I_{l(i+1)}$  can in its turn be defined as  $I_{l(i+1)} + I_{c(i+2)}$ . All further  $I_{L_i}$  (for  $i=3,4,\dots,n-1$ ) can be defined in a similar way. The definition of the final component in respect to  $I_{L_n}$  is however slightly different, since it is the same as  $I_{c_n}$  and therefore  $C_n \dot{V}_{c_n}$ . Using the obtained equations a mathematical model for a circuit of model type 1.2 can be represented in state-space form (see appendix equation B.5)

#### 4.1.2 Model electrical circuit type 1.3

The second variant of standard electrical circuit type 1 is type 1.3. This circuit is again similar to type 1 except for the positioning of the resistors. In the model of electrical circuit type 1.3, the resistor in series with the inductor is taken out (see figure 4.3). As previously mentioned a resistor is always present next to a voltage source this is also the case here. This makes this circuit equal to type 1 for the smallest possible form. Only from the second "group" on the resistor in series with the inductor is taken out and the resulting state-space representation is different.

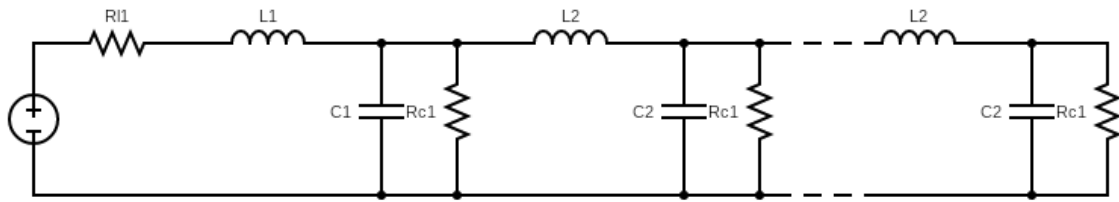


Figure 4.3: Standard circuit type 1.3

As mentioned in contrast to type 1 for all additional components of this circuit no  $R_l$  is included. Applying a similar approach to obtain a the state-space representation of this model is used as with type 1. The mathematical representation for a second component is as a result as follow:

$$U = L_1 \dot{I}_{l1} + I_{l1} R_{l1} + L_2 \dot{I}_{l2} + V_{c2} \quad (4.20)$$

$$I_{l1} = C_1 \dot{V}_{c1} + \frac{V_{c1}}{R_{c1}} + C_2 \dot{V}_{c2} + \frac{V_{c2}}{R_{c2}} \quad (4.21)$$

$$\begin{pmatrix} L_1 \dot{I}_{l1} \\ L_2 \dot{I}_{l2} \\ C_1 \dot{V}_{c1} \\ C_2 \dot{V}_{c2} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & \frac{1}{R_{c1}} & 0 \\ 0 & 1 & 0 & \frac{1}{R_{c2}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ I_{l2} \\ V_{c1} \\ V_{c2} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} U \quad (4.22)$$

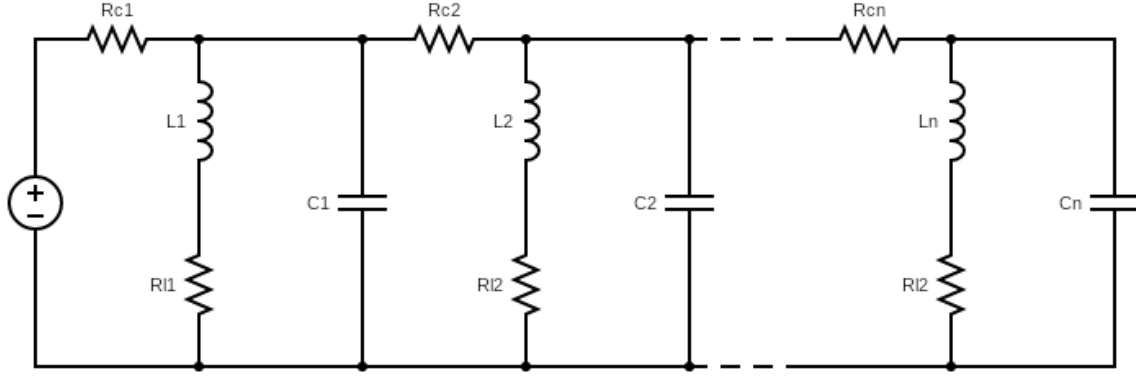
By analyzing this model type it quickly becomes clear state-space of model type 1.3 is equal to the state-space of type 1 with all  $R_l$  equal to zero except for  $R_{l1}$  giving a state-space of the model in the form of equation B.6

#### 4.1.3 The generation of models containing component type 1,1.2 and 1,3

Using these three standard forms a Matlab script was written to create a mathematical model of these circuits. The Matlab script is made in such a way that output will provide all elements needed for creating a state-space representation like 4.7. The Matlab model can be found in appendix D.3. This model can present all possible combinations of models type 1, 1.2 and 1.3.

## 4.2 Model electrical circuit type 2

In the model type 2 we look at a capacitor in parallel over a inductor. In this case an inductor is still in series with a resistor, however the inductor is connected in-series to a resistor while in parallel to the capacitors (see image 4.4)



**Figure 4.4: Standard circuit type 2**

The smallest option for this circuit is represented in the following three equations (4.23, 4.24 and 4.25) using the laws of Kirchhoff and Ohm.

$$I_{Rc1} = I_{l1} + \dot{V}_{c1} C_1 \quad (4.23)$$

$$U = R_{c1} I_{Rc1} + V_{c1} \quad (4.24)$$

$$V_{c1} = \dot{I}_{l1} L_1 + I_{l1} R_{l1} \quad (4.25)$$

Reordering equation 4.25 gives an expression for  $\dot{I}_{l1} L_1$ . Using equation 4.23 and substituting this equation in 4.24 allowed an expression for  $\dot{V}_{c1} C_1$  to be obtained in terms of  $R_{c1}, R_{l1}, I_{l1}, V_{c1}$ . These equations are obtained represented in equations 4.26 and 4.27, which can be put into a state-space form of 4.32

$$\dot{V}_{c1} C_1 = I_{l1} - \frac{V_{c1}}{R_{c1}} + \frac{U}{R_{c1}} \quad (4.26)$$

$$\dot{I}_{l1} L_1 = V_{c1} - I_{l1} R_{l1} \quad (4.27)$$

$$\begin{pmatrix} L_1 \dot{I}_{l1} \\ C_1 \dot{V}_{c1} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 1 \\ -1 & \frac{1}{R_{c1}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ V_{c1} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{R_{c1}} \end{pmatrix} U \quad (4.28)$$

Like type 1 a model of type 2 is also easily Up-scaled. This is done by adding a component containing a  $R_{c2}, R_{l2}, L_2$  and  $C_2$  will influence equations 4.23 and as a result the equation will have a additional component  $I_{Rc1}$  (see equation 4.31). With respect to the equations defining the voltage of source  $U$  a new additional definition will exist where:

$$V_{c1} = R_{c2}(I_{l2} + \dot{V}_{c2} C_2) + V_{c2} \quad (4.29)$$

$$V_{c2} = \dot{I}_{l2} L_2 + R_{l2} I_{l2} \quad (4.30)$$

new expression for  $I_{Rc1}$ ;

$$I_{r1} = \dot{V}_{c1} C_1 + I_{l1} + \dot{V}_{c2} C_2 + I_{l2} \quad (4.31)$$

Expressing  $\dot{I}_{l2} L_2, \dot{I}_{l1} L_1$  and  $\dot{V}_{c2} C_2$  in terms of  $R_c, R_l, L$  and  $C$  is easily done by reordering equations 4.33, 4.25 and 4.35 respectively. For obtaining  $\dot{V}_{c1} C_1$  some additional substitution has to be done. Using the found expression for  $\dot{V}_{c2}$  in equation 4.31 and substituting this equation in equation 4.24 allows  $\dot{V}_{c1} C_1$  to be defined in term of  $R_c, R_l, L$  and  $C$ . The resulting equations can again be written in a state space form (see equation 4.32).

$$\begin{pmatrix} L_1 \dot{I}_{l1} \\ L_2 \dot{I}_{l2} \\ C_1 \dot{V}_{c1} \\ C_2 \dot{V}_{c2} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 0 & 1 & 0 \\ 0 & -R_{l2} & 0 & 1 \\ -1 & 0 & -\frac{1}{R_{c1}} - \frac{1}{R_{c2}} & \frac{1}{R_{c2}} \\ 0 & -1 & \frac{1}{R_{c2}} & -\frac{1}{R_{c2}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ I_{l2} \\ V_{c1} \\ V_{c2} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{R_{c1}} \\ 0 \end{pmatrix} U \quad (4.32)$$

