This is to certify that the
dissertation entitled

NETWORKED ASSEMBLY OF NONLINEAR PHYSICAL SYSTEM MODELS

presented by

Elliot Motato

has been accepted towards fulfillment
of the requirements for the

Ph.D. degree in Mechanical Engineering

Clark Radcliffe
Major Professor's Signature

January 11, 2007
Date

DATE DUE	DATE DUE	DATE DUE

NETWORKED ASSEMBLY OF NONLINEAR PHYSICAL SYSTEM MODELS

By

Elliot Motato

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Mechanical Engineering

2007

ABSTRACT

NETWORKED ASSEMBLY OF NONLINEAR PHYSICAL SYSTEM MODELS

By

Elliot Motato

Global engineering design requires efficient model integration. This process is characterized by the capability of performing an automatic and recursive model assembly. Model integration is a complicated issue when complex assemblies are involved. Model reuse is the key for handling assembly complexity. The ability to reuse and exchange models relies on a standard format [1]. This condition is fundamental to avoid model reformulation.

The Modular Modeling Method (MMM) [2] is a recursive model integration algorithm designed to achieve automatic integration of linear models. MMM linear standard models are dynamic matrices with a unique input-output variable representation. MMM characteristics facilitates the networked integration of large complex linear models.

The objective of this work is to extend the model assembly properties of MMM to assemble nonlinear physical models. This work can divided in two parts. In the first part, the problem of assembling multi-port nonlinear physical models that perform at a constant operating point is solved. In the second part, the problem of assembly nonlinear physical models that perform at a region of operation is solved. Once achieved, the MMM nonlinear methodology will contribute to enable cooperative global engineering design, by facilitating the automatic integration and the simulations of nonlinear physical engineered artifacts which are designed and provided from different and possibly geographical distant locations.

To my dears Sandra and Sarah.

ACKNOWLEDGMENTS

I would like to thank my Advisor Dr. Clark Radcliffe for his continuous guidance and endless support during my years at Michigan State University.

My thanks to Dr. Khalil, Dr. Gokcek, Dr. Sticklen and Dr. Mukerjee for serving as members of my Ph.D committee. They provide useful suggestions to improve this work.

My thanks to Umar Farooq, Jimmy Issa, Jeff Roads, Nanda Methil, Jorge Villa and Wesley Zanardelli for providing me valuable help through their friendship.

Finally, I would like to thank the graduate secretary Aida Montalvo for her help.

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 Global Engineering Design

Engineering analysts are in the process of creating a global engineering strategy that is able to integrate product design, product development, marketing analysis, and manufacturing process [3]. Today's engineers can communicate with of suppliers through supply chain management systems over the Internet [4]. The trend is to use the Internet as the media to achieve this global engineering design strategy.

Product design in the context of a global engineering strategy demands the assembly of dynamic system models from component models retrieved over the Internet. This process requires four important characteristics: 1. Multi-energy domain component models must have a unique standard format, 2. The exchange of model information must be executed in a single-query transmission, 3. The models must describe only external behavior, and 4. The assembly process must be recursive. These four characteristics make global engineering design practical.

A unique, standard model format is the key to handling model exchange through model reuse [4]. A modeling methodology using a unique model representation facilitates model query standardization. A unique model format query prevents model reformulation and decreases model exchange time computation. The Finite Element Method (FEM) [5], is a modeling methodology that uses a unique model format. FEM

uses the same format to represent both elements and the system assembled from those elements. A modeling method that uses a unique standard format is called *modular.*

Single-query exchange of model information reduces network traffic during the assembly of dynamic models through the Internet. A single-query exchange retrieves the full component model using a single request and answer on the Internet. Repeated queries increase network traffic dramatically. The Differential Algebraic Equation (DAE) approach [6] is an example of a model format, which can be exchanged in a single-query network transmission as a set of ordinary differential equations (ODEs) and algebraic constraints. Global engineering strategy demands assembly processes that reduce network load through exchange of model information with a single-query. A model assembly method that has this characteristic is called *single-query.*

An input-output model predicts external behavior using only input and output variables to protect internal proprietary design details. Because design is a dominant cost of new product development, internal product design details must be protected from competitors. These design details might include, the components used in the assembly, the order of connection of those components, the physical parameters of the components and the performance of each component. Protection of proprietary information is critical to the commercial acceptance of any model exchange system. Gu and Asada in [4] have used input and output variables to allow the co-simulation of a collection of dynamic sub-simulators without disclosing proprietary information. A model assembly method that uses models that predict external behavior using external input and output variables is called *external.*

A recursive model assembly process uses standard format component models to produce an assembly model in the same format. Once model assembly recursion is established at a single level, the model assembly method is easily extended to higher-level, more complex system models. DAE [6] is an example of an assembly methodology that recursively obtains DAE system models from either DAE elements

or DAE subsystems. A model assembly process that uses standard format component models to produce an assembly model in the same format is called *recursive.*

All four characteristics are simultaneously required for a successful global engineering model assembly method. Co-simulation, [4], is external but is not single-query because network iteration is required to run dynamic sub-simulators. The DAE approach in [6], is recursive and single-query but is not external because DAE models provide internal information about the assembly components, component connectivity and internal parameters. Bond graphs [7] are modular and recursive but not external because they provide information about assembly components, component connectivity and internal parameters. FEM [5] is modular and single-query but not recursive. Assembly of FEM models generally requires global reformulation to guarantee geometric nodal compatibility. A multi-energy domain, modular, single-query, external and recursive, model assembly methodology is needed.

