university of groningen / faculty of science and engineering

Reduction of large-scale electrical models

Bachelor's Project Thesis

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

Abstract: The model order reduction method developed by Borja, Scherpen en Fujimoto [4] called extended balanced truncation (EBT) allows one to reduce a model while preserving its physical interpretation. The method in question can potentially be applied in the context of large-scale linear electrical networks (LSENs). 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 LSENs and consequently applying EBT. To validate if the method was successful, the reduced-order models are reconstructed in Simulink and compared to another reduction method: generalized balanced truncation (GBT). The result shows that EBT has the ability to preserve this physical interpretation. Moreover, EBT can reduce LSENs with an error significantly lower to GBT, and is able to reduce larger portions of the original model.

1 Introduction

In the technological world of today, most processes or systems are described by mathematical models, and simulations are used to predict the behavior 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" [8], and offer a solution to the above-mentioned issue. Various techniques have been developed during the last decades [3]. A relevant context with the need for model reduction is large-scale electrical networks (LSENs). Electric grids exhibit several typical features of complex networks[11] and are one example of these Large-scale electrical networks (LSENs) with often large dimensions where simulation of the whole networks takes significant time. As a result, model order reduction is a relevant topic in this context. One example for reducing a model is by balanced truncation. This method like many other reduction methods often makes use of a state-space representation for creating a reduced-order model. The basic method 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 to reduce the error bound of these reduced-order models [7]. Unfortunately, these reduction methods often result in a model without a physical interpretation. Which makes it difficult to interpret the model. Borja, Scherpen en Fujimoto used a combination of GBT and port-Hamiltonians (PH) systems to develop a new reduction method called extended balanced truncation (EBT) [4]. 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. the PH system modeling generally encodes more structural information about the physical system than just passivity [10]. Moreover, the port-Hamiltonian systems modeling can be regarded to bridge the gap between passive system models and explicit physical network realizations[10]. This project aims to investigate the method for creating a reduced-order model as described by Borja, Scherpen, and Fujimoto [4] and its possibilities for the application to LSENs. The project aims to validate the theoretical framework for applying the method in question in the relevant context of LSENs through simulations using Simulink. As a result, the simulations should prove the method guarantee an acceptable error ratio while preserves the physical interpretation of the reduced-order model. An acceptable arrow ratio is dependent on the size of the truncated part and between zero and 5 percent.

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

3 Generating the Models

Before it is possible to apply theoretical framework in question in the relevant context, models of Large-scale electrical networks need to be obtained. This is done by generating these models using a Matlab script. This Matlab script is written using a combination of Kirchhoff's circuit laws for linear(Equation 3.1 and 3.2) and parallel circuits (Equation 3.3 and 3.4) and Ohm's law for current over an capacitors and voltage over an inductors (Equation 3.5 and 3.6)

$$\sum_{i=1}^{n} V_i = 0 \tag{3.1}$$

$$I_1 = I_2 = ... = I_n \tag{3.2}$$

$$V_1 = V_2 = ... = V_n \tag{3.3}$$

$$\sum_{i=1}^{n} I_i = 0 \tag{3.4}$$

$$V_l = \dot{I}_l \, L \tag{3.5}$$

$$I_c = \dot{V}_c \, C \tag{3.6}$$

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

EBT uses a state-space representation in a Port-Hamiltonian form. where:

$$\sum\nolimits_H : \begin{cases} \dot{x} & = \; (J-R)Hx + Bu \\ y & = \; B^T Hx \\ H(x) & = \; \frac{1}{2} x^T Hx \end{cases} \tag{3.7}$$

in a Port-Hamiltonian representation, the Hamiltonian, H(x) is the total energy of the system [9], with $H = H > 0$; and $R = R^T 0, J = JT$ [4]. 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 and C. The matrix R consists of information regarding the resisting elements.

in order to easily obtain the matrices J,R,H and B the model are generated by expressing $\dot{V}_{ci} \, C$ and $\dot{I}_{li} \, C$ 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:

$$\begin{pmatrix} L & 0 \\ 0 & C \end{pmatrix} \begin{pmatrix} \dot{I}_l \\ \dot{V}_c \end{pmatrix} = \begin{pmatrix} R_l & J_1 \\ -J_1^T & R_c \end{pmatrix} \begin{pmatrix} I_l \\ V_c \end{pmatrix} + \begin{pmatrix} B \end{pmatrix} U \tag{3.8}$$

where:

$$\begin{pmatrix} L & 0 \\ 0 & C \end{pmatrix} = H \tag{3.9}$$

$$\begin{pmatrix} I_l \\ V_c \end{pmatrix} = x \tag{3.10}$$

$$\begin{pmatrix} R_l & J_1 \\ -J_1^T & R_c \end{pmatrix} = J - R \tag{3.11}$$

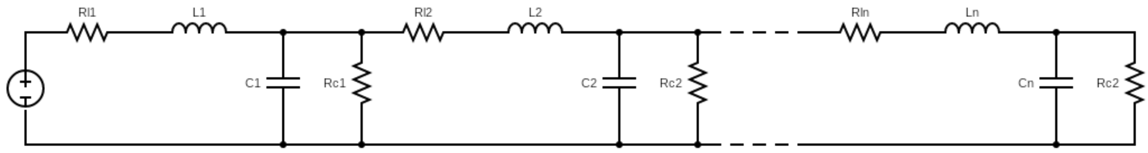

Figure 3.1: Standard circuit type 1

3.1 Model electrical circuit type 1

The first standard electrical network is represented in figure 3.1. 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 3.1 , 3.2 , 3.3 and 3.4) and Ohm's law (Equation 3.5 and 3.6) we obtain the following two equations to represent this model:

$$U = V_{c1} + L_1\,\dot{I}_{l1} + I_{l1}\,R_{l1} \tag{3.12}$$

$$I_{l1} = C_1\,\dot{V}_{c1} + \frac{V_{c1}}{R_{c1}} \tag{3.13}$$

Reordering these equations will allow us to easily create a state-space representation of this model (equations 3.14, 3.15 and 3.16).