Increasing the size of a circuit type 2 to  $n$  components can be done in a similar way. The expressions for all  $\dot{I}_{li} L_i$  with  $i = 1, 2, \dots, n$  can always be defined as:

$$\dot{I}_{li} L_i = V_{ci} - R_{li} I_{li} \quad (4.33)$$

For determining  $\dot{V}_{ci} C_i$  with  $i = 2, 3, \dots, n$  (so not for  $i=1$ ) we can state:

$$\dot{V}_{ci} C_i = \frac{V_{c(i-1)}}{R_{ci}} - \frac{V_{ci}}{R_{ci}} - I_{li} \quad (4.34)$$

And  $\dot{V}_{c1} C_1$

$$\dot{V}_{c1} C_1 = -I_{l1} - \frac{V_{c1}}{R_{c1}} - \frac{V_{c1}}{R_{c2}} + \frac{V_{c2}}{R_{c2}} + \frac{U}{R_{c1}} \quad (4.35)$$

#### 4.2.1 The generation of models containing component type 2

Like with type 1 the model type two has also the ability to have components without a resistor. However, Taking out components ( $R_c$ ) will result in two parallel capacitors. In these cases, the equivalent capacitance over these capacitors is equal to the sum of the parallel capacitors. And the same holds for the indicators. As a result, if  $R_{ci} = 0$ , the model can be reduced without any loss of accuracy and situations in which this is the case are considered not to be relevant for this study. Taking the resistor  $R_{li}$  (positioned in series with an inductor) out of the circuit, however, will not result in a possibility to reduce the model without reducing the accuracy. The effect on the circuit will be the equivalent to the same state-space representation where  $R_{li}$  is equal to 0. With this in mind, a Matlab script was written to obtain a matrix H, J, R, and B for a circuit in the form of model type 2 (see appendix D.4).

### 4.3 Model electrical circuit type 3

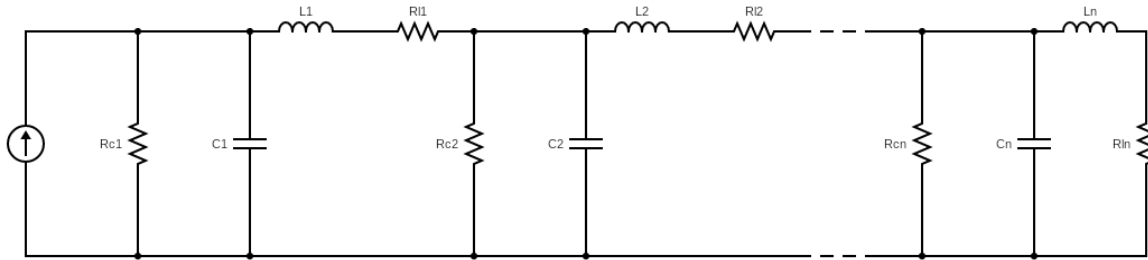


Figure 4.5: Standard circuit type 3

The third and final model of electrical circuits that are considered makes use of a current source. the model is represented in 4.5. Taking again a similar approach as with analyzing model type 1 and 2, we find the state space of this model is represented similar to model type 1. The only difference being is the input. In model type 3 the input affects  $\dot{V}_{c1} C_1$  instead of  $\dot{I}_{l1} L_1$  and naturally, the input is expressed in units of amps (current) instead of a potential difference (volts). The state-space of this model is represented in equation B.8 and the Matlab code for generating model type3 in D.5

## 5 Extended balanced truncation

Consider a continuous-time linear time-invariant (CTLTI) system described as:

$$\sum : \begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx \end{cases} \quad (5.1)$$

Where  $x \in R^n$  is the state-vector, for  $m \leq n$ ,  $u \in R^m$  is the input vector and  $y \in R^q$  denotes the output vector. Accordingly,  $A \in R^{n \times n}$ ,  $B \in R^{n \times m}$  and  $C \in R^{n \times m}$ . Assume that the system as described in 5.1 is asymptotically stable. As a result the so-called generalized observability gramians  $Q \in R^{n \times n}$  are a positive semi-definite solutions to the following Lyapunov inequality;[5]

$$QA + A^T Q + C^T C \leq 0 \quad (5.2)$$

Analogously, the generalized controllability gramians  $\check{P} \in R^{n \times n}$  are given by positive semi-definite solutions to

$$A\check{P} + \check{P}A^T + BB^T \leq 0 \quad (5.3)$$

In particular, when 5.2 and 5.3 are equalities, the matrices  $Q$  and  $\check{P}$  are known as the standard observability and controllability Gramian, respectively. For further details, we refer the reader to [2]. Extended balanced truncation (EBT), is a reduction method that uses the so-called generalized observability Gramians  $Q \in R^{n \times n}$  and generalized controllability Gramians  $\check{P} \in R^{n \times n}$  [2]

## 5.1 Generalized Balanced Truncation

Generalized balanced truncation (GBT) aims to find a matrix  $W$  that solves[5]:

$$WQ\check{P}W^{-1} = \Lambda_{QP}^2 \quad (5.4)$$

Before truncating the system, it first needs to be balanced. Balancing is based on obtaining a matrix  $\Lambda_{QP}$  for which equation 5.4 holds. Here  $\Lambda_{QP}$  is a diagonal matrix containing all singular values of the matrix  $Q$  and  $\check{P}$ . Moreover, the singular values in matrix  $\Lambda_{QP}$  are ordered with respect to size. In other words  $\Lambda_{QP} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$  with  $\sigma_i \geq \sigma_{i+1}$ . Obtaining this matrix  $\Lambda_{QP}$  is done by singular value decomposition (SVD) (see section 3 *Preliminary 2*).

$$\check{P} = Q = \Lambda_{QP} \quad (5.5)$$

Since the multiplying any matrix by its inverse equals the identity matrix, and a multiplication of any matrix by the identity matrix equals the original matrix (see section 3 *Preliminary 4*), it is possible to write 5.4 as 5.6 and 5.5 as 5.7 and 5.8:

$$WQW^T W^{-T} \check{P}W^{-1} = \Lambda_{QP}^2 \quad (5.6)$$

$$WQW^T = \Lambda_{QP} \quad (5.7)$$

$$W^{-T} \check{P}W^{-1} = \Lambda_{QP} \quad (5.8)$$

Assuming there exist a matrix  $\phi_Q$  for all possible matrices  $Q$  (see equation 5.9) we can also express  $Q\check{P}$  and  $\Lambda_{QP}$  as in equations 5.10 and 5.11.

$$Q = \phi_Q^T \phi_Q \quad (5.9)$$

$$Q\check{P} = \phi_Q \check{P} \phi_Q^T = U_{QP} \Lambda_{QP}^2 U_{QP}^T \quad (5.10)$$

$$\Lambda_{QP} = \Lambda_{QP}^{-\frac{1}{2}} U^T \phi_Q \check{P} \phi_Q^T U \Lambda_{QP}^{\frac{1}{2}} \quad (5.11)$$

Using these equations (5.8, 5.10 and 5.11) we are able to obtain an expression for  $W$  at last.

$$W = \Lambda_{QP}^{\frac{1}{2}} U^T \phi_Q^{-T} \quad (5.12)$$

When  $W$  is obtained the system is balanced (see 5.13) and the bottom states are truncated.  $Ab$ ,  $Bb$  and  $Cb$  represent the balanced matrices.

$$\begin{aligned} Ab &= W^{-1} A W \\ Bb &= W^{-1} B \\ Cb &= C W \end{aligned} \quad (5.13)$$



## 5.2 Extended Balanced Truncation

With the new method, a similar approach is taken. First the system needs to be balanced such that equation 5.5 with  $\Lambda_{QP} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$  with  $\sigma_i \geq \sigma_{i+1}$ . Afterwards, a matrix  $W$  needs to be found to obtain a matrix balanced system which can later be truncated. The main difference is as previously mentioned in using a PH representation of the system. With EBT the reduction of a model based on the hamiltonian matrix contains all information of the energy storing elements of a circuit.  $\check{P}$  and  $Q$  are defined as follow:

$$\check{P} = \delta_o H^{-1} \quad (5.14)$$

$$Q = \delta_c H \quad (5.15)$$

Here  $\delta$  is scalar obtained by solving equality of equation 5.2 and 5.3. The value for  $\delta$  increasing slightly so the left side of the equation is a negative definite matrix, and constraint 5.2 and 5.3 are not violated due to rounding error. If possible both values for the  $\delta$  are said to be equal. Using SVD to obtain  $\Lambda_{QP}$  in this is however difficult. Since  $Q$  and  $\check{P}$  are a scalar multiplied by  $H$  and its inverse,  $Q\check{P} = \delta_o \delta_c I$  (see section 3 *Preliminary 4*). As a result the SVD of  $Q\check{P}$  would result in  $\Lambda_{QP} = \delta_o \delta_c I$ . This result all relevant information is as a result lost rather balancing the system. To tackle this problem EBT uses extended gramians  $S$  and  $T$  (defined in equation 5.16 and 5.17) instead of  $Q$  and  $\check{P}$ . Here  $\Gamma_o$  and  $\Gamma_c$  are matrices taken to be a diagonal while baring a strong resemblance but are not equal to  $Q$  and  $\check{P}$  respectively, and  $\beta$  and  $\alpha$  are scalars.

