university of groningen / faculty of science and engineering

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: Borja, Scherpen and Fujimoto developed a theoretical framework for the application of a model order reduction method called extended balanced truncation [6]. This model reduction technique allows one to reduce a linear time-invariant model while preserving its physical structure. 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 this reduction method in the mentioned context. This is done by first generating mathematical models representing large-scale linear electrical networks using Matlab. Consequently extended balanced truncation is applied using the framework described in [6]. To do so, again a Matlab script was written that allowed to persistently apply the framework, eliminating deviation due to human error. To, validate if the method was successful, the reduced-order models are reconstructed in Simulink and plotted to compare the input-output relation. For a frame of reference, another reduction method (generalised balanced truncation) was applied on the same models and compared to extended balanced truncation.

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 [3]. Unfortunately, the simulation of a whole system is often difficult and sometimes not feasible due to their often large dimensions [4]. As a result "Model order reduction is critical for engineers and scientists" [15], and offers a solution, by reducing the dimensions of a model enabling simulations to be performed while requiring a smaller amount of computing time.

Since the work of Moore [18] various techniques have been developed [4]. One example of a method for reducing a model is balanced truncation (BT) [20]. This method like many others 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 as a tool for balancing (or re-ordering) a system. These gramians are the solutions to Lyapunov inequalities and are used to reduce the error bound of these reduced-order models [14, 7]. 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) [19]. 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 [22]. Moreover, the PH system modelling can be regarded to bridge the gap between passive system models and explicit physical network realizations[22]. Scherpen and Fujimoto further developed this method for application on continuous-time linear time-invariant (CTLTI) systems in [21]. Based on this research Borja, Scherpen and Fujimoto were able to continue building on this research to develop a framework for applying this reduction method of EBT on CTLTI systems [6].

A relevant context with the need for model reduction is that of large-scale electrical networks (LSENs). The electricity grids exhibit several typical features of complex networks[24] and are one example of these

LSENs with often large dimensions where simulation of the whole network takes significant time. As a result, model order reduction is considered a relevant topic in this context and the Framework developed in [6] could further improve the capabilities for modelling LSENs.

This project aims to investigate the method for creating a reduced-order model as described in [21] and its possibilities for the application to LSENs by applying the theoretical framework described in [6]. One of the main objectives is 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 guarantees an acceptable error ratio while preserving the physical interpretation of the reduced-order model. The acceptable error ratio is dependent on the size of the truncated part. However, for practical purpose, we consider that such an error must be less than 5%.

For performing this research the programming language Matlab was used [1] and the mentioned simulations were done using [13].

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_0 represents the current as an input from a current source
I represents the identity matrix
$Matrix\,A \geq 0$ notes a positive-semi-definite matrix
σ represents a singular value

3 Preliminaries

The research in this paper uses a variety of principles and originated from linear algebra. For a full understanding of the framework developed in [6] a brief description of these principles is 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 [5].

Preliminary 2 Singular value decomposition (SVD) is a method used for separating a matrix into its key features [8]. The SVD of a matrix consist of three matrices; an orthogonal matrix U, Λ and orthogonal matrix V^T, where Λ contains all singular values of the original matrix, ordered in its diagonal [8] (see equation 3.1).

$$svd(A) = U_A\,\Lambda_A\,V_A^T \tag{3.1}$$

Matlab is easily able to obtain the SVD of a matrix using the method as described in [2]. It is important to note that the singular values of a matrix are equivalent to the absolute values of its eigenvalues. In addition, if a matrix is strictly diagonal and positive-definite then all its eigenvalues are stored in the diagonal of Λ. Moreover, when a matrix is diagonal the matrices U_A and V_A are said to be equal.

Preliminary 3 The method of EBT requires similarity and 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 $C^1\,A\,C \,=\, B$ [11].
And

$A \in R^{2nx2n}$, symplectically congruent to $B \in R^{2nx2n}$ if there exists a symplectic matrix $C \in R^{2nx2n}$ such that $C^T\,A\,C \,=\, B$ [11].

Preliminary 4 Some more generally used definition in linear algebra are obtained from Horn and Johnson [16]. 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 [6] and is defined as flow;

$$A * A^{-1} \,=\, I \tag{3.2}$$

$$A * I = A^{-1} \tag{3.3}$$

4 Modeling of electrical networks

Before it is possible to apply the theoretical framework in question to the relevant context, models of Large-scale electrical networks need to be obtained. For this purpose a Matlab script is written that can generating random models representing LSENs. The theory used for creating these models is based on the research of Castaños, Jayawardhana, Ortega and García-Canseco [9] and Jeltsema [17]. 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)

$$\sum_{i=1}^{n} V_i = 0 \tag{4.1}$$

$$I_1 = I_2 = ... = I_n \tag{4.2}$$

$$V_1 = V_2 = ... = V_n \tag{4.3}$$

$$\sum_{i=1}^{n} I_i = 0 \tag{4.4}$$

$$V_l = \dot{I}_l\, L \tag{4.5}$$

$$I_c = \dot{V}_c\, C \tag{4.6}$$

By applying these laws to several standard forms of electrical circuits a mathematical representation of these electrical circuits was obtained. To allow the model to be generated in this way assumption 1 was made, stating:

Assumption 1 If a circuit has a voltage source, there is always one resister directly in series with this voltage source.

As described later 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 [23], with $H = H^T > 0$; and $R = R^T \geq 0, J = J^T$ [6]. In other words, The matrix H contains all information regarding the energy storing elements. In the case of an RLC circuit, this means all components of L (inductors) and C (capacitors). The matrix R consists of information regarding the resistive elements.

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

In order to easily obtain the matrices J, R, H and B the models 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 result an expression as follow to be obtained:

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

where:

$$\begin{pmatrix} L & 0 \\ 0 & C \end{pmatrix} = H^{-1} \tag{4.9}$$

$$\begin{pmatrix} I_l \\ V_c \end{pmatrix} = x \tag{4.10}$$

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

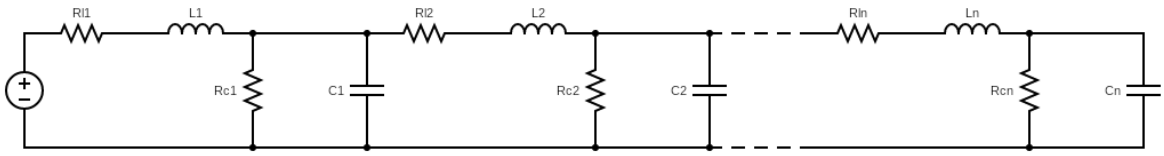

Figure 4.1: Standard circuit type 1 [12]

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) it is possible to obtain the following two equations to represent this model:

$$U = V_{c1} + L_1\, \dot{I}_{l1} + I_{l1}\, R_{l1} \tag{4.12}$$

$$I_{l1} = C_1\, \dot{V}_{c1} + \frac{V_{c1}}{R_{c1}} \tag{4.13}$$

Reordering these equations will allow for a state-space representation of this model to be created in the above mentioned desired form (equations 4.14, 4.15 and 4.16).

$$L_1\, \dot{I}_{l1} = U - V_{c1} - I_{l1}\, R_{l1} \tag{4.14}$$

$$C_1\, \dot{V}_{c1} = I_{l1} + \frac{V_{c1}}{R_{c1}} \tag{4.15}$$

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

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 substituted by $V_{c(i+1)} \,/\, R_{c(i+1)} + C_{(i+1)}\, \dot{V}_{c(i+1)}$. As a result, all circuit of model type 1 can be represented as equation A.1 and in the corresponding state-space form equation A.2 (see appendix)

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). 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 law (Equation 4.5 and 4.6) one is able to obtain a mathematical representation of the circuit, which can also be represented in state-space form (Equation 4.17, 4.18 and 4.19).