MMM [8]-[9] is modeling strategy that satisfies global engineering design requirements. MMM uses two standard model formats. Dynamic matrices are used in the networked distribution and assembly of linear models. Transfer function matrices are used in model simulation. MMM is characterized by the advantages that result from using these two model representations. This work is an extension of the MMM for assembly of nonlinear physical models represented through Volterra models. Using this extension, nonlinear dynamic models of assemblies can be built and distributed while hiding the topology and characteristics of their structural subassemblies.

This work is divided in four parts. In the first part, the linear MMM procedure, the standard linear model formats and the physical constraints to assemble physical models are reviewed. In the second part, the problem of assembling nonlinear physical models that operate at a constant point is solved. In the third part, a procedure to obtain Volterra models from port based nonlinear differential equation is explained. Finally in the fourth part, the MMM procedure for assembling Volterra

models performing at a region of operation is derived.

1.2 Contributions of This Work

This work makes six (6) significant, original contributions to mechanical engineering. These contributions are:

1. An extension of the Modular Modeling Methodology (MMM) to nonlinear systems.

2. The recognition that two model formats are necessary for physical system model assembly and simulation.

3. A method to assemble physical port-based affine ODEs around an equilibrium operating point.

4. A method to obtain subsystems operating points from the system assembly operating point outputs.

5. A method to obtain frequency domain Volterra system models from MIMO port-based ODEs.

6. A method to assemble physical MIMO port-based Volterra system models.

Each of these contributions is individually important to the development of a Global Engineering Design (GED) strategy.

The extension of the MMM to nonlinear systems is important. This extension allows the application of the MMM to assemble, distribute and simulate system models for a large class of nonlinear physical systems. This class is consists of nonlinear physical systems that are analytic or that have fading memory. Before this extension was developed, the application of the MMM was limited to the assembly, distribution and simulation of linear physical system models. The extension of the MMM to nonlinear systems is discussed throughout the thesis. Model distribution is discussed in the Chapter Two. Nonlinear subsystem models assembly is discussed in Chapters Three, Four and Five. Finally, nonlinear assembled system simulation is discussed at

the end of the Chapter Five.

The recognition that two model formats are necessary for physical system model assembly and simulation is important. As discussed below, the use of two model formats decreases the computational time required to assemble and simulate physical system models. These two model formats are the linear/nonlinear transfer function representation and the linear/nonlinear dynamic representation. The transfer function representation is the traditional format for simulation. This thesis shows for the first time, that in a constraint-based model assembly, the Transfer Function representation is not appropriate for MMM assembly because a matrix inversion is required. Matrix inversion demands substantial computational time, a clear disadvantage in a global distributed environment. Additionally, for this inverse to exist, system models are limited to have independent inputs and independent outputs. This is a big limitation on allowable models that excludes many dynamic system models. In contrast, if the format used for assembly is the dynamic representation, no matrix inversion is required and system models with dependent inputs and dependent outputs can also be assembled. The recognition that the two model formats are necessary for efficient physical system model assembly and simulation is discussed in the Chapter 2.

A method to assemble physical port-based affine ODEs around an equilibrium operating point is important. Affine systems often result of local linearization about an operating point. In this case, the local system model is linear in deviation variables and non-linear in physical variables. Because assembly constraints are always given in terms of physical variables, an explicit method to apply physical assembly constraints for affine physical models was first developed in this work. This new method addresses one of the most common nonlinear systems in mechanical engineering. Using this method many practical physical nonlinear system models can be obtained by assembling the affine approximations of their subsystems models. In addition, this method is computationally efficient because it is recursive and does not require

matrix inversions. These characteristics make this affine assembly procedure original and ideal for being used in the MMM. A method to assemble physical port-based affine ODEs around an equilibrium operating point is discussed in the Chapter 3.

The explicit method to obtain subsystems operating points from the system assembly operating point outputs is important. This method for the first time provides an explicit, closed form solution to the general operating point problem. In a recursive, closed, form, the method specifies information required to roll-down an operating point specification through every subsystem to all the lowest level components of a system. For each lowest level component, the method provides the exact port output values where the operating point representation of the nonlinear component model should be developed. The method then provides an explicit method for computing operating point inputs of the component level and use these inputs to compute the operating point inputs of the assembled system model. This method is demonstrated through Affine approximations, one of the standard formats used in the MMM nonlinear extensions to nonlinear system models. Determining the operating point inputs and outputs of nonlinear subsystems in a system assembly is a classical problem [4]. This method provides an explicit solution to this problem for an important class of mechanical engineering system models: port-based physical system assemblies. The method to obtain subsystems operating points from the operating outputs of an assembly is discussed in Chapter 3.