$$S = Q(\alpha Q + \Gamma_o)^{-1} Q \quad (5.16)$$

$$T = (\beta \check{P} + \Gamma_c)^{-1} \quad (5.17)$$

Matrix  $S$  and  $T$  are solutions to the linear matrix inequalities (LMIs) 5.18 and 5.19. The scalars and matrices that result from the definition of  $S$  and  $T$  need to be determined in such a way that the LMIs hold. in these LMIs  $X_o$ ,  $A_o$  and  $A_c$  are defined as in C.1 and C.2.

$$\begin{pmatrix} X_o & Q - A_o^T S \\ Q - S^T A_o & S + S^T \end{pmatrix} \geq 0 \quad (5.18)$$

$$\begin{pmatrix} -PA - A^T P & -P + A^T T & -2PB \\ -P + T^T A_c & T + T^T & 2T^T B \\ -2B^T P & 2B^T T & 4I_m \end{pmatrix} \geq 0 \quad (5.19)$$

When matrices  $S$  and  $T$  are obtained the system is balanced in and reduced in the same way as with GBT, by finding a matrix  $W$ . Now using  $S$ ,  $T$  and  $\Lambda_{ST}$  instead of  $Q$ ,  $\check{P}$  and  $\Lambda_{QP}$

## 6 Application of EBT

With the method for creating mathematical representations of LSEs and obtaining insight into framework for applying EBT, it is possible to investigate the reduction method in the context of LSEs. At first, values have to be determined for  $\delta_c$ ,  $\delta_o$ ,  $\beta$ ,  $\Gamma_c$  and  $\Gamma_o$ . It is assumed there exists an optimal value for these variables whit whom the resulting reduced-order models contain the smallest deviation from the original. However, determining these optimal values is not the objective of this paper and might be relevant for future research. Nevertheless, as can be learned from econometric an optimal solution often can be found on the boundary conditions [9]. Using this idea, the value for  $\delta_o$  is determined by establishing the smallest possible value that allows  $X_o$  to be a positive-semi-definite matrix. Or in other words, taking the smallest possible  $\delta_o$  so the lowest eigenvalue of  $X_o$  equals 0. Lastly, to avoid violating the constraints due to round-off error  $\delta_o$  is increased slightly. If using the same value for  $\delta_c$  as  $\delta_o$  allows  $X_c$  to be a positive-semi-definite matrix,  $\delta_c$  is taken to be equal to  $\delta_o$ . Otherwise,  $\delta_c$  is calculated similarly to  $\delta_o$ , by taking the smallest possible  $\delta_c$  so the lowest eigenvalue of  $X_c$  equals 0.

The other values that need to be obtained are  $\beta$ ,  $\Gamma_c$ , and  $\Gamma_o$ . Again an approximation of the boundary conditions is taken. In this case, these conditions should relate to 5.18 and 5.19. As mentioned in section 5,  $\Gamma$  marks a close resemblance to  $\check{P}$  and  $Q$ . For testing the model  $\Gamma_c$  and  $\Gamma_o$  are determined as following:

$$\Gamma_c = \epsilon_c \check{P} \zeta \quad (6.1)$$

$$\Gamma_o = \epsilon_o Q \zeta \quad (6.2)$$

$$\zeta = \text{diagonal}(1.1^1, 1.1^2, 1.1^3, \dots, 1.1^n, 1.1^1, 1.1^2, 1.1^3, \dots, 1.1^n) \quad (6.3)$$

For defining  $\zeta$ ,  $n = \text{number of inductors or capacitors}$ . Using these definitions,  $\beta$  is set to equal 1 and increased with a multiplication of ten until the constraint 5.19 is solved with  $\epsilon_c$  fixated to equal one. Afterwards the this rough boundary condition of  $\beta$  is taken and the same constraint is solves, now altering the value for  $\epsilon_c$ . Initially,  $\epsilon_c$  is still set equals 1, and is increased with steps of five until one additional increase of the same size violates constraint 5.19. Using the same approach for defining  $\epsilon_c$  we define  $\epsilon_o$  now solving constraint 5.18 and setting  $\alpha = \beta$ .

A Matlab code is made that calculates these values for  $\delta_c$ ,  $\delta_o$ ,  $\beta$ ,  $\Gamma_c$  and  $\Gamma_o$ , and reduces the model using EBT as well as GBT (For the complete script see appendix D).

## 7 Reduced model evaluation

To validate if EBT can reduce the model and still preserve the original structure, eight different models have been constructed and reduces. The dimensions, reduction and error bound of these models are displayed in Table A.1, A.2 and A.3. To study the behaviour of these reduced models a plot is made (see appendix A). The upper plot displays the outputs of the original system and the reduced system (EBT and GBT), the middle plot shows the deviation of the EBT- and GBT- model from the original model and The third plot shows the input function is shown. In addition, a plot of the error bound is created for five other models with 100 capacitors and 100 redactors, with respect to the percentage of reduction. These can be seen in figure; A.1, A.2, and A.3. The calculation of the error-bound uses the method of Willems [22] (all data can be found at [https://phw-h.github.io/IDP\\_extended\\_balanced\\_truncation/](https://phw-h.github.io/IDP_extended_balanced_truncation/)).

### 7.1 Results

The obtained data support the findings of Sandberg in [16], since it quickly becomes clear that EBT is not only able to provide a reduced model with preservation of the physical interpretation, but also returns a reduced model with a lower error bound in comparison to GBT. The models reduced with EBT up to 80%, show a error bound that is almost to be negligible. Further reduction however seems to drastically worsen the accuracy of the model. One of the reasons for GBT is frequently drastically un-accurate is due to its loss of all scalars (that are not equal to zero) in its B matrix. As a result, its reduced system no longer responds to any input. Lastly, simulations of model type 3 in simulating frequently gave an error. This was due to a high value for the resistance in  $R_c(1)$ . The error occurred when running the original model and was thus considered not to be a result of a flaw in the reduction technique but. The needs for a restriction of the maximal resistance in  $R_c(1)$  can possibly be explained by investigating the mathematical model. With a relatively high resistance in  $R_c(1)$  the input of the model will be scaled by one over this resistance. Making the input of the system have a significantly small effect on the output. This makes the input inefficient and makes the model unfavourable to use. Moreover, As a result of the consequently relative small influence of the input on the system, round-of errors are likely to occur. It should be noted that these findings only for the model types considered in this paper, and therefore it does not mean they are valid for all models of CTLTI-LSNs.

## 8 Conclusions and Discussion

The framework for applying EBT developed by Borja, Scherpen and Fujimoto [5] (EBT) shows great promise. The methods return reduced models with an almost negligible error-bound. Only when the model is reduced by 80% or more the error-bound start to increase. The framework for the application of the technique, unfortunately, has no optimal format and as a result, the findings in this paper do not show a complete picture. Future research to could aim to find a way of determining the optimal values for  $\delta_c$ ,  $\delta_o$ ,  $\Gamma_c$ ,  $\Gamma_o$ ,  $\beta$  and  $\alpha$  to better establish the possibilities for applying EBT to CTLTI-LSNs. In addition, this research focuses on the use of three standard types of electrical circuits. Making the findings not hold for all electrical circuits. Lastly, the small error bound of the reduction method of EBT raises some suspicion. With the error bound occasionally equal to zero, suggest the use of Matlab and Simulink might not be optimal for validating this reduction method. It is hypothesised Matlab is not capable of handling significantly small numbers.