$$L_1\,\dot{I}_{l1} = U - V_{c1} - I_{l1}\,R_{l1} \tag{3.14}$$

$$C_1\,\dot{V}_{c1} = I_{l1} + \frac{V_{c1}}{R_{c1}} \tag{3.15}$$

$$\begin{pmatrix} L_1\,\dot{I}_{l1} \\ C_1\,\dot{V}_{c1} \end{pmatrix} = \begin{pmatrix} -R_{l1} & -1 \\ 1 & -\frac{1}{R_{c1}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ V_{c1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} U \tag{3.16}$$

In this standard model of an electrical circuit, addition a component containing L, C, R_l, R_c will influence Equations 3.13 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 3.12. 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 A.1 and the corresponding state-space form A.2

3.1.1 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 3.2).

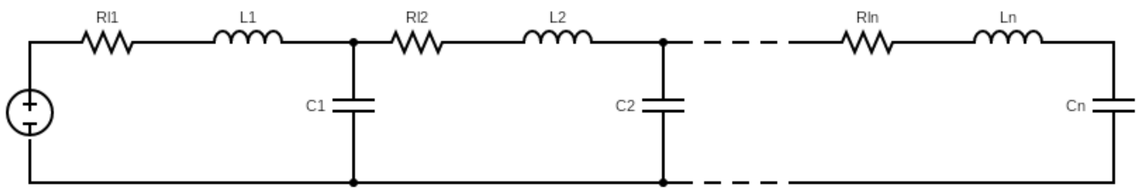

Figure 3.2: Standard circuit type 1.2

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 3.1 , 3.2 , 3.3 and 3.4) and Ohm's laws (Equation 3.5

and 3.6) we are able to a mathematical representation of the circuit, which also can be represented in state-space form.

$$U = V_{c1} + L_1\,\dot{I}_{l1} + I_{l1}\,R_{l1} \tag{3.17}$$

$$I_{l1} = C_1\,\dot{V}_{c1} \tag{3.18}$$

$$\begin{pmatrix} L_1\,\dot{I}_{l1} \\ C_1\,\dot{V}_{c1} \end{pmatrix} = \begin{pmatrix} -R_{l1} & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} I_{l1} \\ V_{c1} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} U \tag{3.19}$$

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 3.17 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 3.18 a term I_{l2} is added to the right side of the equation. Moreover, I_{l2} can in its turn be defined as $I_{l2} + I_{c3}$. All further I_{Li} (for i=3,4,...,n-1)can be defined in a similar way. The definition of I_{ln} is however is the same as I_{cn} being $C_n\,\dot{V}_{cn}$). 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 A.5)

3.1.2 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 is in series with the inductor is taken out (see figure 3.3). 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.

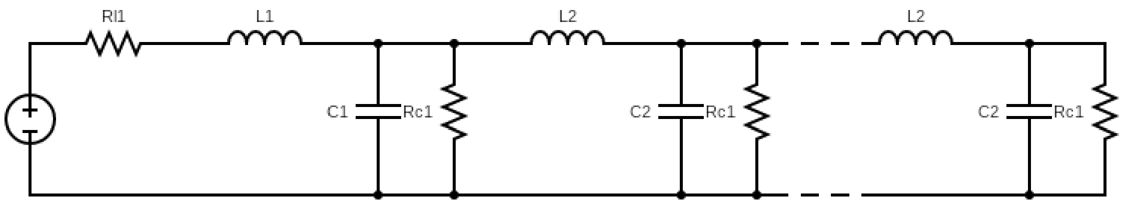

Figure 3.3: Standard circuit type 1.3

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

$$U = L_1\,\dot{I}_{l1} + I_{l1}\,R_{l1} + L_2\,\dot{I}_{l2} + V_{c2} \tag{3.20}$$

$$I_{l1} = C_1\,\dot{V}_{c1} + \frac{V_{c1}}{R_{c1}} + C_2\,\dot{V}_{c2} + \frac{V_{c2}}{R_{c2}} \tag{3.21}$$

$$\begin{pmatrix} L_1\,\dot{I}_{l1} \\ L_2\,\dot{I}_{l2} \\ C_1\,\dot{V}_{c1} \\ C_2\,\dot{V}_{c2} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & -1 & \frac{1}{R_{c1}} & 0 \\ 0 & 1 & 0 & \frac{1}{R_{c2}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ I_{l2} \\ V_{c1} \\ V_{c2} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} U \tag{3.22}$$

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 0 except for R_{l1} giving a state-space of the model in the form of equation A.6

3.1.3 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 3.7. The Matlab model can be found in appendix C.3. this model can present all possible combinations of models type 1,1.2 and 1.3.