The method developed here to generate frequency domain Volterra system models from MIMO external port-based ODEs is important. Even though a similar procedure for the SISO case is available in the literature [10], a method for MIMO port-based, external nonlinear systems has not been published. The MIMO procedure requires a nonlinear vector operator to obtain the frequency domain kernels. This nonlinear operator is not required in the SISO case. An advantage of using this MIMO method is that the two standard MMM nonlinear formats can be easily obtained. The first

format is the Volterra transfer function representation. This format is traditionally used in model analysis and simulation. The second format is the Volterra dynamic representation. This format is used in the assembly of nonlinear system models. The Volterra dynamic representation was recognized and defined for first time in this work. The Volterra dynamic representation is important in the MMM because this external port based model format protects internal design details. This procedure generates the Volterra dynamic port-based system models in an explicit form for the first time. The method to obtain frequency domain port-based Volterra system models from MIMO port-based ODEs is discussed in Chapter 4.

The method to assemble physical MIMO external port-based Volterra system models is an original contribution to mechanical engineering because it is a direct procedure to assemble nonlinear, port-based, Volterra system models using port-based physical constraints. Volterra representations are appropriate to the MMM approach to distributed model assembly because they protect internal design details. This method extends the MMM approach to nonlinear port-based models through a method that is computationally efficient because it is recursive and does not require matrix inverses. These properties make this Volterra-based assembly procedure ideal for distribution and assembly of nonlinear physical engineering system models. The method to assemble physical MIMO port-based Volterra representations is described in the chapter 5.

1.3 Limitations of the Nonlinear MMM Extension

The nonlinear Modular Model distribution and assembly methodology has two (2) limitations. These limitations are:

1. The nonlinear subsystems must be modeled using port-based Volterra representations.

2. The assembled system model is valid if the subsystem models used in the assembly are valid.

The nonlinear subsystems must be modeled using port-based Volterra representations because the formats used in the nonlinear MMM extension developed here are port-based Volterra models. Not all nonlinearities can be represented using Volterra models. A non-linearity can be represented using a Volterra model if it is analytic or if it has a fading memory. The analytic requirement for Volterra models was first recognized by Brilliant [11] and later by Sandberg [12]. Because many common nonlinear systems are modeled with non-analytic non-linearities, Boyd [13] reduced this limitation by showing that for non-analytic nonlinear operators with fading memory, a Volterra representation can be obtained for a useful set of bounded input signals. Although port-based Volterra models are not possible for all nonlinear systems, SISO Volterra models are common in engineering literature and have been used for a variety of engineering applications [14],[15] and [16].

The assembled system model is valid if the subsystem models used in the assembly are valid. In the MMM process it is assumed that each provided Volterra subsystem model is valid at a specific input-output operating condition. When these subsystem models are assembled using the MMM procedure, the input-output operating conditions for each subsystem in the assembly do not change, then the resultant assembled model is valid. Component operating conditions do not change because the MMM method provides an explicit method for computing the operating point inputs of the assembly required to operate the components at the desired input-output conditions. This explicit method is explained in the Chapter 3 of this thesis. It is important to clarify that the validity of the Volterra subsystem models is not address in this work. This work only deal with the problem of assembly and distribution of nonlinear models.

CHAPTER 2

Background

2.1 Internet Distribution of Models

The Internet has enabled a global engineering design strategy where collaborative design teams perform irrespective of physical distance. Models of components provided from different sources must be distributed and assembled to generate system models. This section will discuss the functionality of the networked software agents required to implement the model information flow.

The performance of a networked model distribution system (Figure 2.1) relies on the operation of autonomous and flexible computational systems called Agents [17]. Four agent classes are used: Component agents, Assembly agents, the Agent Registry and the Query Ontology. Initially an *User* connected to the Internet, consults to an *Agent Registry* for the locations of *Assembly* and *Component Agents* on the network. The *User* uses standardized queries whose format is obtained from the *Query Ontology* which publishes an organized list of these queries. An assembly agent assembles models by using queries to lower level components. Assembly and Component agents, respond to queries with models in the standard model format specified by the Ontology.

The networked distribution of models is hierarchical. This characteristic is the result of the agent's capabilities to perform either as a client, as a server or both.