## References

- [1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 3 edition, aug 1999.
- [2] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society of Industrial and Applied Mathematics, Philadelphia, illustrate edition, 2005.
- [3] M. Beitelshmidt and P. Koutsovasilis. Comparison of model reduction techniques for large mechanical systems-A study on an elastic rod. *Multibody System Dynamics*, pages 111–128, 2008.
- [4] R. Bhatia. *Positive definite matrices*. Princeton, 2009.
- [5] P. Borja, J. M. A. Scherpen, and K. Fujimoto. Extended balancing of continuous LTI systems: a structure-preserving approach. Technical report.
- [6] R. C. Brown and D. B. Hinton. Lyapunov Inequalities and their Applications. In *Survey on Classical Inequalities*, pages 1–25. Springer Netherlands, 2000.
- [7] S. L. Brunton and J. N. Kutz. *Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control*. 2017.
- [8] F. Castaños, B. Jayawardhana, R. Ortega, and E. García-Canseco. Proportional plus integral control for set-point regulation of a class of nonlinear RLC circuits. *Circuits, Systems, and Signal Processing*, 28(4):609–623, 2009.
- [9] B. D. Craven. Boundary conditions optimal control. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 30(3):343–349, 1989.
- [10] R. J. De La Cruz and H. Faßbender. *On the diagonalizability of a matrix by a symplectic equivalence, similarity or congruence transformation*, volume 496. Elsevier Inc., 2016.
- [11] G. E. Dullerud and F. Paganini. *A course in robust control theory: a convex approach*. Springer Science & Business media, 36 edition, 2013.
- [12] L. Fortuna, G. Nunnari, and A. Gallo. *Model Order Reduction Techniques with Applications in Electrical Engineering*. Springer London, 1992.
- [13] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, New York, 2 edition, 1985.
- [14] D. Jeltsema. Modeling and Control of Nonlinear Networks A Power-Based Perspective. *Perspective*, (August):231, 2005.
- [15] B. C. Moore. Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, 1981.
- [16] H. Sandberg. Model reduction of linear systems using extended balanced truncation. In *American Control Conference*, pages 4654–4659, Seattle, 2008.
- [17] H. Sandberg and A. Rantzer. Balanced Truncation of Linear Time-Varying Systems. *IEEE Transactions on Automatic Control*, 49(2):217–229, 2004.
- [18] J. M. Scherpen and K. Fujimoto. Extended balanced truncation for continuous time LTI systems. *2018 European Control Conference, ECC 2018*, pages 2611–2615, 2018.
- [19] A. Van Der Schaft. *Communications and Control Engineering L2-Gain and Passivity Techniques in Nonlinear Control*. 2017.
- [20] A. J. Van Der Schaft and R. V. Polyuga. Structure-preserving model reduction of complex physical systems. *IEEE Conference on Decision and Control*, pages 4322–4327, 2009.
- [21] X. Wei, S. Gao, T. Huang, T. Wang, and W. Fan. Identification of two vulnerability features: A new framework for electrical networks based on the load redistribution mechanism of complex networks. *Complexity*, 2019.
- [22] J. C. Willems. MODEL REDUCTION by BALANCING (slide), 2002.

## A Appendix Tables and Figures

| Model | Input      | Reduction | Dimintions  | Error-Bound GBT | Error-Bound EBT |
|-------|------------|-----------|-------------|-----------------|-----------------|
| 1     | step 50    | 25%       | C:20 L:20   | 1.6685e-7       | 0               |
| 2     | step 50    | 25%       | C:40 L:40   | 2.1235e-5       | 0               |
| 3     | step 50    | 25%       | C:60 L:60   | 2.1920e-5       | 5.4210e-20      |
| 4     | sinusoidal | 25%       | C:40 L:40   | 3.7724e-4       | 2.1684e-18      |
| 5     | step 50    | 50%       | C:20 L:20   | 4.3211e-4       | 5.7762e-13      |
| 6     | step 50    | 50%       | C:40 L:40   | 7.2875e-4       | 6.7763e-21      |
| 7     | sinusoidal | 50%       | C:40 L:40   | 2.1920e-5       | 1.0842e-19      |
| 8     | sinusoidal | 25%       | C:100 L:100 | 5.7047e-4       | 4.3368e-19      |

Table A.1: Error bound of GBT and EBT in relation to one model of type 1 and its dimensions.

| Model | Input      | Reduction | Dimintions  | Deviation General | Deviation Extended |
|-------|------------|-----------|-------------|-------------------|--------------------|
| 2.1   | step 50    | 25%       | C:20 L:20   | 1.3611e-05        | 1.3553e-20         |
| 2.2   | step 50    | 25%       | C:40 L:40   | 6.4766e-04        | 1.7381e-18         |
| 2.3   | step 50    | 25%       | C:60 L:60   | 0.0016            | 6.7769e-21         |
| 2.4   | sinusoidal | 25%       | C:40 L:40   | 7.5052e-06        | 6.7761e-21         |
| 2.5   | step 50    | 50%       | C:20 L:20   | 2.1852e-05        | 5.5896e-13         |
| 2.6   | step 50    | 50%       | C:40 L:40   | 73.4352e-04       | 3.7188e-26         |
| 2.7   | sinusoidal | 50%       | C:40 L:40   | 0.0131            | 3.8790e-18         |
| 2.8   | sinusoidal | 25%       | C:100 L:100 | 3.0941e-04        | 3.4286e-19         |

Table A.2: Error bound of GBT and EBT in relation to one model of type 2 and its dimensions.

| Model | Input      | Reduction | Dimintions  | Deviation General | Deviation Extended |
|-------|------------|-----------|-------------|-------------------|--------------------|
| 3.1   | step 50    | 25%       | C:20 L:20   | 3.9786            | 9.0366e-16         |
| 3.2   | step 50    | 25%       | C:40 L:40   | 0.3760            | 2.7756e-17         |
| 3.3   | step 50    | 25%       | C:60 L:60   | 0.1224            | 2.6646e-15         |
| 3.4   | sinusoidal | 25%       | C:40 L:40   | 0.0015            | 2.2276e-16         |
| 3.5   | step 50    | 50%       | C:20 L:20   | 4.9462            | 6.2172e-15         |
| 3.6   | step 50    | 50%       | C:40 L:40   | 0.0622            | 8.8818e-16         |
| 3.7   | sinusoidal | 50%       | C:40 L:40   | 27.9051           | 6.6169e-24         |
| 3.8   | sinusoidal | 25%       | C:100 L:100 | 0.0217            | 8.8818e-16         |

Table A.3: Error bound of GBT and EBT in relation to one model of type 3 and its dimensions.