3.2 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 (see image 3.4))

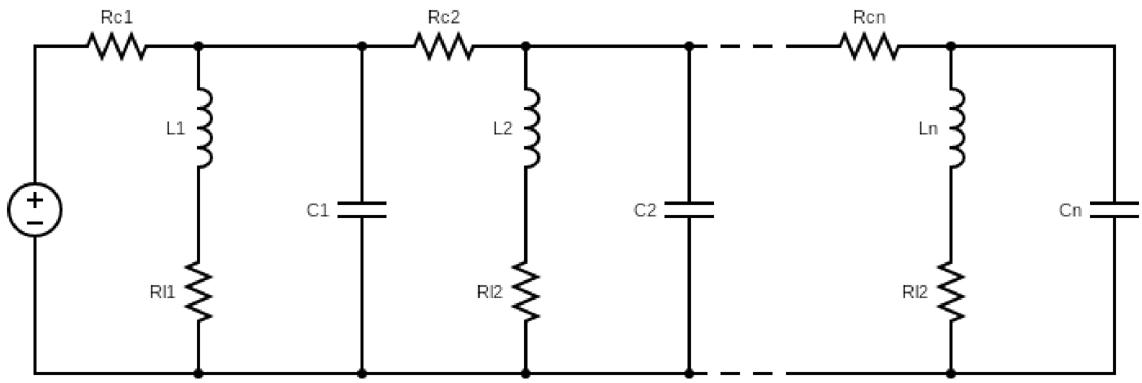

Figure 3.4: Standard circuit type 2

in this case, the smallest option for this circuit is represented in the following three equations (3.23, 3.24 and 3.25) using the laws of Kirchhoff and Ohm.

$$I_{Rc1} = I_{l1} + \dot{V}_{c1}\, C_1 \tag{3.23}$$

$$U = R_{c1}\, I_{Rc1} + V_{c1} \tag{3.24}$$

$$V_{c1} = \dot{I}_{l1}\, L_1 + I_{l1}\, R_{l1} \tag{3.25}$$

Reordering equation equation 3.25 gives an expression for $\dot{I}_{l1}\, L_1$. Using equation 3.23 and substituting this equation in 3.24 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 3.26 and 3.27, which can be put into a state-space form of 3.32

$$\dot{V_{c1}}\, C_1 = I_{l1} - \frac{V_{c1}}{R_{c1}} + \frac{U}{R_{c1}} \tag{3.26}$$

$$\dot{I}_{l1}\, L_1 = V_{c1} - I_{l1}\, R_{l1} \tag{3.27}$$

$$\begin{pmatrix} L_1\, \dot{I}_{l1} \\ C_1\, \dot{V}_{c1} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 1 \\ -1 & \frac{1}{R_{c1}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ V_{c1} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{R_{c1}} \end{pmatrix} U \tag{3.28}$$

Like type 1 a model of type 2 is also easily Up-scalled. Adding a component containing a R_{c2}, R_{l2}, L_2 and C_2 will influence equations 3.23 and as a result the equation will have a additional component I_{rc1} (see equation 3.31). With respect to the equations defining the voltage of source U a new additional definition will exist where:

$$V_{c1} = R_{c2}(I_{l2} + \dot{V}_{c2}\, C_2) + V_{c2} \tag{3.29}$$

$$V_{c2} \;=\; \dot{I}_{l2}\, L_2 + R_{l2}\, I_{l2} \tag{3.30}$$

new expression for I_{Rc1};

$$I_{r1} = \dot{V}_{c1}\,C_1 + I_{l1} + \dot{V}_{c2}\,C_2 + I_{l2} \tag{3.31}$$

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 reordering equations 3.33 , 3.25 and 3.35 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 3.31 and substituting this equation in equation 3.24 will allow $\dot{V}_{c1}\,C_1$ 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 3.32).

$$\begin{pmatrix} L_1\,\dot{I}_{l1} \\ L_2\,\dot{I}_{l2} \\ C_1\,\dot{V}_{c1} \\ C_2\,\dot{V}_{c2} \end{pmatrix} = \begin{pmatrix} -R_{l1} & 0 & 1 & 0 \\ 0 & -R_{l2} & 0 & 1 \\ -1 & 0 & -\frac{1}{R_{c1}} - \frac{1}{R_{c2}} & \frac{1}{R_{c2}} \\ 0 & -1 & \frac{1}{R_{c2}} & -\frac{1}{R_{c2}} \end{pmatrix} \begin{pmatrix} I_{l1} \\ I_{l2} \\ V_{c1} \\ V_{c2} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{1}{R_{c1}} \\ 0 \end{pmatrix} U \tag{3.32}$$

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 $i = 1, 2, ..., n$ can always be defined as

$$\dot{I}_{li}\,L_i = V_{ci} - R_{li}\,I_{li} \tag{3.33}$$

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

$$\dot{V}_{ci}\,C_i = \frac{V_{c(i-1)}}{R_{ci}} - \frac{V_{ci}}{R_{ci}} - I_{li} \tag{3.34}$$

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

$$\dot{V}_{c1}\,C_1 = -I_{l1} - \frac{V_{c1}}{R_{c1}} - \frac{V_{c1}}{R_{c2}} + \frac{V_{c2}}{R_{c2}} + \frac{U}{R_{c1}} \tag{3.35}$$

3.2.1 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. 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 C.4).

3.3 Model electrical circuit type 3

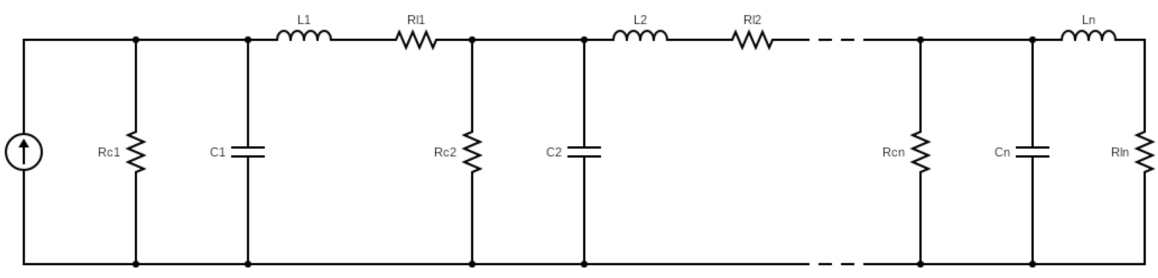

Figure 3.5: Standard circuit type 3

The third and final model of electrical circuits makes use of a current source. the model is represented in 3.5. Taking 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 has an effect on the $\dot{V}_{c1}\,C_1$ instead of $\dot{I}_{l1}\,L_1$ and naturally expressed in current instead of a voltage. the state space of this model is represented in equation A.8 and the Matlab code for generating model type3 in C.5