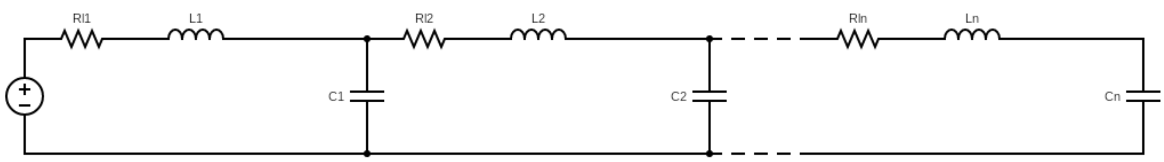

Figure 4.2: Standard circuit type 1.2 [12]

$$U = V_{c1} + L_1\,\dot{I}_{l1} + I_{l1}\,R_{l1} \tag{4.17}$$

$$I_{l1} = C_1\,\dot{V}_{c1} \tag{4.18}$$

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

With electrical circuit 1.2, adding a component containing L, C and R_l requires an additional equations. The equations in question are an alteration of equations 4.17 and are ascertained 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 it is needed to add a term I_{li+1} to the right side of the equation per additional component. Later, I_{li+1} can in its turn be defined as $I_{li+2} + I_{ci+1}$. All further I_{li} (for i=3,4,...,n-1)can be defined in a similar way. The definition of the final component in respect to I_{ln} is however slightly different. Since in this final component the current in $I_{ln} = I_{cn}$ and therefore also equal to $C_n\,\dot{V}_{cn}$. 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).

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 its 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 it is assumed a resistor is always present next to a voltage source, this is also the case here. As a result, circuit type 1.3 is the same as type 1 for the smallest possible form. Only from the second "component" a resistor in series with the inductor is taken out and the resulting state-space representation is different.

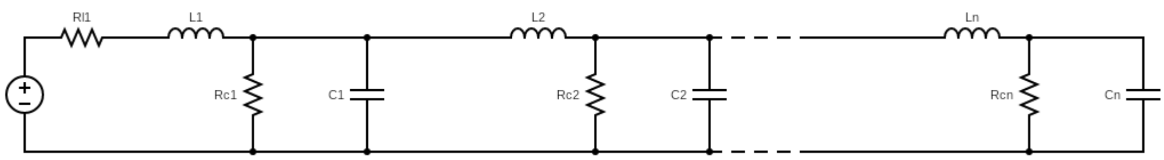

Figure 4.3: Standard circuit type 1.3 [12]

A similar approach for obtaining 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:

$$U = L_1\,\dot{I}_{l1} + I_{l1}\,R_{l1} + L_2\,\dot{I}_{l2} + V_{c2} \tag{4.20}$$

$$I_{l1} = C_1\,\dot{V}_{c1} + \frac{V_{c1}}{R_{c1}} + C_2\,\dot{V}_{c2} + \frac{V_{c2}}{R_{c2}} \tag{4.21}$$

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

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 A.6

4.1.3 Modeling 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 its output will provide all elements needed for creating a state-space representation similar to 4.7 and can be found in appendix B.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 model type 2 a look is taken at a situation where a capacitor is in parallel over an inductor. The inductor is still in series with a resistor (see image 4.4)

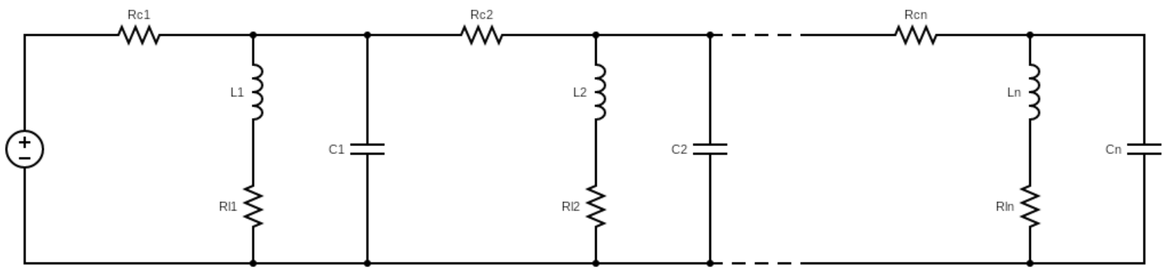

Figure 4.4: Standard circuit type 2 [12]

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.

$$I_{Rc1} = I_{l1} + \dot{V}_{c1}\,C_1 \tag{4.23}$$

$$U = R_{c1}\,I_{Rc1} + V_{c1} \tag{4.24}$$

$$V_{c1} = \dot{I}_{l1}\,L_1 + I_{l1}\,R_{l1} \tag{4.25}$$

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 $R_{c1}, R_{l1}, I_{l1}, V_{c1}$ (represented in equations 4.26 and 4.27) which can than be put into a state-space form (see 4.28).

$$\dot{V_{c1}}\,C_1 = I_{l1} - \frac{V_{c1}}{R_{c1}} + \frac{U}{R_{c1}} \tag{4.26}$$

$$\dot{I}_{l1}\,L_1 = V_{c1} - I_{l1}\,R_{l1} \tag{4.27}$$

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

Like type 1 it is possible to upscale a model of type 2. Adding a component containing a R_{c2}, R_{l2}, L_2 and C_2 will influence equations 4.23 by giving the equation an additional component I_{Rc1} (see equation 4.29). With respect to the equations defining the voltage of source U a new additional definition will exist (4.29) where:

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

$$V_{c2} \;=\; \dot{I}_{l2}\,L_2 + R_{l2}\,I_{l2} \tag{4.30}$$

$$I_{Rc1} = \dot{V}_{c1}\,C_1 + I_{l1} + \dot{V}_{c2}\,C_2 + I_{l2} \tag{4.31}$$

With these expressing $\dot{I}_{l2}\,L_2$, $\dot{I}_{l1}\,L_1$ and $\dot{V}_{c2}\,C_2$ in can be expressed in terms of R_c, R_l, L and C by reordering equations 4.35 , 4.25 and 4.29 respectively. For obtaining $\dot{V}_{c1}\,C_1$ some additional substitution is required. Taking the found expression for $\dot{V}_{c2}\,C_2$ resulting form equation 4.29 and substituting this in equation **??** allows $\dot{V}_{c1}\,C_1$ in its turn to be defined in term of R_c, R_l, L and C. These equations can again be written in a state-space form (see equation 4.32).

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

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

$$\dot{I}_{li}\, L_i = V_{ci} - R_{li}\, I_{li} \tag{4.33}$$

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

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

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

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

4.2.1 The generation of models containing component type 2

Like with type 1 the model type 2 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 capacitance over these two 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. A situations in which this is the case is considered not to be relevant for this study. Since an optimal way for reducing this system already exists. 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 mathematical model of this circuit will be equivalent to the same state-space representation where R_{li} is equal to 0 (the general definition of the model can be seen at A.7). With this in mind, a Matlab script was again written to obtain a matrix H, J, R, and B for a circuit in the form of model type 2 (see appendix B.4).

4.3 Model electrical circuit type 3

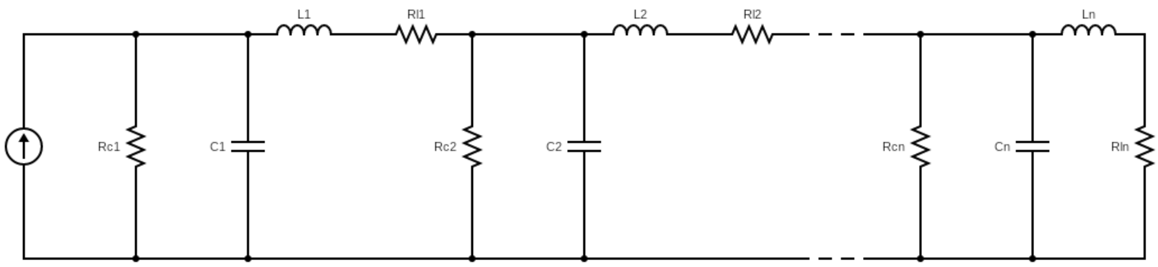

Figure 4.5: Standard circuit type 3 [12]

The third and final model of electrical circuit that is 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, one can find the state-space of this model. With studying this model type it quickly becomes clear its mathematical representation is similar to that of model type 1. The only difference being the input-matrix B. 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 (or current) instead of a volts (or potential difference). The state-space of this model is represented in equation A.8 and the Matlab code for generating model type 3 in B.5

5 Extended balanced truncation

As previously mentioned, this study considers a continuous-time linear time-invariant (CTLTI) system described as in equations 4.7. 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 $C \in R^{nxm}$. Assuming this system is asymptotically stable, the so-called generalized observability gramian $Q \in R^{nxn}$ is a positive semi-definite solutions to the following Lyapunov inequality;[6]

$$QA + A^T Q + C^T C \leq 0 \tag{5.1}$$

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

$$A\breve{P} + \breve{P}A^T + BB^T \leq 0 \tag{5.2}$$

In particular, when 5.1 and 5.2 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 [3].