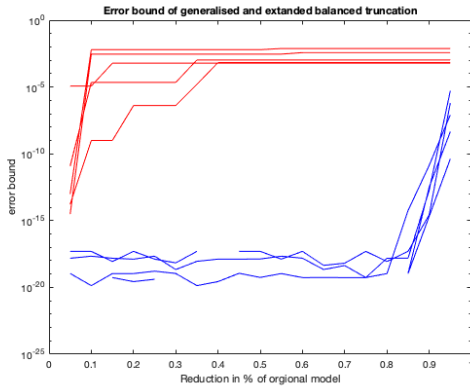


Figure A.1: Error bound model type 1

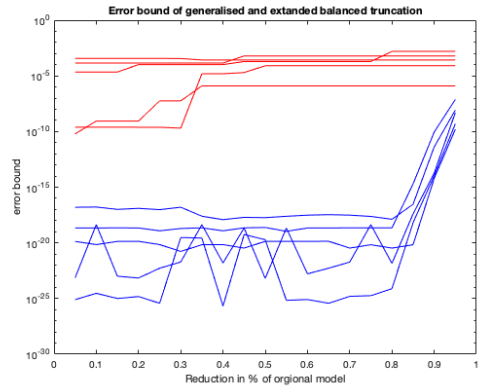


Figure A.2: Error bound model type 2

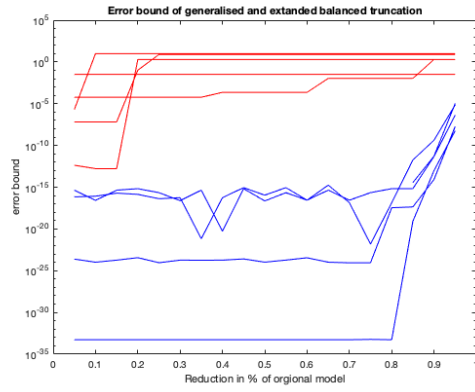


Figure A.3: Error bound model type 3

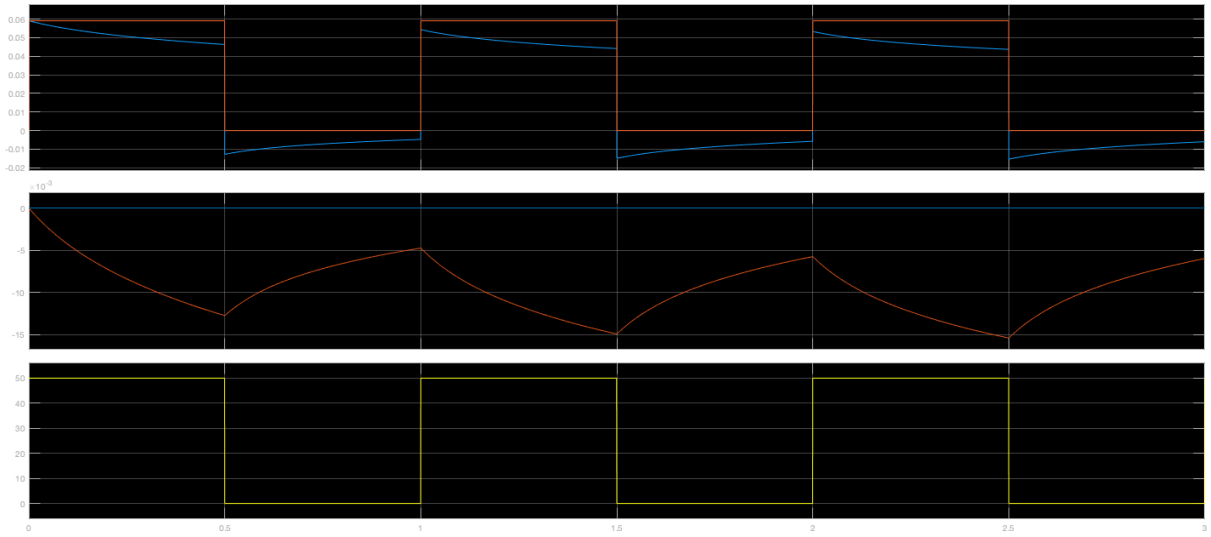


Figure A.4: Model simulation of model 1.1

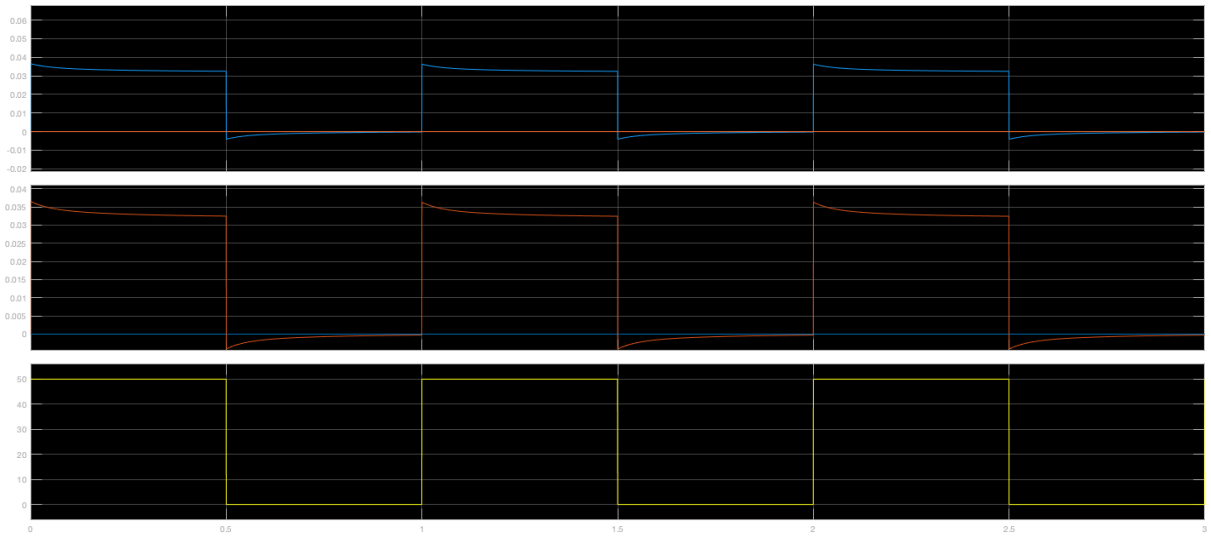
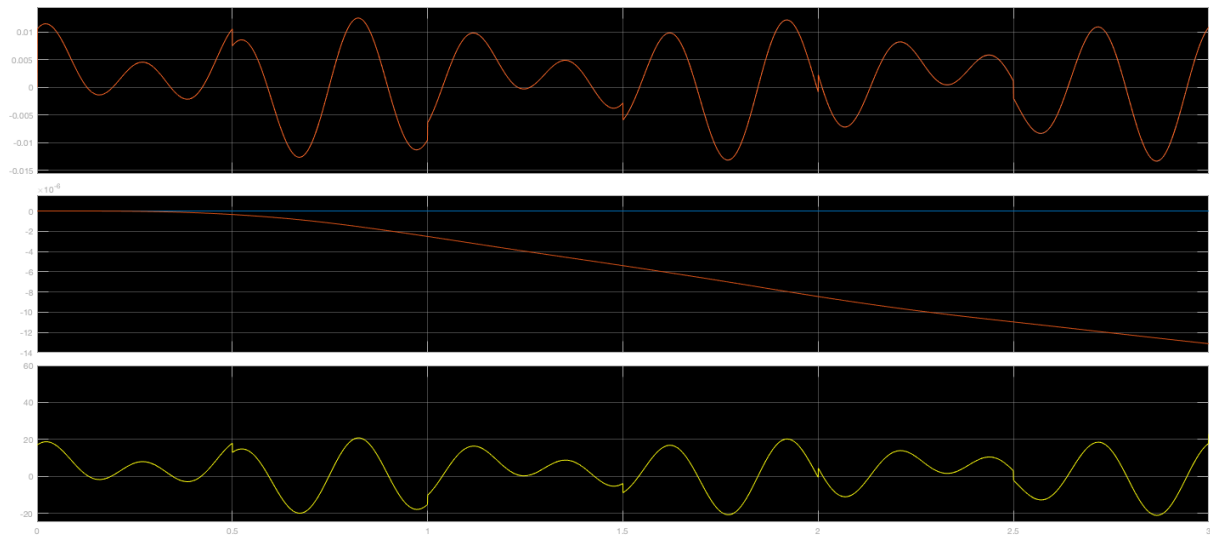
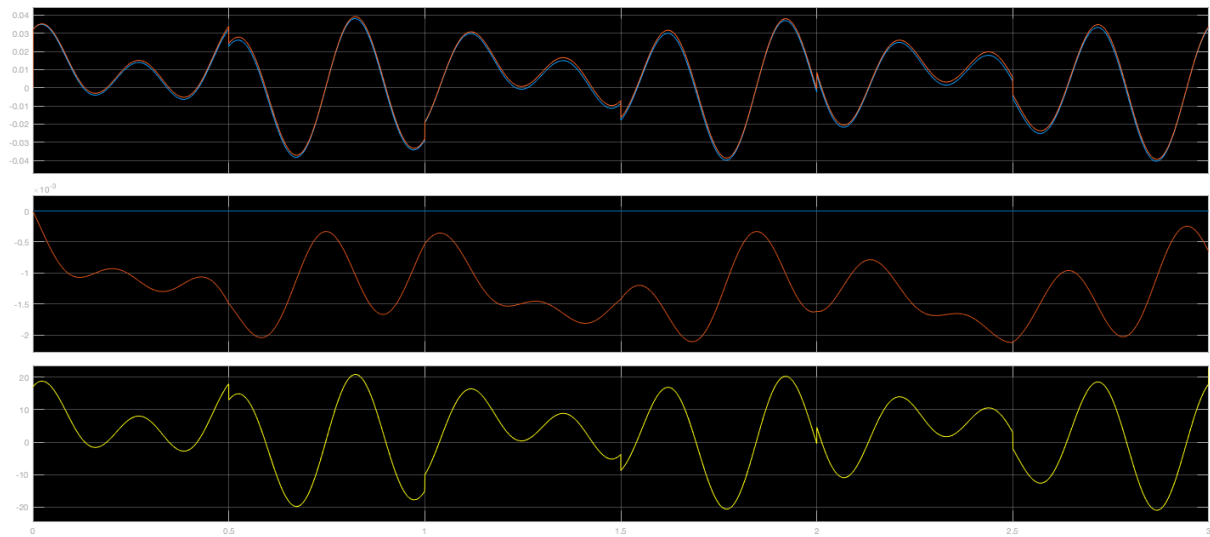