4 Extended balanced truncation

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

$$\sum : \begin{cases} \dot{x} &= Ax + Bu \\ y &= Cx \end{cases} \tag{4.1}$$

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, $A \in R^{nxn}$, $B \in R^{nxm}$ and $C \in R^{nxm}$. Assume that the system as described in 4.1 is asymptotically stable. Thus the so-called generalized observability Gramians $Q \in R^{nxn}$ are a positive semi-definite solutions to the following Lyapunov inequality;

$$QA + A^T Q + C^T C \leq 0 \tag{4.2}$$

Analogously, the generalized controllability Gramians $\breve{P} \in R^{nxn}$ are given by positive semi-definite solutions to

$$A\breve{P} + \breve{P}A^T + BB^T \leq 0 \tag{4.3}$$

In particular, when 4.2 and 4.3 are equalities, the matrices Q and $\breve{P}$ 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 observably Grampians $Q \in \mathbb{R}^{nxn}$ and generalized controllability Grampians $\breve{P} \in \mathbb{R}^{nxn}$ [2]

4.1 Generalized Balanced Trucation

Generalized balanced truncation (GBT) aims to find a matrix W that that solves:

$$WQ\breve{P}W^{-1} = \Lambda_{QP}^2 \tag{4.4}$$

Here Λ_{QP} is a diagonal matrix containing all singular values of the matrix Q and $\breve{P}$. Before truncating the system, it first needs to be balanced. This is done when equation 4.5 holds

$$\breve{P} = Q = \Lambda_{QP} \tag{4.5}$$

To obtain Λ_{QP} Singular value Decomposition (SVD) can be used. SVD is a method to separates a matrix into its key features [5]. The SVD of a matrix consist of three matrices, A orthogonal matrix U, Λ and orthogonal matrix V^T, where Λ contains all singular values of the corresponding matrix, ordered in its diagonal [5] (see equation 4.6).

$$svd(A) = U_A \, \Lambda_A \, V_A^T \tag{4.6}$$

Matlab is easily able to obtain the SVD of a matrix using the method as described by [1]. If a matrix is a strictly diagonal and positive-definite and 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 $U_A = V_A$ 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, it is possible to write 4.4 as 4.7 and 4.5 as 4.8 and 4.9:

$$WQW^T W^{-T} \breve{P} W^{-1} = \Lambda_{QP}^2 \tag{4.7}$$

$$WQW^T = \Lambda_{QP} \tag{4.8}$$

$$W^{-T} \breve{P} W^{-1} = \Lambda_{QP} \tag{4.9}$$

Assuming there exist a matrix ϕ_Q for all possible matrices Q (see equation 4.10) we can also express $Q\,\breve{P}$ and λ_{QP} as in equations 4.11 and 4.12.

$$Q = \phi_Q^T \phi_Q \tag{4.10}$$

$$Q\breve{P} = \phi_Q \breve{P} \phi_Q^T = U_{QP} \, \Lambda_{QP}^2 \, U_{QP}^T \tag{4.11}$$

$$\Lambda_{QP} = \Lambda_{QP}^{-\frac{1}{2}} \, U^T \, \phi_Q \, \breve{P} \, \phi_Q^T \, U \, \Lambda_{QP}^{\frac{1}{2}} \tag{4.12}$$

Using these equations (4.9, 4.11 and 4.12) we are able to obtain an expression for W at last.

$$W = \Lambda_{QP}^{\frac{1}{2}} \, U^T \, \phi_Q^{-T} \tag{4.13}$$

4.2 Extended Balanced Truncation

With the new method, a similar approach is taken. 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. $\breve{P}$ and Q are defined as follow:

$$\breve{P} = \delta_o H^{-1} \tag{4.14}$$

$$Q = \delta_c H \tag{4.15}$$

Here δ is scalar obtained by solving equation 4.2 and increasing the value slightly so the left side of the equation is a negative definite matrix. If possible both values for the δ are said to be equal. Using SVD to obtain Λ_{PQ} in this is however difficult. Since Q and $\breve{P}$ are a scalar multiplied by H and its inverse, $Q\,\breve{P} = \delta_o\,\delta_c\,I$. As a result the SVD of $\breve{P}Q$ would result in $\Lambda_{QP} = \delta_o\,\delta_c\,I$. This means all relevant information is rather lost than balanced. To tackle this problem EBT uses a matrix S and T (defined in equation 4.16 and 4.17). Here matrices Γ_o and Γ_c are taken to be diagonal and bare a strong relation to Q and $\breve{P}$ respectively.

$$S = Q(\alpha\, Q + \Gamma_o)^{-1}\, Q \tag{4.16}$$

$$T = (\beta\, \breve{P} + \Gamma_c)^{-1} \tag{4.17}$$

4.3 General remarks

With EBT and GBT, multiple boundary conditions and constraints exist when applying these methods. Moreover, there exist multiple definitions for other variables and matrices. And for a more detailed explanation of these two methods, and the definition of all factors see [4]. A general overview of these constraints are included in appendixB

5 Application of EBT

After having obtained a method for creating mathematical representations of LSENs and obtaining insight into how the method of EBT is applied, it is possible to investigate and apply the reduction method in the context of LSENs. At first, values have to be determined for δ_c, δ_o, β, Γ_c and Γ_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 are 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 [6]. Using this idea, the value for δ_o is determined by establishing with what smallest possible number that allows constraint B.1 to holds. Or in other words, taking the smallest possible δ_o so the lowest eigenvalue of X_o equals 0. Lastly, to avoid violating the constraints due to round-off error δ_o is increased slightly. If using the same value for δ_c as δ_o holds for constraint B.2, δ_c is taken to be equal to δ_o. Otherwise, δ_c is calculated similarly to δ_o, by taking the smallest possible δ_c so the lowest eigenvalue of X_c equals 0.