5.1 Generalized Balanced Truncation

Before truncating the system, it first needs to be balanced. Balancing is based on obtaining a matrix Λ_{QP} for which equation 5.4 holds. To do so, generalized balanced truncation (GBT) aims to find a matrix W that solves 5.3 which is later used to balance the system[6]:

$$WQ\breve{P}W^{-1} = \Lambda_{QP}^2 \tag{5.3}$$

$$\breve{P} = Q = \Lambda_{QP} \tag{5.4}$$

Here Λ_{QP} is a diagonal matrix containing all singular values of the matrix Q and $\breve{P}$. Moreover, the singular values in matrix Λ_{QP} are ordered with respect to size, with large singular values in the top right. In other words $\Lambda_{QP} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_n)$ with $\sigma_i \geq \sigma_{i+1}$. Obtaining this matrix Λ_{QP} is done by singular value decomposition (SVD) (see section 3 *Preliminary 2*).

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 5.3 as 5.5, and 5.4 as 5.6 and 5.7:

$$WQW^TW^{-T}\breve{P}W^{-1} = \Lambda_{QP}^2 \tag{5.5}$$

$$WQW^T = \Lambda_{QP} \tag{5.6}$$

$$W^{-T}\breve{P}W^{-1} = \Lambda_{QP} \tag{5.7}$$

Assuming there exist a matrix ϕ_Q for all possible matrices Q (see equation 5.8) it is possible express $Q\,\breve{P}$ and Λ_{QP} as in equations 5.9 and 5.10.

$$Q = \phi_Q^T\phi_Q \tag{5.8}$$

$$Q\breve{P} = \phi_Q\breve{P}\phi_Q^T = U_{QP}\,\Lambda_{QP}^2\,U_{QP}^T \tag{5.9}$$

$$\Lambda_{QP} = \Lambda_{QP}^{-\frac{1}{2}}\,U^T\,\phi_Q\,\breve{P}\,\phi_Q^T\,U\,\Lambda_{QP}^{\frac{1}{2}} \tag{5.10}$$

Using these equations (5.7, 5.9 and 5.10) we are able to obtain an expression for W at last.

$$W = \Lambda_{QP}^{\frac{1}{2}}\,U^T\,\phi_Q^{-T} \tag{5.11}$$

When W is obtained the system is balanced (see 5.12) and the bottom states can be truncated. *Note,* Ab, Bb and Cb represent the balanced matrices of a state-space representation.

$$\begin{aligned} Ab &= W^{-1}A\,W \\ Bb &= W^{-1}B \\ Cb &= C\,W \end{aligned} \tag{5.12}$$

5.2 Extended Balanced Truncation

With the method of EBT, a similar approach is taken as with GBT. First, the system needs to be balanced and a matrix W needs to be found. Later the balanced system can be truncated. The main difference is, as previously mentioned, in using a PH representation of the system. EBT defines the generalised gramians $\breve{P}$ and Q as follow:

$$\breve{P} = \delta_o H^{-1} \tag{5.13}$$

$$Q = \delta_c H \tag{5.14}$$

Here δ is scalar solving the equality of equation 5.1 and 5.2. If possible both values for the δ are said to be equal. With the use of the diagonal matrix H, using SVD to obtain Λ_{QP} is however difficult. Since Q

and $\breve{P}$ are a scalar multiplied by H and its inverse, $Q\,\breve{P} = \delta_o\,\delta_c\,I$ (see section 3 *Preliminary 4*). As a result the SVD of $Q\,\breve{P}$ would return a $\Lambda_{QP} = \delta_o\,\delta_c\,I$. This means all relevant information is lost rather than that the system is balanced. To tackle this problem EBT uses extended gramians S and T (defined in equation 5.15 and 5.16). Here Γ_o and Γ_c are matrices taken to be a diagonal while baring a strong resemblance, but are not equal to Q and $\breve{P}$ respectively, and β and α are scalars.

$$S = Q(\alpha\,Q + \Gamma_o)^{-1}\,Q \tag{5.15}$$

$$T = (\beta\,\breve{P} + \Gamma_c)^{-1} \tag{5.16}$$

Matrices S and T are solutions to the linear matrix inequalities (LMIs) 5.17 and 5.18. 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. For The definitions of matrices X_o, A_o and A_c In these LMIs see A.9 and A.10.

$$\begin{pmatrix} X_o & Q - A_o^T\,S \\ Q - S^T\,A_o & S + S^T \end{pmatrix} \geq 0 \tag{5.17}$$

$$\begin{pmatrix} -P\,A - A^T\,P & -P + A_c^T\,T & -2\,P\,B \\ -P + T^T\,A_c & T + T^T & 2\,T^T\,B \\ -2\,B^T\,P & 2\,B^T\,T & 4\,I_m \end{pmatrix} \geq 0 \tag{5.18}$$

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

6 Application of EBT

With the method for creating mathematical representations of LSENs and the obtaining insight into framework for applying EBT, it is possible to investigate 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 with whom the resulting reduced-order models contain the smallest deviation with respect to the original model. 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 is found on the boundary conditions [10]. Using this idea, the value for δ_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 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 allowes X_c to be a positive-semi-definite matrix, 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. Here an approximation of the boundary conditions is used. In this case, these conditions relate to LMI 5.17 and 5.18. As mentioned in section 5, Γ marks a close resemblance to $\breve{P}$ and Q. For validation of the framewok as discribed in [6], Γ_c and Γ_o are determined as following:

$$\Gamma_c = \epsilon_c\,\breve{P}\,\zeta \tag{6.1}$$

$$\Gamma_o = \epsilon_o\,Q\,\zeta \tag{6.2}$$

$$\zeta = diagonal(1.1^1, 1.1^2, 1.1^3, ...1.1^n, 1.1^1, 1.1^2, 1.1^3, ...1.1^n) \tag{6.3}$$

For defining ζ , $n = number\,of\,inductors\,or\,capasitors$. Using these definitions, β is set to equal 1 and increased with a multiplication of ten until the LMI 5.18 is solved with ϵ_c fixated to equal one. Afterwards this rough boundary condition of β is fixed and the same constraint is solves, now altering the value for ϵ_c. Initially, ϵ_c is again set equals 1, and is increased with steps of five until one additional increase of five violates LMI 5.18. Using the same approach for defining ϵ_c a definition for ϵ_o is calculated, now solving LMI 5.17 and setting $\alpha = \beta$.

A Matlab script is written that calculates these values for δ_c, δ_o, β, Γ_c and Γ_o, and the other mathematical calculations needed to reduces the model using EBT as well as GBT (For the complete script see appendix B).