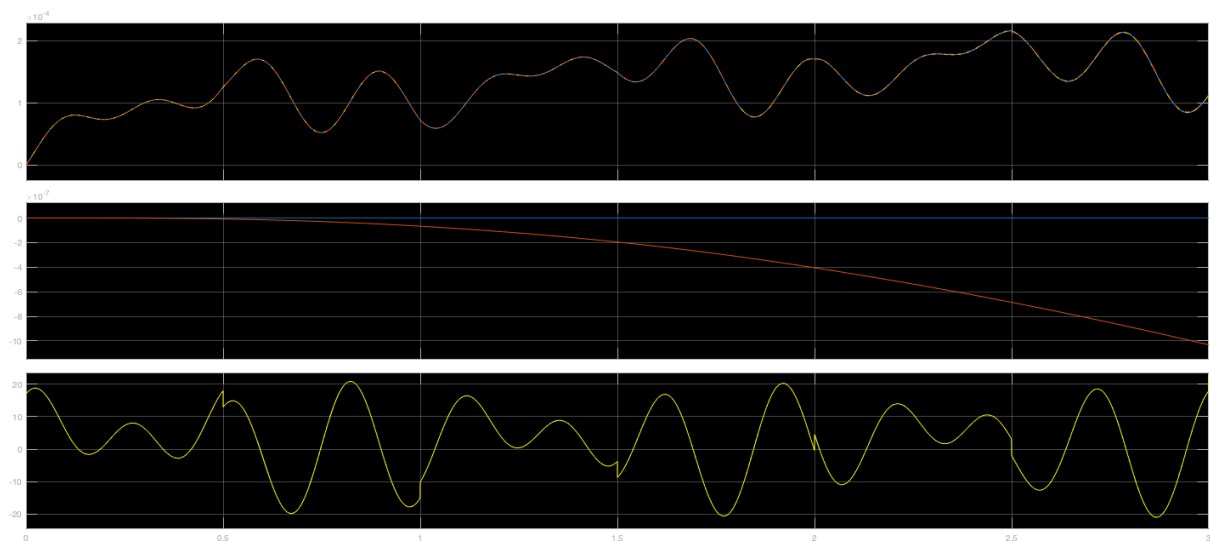
Figure A.5: Model simulation of model 1.2



**Figure A.6: Model simulation of model 1.3**



**Figure A.7: Model simulation of model 1.4**



**Figure A.8: Model simulation of model 1.5**

## B Appendix A. Mathematical representation of electrical circuits

$$L_1 \dot{I}_{l1} = U - V_{c1} - I_{l1} R_{l1}$$

$$L_2 \dot{I}_{l2} = V_{c1} - V_{c2} - I_{l2} R_{l2}$$

$$L_n \dot{I}_{ln} = V_{c1} + V_{c2} + \dots - V_{cn} - I_{ln} R_{ln}$$

$$C_1 \dot{V}_{c1} = \frac{V_{c1}}{R_{c1}} - I_{l1} + I_{l2} + \dots + I_{ln} \quad (\text{B.1})$$

$$C_1 \dot{V}_{c2} = \frac{V_{c2}}{R_{c2}} + I_{l1} - I_{l2} + \dots + I_{ln}$$

$$C_1 \dot{V}_{cn} = \frac{V_{cn}}{R_{cn}} + I_{l1} + I_{l2} + \dots - I_{ln}$$

---


$$\begin{pmatrix} L_1 \dot{I}_{l1} \\ L_2 \dot{I}_{l2} \\ \vdots \\ L_n \dot{I}_{ln} \\ C_1 \dot{V}_{c1} \\ C_2 \dot{V}_{c2} \\ \vdots \\ C_n \dot{V}_{cn} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 0 & \dots & 0 & -1 & 0 & \dots & 0 \\ 0 & -R_{l2} & \dots & 0 & 1 & -1 & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & -R_{ln} & 0 & 0 & \dots & -1 \\ 1 & -1 & \dots & 0 & -\frac{1}{R_{c1}} & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & -\frac{1}{R_{c2}} & \dots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & 0 & 0 & \dots & -\frac{1}{R_{cn}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ I_{l2} \\ \vdots \\ I_{ln} \\ V_{c1} \\ V_{c2} \\ \vdots \\ V_{cn} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} U \quad (\text{B.2})$$


---

$$\begin{pmatrix} L & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} \dot{I}_l \\ \dot{V}_c \end{pmatrix} = \begin{pmatrix} R_l & J_1 \\ -J_1^T & R_c \end{pmatrix} \begin{pmatrix} I_l \\ V_c \end{pmatrix} + \begin{pmatrix} B \end{pmatrix} U \quad (\text{B.3})$$

$$\begin{aligned} 0 &= 0_{n,n} \\ L &= \text{diag}(L_1, L_2, \dots, L_n) \\ C &= \text{diag}(C_1, C_2, \dots, C_n) \\ \dot{I}_l &= (\dot{I}_{l1}, \dot{I}_{l2}, \dots, \dot{I}_{ln})^T \\ \dot{V}_c &= (\dot{V}_{c1}, \dot{V}_{c2}, \dots, \dot{V}_{cn})^T \\ J_1 &= \begin{pmatrix} -1 & 0 & 0 & \dots & 0 \\ 1 & -1 & 0 & \dots & 0 \\ 0 & 1 & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & -1 \end{pmatrix} \\ I_l &= (I_{l1}, I_{l2}, \dots, I_{ln})^T \\ V_c &= (V_{c1}, V_{c2}, \dots, V_{cn})^T \\ R_l &= \text{diag}(-R_{l1}, -R_{l2}, \dots, -R_{ln}) \\ R_c &= \text{diag}\left(-\frac{1}{R_{c1}}, -\frac{1}{R_{c2}}, \dots, -\frac{1}{R_{cn}}\right) \\ B &= (1, 0, \dots, 0, 0, 0, \dots, 0)^T \end{aligned} \quad (\text{B.4})$$


---

$$J_1 = \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \quad (B.5)$$

$$Rl = \text{diag}(-R_{l1}, -R_{l2}, \dots, -R_{ln})$$

$$Rc = 0_{n,n}$$

$$B = (1, 0, \dots, 0, , 0, 0, \dots, 0)^T$$


---

$$J_1 = \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \quad (B.6)$$

$$Rl = \text{diag}(-R_{l1}, 0, 0, \dots, 0)$$

$$Rc = \text{diag}\left(-\frac{1}{R_{c1}}, -\frac{1}{R_{c2}}, \dots, -\frac{1}{R_{cn}}\right)$$

$$B = (1, 0, \dots, 0, , 0, 0, \dots, 0)^T$$


---

$$J_1 = \begin{pmatrix} 1 & 0 & 0 & .. & 0 \\ 0 & 1 & 0 & .. & 0 \\ 0 & 0 & 1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & 1 \end{pmatrix}$$

$$Rl = \text{diag}(-R_{l1}, -R_{l2}, \dots, -R_{ln})$$

$$Rc = \begin{pmatrix} -\frac{1}{R_{c1}} - \frac{1}{R_{c2}} & \frac{1}{R_{c2}} & 0 & .. & 0 \\ \frac{1}{R_{c2}} & -\frac{1}{R_{c2}} & \frac{1}{R_{c3}} & .. & 0 \\ 0 & \frac{1}{R_{c3}} & -\frac{1}{R_{c3}} & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -\frac{1}{R_{cn}} \end{pmatrix} \quad (B.7)$$

$$B = (0, 0, \dots, 0, , \frac{1}{R_{c1}}, 0, \dots, 0)^T$$


---

$$0 = 0_{n,n}$$

$$L = \text{diag}(L_1, L_2, \dots, L_n)$$

$$C = \text{diag}(C_1, C_2, \dots, C_n)$$

$$\dot{I}_l = (\dot{I}_{l1}, \dot{I}_{l2}, \dots, \dot{I}_{ln})^T$$

$$\dot{V}_c = (\dot{V}_{c1}, \dot{V}_{c2}, \dots, \dot{V}_{cn})^T$$

$$J_1 = \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \quad (B.8)$$

$$I_l = (I_{l1}, I_{l2}, \dots, I_{ln})^T$$

$$V_c = (V_{c1}, V_{c2}, \dots, V_{cn})^T$$

$$Rl = \text{diag}(-R_{l1}, -R_{l2}, \dots, -R_{ln})$$

$$Rc = \text{diag}\left(-\frac{1}{R_{c1}}, -\frac{1}{R_{c2}}, \dots, -\frac{1}{R_{cn}}\right)$$

$$B = (1, 0, \dots, 0, , 0, 0, \dots, 0)^T$$



## C Appendix EBT constraints

$$\begin{aligned}
Q &= \delta_o * H; \\
A_o &= \alpha I_n + A \\
X_o &= -Q * A - A' * Q - C' * C; \\
\alpha &> 0 \\
X_o &\geq 0;
\end{aligned} \tag{C.1}$$


---

$$\begin{aligned}
\check{P} &= \delta_c * H^{-1}; \\
A_c &= \beta I_n + A \\
X_c &= -Q * A - A' * Q - C' * C; \\
\beta &\geq 0 \\
X_c &\geq 0;
\end{aligned} \tag{C.2}$$


---

$$2(\beta \check{P} + \Gamma_c) - B B^T - (-\Gamma_c \check{P} + A + B B^T \check{P}) X_c^{-1} (-\check{P} \Gamma_c + A^T + \check{P} B B^T) \geq 0 \tag{C.3}$$


---

$$2(\alpha Q + \Gamma_o) - (\Gamma_o - Q A) X_o^{-1} (\Gamma_o - A^T Q) \geq 0 \tag{C.4}$$