The other values that need to be obtained are β , Γ_c, and Γ_o. Again these values are obtained seeking for a boundary condition of a constraint, in this case, being B.3 and B.4. As mentioned in section 4, Γ marks a close resemblance to $\breve{P}$ and Q. For testing the model Γ_c and Γ_o are determined as following:

$$\Gamma_c = \epsilon_c\, \breve{P}\, \zeta \tag{5.1}$$

$$\Gamma_o = \epsilon_o\, Q\, \zeta \tag{5.2}$$

$$\zeta = diagonal(1.1^1, 1.1^2, 1.1^3, ...1.1^n, 1.1^1, 1.1^2, 1.1^3, ...1.1^n) \tag{5.3}$$

For defining ζ , $n = number\, of\, inductors\, /\, capasitors$. Using these definitions, β is set to equal 1 and increased with a multiplication of ten until the constraint B.3 is solved with ϵ_c fixated to equal one. This provides a rough estimate of the minimum value for β. Afterward fixating this minimum value for β the same constraint is solves, now altering the value for ϵ_c. Initially, ϵ_c is still set equals 1, and is increased with steps of five until one additional increase of the same size violates constraint B.3. Using the same approach for defining ϵ_c we define ϵ_o now solving constraint B.4 and setting $\alpha = \beta$.

A Matlab code is made that calculates these values for δ_c, δ_o, β, Γ_c and Γ_o, and reduces the model using EBT as well as GBT (see appendix **??**).

6 reduced model evaluation

Models have been created and simulated in simulink. However, I still need not write about the findings.

7 Conclusions

to be written

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, 3 edition, aug 1999.

[2] Athanasios C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society of Industrial and Applied Mathematics, Philadelphia, illustrate edition, 2005.

[3] Michael Beitelschmidt and Panagiotis Koutsovasilis. Comparison of model reduction techniques for large mechanical systems-A study on an elastic rod. 2008.

[4] Pablo Borja, Jacquelien M A Scherpen, and Kenji Fujimoto. Extended balancing of continuous LTI systems: a structure-preserving approach. Technical report.

[5] S. L. Brunton and J. N. Kutz. *Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control*. 2017.

[6] B. D. Craven. Boundary conditions optimal control. *The Journal of the Australian Mathematical Society. Series B. Applied Mathematics*, 30(3):343–349, 1989.

[7] Geir E. Dullerud and Fernando Paganini. *A course in robust control theory: a convex approach*. Springer Science & Business media, 36 edition, 2013.

[8] L. Fortuna, G. Nunnari, and A. Gallo. *Model Order Reduction Techniques with Applications in Electrical Engineering*. Springer London, 1992.

[9] A. J. Van Der Schaft and R. V. Polyuga. Structure-preserving model reduction of complex physical systems. *Proceedings of the IEEE Conference on Decision and Control*, pages 4322–4327, 2009.

[10] Arjan Van Der Schaft. *Communications and Control Engineering L2-Gain and Passivity Techniques in Nonlinear Control*. 2017.

[11] Xiaoguang Wei, Shibin Gao, Tao Huang, Tao Wang, and Wenli Fan. Identification of two vulnerability features: A new framework for electrical networks based on the load redistribution mechanism of complex networks. *Complexity*, 2019, 2019.

A Appendix A. Mathematical representation of electrical circuits

$$\begin{aligned}
L_1\,\dot{I}_{l1} &= U - V_{c1} - I_{l1}\,R_{l1}\\
L_2\,\dot{I}_{l2} &= V_{c1} - V_{c2} - I_{l2}\,R_{l2}\\
L_n\,\dot{I}_{ln} &= V_{c1} + V_{c2} + ... - V_{cn} - I_{ln}\,R_{ln}\\
C_1\,\dot{V}_{c1} &= \frac{V_{c1}}{R_{c1}} - I_{l1} + I_{l2} + ... + I_{ln}\\
C_1\,\dot{V}_{c2} &= \frac{V_{c2}}{R_{c2}} + I_{l1} - I_{l2} + ... + I_{ln}\\
C_1\,\dot{V}_{cn} &= \frac{V_{cn}}{R_{cn}} + I_{l1} + I_{l2} + ... - I_{ln}
\end{aligned} \tag{A.1}$$

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

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

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

$$
\begin{aligned}
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
Rl &= diag(-R_{l1}, -R_{l2}, ..., -R_{ln}) \\
Rc &= 0_{n,n} \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
Rl &= diag(-R_{l1}, 0, 0, ..., 0) \\
Rc &= diag(-\frac{1}{R_{c1}}, -\frac{1}{R_{c2}}, ..., -\frac{1}{R_{cn}}) \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned}
\tag{A.6}
$$

$$
\begin{aligned}
J_1 &= \begin{pmatrix} 1 & 0 & 0 & .. & 0 \\ 0 & 1 & 0 & .. & 0 \\ 0 & 0 & 1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & 1 \end{pmatrix} \\
Rl &= diag(-R_{l1}, -R_{l2}, ..., -R_{ln}) \\
Rc &= \begin{pmatrix} -\frac{1}{R_{c1}} - \frac{1}{R_{c2}} & \frac{1}{R_{c2}} & 0 & .. & 0 \\ \frac{1}{R_{c2}} & -\frac{1}{R_{c2}} & \frac{1}{R_{c3}} & .. & 0 \\ 0 & \frac{1}{R_{c3}} & -\frac{1}{R_{c3}} & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -\frac{1}{R_{cn}} \end{pmatrix} \\
B &= (0, 0, ..., 0, , \frac{1}{R_{c1}}, 0, ..., 0)^T
\end{aligned}
\tag{A.7}
$$