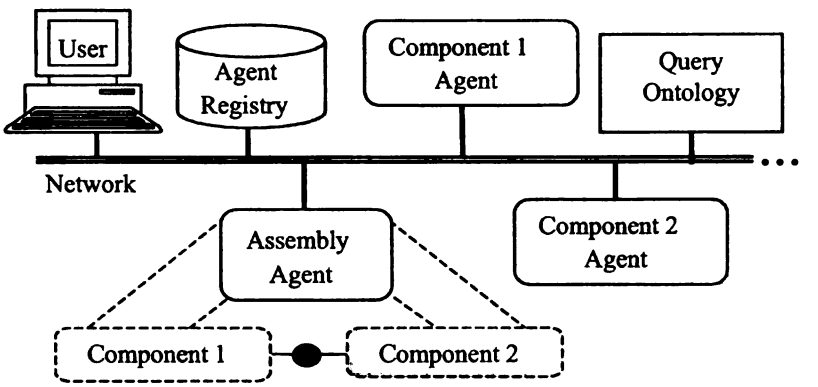

Figure 2.1. Physical View of a Networked Modeling System

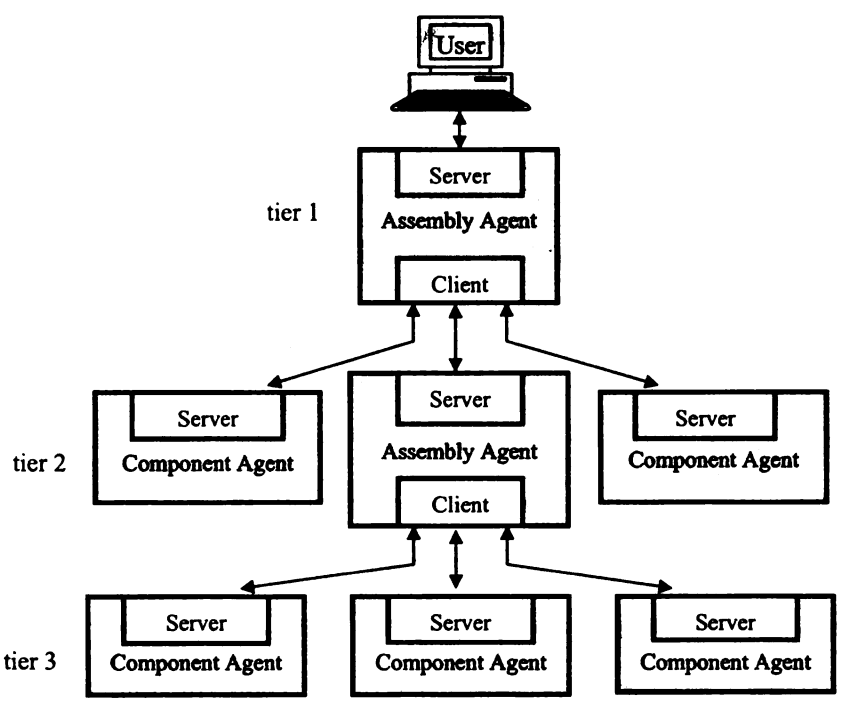

Figure 2.2. Logical Information Flow View of a Networked Engineering System.

The User can perform only as a client, requesting model information from lower-level agents. Assembly agents can perform either as clients requesting model information from lower-level agents or as servers providing model information to higher-level agents. Finally, component agents can perform only as servers, providing model information to higher-level agents.

In a three level network example (Figure 2.2), the User requests model information from the server in the tier 1 assembly agent. The tier 1 client requests model information from servers in tier 2 agents. The tier 2 client requests models from tier 3 component agent servers. Starting at the lower tier in the system, model information is provided by servers to higher-level assembly agents that assemble them into yet higher-level assembly models. These assembled models are then provided to agent servers that respond to queries from above.

2.2 The Standard Linear Physical Model Formats

A physical system is an entity separated from the environment that interchanges energy through a boundary [7]. Physical systems are composed of interacting components that perform in a synchronized way to generate an energy flow. This energy flow is transferred through physical connections consisting of input-output pairs called ports [18]. The product between the input and the output variables of a port defines the energy flow through the port. Positive energy flow through a port is defined as the work done on the system. The total energy flow in a physical system is the sum of all the energy flows through each of its ports.

Physical systems can be modeled using a port-based approach. Port-based models always have an equal number of inputs and outputs because an input-output pair defines each port [7]. Port-based models are considered external models if they are given as functions of external port variables. A valid external port-based model has independent external work ports. This characteristic requires the number of model

equations to be equal to the number of system ports outputs. An external, time-invariant, port-based dynamic linear physical system model having r external ports has r equations in the form

$$\left[\mathbf{N}_0\mathbf{y}(t) + \cdots + \mathbf{N}_i\left(\frac{d^i\mathbf{y}(t)}{dt^i}\right) + \cdots\right] - \left[\mathbf{M}_0\mathbf{y}(t) + \cdots + \mathbf{M}_j\left(\frac{d^j\mathbf{y}(t)}{dt^j}\right) + \cdots\right] \quad (2.1)$$

where, $\mathbf{N}_i$ $\forall(\ 0 \leq i \leq n)$ is a $(r \times r)$ matrix of constants, $\mathbf{M}_j$ $\forall(\ 0 \leq j \leq m)$ is a $(r \times r)$ matrix of constants, $\mathbf{y}$(t) is the $(r \times 1)$ system output vector and $\mathbf{u}(t)$ is the $(r \times 1)$ system input vector. Causal physical system models require $m \leq n$ [19]. Applying the Laplace Transform to the linear ODE (2.1) and factoring common terms,

$$\left[\mathbf{N}_0 s^0 + \cdots + \mathbf{N}_n s^n\right]\mathbf{Y}(s) = \left[\mathbf{M}_0 s^0 + \cdots + \mathbf{M}_m s^m\right]\mathbf{U}(s) \quad (2.2)$$

where $\mathbf{Y}(s)$ and $\mathbf{U}(s)$ are respectively the Laplace transform of the physical system output $\mathbf{y}(t)$ and input $\mathbf{u}(t)$. Defining the two polynomial matrices

$$\begin{aligned}\mathbf{N}(s) &= \left[\mathbf{N}_0 s^0 + \mathbf{N}_1 s + \cdots + \mathbf{N}_n s^n\right] \\ \mathbf{M}(s) &= \left[\mathbf{M}_0 s^0 + \mathbf{M}_1 s + \cdots + \mathbf{M}_m s^m\right]\end{aligned} \quad (2.3)$$

and substituting them into (2.2), yields

$$\mathbf{N}(s)\mathbf{Y}(s) = \mathbf{M}(s)\mathbf{U}(s) \quad (2.4)$$

where $\mathbf{N}(s)$ and $\mathbf{M}(s)$ are $(r \times r)$ matrices of polynomials. Two model representations can be derived from (2.4). These are the dynamic model representation [20] and the transfer function model representation [21].