## D Matlab code

This appendix contain all the Matlab scripts. These scripts and the Simulink model can all be found and downloaded form: [https://github.com/PHW-H/IDP\\_Simulink\\_2021](https://github.com/PHW-H/IDP_Simulink_2021)

### D.1 Model reduction script

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ——— Model ——— %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5 clear all
6 clc
7
8 %input model MANUAL
9 % Rc=[]; %values for risistance in Rc (in Ohm)
10 % Rl=[]; %values for risistance in Rl (in Ohm)
11 % Cc=[]; %values for capacitance in C (in Farad)
12 % Ll=[]; %values for inductance in L (in Henry)
13 % ModT=[]; %Model type
14 % M=[]; %reduction (in percentages)
15 % n=size(L);
16 % n=n(1,1);
17
18 %generat random model
19 % n=20; %dimentions of the system (number of conductors and capasitors)
20 % setMT=1; %determine model type or if=0 generates random model type
21 % setRe=0.25; %determine reduction in % or if=0 generates random reduction
    in %
22 % saveM=0; % if=0 saves model in 'Model_auto_save'
23 % [Rl,Rc,Cc,Ll,ModT,M] = Random_model-generator(n,setMT,setRe,saveM);
24
25 %load existing model
26 load ('Model_auto_save');
27
28 %%
29 if ModT==1
30     [H,R,J,B] = Modeltype41(Rl,Rc,Ll,Cc);
31 elseif ModT==2
32     [H,R,J,B] = Modeltype42(Rl,Rc,Ll,Cc);
33 elseif ModT==3
34     [H,R,J,B] = Modeltype43(Rl,Rc,Ll,Cc);
35 else
36     'error model type does not exist'
37     return
38 end
39 F =J+R;
40 A =F*H;
41 Hi =inv(H);
42
43 beta=1;
44 M =2*M*n;
45
46 C = B'*H;
47
48 if ModT==2
49     do=0.10;
50     Q = do*H;
```

```

51     Xo = -Q*A-A'*Q-C'*C;
52     EIGXo=eig(Xo);
53     i=max(EIGXo);
54     while i<0
55         do=do*1.5;
56         Q = do*H;
57         C= B'*H;
58         Xo = -Q*A-A'*Q-C'*C;
59         EIGXo=eig(Xo);
60         i=max(EIGXo);
61     end
62 else
63     syms 'do';
64     Q = do*H;
65     Xo = -Q*A-A'*Q-C'*C;
66
67     EIGXo = eig(Xo);
68     i = 1;
69     while i<=2*n % find definition delta_o
70         EIGXo(i)=solve(EIGXo(i)==0,do);
71         i=i+1;
72     end
73     do = double(max(EIGXo))+0.0001;
74 end
75 Q = do*H;
76 Qi=inv(Q);
77 Xo = -Q*A-A'*Q-C'*C;
78
79 dc = do;
80
81 Pi = dc*Hi;
82 Xc = -A*Pi-Pi*A'-(B*B');
83
84 if Xc>=0 % find definition delta_c if needed
85     syms 'dc'
86     Pi=dc*Hi;
87     Xc=-A*Pi-Pi*A'-(B*B');
88     EIGXc=eig(Xc);
89     i=1;
90     while i<=2*n
91         EIGXc(i)=solve(EIGXc(i)==0,dc);
92         i=i+1;
93     end
94     dc=double(max(EIGXc))+0.0001;
95     Pi=dc*Hi;
96 end
97
98 epsc = 1;
99 epso = 1;
100
101 i=1;
102 while i<=n %define zeta
103     zeta_c(i,i)=Pi(i,i)*(1.1^i);
104     zeta_o(i,i)=Q(i,i)*(1.1^i);
105     zeta_c(n+i,n+i)=Pi(n+i,n+i)*(1.1^i);
106     zeta_o(n+i,n+i)=Q(n+i,n+i)*(1.1^i);
107     i=i+1;
108 end

```

```

109
110 GAMc=-epsc*zeta_c;
111 GAMo=zeta_o;
112
113 Thc=(-GAMc+A*Pi+B*B')*inv(Xc)*(-GAMc+Pi*A'+B*B');
114 condc=2*(beta*Pi+GAMc)-Thc;
115 con1=min(eig(condc)); % checking (all the eigenvalues must be positive)
116 i=1;
117 while con1<=0 %find smalles possible beta
118     beta=beta*10;
119     condc=2*(beta*Pi+GAMc)-Thc;
120     con1=min(eig(condc));
121     i=i+1;
122     if i>11; %set maximum value for beta (if beta to large -> rounding
123         errors)
124         'error_beta'
125         return
126     end
127 end
128 epsc1=epsc;
129
130 i=1;
131 while con1>=0 %obtaining max value for epsilon_c
132     epsc=epsc1;
133     epsc1=(5*i);
134     GAMc=-epsc1*zeta_c;
135     Thc=(-GAMc+A*Pi+B*B')*inv(Xc)*(-GAMc+Pi*A'+B*B');
136     condc=2*(beta*Pi+GAMc)-Thc;
137     con1=min(eig(condc));
138     i=i+1;
139     if epsc1>=beta %epsc has to be smaller then beta
140         'error_epsc'
141         return
142     end
143 end
144
145 GAMc=-epsc*zeta_c;
146
147 alpha=beta;
148
149 Tho=(GAMo-Q*A)*inv(Xo)*(GAMo-A'*Q);
150
151 condo=2*(alpha*Q+GAMo)-Tho;
152 con2=min(eig(condo)); %checking (all the eigenvalues must be positive)
153
154 epsol=epso;
155
156 i=1;
157 while con2 >= 0
158     epso=epsol;
159     epsol=(5*i);
160     GAMo=epsol*zeta_o;
161     Tho=(GAMo-Q*A)*inv(Xo)*(GAMo-A'*Q);
162     condo=2*(alpha*Q+GAMo)-Tho;
163     con2=min(eig(condo));
164     i=i+1;
165     if epso>=beta %epsc has to be smaller then beta

```

```

166         'error_epso'
167     return
168 end
169 end
170 GAMo=epso*zeta_o;
171
172 %%
173 %Define Ti
174
175 Ti=beta*Pi+GAMc;
176
177 min(eig(Ti)); % checking (all the eigenvalues must be positive)
178
179 Thc=(-GAMc+A*Pi+B*B')*inv(Xc)*(-GAMc+Pi*A'+B*B');
180 condc=2*(Ti)-Thc;
181 con1=min(eig(condc)); % checking (all the eigenvalues must be positive)
182 if con1<=0
183     con1
184     'error1'
185     return
186 end
187
188 %%
189
190 % Define S
191
192 S=inv(alpha*Qi+Qi*GAMo*Qi);
193
194 Tho=(GAMo-Q*A)*inv(Xo)*(GAMo-A'*Q);
195
196 condo=2*(alpha*Q+GAMo)-Tho;
197 con2=min(eig(condo)); %checking (all the eigenvalues must be positive)
198 if con2<=0
199     con2
200     'error2'
201     return
202 end
203 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
204 %% Transformation Extended
205
206 %splitting Ti and S in C and L related parts
207 TiL=Ti(1:n,1:n);
208 TiC=Ti(n+1:2*n,n+1:2*n);
209 SL=S(1:n,1:n);
210 SC=S(n+1:2*n,n+1:2*n);
211
212 PhTiC=chol(TiC);
213 [UTSC,S2TSC]=svd(PhTiC*SC*PhTiC');
214 STSC=sqrt(S2TSC);
215 WEC=PhTiC'*UTSC*sqrt(inv(STSC));
216 WECi=inv(WEC);
217
218 PhTiL=chol(TiL);
219 [UTSL,S2TSL]=svd(PhTiL*SL*PhTiL');
220 STSL=sqrt(S2TSL);
221 WEL=PhTiL'*UTSL*sqrt(inv(STSL));
222 WELi=inv(WEL);
223

```