7 Reduced model evaluation

To validate the framework for applying EBT [6] and determine if it is can reduce the model while still preserving the original structure, eight different models have been constructed and reduces for all model types. The dimensions, reduction and error bound of these models are displayed in Table C.1, C.2 and C.3. To study the behaviour of these reduced models a plot is made (see appendix C, D, E and F). 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. 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 for all three model types. (For all plots, the lines related to EBT are shown in blue, The lines related to GBT are shown in Red, and graphs related to the original system are displayed in green) These can be seen in figure; C.1, C.2, and C.3. The calculation of the error-bound uses the method as described by Willems [25]

7.1 Results

The obtained data support the findings of Sandberg in [19], and 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. With the use of EBT the models reduced up to 80%, show a error bound that which can almost 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 frequently gave an error. This was due to a high value for the resistance in Rc_1. The error occurred when running the original model in Simulink and was thus considered not to be a result of a flaw in the reduction technique. The needs for a restriction for the maximal resistance in Rc_1 can be explained by investigating the mathematical model. With a relatively high resistance in Rc_1 the input of the model will be scaled by one over this resistance of Rc_1. Making the input of the system have a significantly small effect on the output. This makes the input inefficient and the model unfavourable to use in a real live application. It should be noted that these findings only hold for the model types considered in this paper, and therefore it does not mean they are valid for all models of CTLTI-LSENs.
(All data, models, reduced models, Matlab scripts and Simulink simulations can be found at: `https://phw-h.github.io/IDP_extended_balanced_truncation/`).

8 Conclusions and Discussion

The framework for applying EBT developed by Borja, Scherpen and Fujimoto [6] (EBT) shows great promise. The methods return reduced models with an almost negligible error-bound when applied to CTLTI-LSENs, and only when the model is reduced by 80% or more the error-bound start to increase. The framework for the application of the reduction technique, unfortunately, has no optimal available method for determining variables; δ_c, δ_o, Γ_c, $\Gamma_o, \beta \, and \, \alpha$. As a result, the findings in this paper do not show a complete picture. Future research could aim to develop a method for determining these values further improving the possible benefits of the framework discribed in [6]. In addition, this research focuses on the use of three standard types of electrical circuits. Making the findings not hold for all CTLTI electrical circuits. Lastly, the small error bound of the reduction method of EBT raises some suspicion. With an error bound occasionally equal to zero, suggest the used models in Matlab might not be optimal for validating this reduction method.

References

[1] *MATLAB version 9.10.0.1602886 (R2021a)*. The MathWorks Inc., Natick, Massachusetts, 2021.

[2] 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.

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

[4] M. Beitelschmidt 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.

[5] R. Bhatia. *Positive definite matrices.* Prinston, 2009.

[6] P. Borja, J. M. A. Scherpen, and K. Fujimoto. Extended balancing of continuous LTI systems: a structure-preserving approach. Technical report.

[7] R. C. Brown and D. B. Hinton. Lyapunov Inequalities and their Applications. In *Survey on Classical Inequalities*, pages 1–25. Springer Netherlands, 2000.

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

[9] 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.

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

[11] 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.

[12] C. Diagram. Circuit Diagram editor, 2021. `https://www.circuit-diagram.org/editor` (accessed:2021.04.1).

[13] S. Documentation. Simulation and model-based design, 2020.

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

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

[16] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, New York, 2 edition, 1985.

[17] D. Jeltsema. Modeling and Control of Nonlinear Networks A Power-Based Perspective. *Perspective*, (August):231, 2005.

[18] B. C. Moore. Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction. *IEEE Transactions on Automatic Control*, 26(1):17–32, 1981.

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

[20] H. Sandberg and A. Rantzer. Balanced Truncation of Linear Time-Varying Systems. *IEEE Transactions on Automatic Control*, 49(2):217–229, 2004.

[21] J. M. Scherpen and K. Fujimoto. Extended balanced truncation for continuous time LTI systems. *2018 European Control Conference, ECC 2018*, pages 2611–2615, 2018.

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

[23] 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.

[24] 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.

[25] J. C. Willems. Model Reduction by Balancing (slide), 2002.

A Appendix. Mathematical representation of electrical circuits

$$\begin{aligned}
L_1\,\dot{I}_{L1} &= U - V_{C1} - I_{L1}\,R_{L1} \\
L_2\,\dot{I}_{L2} &= V_{C1} - V_{C2} - I_{L2}\,R_{L2} \\
L_n\,\dot{I}_{Ln} &= V_{C1} + V_{C2} + ... - V_{Cn} - I_{Ln}\,R_{Ln} \\
C_1\,\dot{V}_{C1} &= \frac{V_{C1}}{R_{C1}} - I_{L1} + I_{L2} + ... + I_{Ln} \\
C_2\,\dot{V}_{C2} &= \frac{V_{C2}}{R_{C2}} + I_{L1} - I_{L2} + ... + I_{Ln} \\
C_n\,\dot{V}_{Cn} &= \frac{V_{Cn}}{R_{Cn}} + I_{L1} + I_{L2} + ... - I_{Ln}
\end{aligned} \tag{A.1}$$

$$\begin{pmatrix} L_1\,\dot{I}_{L1} \\ L_2\,\dot{I}_{L2} \\ \vdots \\ L_n\,\dot{I}_{Ln} \\ \\ C_1\,\dot{V}_{C1} \\ C_2\,\dot{V}_{C2} \\ \vdots \\ C_n\,\dot{V}_{Cn} \end{pmatrix} = \begin{pmatrix} -R_{L1} & 0 & .. & 0 & -1 & 0 & .. & 0 \\ 0 & -R_{L2} & .. & 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & .. & -R_{Ln} & 0 & 0 & .. & -1 \\ \\ 1 & -1 & .. & 0 & -\frac{1}{R_{C1}} & 0 & .. & 0 \\ 0 & 1 & .. & 0 & 0 & -\frac{1}{R_{C2}} & .. & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & .. & 1 & 0 & 0 & .. & -\frac{1}{R_{Cn}} \end{pmatrix} \begin{pmatrix} I_{L1} \\ I_{L2} \\ \vdots \\ I_{Ln} \\ \\ V_{C1} \\ V_{C2} \\ \vdots \\ V_{Cn} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} U \tag{A.2}$$

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

$$\begin{aligned}
0 &= 0_{n,n} \\
L &= diag(L_1, L_2, ..., L_n) \\
C &= diag(C_1, C_2, ..., C_n) \\
\dot{I}_l &= (\dot{I}_{L1}, \dot{I}_{L2}, ..., \dot{I}_{Ln})^T \\
\dot{V}_c &= (\dot{V}_{C1}, \dot{V}_{C2}, ..., \dot{V}_{Cn})^T \\
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
I_l &= (I_{L1}, I_{L2}, ..., I_{Ln})^T \\
V_c &= (V_{C1}, V_{C2}, ..., V_{Cn})^T \\
Rl &= diag(-R_{L1}, -R_{L2}, ..., -R_{Ln}) \\
Rc &= diag(-\frac{1}{R_{C1}}, -\frac{1}{R_{C2}}, ..., -\frac{1}{R_{Cn}}) \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned} \tag{A.4}$$

$$
\begin{aligned}
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
Rl &= diag(-R_{L1}, -R_{L2}, ..., -R_{Ln}) \\
Rc &= 0_{n,n} \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned}
\tag{A.5}
$$

$$
\begin{aligned}
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
Rl &= diag(-R_{L1}, 0, 0, ..., 0) \\
Rc &= diag(-\frac{1}{R_{C1}}, -\frac{1}{R_{C2}}, ..., -\frac{1}{R_{Cn}}) \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned}
\tag{A.6}
$$

$$
\begin{aligned}
J_1 &= \begin{pmatrix} 1 & 0 & 0 & .. & 0 \\ 0 & 1 & 0 & .. & 0 \\ 0 & 0 & 1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & 1 \end{pmatrix} \\
Rl &= diag(-R_{L1}, -R_{L2}, ..., -R_{Ln}) \\
Rc &= \begin{pmatrix} -\frac{1}{R_{C1}} - \frac{1}{R_{C2}} & \frac{1}{R_{C2}} & 0 & .. & 0 \\ \frac{1}{R_{C2}} & -\frac{1}{R_{C2}} & \frac{1}{R_{c3}} & .. & 0 \\ 0 & \frac{1}{R_{c3}} & -\frac{1}{R_{c3}} & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -\frac{1}{R_{Cn}} \end{pmatrix} \\
B &= (0, 0, ..., 0, , \frac{1}{R_{C1}}, 0, ..., 0)^T
\end{aligned}
\tag{A.7}
$$