The dynamic model representation

$$\mathbf{P}(s)\mathbf{Y}(s) = \mathbf{U}(s) \quad (2.5)$$

uses the $(r \times r)$ matrix

$$\mathbf{P}(s) = [\mathbf{M}(s)]^{-1}\mathbf{N}(s) \quad (2.6)$$

This first model representation is a dynamic extension of the traditional format used in dynamic structural analysis [5]. The dynamic matrix $\mathbf{P}$(s) exists only if $\mathbf{M}$(s) is

nonsingular. When $\mathbf{M}$(s) is nonsingular, the effects of the port inputs in the equations are independent. Model representation (2.5) exists even if the matrix $\mathbf{N}$(s) is singular, allowing models where the effects of their outputs in the equations are dependent.

The transfer function model representation

$$\mathbf{Y}(s) = \mathbf{G}(s)\mathbf{U}(s) \tag{2.7}$$

uses the $(r \times r)$ matrix

$$\mathbf{G}(s) = [\mathbf{N}(s)]^{-1}\,\mathbf{M}(s) \tag{2.8}$$

This second representation is the traditional format used in system dynamic analysis [22]. The transfer function matrix $\mathbf{G}$(s) exists only if $\mathbf{N}$(s) is nonsingular. When $\mathbf{N}$(s) is nonsingular, the effects of the outputs in the equations are independent. Model representation (2.7) exists even if the matrix $\mathbf{M}$(s) is singular, allowing models where the effects of their inputs in the equations are dependent.

The linear state space model representation [22],

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \end{aligned} \tag{2.9}$$

is another commonly form used to obtain the external model representation (2.4). A state space model representing a physical system with r ports and p internal states uses a $(r \times 1)$ output vector $\mathbf{y}$(t), a $(r \times 1)$ input vector $\mathbf{u}$(t) and a $(p \times 1)$ state vector $\mathbf{x}$(t). Space state models are internal representations because they are functions of internal state variables. The internal state variables can be removed to obtain an external representation in the form (2.4). The transfer function model representation can be derived from (2.9) by using the well-known equation

$$\mathbf{Y}(s) = \left[\mathbf{C}\,[s\mathbf{I} - \mathbf{A}]^{-1}\,\mathbf{B} + \mathbf{D}\right]\mathbf{U}(s) \tag{2.10}$$

The equivalent model format (2.4), using equation (2.10) is

$$\mathbf{I}|[s\mathbf{I} - \mathbf{A}]| = [\mathbf{C}\;adj(s\mathbf{I} - \mathbf{A})\,\mathbf{B} + |[s\mathbf{I} - \mathbf{A}]|\mathbf{D}]\,\mathbf{U}(s) \tag{2.11}$$

where *adj* is the matrix adjoint operator. Comparing (2.4) and (2.11),

$$\mathbf{N}(s) = |[s\mathbf{I} - \mathbf{A}]| \tag{2.12}$$

$$\mathbf{M}(s) = [\mathbf{C}adj\,[s\mathbf{I} - \mathbf{A}]\,\mathbf{B} + |[s\mathbf{I} - \mathbf{A}]|\mathbf{D}] \tag{2.13}$$

In general, external model representations derived from (2.1) can describe systems with outputs that produce dependent effects on the port equations (the matrix **N**(s) can be singular). In contrast, the external model representations derived from the state space representation (2.9) do not allow this characteristic because the resultant matrix **N**(s) is always nonsingular since $\mathbf{N}(s) = |s\mathbf{I} - \mathbf{A}| \neq 0$.

Port-based physical systems can be modeled by the dynamic model representation (2.5) or by the transfer function model representation (2.7). If the matrices **M**(s) and **N**(s) are nonsingular, both model representations can be obtained either by using the external model (2.4) or by using the state space model (2.9). These two model representations are the standard formats used by the linear MMM for distribution and assembly of models.

In the next section, the two properties that characterized the assembly of physical systems are introduced. These properties are represented mathematically by two constraints equations. These equations are used in the linear and nonlinear MMM assembly method.

2.3 Physical Assembly Constraints

Physical model assembly is characterized by two conditions. The first condition is associated with the word *assembly.* The assembly of two or more system models means their response or outputs variables are equal at the geometric location of the physical assembly. This condition will define the relationship between port output variables. The second condition is associated with the word "physical". All physical system connections must satisfy conservation of energy. The assembly of physical systems

requires both conditions at physical connections: equal system output response and energy conservation.

The connection of physical subsystem models is executed by joining port variables that exchange energy. The output and input pair of each port is determined using the two concepts characterizing the assembly of physical systems. The output variable is selected as the variable physically constrained to be equal to other output variables when ports are connected. The input variable is selected, as the port variable required to compute units of energy when it is multiplied by that port's output. Energy conservation requires all connected ports to be in the same energy domain.