```

224 WE=[WEL zeros(n,n); zeros(n,n) WEC];
225 WEi=inv(WE);
226
227 %% Transformation Generalized
228
229 PiL=Pi(1:n,1:n);
230 PiC=Pi(n+1:2*n,n+1:2*n);
231 QL=Q(1:n,1:n);
232 QC=Q(n+1:2*n,n+1:2*n);
233
234 PhPiC=chol(PiC);
235 [UQPC,S2QPC]=svd(PhPiC*QC*PhPiC');
236 SQPC=sqrt(S2QPC);
237 WGC=PhPiC'*UQPC*sqrt(inv(SQPC));
238 WGCi=inv(WGC);
239
240 PhPiL=chol(PiL);
241 [UQPL,S2QPL]=svd(PhPiL*QL*PhPiL');
242 SQPL=sqrt(S2QPL);
243 WGL=PhPiL'*UQPL*sqrt(inv(SQPL));
244 WGLi=inv(WGL);
245
246 WG=[WGL zeros(n,n); zeros(n,n) WGC];
247 WGi=inv(WG);
248
249 %%
250
251 e=@(k,n) [zeros(k-1,1);1;zeros(n-k,1)];
252 % M is the number of state you want to truncate
253 M=M;
254 K=(n*2)-M;
255 aux1=[e(1,2*n)];
256 aux2=[e(n+1,2*n)];
257 i=2;
258 while i <= 0.5*K
259     aux3=[e(i,2*n)];
260     aux4=[e(n+i,2*n)];
261     aux1=[aux1,aux3];
262     aux2=[aux2,aux4];
263     i=i+1;
264 end
265 aux=[aux1,aux2];
266
267 % Reduced via extended
268
269 Ah=WE\A*WE;
270 Hh=WE'*H*WE;
271 Bh=WE\B;
272 C=B'*H;
273 Ch=C*WE;
274 Ar=aux'*Ah*aux;
275 Br=aux'*Bh;
276 Cr=Ch*aux;
277 Hr=aux'*Hh*aux;
278
279 % Reduced via generalized
280
281 Ahg=WG\A*WG;

```

```

282 Hhg=WG'*H*WG;
283 Bhg=WG\B;
284 Chg=C*WG;
285 Arg=aux'*Ahg*aux;
286 Brg=aux'*Bhg;
287 Crg=Chg*aux;
288 Hrg=aux'*Hhg*aux;
289
290 %%
291
292 lCn = eig(STSC)/max(eig(STSC));
293 lLn = eig(STSL)/max(eig(STSL));
294
295 lCni = flip(lCn);
296 lLni = flip(lLn);
297
298 %plot eigenvalues
299 % figure
300 % plot(lCni,'bO','LineWidth',2)
301 % grid on
302 % title('Eigenvalues of $\Lambda_{ST-1}$','Interpreter','latex')
303 % xticks([0:n])
304 % figure
305 % plot(lLni,'rO','LineWidth',2)
306 % grid on
307 % title('Eigenvalues of $\Lambda_{ST-2}$','Interpreter','latex')
308 % xticks([0:n])
309
310 %%
311
312 % Error system
313
314 Ae = [Ah zeros(2*n,2*n-M); zeros(2*n-M,2*n) Ar];
315 Be = [Bh; Br];
316 Ce = [Ch -Cr];
317
318 Aeg = [Ahg zeros(2*n,2*n-M); zeros(2*n-M,2*n) Arg];
319 Beg = [Bhg; Brg];
320 Ceg = [Chg -Crg];
321
322 %%
323
324 % H inf normst
325
326 % extended
327
328 fsys = ss(A,B,C,0);
329 bsys = ss(Ah,Bh,Ch,0);
330 rsys = ss(Ar,Br,Cr,0);
331 esys = ss(Ae,Be,Ce,0);
332
333 [ninff,fpeakf] = hinfnorm(fsys);
334 [ninfb,fpeakb] = hinfnorm(bsys);
335 [ninfrr,fpeakr] = hinfnorm(rsys);
336 [ninfe,fpeake] = hinfnorm(esys);
337
338 bgsys = ss(Ahg,Bhg,Chg,0);
339 rgsys = ss(Arg,Brg,Crg,0);

```

```

340  egsys = ss(Aeg,Beg,Ceg,0);
341
342  [ninfbg,fpeakbg] = hinfnorm(bgsys);
343  [ninfrg,fpeakrg] = hinfnorm(rgsys);
344  [ninfeg,fpeakeg] = hinfnorm(egsys);
345
346  [ninfe;ninfeg]

```



## D.2 Model Generator

```
1 %Random model generator
2 function [Rl,Rc,Cc,Ll,ModT,M] = Random_model_generator(n,setMT,setRe,saveM)
3
4 Rl=randi([0 2000],n,1);
5 Rc=randi([0 2000],n,1);
6 Cc=randi([1 5000],n,1);
7 Cc=Cc*10-6;
8 Ll=randi([50 15000],n,1);
9 Ll=Ll*10-6;
10 if setMT==0
11     ModT=randi([1 3]);
12 else
13     ModT=setMT;
14 end
15 if setRe==0
16     M=randi([0.1 0.5]);
17 else
18     M=setRe;
19 end
20 if saveM==0
21     save('Model_auto_save','Rl','Rc','Ll','Cc','ModT','M','n')
22 end
23 end
```

### D.3 Matlab Code model type 1

```

1 function [H,R,J,B] = Modeltype41(Rl,Rc,Ll,Cc)
2 R=[Rl, Rc];
3
4 n=size(Ll);
5 n=n(1,1);
6
7 B=zeros([2*n 1]);
8 B(1,1)=1;
9
10 c=size(Cc);
11 c=c(1,1);
12 rl=size(Rl);
13 rl=rl(1,1);
14 rc=size(Rc);
15 rc=rc(1,1);
16
17 if n~=c && n~=rl && n~=rc
18     'dimentions do not match'
19     return
20 end
21 %% creating matrix F
22
23 i=1;
24 while i<=n
25     A11(i,i)=R(i,1);
26     if R(i,2)==0
27         A22(i,i)=0;
28     else
29         A22(i,i)=-1/R(i,2);
30     end
31     H(i,i)=1/Ll(i);
32     H(i+n,i+n)=1/Cc(i);
33     i=i+1;
34 end
35
36 a=ones(1,n);
37 b=ones(1,n-1);
38 A121=diag(-a);
39 A122=diag(b,-1);
40 A12=A121+A122;
41 O=zeros(n,n);
42
43 A122=diag(b,-1);
44 A21=-1*A12.';
45 R=[A11,O;O,A22];
46 J=[O,A12;A21,O];
47 end

```

## D.4 Matlab Code model type 2

```

1  function [H,R,J,B] = Modeltype42(Rl,Rc,Ll,Cc)
2  R=[Rl, Rc];
3
4  n=size(Cc);
5  n=n(1,1);
6
7  B=zeros(2*n,1);
8  B(n+1,1)=1/R(1,2);
9
10 l=size(Ll);
11 l=l(1,1);
12 rl=size(Rl);
13 rl=rl(1,1);
14 rc=size(Rc);
15 rc=rc(1,1);
16
17 p=min(Rc);
18 if n~=c && n~=rl && n~=rc
19     'dimentions do not match'
20     return
21 end
22 %% creating matrix F
23 A11(1,1)=R(1,1);
24 A22(n,n)=-1/R(n,2);
25 A22(n-1,n)=1/R(n,2);
26 A22(n,n-1)=1/R(n,2);
27 H(1,1)=1/Ll(1);
28 H(1+n,1+n)=1/Cc(1);
29 i=2;
30 j=n-1;
31 k=j;
32 while i<=n
33     A11(i,i)=R(i,1);
34     A22(j,j)=-1/R(j+1,2)-1/R(j,2);
35     while k>1
36         A22(j-1,j)=1/R(j,2);
37         A22(j,j-1)=1/R(j,2);
38         k=k-1;
39     end
40     H(i,i)=1/Ll(i);
41     H(i+n,i+n)=1/Cc(i);
42     j=n-i;
43     k=j;
44     i=i+1;
45 end
46
47 a=ones(1,n);
48 A12=diag(a);
49 A21=-1*A12.';
50
51 O=zeros(n,n);
52 R=[A11,O;O,A22];
53 J=[O,A12;A21,O];
54 end

```

## D.5 Matlab Code model type 3

```

1  function [H,R,J,B] = Modeltype43(Rl,Rc,Ll,Cc)
2  R=[Rl, Rc];
3
4  n=size(Ll);
5  n=n(1,1);
6
7  B=zeros(2*n,1);
8  B(n+1)=1;
9
10 c=size(Cc);
11 c=c(1,1);
12 rl=size(Rl);
13 rl=rl(1,1);
14 rc=size(Rc);
15 rc=rc(1,1);
16
17 if n~=c && n~=rl && n~=rc
18     'dimentions do not match'
19     return
20 end
21 %% creating matrix F
22 i=1;
23 while i<=n
24     A11(i,i)=R(i,1);
25     if R(i,2)==0
26         A22(i,i)=0;
27     else
28         A22(i,i)=-1/R(i,2);
29     end
30     H(i,i)=1/Ll(i);
31     H(i+n,i+n)=1/Cc(i);
32     i=i+1;
33 end
34
35 a=ones(1,n);
36 b=ones(1,n-1);
37 A121=diag(-a);
38 A122=diag(b,-1);
39 A12=A121+A122;
40 O=zeros(n,n);
41
42 A122=diag(b,-1);
43 A21=-1*A12.';
44 R=[A11,O;O,A22];
45 J=[O,A12;A21,O];
46 end

```