$$
\begin{aligned}
0 &= 0_{n,n} \\
L &= diag(L_1, L_2, ..., L_n) \\
C &= diag(C_1, C_2, ..., C_n) \\
\dot{I}_l &= (\dot{I}_{l1}, \dot{I}_{l2}, ..., \dot{I}_{ln})^T \\
\dot{V}_c &= (\dot{V}_{c1}, \dot{V}_{c2}, ..., \dot{V}_{cn})^T \\
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
I_l &= (I_{l1}, I_{l2}, ..., I_{ln})^T \\
V_c &= (V_{c1}, V_{c2}, ..., V_{cn})^T \\
Rl &= diag(-R_{l1}, -R_{l2}, ..., -R_{ln}) \\
Rc &= diag(-\frac{1}{R_{c1}}, -\frac{1}{R_{c2}}, ..., -\frac{1}{R_{cn}}) \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned}
\tag{A.8}
$$

B Appendix EBT constraints

$$\begin{aligned} Q &= \delta_o * H; \\ Xo &= -Q * A - A' * Q - C' * C; \\ Xo &\geq 0; \end{aligned} \tag{B.1}$$

$$\begin{aligned} \breve{P} &= \delta_c * H^{-1}; \\ Xc &= -Q * A - A' * Q - C' * C; \\ Xc &\geq 0; \end{aligned} \tag{B.2}$$

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

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

C 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`

C.1 Model reduction script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ------ Model ------ %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
clc

%input model MANUAL
% Rc=[]; %values for risistance in Rc (in Ohm)
% Rl=[]; %values for risistance in Rl (in Ohm)
% Cc=[]; %values for capasitance in C (in Farad)
% Ll=[]; %values for inductance in L (in Henry)
% ModT=[]; %Model type
% M=[]; %reduction (in percentages)
% n=size(L);
% n=n(1,1);

%generat random model
% n=20; %dimentions of the system (number of conductors and capasitors)
% setMT=1; %determine model type or if=0 generates random model type
% setRe=0.25; %determine reduction in % or if=0 generates random reduction in %
% saveM=0; % if=0 saves model in 'Model_auto_save'
% [Rl,Rc,Cc,Ll,ModT,M] = Random_model_generator(n,setMT,setRe,saveM);

%load existing model
load ('Model_auto_save');

%%
if ModT==1
    [H,R,J,B] = Modeltype41(Rl,Rc,Ll,Cc);
elseif ModT==2
    [H,R,J,B] = Modeltype42(Rl,Rc,Ll,Cc);
elseif ModT==3
    [H,R,J,B] = Modeltype43(Rl,Rc,Ll,Cc);
else
    'error model type does not exist'
    return
end
F   =J+R;
A   =F*H;
Hi  =inv(H);

beta=1;
M    =2*M*n;

C   = B'*H;

if ModT==2
    do=0.10;
```

```
    Q = do*H;
    Xo = -Q*A-A'*Q-C'*C;
    EIGXo=eig(Xo);
    i=max(EIGXo);
    while i<0
        do=do*1.5;
        Q = do*H;
        C= B'*H;
        Xo = -Q*A-A'*Q-C'*C;
        EIGXo=eig(Xo);
        i=max(EIGXo);
    end
else
    syms 'do';
    Q  = do*H;
    Xo = -Q*A-A'*Q-C'*C;

    EIGXo = eig(Xo);
    i = 1;
    while i<=2*n % find definition delta_o
        EIGXo(i)=solve(EIGXo(i)==0,do);
        i=i+1;
    end
    do = double(max(EIGXo))+0.0001;
end
Q  = do*H;
Qi=inv(Q);
Xo = -Q*A-A'*Q-C'*C;

dc = do;

Pi = dc*Hi;
Xc = -A*Pi-Pi*A'-(B*B');

if Xc>=0 % find definition delta_c if needed
    syms 'dc'
    Pi=dc*Hi;
    Xc=-A*Pi-Pi*A'-(B*B');
    EIGXc=eig(Xc);
    i=1;
    while i<=2*n
        EIGXc(i)=solve(EIGXc(i)==0,dc);
        i=i+1;
    end
    dc=double(max(EIGXc))+0.0001;
    Pi=dc*Hi;
end

epsc = 1;
epso = 1;

i=1;
while i<=n %define zeta
    zeta_c(i,i)=Pi(i,i)*(1.1^i);
    zeta_o(i,i)=Q(i,i)*(1.1^i);
    zeta_c(n+i,n+i)=Pi(n+i,n+i)*(1.1^i);
    zeta_o(n+i,n+i)=Q(n+i,n+i)*(1.1^i);
    i=i+1;
```

```
end

GAMc=-epsc*zeta_c;
GAMo=zeta_o;

Thc=(-GAMc+A*Pi+B*B')*inv(Xc)*(-GAMc+Pi*A'+B*B');
condc=2*(beta*Pi+GAMc)-Thc;
con1=min(eig(condc)); % checking (all the eigenvalues must be positive)
i=1;
while con1<=0 %find smalles possible beta
    beta=beta*10;
    condc=2*(beta*Pi+GAMc)-Thc;
    con1=min(eig(condc));
    i=i+1;
    if i>11; %set maximum value for beta (if beta to large -> rounding errors)
        'error_beta'
        return
    end
end

epsc1=epsc;

i=1;
while con1>=0 %obtaining max value for epsilon_c
    epsc=epsc1;
    epsc1=(5*i);
    GAMc=-epsc1*zeta_c;
    Thc=(-GAMc+A*Pi+B*B')*inv(Xc)*(-GAMc+Pi*A'+B*B');
    condc=2*(beta*Pi+GAMc)-Thc;
    con1=min(eig(condc));
    i=i+1;
    if epsc1>=beta %epsc has to be smaller then beta
        'error_epsc'
        return
    end
end