Energy Domain	Output (y)	Input (u)
Electrical	Potential	Charge
Mechanical Translation	Displacement	Force
Mechanical Rotation	Angle	Torque
Hydraulic	Pressure	Volume
Acoustic	Sound Pressure	Volume
Heath Transfer	Temperature	Heat flux

Table 2.1. Input-Output MMM Causality for Different Energy Domains

Input and output port variable pairs for different energy domains are shown in Table 2.1. Connected port outputs measured with respect to the assembly's system reference are equal. Ports assembled in the electric energy domain require equal potentials. Ports assembled in the mechanical energy domain require either equal angular or linear displacement. Ports assembled in the hydraulic or acoustic energy domain require equal pressure. Ports assembled in the heat transfer energy domain require equal temperatures. All these required conditions describe physical assembly in their respective energy domain. The associated port inputs in each energy domain permit the computation of energy. This standard facilitates a the model assembly process. The naturally constrained port variables allow the use of standard physical assembly constraints to assemble models.

Joins are used to enforce these assembly constraints. A join is not a model of

a physical connection subsystem and does not store nor dissipate energy. Joins are mathematical mechanisms that provide the proper physical connection constraints for connecting the ports of physical components within a single energy domain [8]. A join is graphically represented as a circle enclosing the letter J for *Join* (Figure 2.3). The lines represent ports with the direction of positive energy flow indicated by the arrowheads.

Two kinds of control volumes are defined in any assembly. The component control volume (c) interchanges energy with the assembly control volume (a) through the component output $\mathbf{y}_c(t)$ and input $\mathbf{u}_c(t)$ port variables. The assembly control volume interchanges energy with the environment thorough the assembly output $\mathbf{y}_a(t)$ and input $\mathbf{u}_a(t)$ port variables.

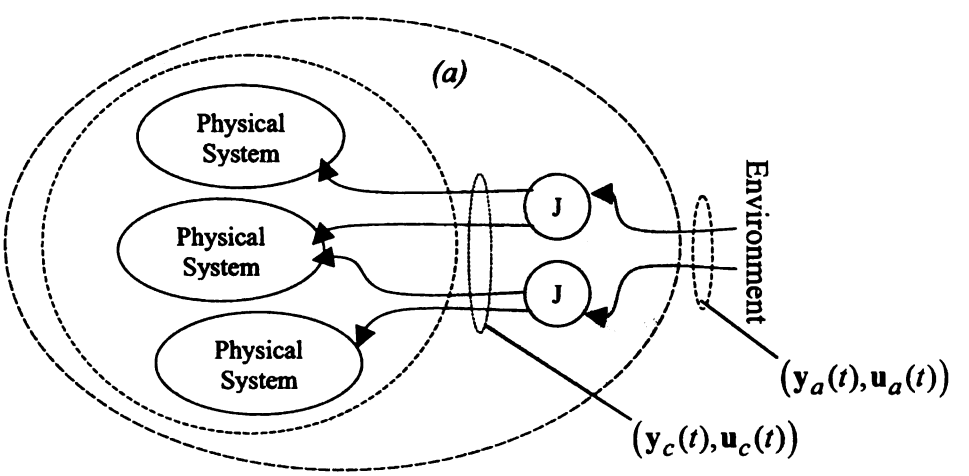

Figure 2.3. Unconstrained and Assembly Control Volumes

A simple relationship determines the number of assembly variables. If a set of components with r port variables uses f connections to connect p component ports, the assembled system has $l = r + f - p$ pairs of port variables. In a practical assembly $f \leq p$ and $l \leq r$. The dimension of the component and assembly port variable vectors are respectively $(r \times 1)$ and $(l \times 1)$.

The two conditions that characterize a physical systems assembly can be repre-

sented using two equations. The first equation constrains connected outputs to be equal. For a join connecting d outputs,

$$y_1(t) = \cdots = y_i(t) = \cdots = y_d(t) = y_a(t) \tag{2.14}$$

where $y_i(t)$ is the i^{th} connected component output at the join and $y_a(t)$ is the output assembly variable at the join. For both joins in (Figure 2.3), $d = 2$. For a set of l joins generating l assembly output variables from r component ports, the relationship (2.14) between the r component outputs and the l assembly outputs is,

$$\mathbf{y}_c(t) = \mathbf{S}\mathbf{y}_a(t) \tag{2.15}$$

where $\mathbf{S}$ is a $(r \times l)$ matrix of constants called the constraint matrix. The second equation, constrains the energy stored at all the connections to be zero when the work done internally on the component ports equals the external work done on the assembly,

$$\mathbf{u}_c^T(t)\mathbf{y}_c(t) = \mathbf{u}_a^T(t)\mathbf{y}_a(t) \tag{2.16}$$

This equation states that the total energy supplied to the component port equals the total energy supplied to the assembly port. The equation (2.16) includes input and output variables and is difficult to use. An equation easier to use relates only component and assembly inputs and can be derived substituting (2.15) into (2.16),

$$\mathbf{u}_c^T(t)\mathbf{S}\mathbf{y}_a(t) = \mathbf{u}_a^T(t)\mathbf{y}_a(t) \tag{2.17}$$

From equation (2.17) it follows that,

$$\mathbf{S}^T\mathbf{u}_c(t) = \mathbf{u}_a(t) \tag{2.18}$$

Applying the Laplace Transform to (2.15) and (2.18),

$$\mathbf{Y}_c(s) = \mathbf{S}\mathbf{Y}_a(s) \tag{2.19a}$$

$$\mathbf{S}^T\mathbf{U}_c(s) = \mathbf{U}_a(s) \tag{2.19b}$$

Equations (2.19a) and (2.19b) are the two constraints used to assemble physical models. This two constraints are satisfied by any assembly, independently if the components are linear or nonlinear.