$$
\begin{aligned}
0 &= 0_{n,n} \\
L &= diag(L_1, L_2, ..., L_n) \\
C &= diag(C_1, C_2, ..., C_n) \\
\dot{I}_l &= (\dot{I}_{L1}, \dot{I}_{L2}, ..., \dot{I}_{Ln})^T \\
\dot{V}_c &= (\dot{V}_{C1}, \dot{V}_{C2}, ..., \dot{V}_{Cn})^T \\
J_1 &= \begin{pmatrix} -1 & 0 & 0 & .. & 0 \\ 1 & -1 & 0 & .. & 0 \\ 0 & 1 & -1 & .. & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & .. & -1 \end{pmatrix} \\
I_l &= (I_{L1}, I_{L2}, ..., I_{Ln})^T \\
V_c &= (V_{C1}, V_{C2}, ..., V_{Cn})^T \\
Rl &= diag(-R_{L1}, -R_{L2}, ..., -R_{Ln}) \\
Rc &= diag(-\frac{1}{R_{C1}}, -\frac{1}{R_{C2}}, ..., -\frac{1}{R_{Cn}}) \\
B &= (1, 0, ..., 0, , 0, 0, ..., 0)^T
\end{aligned}
\tag{A.8}
$$

$$
\begin{aligned}
A_o &= \alpha\, I_n \,+\, A \\
X_o &= -\,Q\,A \,-\, A^T\,Q \,-\, C^T\,C \\
\alpha &> 0 \\
X_o &\geq 0
\end{aligned}
\tag{A.9}
$$

$$
\begin{aligned}
A_c &= \beta\, I_n \,+\, A \\
X_c &= -P\,A \,-\, A^T\,P \,-\, P\,B\,B^T\,P \\
\beta &\geq 0 \\
X_c &\geq 0
\end{aligned}
\tag{A.10}
$$

B 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`

B.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]
```

B.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
```

B.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
```

B.4 Matlab Code model type 2

```matlab
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
```

B.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
```

C Appendix Tables and Figures

Model	Input	Reduction	Dimentions	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 C.1: Error bound of GBT and EBT in relation to one model of type 1 and its dimensions.

Model	Input	Reduction	Dimentions	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 C.2: Error bound of GBT and EBT in relation to one model of type 2 and its dimensions.

Model	Input	Reduction	Dimentions	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 C.3: Error bound of GBT and EBT in relation to one model of type 3 and its dimensions.

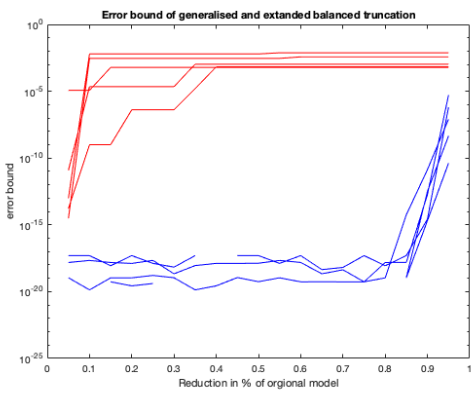

Figure C.1: Error bound model type 1
Generalised (Red), Extended (Blue)

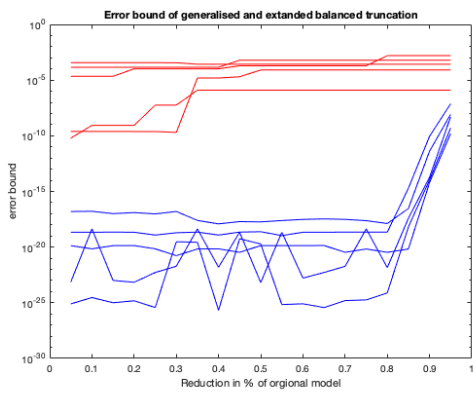

Figure C.2: Error bound Model type 2
Generalised (Red), Extended (Blue)

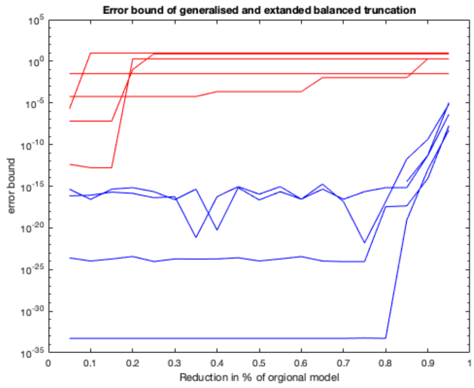

Figure C.3: Error bound Model type 3
Generalised (Red), Extended (Blue)

D Plots model type 1

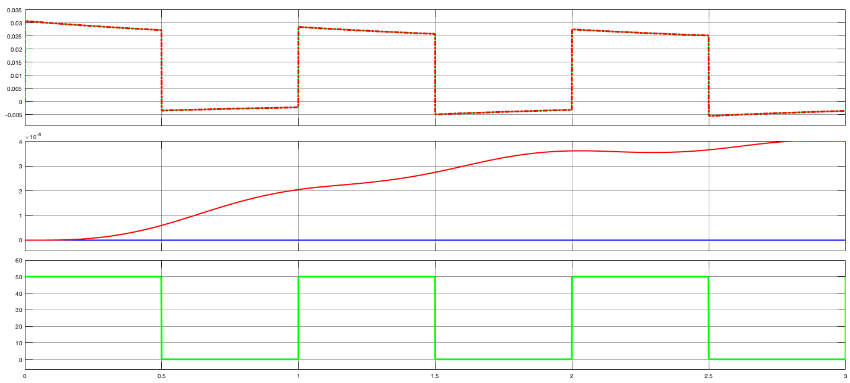

Figure D.1: Model simulation of model 1.1
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

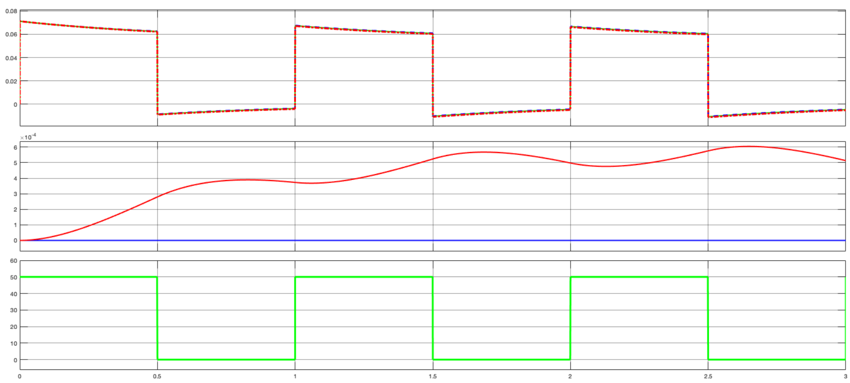

Figure D.2: Model simulation of model 1.2
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

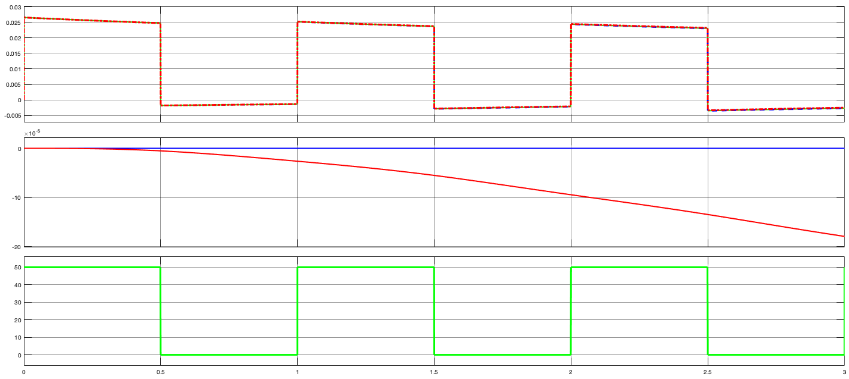

Figure D.3: Model simulation of model 1.3
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