GAMc=-epsc*zeta_c;

alpha=beta;

Tho=(GAMo-Q*A)*inv(Xo)*(GAMo-A'*Q);

condo=2*(alpha*Q+GAMo)-Tho;
con2=min(eig(condo)); %checking (all the eigenvalues must be positive)

epso1=epso;

i=1;
while con2 >= 0
    epso=epso1;
    epso1=(5*i);
    GAMo=epso1*zeta_o;
    Tho=(GAMo-Q*A)*inv(Xo)*(GAMo-A'*Q);
    condo=2*(alpha*Q+GAMo)-Tho;
    con2=min(eig(condo));
    i=i+1;
```

```
    if epso>=beta %epsc has to be smaller then beta
        'error_epso'
        return
    end
end
GAMo=epso*zeta_o;

%%
%Define Ti

Ti=beta*Pi+GAMc;

min(eig(Ti)); % checking (all the eigenvalues must be positive)

Thc=(-GAMc+A*Pi+B*B')*inv(Xc)*(-GAMc+Pi*A'+B*B');
condc=2*(Ti)-Thc;
con1=min(eig(condc)); % checking (all the eigenvalues must be positive)
if con1<=0
    con1
    'error1'
    return
end


%%

% Define S

S=inv(alpha*Qi+Qi*GAMo*Qi);

Tho=(GAMo-Q*A)*inv(Xo)*(GAMo-A'*Q);

condo=2*(alpha*Q+GAMo)-Tho;
con2=min(eig(condo)); %checking (all the eigenvalues must be positive)
if con2<=0
    con2
    'error2'
    return
end
%%%%%%%%%%%%%%%%%%%%
%% Transformation Extended

%splitting Ti and S in C and L related parts
TiL=Ti(1:n,1:n);
TiC=Ti(n+1:2*n,n+1:2*n);
SL=S(1:n,1:n);
SC=S(n+1:2*n,n+1:2*n);

PhTiC=chol(TiC);
[UTSC,S2TSC]=svd(PhTiC*SC*PhTiC');
STSC=sqrt(S2TSC);
WEC=PhTiC'*UTSC*sqrt(inv(STSC));
WECi=inv(WEC);

PhTiL=chol(TiL);
[UTSL,S2TSL]=svd(PhTiL*SL*PhTiL');
STSL=sqrt(S2TSL);
WEL=PhTiL'*UTSL*sqrt(inv(STSL));
```

```
WELi=inv(WEL);



WE=[WEL zeros(n,n); zeros(n,n) WEC];
WEi=inv(WE);

%% Transformation Generalized

PiL=Pi(1:n,1:n);
PiC=Pi(n+1:2*n,n+1:2*n);
QL=Q(1:n,1:n);
QC=Q(n+1:2*n,n+1:2*n);

PhPiC=chol(PiC);
[UQPC,S2QPC]=svd(PhPiC*QC*PhPiC');
SQPC=sqrt(S2QPC);
WGC=PhPiC'*UQPC*sqrt(inv(SQPC));
WGCi=inv(WGC);

PhPiL=chol(PiL);
[UQPL,S2QPL]=svd(PhPiL*QL*PhPiL');
SQPL=sqrt(S2QPL);
WGL=PhPiL'*UQPL*sqrt(inv(SQPL));
WGLi=inv(WGL);


WG=[WGL zeros(n,n); zeros(n,n) WGC];
WGi=inv(WG);


%%

e=@(k,n) [zeros(k-1,1);1;zeros(n-k,1)];
% M is the number of state you want to truncate
M=M;
K=(n*2)-M;
aux1=[e(1,2*n)];
aux2=[e(n+1,2*n)];
i=2;
while i<=0.5*K
    aux3=[e(i,2*n)];
    aux4=[e(n+i,2*n)];
    aux1=[aux1,aux3];
    aux2=[aux2,aux4];
    i=i+1;
end
aux=[aux1,aux2];


% Reduced via extended

Ah=WE\A*WE;
Hh=WE'*H*WE;
Bh=WE\B;
C=B'*H;
Ch=C*WE;
Ar=aux'*Ah*aux;
```

```
Br=aux'*Bh;
Cr=Ch*aux;
Hr=aux'*Hh*aux;

% Reduced via generalized

Ahg=WG\A*WG;
Hhg=WG'*H*WG;
Bhg=WG\B;
Chg=C*WG;
Arg=aux'*Ahg*aux;
Brg=aux'*Bhg;
Crg=Chg*aux;
Hrg=aux'*Hhg*aux;

%%

lCn = eig(STSC)/max(eig(STSC));
lLn = eig(STSL)/max(eig(STSL));

lCni = flip(lCn);
lLni = flip(lLn);

%plot eigenvalues
% figure
% plot(lCni,'bO','LineWidth',2)
% grid on
% title('Eigenvalues of $\Lambda_{ST_{1}}$','Interpreter','latex')
% xticks([0:n])
% figure
% plot(lLni,'rO','LineWidth',2)
% grid on
% title('Eigenvalues of $\Lambda_{ST_{2}}$','Interpreter','latex')
% xticks([0:n])



%%

% Error system

Ae = [Ah zeros(2*n,2*n-M); zeros(2*n-M,2*n) Ar];
Be = [Bh; Br];
Ce = [Ch -Cr];