The assembly of three components is shown (Figure 2.4). These three components have a total of eight ($r = 8$) ports. Component 1 has three ports: **1**,**2** and **3**. Component 2 has 1 port: **4**. Component 3 has four ports: **5**,**6**,**7** and **8**. Five additional ports (**a**, **b**, **c**, **d** and **e**) are defined for each of the five ($f = 5$) joins of the assembly (J_1, J_2, J_3, J_4, and J_5). These joins are used to connect eight ($p = 8$) component ports. The total number of ports for the assembled system is five: ($l = r + f - p = 5$). They are the ports **a**, **b**, **c**, **d** and **e**.

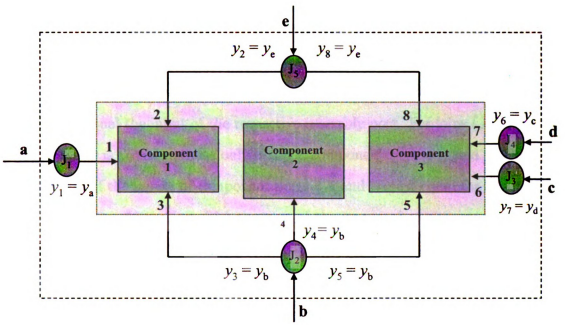

Figure 2.4. Assembly of Three Components Through Five Connections

A constraint is defined for each of the eight connected component port outputs. At the join (J_1), the output of component port **1** is constrained to be equal to the output of assembly port **a**. At the join (J_2), the outputs of component ports **3**, **4** and **5** are constrained to be equal to the output of assembly port **b**. At the join (J_3),

the output of component port **6** is constrained to be equal to the output of assembly port **c**. At the join (J_4), the output of component port **7** is constrained to be equal to the output of assembly port **d**. At the join (J_5), the outputs of component ports **8** and **2** are constrained to be equal to the output of assembly port **e**. The matrix form of the equations that relate component outputs and assembly outputs is,

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_a \\ y_b \\ y_c \\ y_d \\ y_e \end{bmatrix} \tag{2.20a}$$

This equation is an example of (2.15) that defines the constraints between component and assembly outputs through the constraint matrix **S**. A constraint equation is defined for each of the five joins used. At every join the sum of the component port inputs must equal the input of the respective assembly port. For joins (J_1), (J_2) and (J_3) the result is trivial because only a single component port is joined. At the join (J_2) the sum of the inputs of component ports **3**,**4** and **5** must equal the input of the assembly port **b**. At the join (J_5) the sum of the inputs of component ports **2** and **8** must equal the input of the assembly port **e**. The equation that relates component port inputs and assembly port inputs is,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{bmatrix} = \begin{bmatrix} u_a \\ u_b \\ u_c \\ u_d \\ u_e \end{bmatrix} \tag{2.20b}$$

The matrix above equals the transpose of the constraint matrix **S** in (2.20a). The next section shows how the two physical dynamic constraints equations (2.19a) and (2.19b)

are used to assemble external physical system models. Additionally, it is shown that two different model representations, one model representation used for distribution and assembly and the other model representation used for simulation and analysis are required to effectively implement the MMM process.

2.4 The Modular Modeling Assembly Procedure

The Modular Modeling Method (MMM) uses a systematic process to assemble dynamic matrix models (2.5). These standard format models have port pairs standardized through the two concepts that generate constraints (2.19a) and (2.19b). It is shown in this section that the dynamic matrix representation is a convenient form for assembling physical system models from component models. The MMM assembly includes four steps. These steps are: 1) Generation of the unconstrained component model, 2) Generation of the constraint equations, 3) Generation of the assembled model and 4) Generation of the condensed model.

The generation of the unconstrained component model is the first step. This process formulates a matrix of component dynamic models (2.5) in diagonal form. An assembly of k component models with a total number of r ports yields,

$$\mathbf{P}_c(s) = \begin{bmatrix} \mathbf{P}_1(s) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_2(s) & \vdots & \mathbf{0} \\ \vdots & \cdots & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_k(s) \end{bmatrix} \tag{2.21}$$

where $\mathbf{P}_i(s)$, $(1 \leq i \leq k)$ is the $(r_i \times r_i)$ dynamic matrix of the i^{th} component and $\mathbf{P}_c(s)$ is the $(r \times r)$ unconstrained matrix constituted by the dynamic matrices of all the assembly's components. The number of ports in an assembly of k components is,

$$r = \sum_{i=1}^{k} r_i \tag{2.22}$$

The unconstrained component model relates the $(r \times 1)$ component output vector $\mathbf{Y}_c$

to the $(r \times 1)$ component input vector $\mathbf{U}_c$ through the equation,

$$\mathbf{P}_c(s)\mathbf{Y}_c(s) = \mathbf{U}_c(s) \tag{2.23}$$

The generation of dynamic constraint equations in format (2.19a) and (2.19b) is the second step. This process builds the constraint matrix $\mathbf{S}$ that relates unconstrained component variables to constrained assembly variables. The matrix $\mathbf{S}$ is dependent on the connection topology between components. This matrix equates each component output in $\mathbf{Y}_c$ to an assembly output in $\mathbf{Y}_a$ based on the constraints associated with the assembly. An example is shown in the next section.