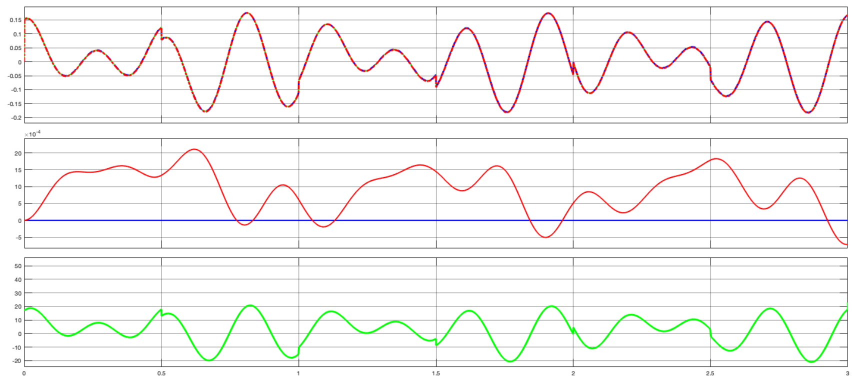

Figure D.4: Model simulation of model 1.4
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

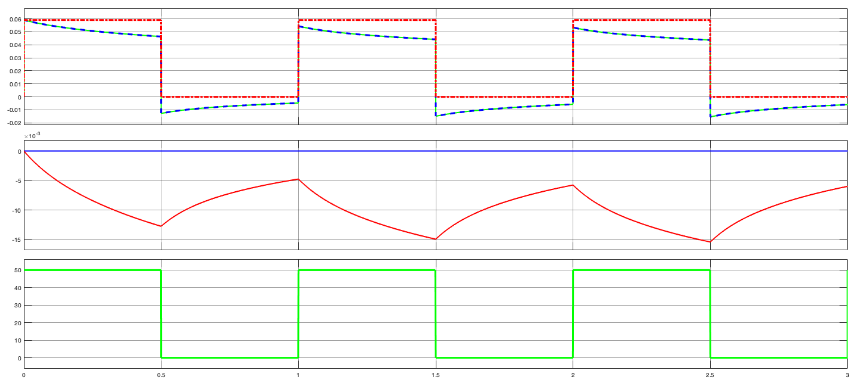

Figure D.5: Model simulation of model 1.5
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

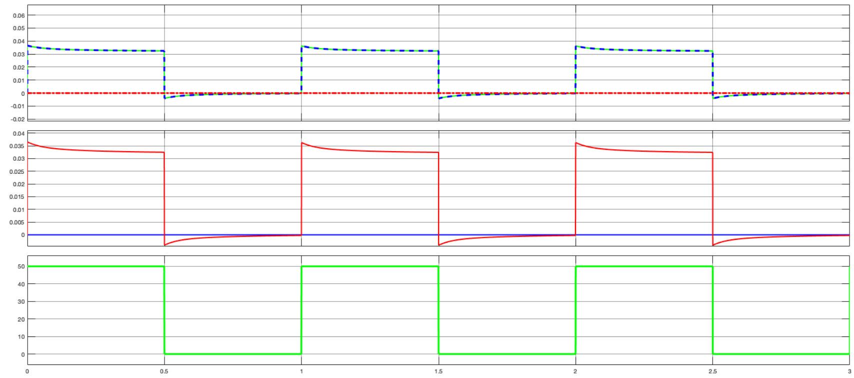

Figure D.6: Model simulation of model 1.6
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

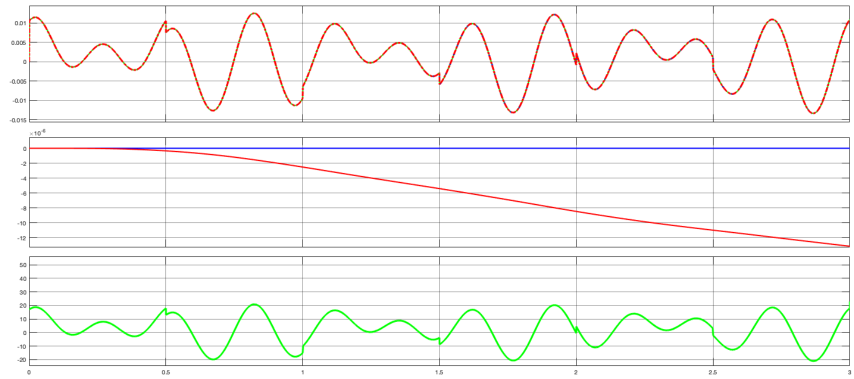

Figure D.7: Model simulation of model 1.7
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

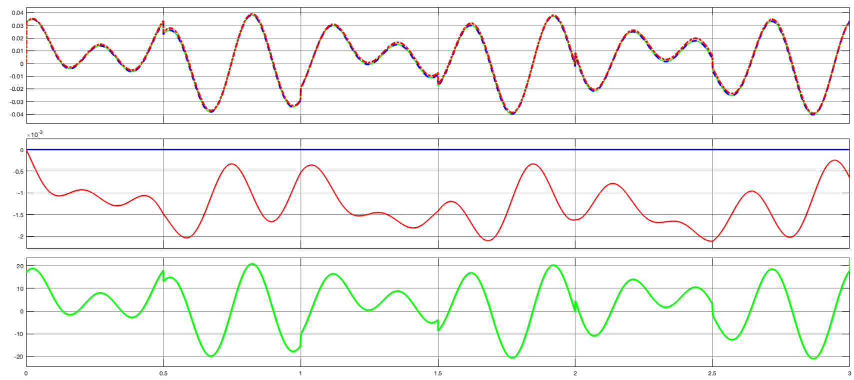

Figure D.8: Model simulation of model 1.8
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

E Plots model type 2

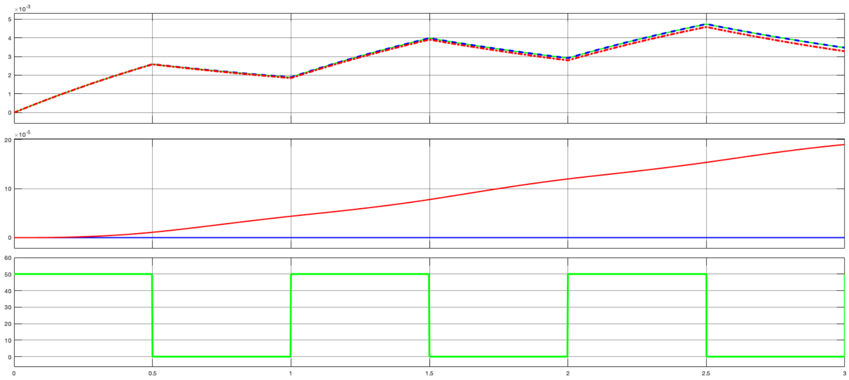

Figure E.1: Model simulation of model 2.1
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

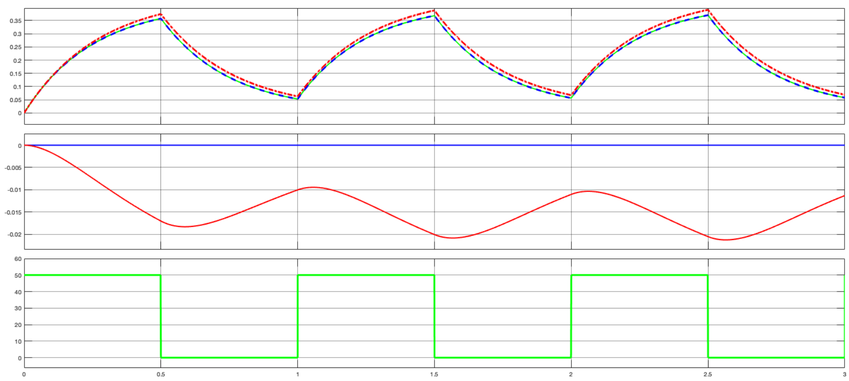

Figure E.2: Model simulation of model 2.2
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