Aeg = [Ahg zeros(2*n,2*n-M); zeros(2*n-M,2*n) Arg];
Beg = [Bhg; Brg];
Ceg = [Chg -Crg];

%%

% H inf normst

% extended

fsys = ss(A,B,C,0);
bsys = ss(Ah,Bh,Ch,0);
rsys = ss(Ar,Br,Cr,0);
```

```
esys = ss(Ae,Be,Ce,0);

[ninff,fpeakf] = hinfnorm(fsys);
[ninfb,fpeakb] = hinfnorm(bsys);
[ninfr,fpeakr] = hinfnorm(rsys);
[ninfe,fpeake] = hinfnorm(esys);

bgsys = ss(Ahg,Bhg,Chg,0);
rgsys = ss(Arg,Brg,Crg,0);
egsys = ss(Aeg,Beg,Ceg,0);

[ninfbg,fpeakbg] = hinfnorm(bgsys);
[ninfrg,fpeakrg] = hinfnorm(rgsys);
[ninfeg,fpeakeg] = hinfnorm(egsys);

[ninfe;ninfeg]
```

C.2 Model Generator

```
%Random model generator
function [Rl,Rc,Cc,Ll,ModT,M] = Random_model_generator(n,setMT,setRe,saveM)

Rl=randi([0 2000],n,1);
Rc=randi([0 2000],n,1);
Cc=randi([1 5000],n,1);
Cc=Cc*10^-6;
Ll=randi([50 15000],n,1);
Ll=Ll*10^-6;
if setMT==0
    ModT=randi([1 3]);
else
    ModT=setMT;
end
if setRe==0
    M=randi([0.1 0.5]);
else
    M=setRe;
end
if saveM==0
    save('Model_auto_save','Rl','Rc','Ll','Cc','ModT','M','n')
end
end
```

C.3 Matlab Code model type 1

```
function [H,R,J,B] = Modeltype41(Rl,Rc,Ll,Cc)
R=[Rl, Rc];

n=size(Ll);
n=n(1,1);

B=zeros([2*n 1]);
B(1,1)=1;

c=size(Cc);
c=c(1,1);
rl=size(Rl);
rl=rl(1,1);
rc=size(Rc);
rc=rc(1,1);

if n~=c && n~=rl && n~=rc
    'dimentions do not match'
    return
end
 %% creating matrix F

i=1;
while i<=n
    A11(i,i)=-R(i,1);
    if R(i,2)==0
        A22(i,i)=0;
    else
        A22(i,i)=-1/R(i,2);
    end
    H(i,i)=1/Ll(i);
    H(i+n,i+n)=1/Cc(i);
    i=i+1;
end

 a=ones(1,n);
 b=ones(1,n-1);
 A121=diag(-a);
 A122=diag(b,-1);
 A12=A121+A122;
 O=zeros(n,n);

 A122=diag(b,-1);
 A21=-1*A12.';
 R=[A11,O;O,A22];
 J=[O,A12;A21,O];
end
```

C.4 Matlab Code model type 2

```
function [H,R,J,B] = Modeltype42(Rl,Rc,Ll,Cc)
R=[Rl, Rc];

n=size(Cc);
n=n(1,1);

B=zeros(2*n,1);
B(n+1,1)=1/R(1,2);

l=size(Ll);
l=l(1,1);
rl=size(Rl);
rl=rl(1,1);
rc=size(Rc);
rc=rc(1,1);

p=min(Rc);
if n~=c && n~=rl && n~=rc
    'dimentions do not match'
    return
end
 %% creating matrix F
 A11(1,1)=-R(1,1);
 A22(n,n)=-1/R(n,2);
 A22(n-1,n)=1/R(n,2);
 A22(n,n-1)=1/R(n,2);
 H(1,1)=1/Ll(1);
 H(1+n,1+n)=1/Cc(1);
 i=2;
 j=n-1;
 k=j;
 while i<=n
     A11(i,i)=-R(i,1);
     A22(j,j)=-1/R(j+1,2)-1/R(j,2);
     while k>1
     A22(j-1,j)=1/R(j,2);
     A22(j,j-1)=1/R(j,2);
     k=k-1;
     end
     H(i,i)=1/Ll(i);
     H(i+n,i+n)=1/Cc(i);
     j=n-i;
     k=j;
     i=i+1;
 end

 a=ones(1,n);
 A12=diag(a);
 A21=-1*A12.';

 O=zeros(n,n);
 R=[A11,O;O,A22];
 J=[O,A12;A21,O];
end
```

C.5 Matlab Code model type 3

```
function [H,R,J,B] = Modeltype43(Rl,Rc,Ll,Cc)
R=[Rl, Rc];

n=size(Ll);
n=n(1,1);

B=zeros(2*n,1);
B(n+1)=1;

c=size(Cc);
c=c(1,1);
rl=size(Rl);
rl=rl(1,1);
rc=size(Rc);
rc=rc(1,1);

if n~=c && n~=rl && n~=rc
    'dimentions do not match'
    return
end
 %% creating matrix F
 i=1;
 while i<=n
    A11(i,i)=-R(i,1);
    if R(i,2)==0
        A22(i,i)=0;
    else
        A22(i,i)=-1/R(i,2);
    end
    H(i,i)=1/Ll(i);
    H(i+n,i+n)=1/Cc(i);
    i=i+1;
 end

 a=ones(1,n);
 b=ones(1,n-1);
 A121=diag(-a);
 A122=diag(b,-1);
 A12=A121+A122;
 O=zeros(n,n);

 A122=diag(b,-1);
 A21=-1*A12.';
 R=[A11,O;O,A22];
 J=[O,A12;A21,O];
end
```