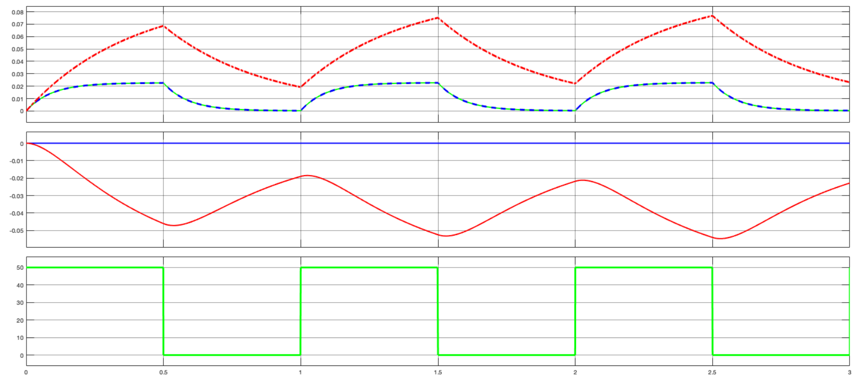

Figure E.3: Model simulation of model 2.3
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

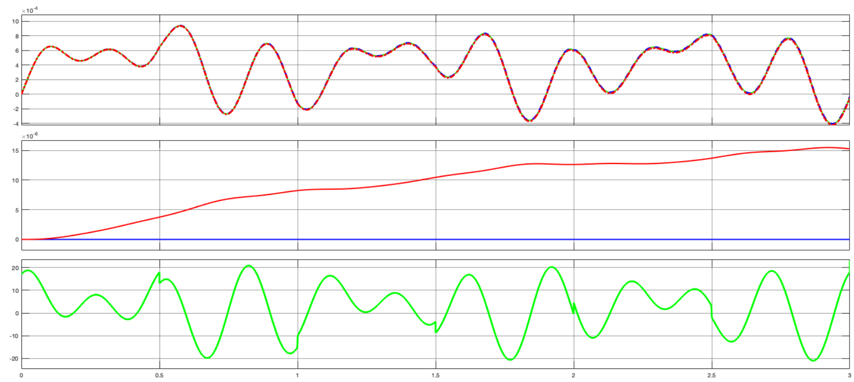

Figure E.4: Model simulation of model 2.4
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

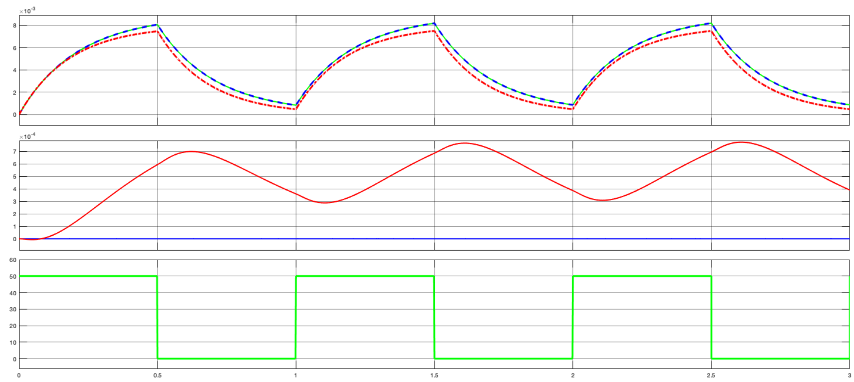

Figure E.5: Model simulation of model 2.5
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

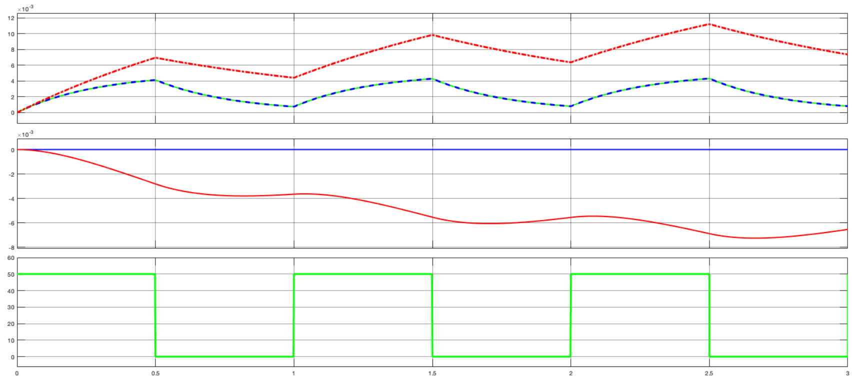

Figure E.6: Model simulation of model 2.6
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

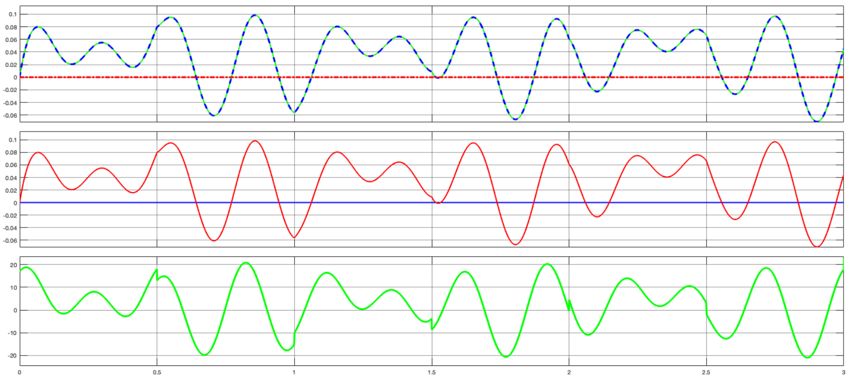

Figure E.7: Model simulation of model 2.7
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

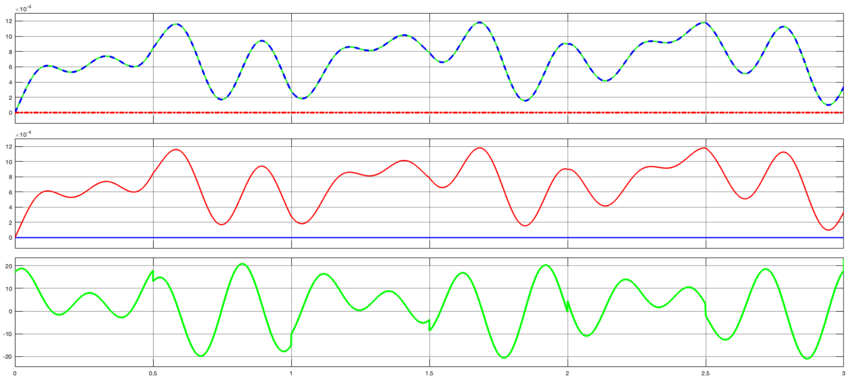

Figure E.8: Model simulation of model 2.8
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

F Plots model type 3

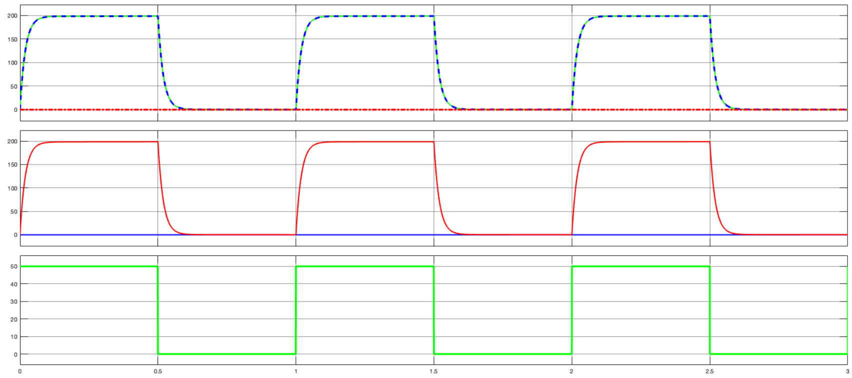

Figure F.1: Model simulation of model 3.1
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

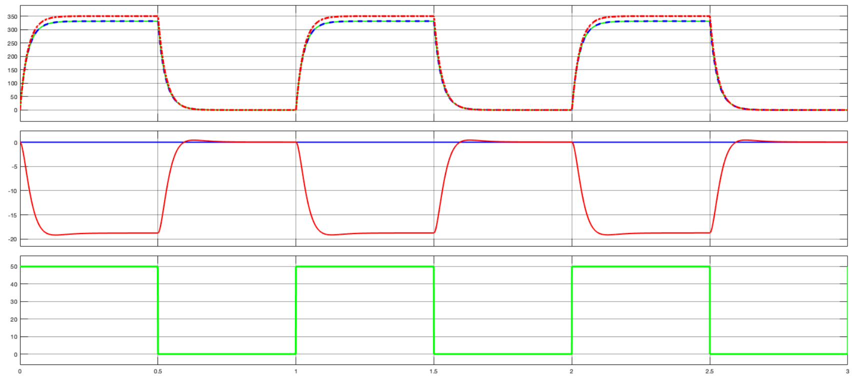

Figure F.2: Model simulation of model 3.2
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

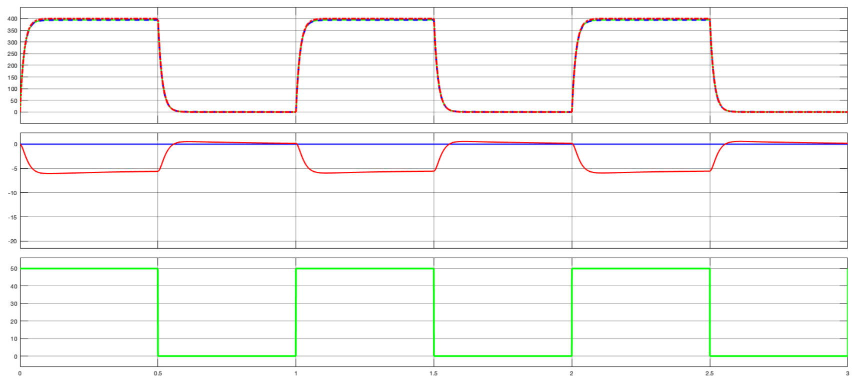

Figure F.3: Model simulation of model 3.3
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

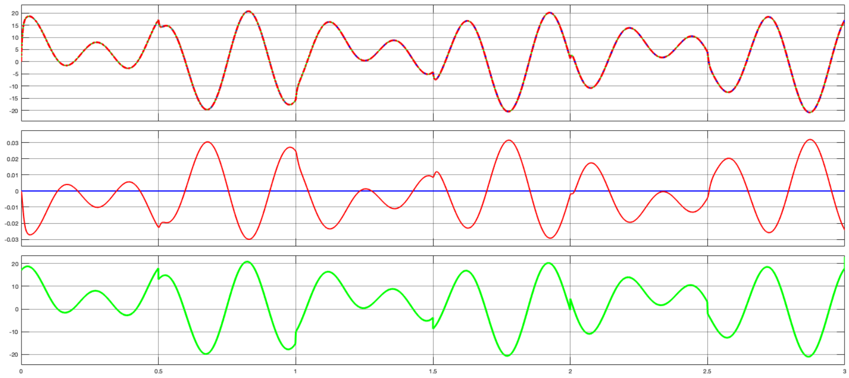

Figure F.4: Model simulation of model 3.4
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

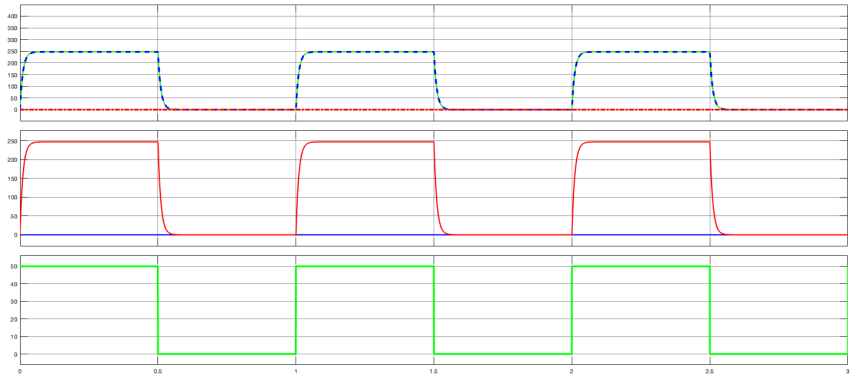

Figure F.5: Model simulation of model 3.5
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

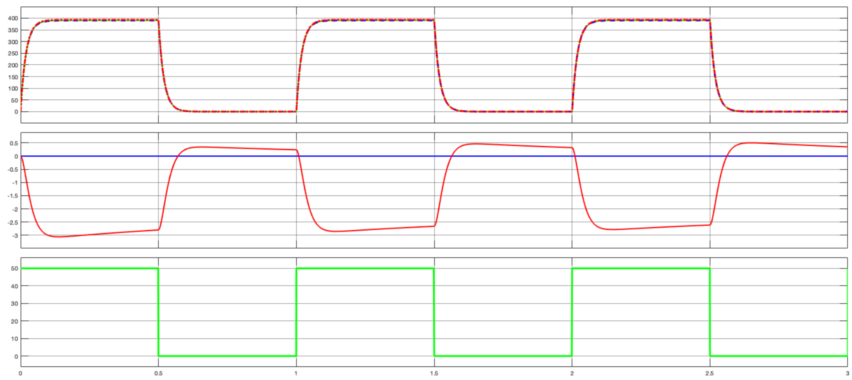

Figure F.6: Model simulation of model 3.6
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

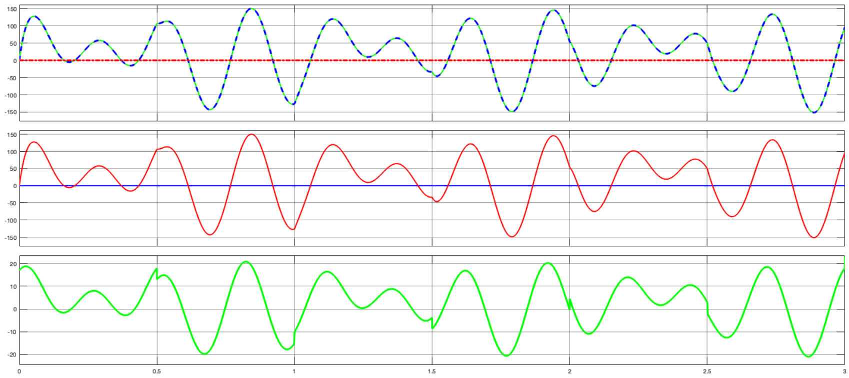

Figure F.7: Model simulation of model 3.7
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)

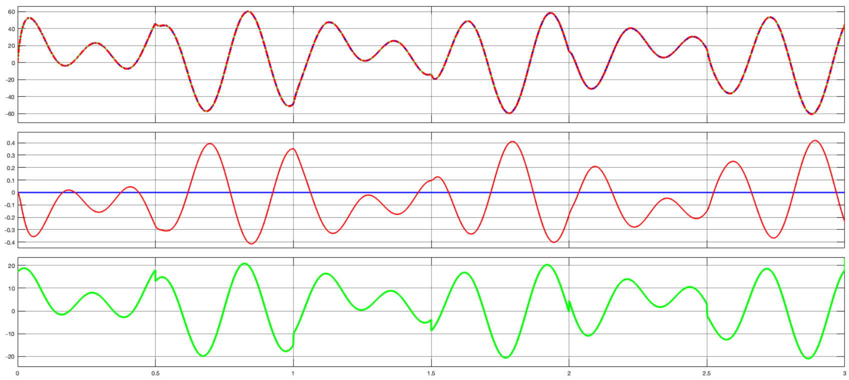

Figure F.8: Model simulation of model 3.8
From top to bottom; Output systems, Deviation from original models, Input function
Generalised (Red), Extended (Blue), Original model (Green)