The generation of the assembled model is the third step. This process is executed by applying output (2.19a) and input (2.19b) constraint equations to (2.23). Initially, (2.19a) is substituted into (2.23) to yield,

$$\mathbf{P}_c(s)\mathbf{S}\mathbf{Y}_a(s) = \mathbf{U}_c(s) \tag{2.24}$$

Multiplying both sides of (2.24) by $\mathbf{S}^T$ and using (2.19b) yields the assembled model,

$$\mathbf{P}_a(s)\mathbf{Y}_a(s) = \mathbf{U}_a(s) \tag{2.25}$$

where

$$\mathbf{P}_a(s) = \mathbf{S}^T\mathbf{P}_c(s)\mathbf{S}$$

is the $(l \times l)$ assembly dynamic matrix and $\mathbf{U}_a(s)$ and $\mathbf{Y}_a(s)$ are the $(l \times 1)$ assembly input and output vectors.

The dynamic model representation (2.5) is particularly effective as the standard assembly MMM format because constraints (2.19a) and (2.19b) can be easily applied. Additionally, the overall process is recursive since the assembled model (2.25) is again in the standard format (2.5). In contrast, the transfer function model (2.7) cannot be used as the assembly MMM format because (2.19a) and (2.19b) cannot be substituted directly. This process requires the $(r \times l)$ left inverse of the matrix $\mathbf{S}$. This matrix

does not exists because in the matrix $\mathbf{S}$ the number of rows is greater than the number of columns. Even though standard format (2.5) is effective for the model assembly process, it is not appropriate for simulations because is not possible to use it to solve directly for the outputs given the inputs. A different model format is required for simulation.

The transfer function model representation (2.7) is chosen as the MMM simulation format because it can directly be used to solve for the outputs given the inputs. This format is obtained by inverting (2.25) to yield,

$$\mathbf{Y}_a(s) = \mathbf{G}_a(s)\mathbf{U}_a(s) \tag{2.26}$$

where

$$\mathbf{G}_a(s) = [\mathbf{P}_a(s)]^{-1}$$

is the transfer function matrix. Model (2.26) is the traditional model format used in the analysis and simulation of dynamic systems [22].

The two different MMM system representations (2.25) and (2.26) are important. The analysis above demonstrates that the MMM modeling format (2.25) is particularly well suited to recursive assembly of models using output constraints but not appropriate for perform model simulation. The analysis above also reveals that the model format (2.26) is particularly well suited for simulation but not appropriate to perform recursive system model assembly. If only one format is used, the advantages of both model representations are not obtained. MMM used the model representation (2.25) for distribution and assembly and the model representation (2.26) for analysis and simulation. This is a clear difference with the traditional methods that use the transfer function model format (2.26) as the unique model representation.

The assembly model (2.25) is formulated in terms of assembly port variables consisting of both internal and external assembly ports. Internal ports are those ports whose inputs are zero during normal use of the model. These zero input ports can

be removed from the model. Additionally, these ports might be removed to protect the proprietary response of the internal model ports. Conversely, external ports must be accessible to the users and should remain in the model. A process to eliminate the internal ports is required. This process is called condensation and the resultant model is only a function of external port variables.

Model condensation is the fourth and final step. This process uses a symmetric transformation matrix $\mathbf{T}$ to reorganize assembly variables into internal and external varaible vectors in the form,

$$\mathbf{T}\begin{bmatrix} \mathbf{Y}_e(s) \\ \mathbf{Y}_i(s) \end{bmatrix} = \mathbf{Y}_a(s) \tag{2.27a}$$

$$\begin{bmatrix} \mathbf{U}_e(s) \\ \mathbf{U}_i(s) \end{bmatrix} = \mathbf{T}^T\mathbf{U}_a(s) \tag{2.27b}$$

where the port input $\mathbf{U}_e(s)$ and output $\mathbf{Y}_e(s)$ are external and the port input $\mathbf{U}_i(s)$ and output $\mathbf{Y}_i(s)$ are internal. To remove internal port variables substitute (2.27a) into (2.27b),

$$\mathbf{P}_a(s)\mathbf{T}\begin{bmatrix} \mathbf{Y}_e(s) \\ \mathbf{Y}_i(s) \end{bmatrix} = \mathbf{U}_a(s) \tag{2.28}$$

Multiplying both sides of (2.28) by $\mathbf{T}^T$ and substituting (2.27b),

$$\mathbf{T}^T\mathbf{P}_a(s)\mathbf{T}\begin{bmatrix} \mathbf{Y}_e(s) \\ \mathbf{Y}_i(s) \end{bmatrix} = \begin{bmatrix} \mathbf{U}_e(s) \\ \mathbf{U}_i(s) \end{bmatrix} \tag{2.29}$$

Defining the matrix,

$$\mathbf{T}^T\mathbf{P}_a(s)\mathbf{T} = \begin{bmatrix} \mathbf{P}_{ee}(s) & \mathbf{P}_{ei}(s) \\ \mathbf{P}_{ie}(s) & \mathbf{P}_{ii}(s) \end{bmatrix} \tag{2.30}$$

yields the form,

$$\begin{bmatrix} \mathbf{P}_{ee}(s) & \mathbf{P}_{ei}(s) \\ \mathbf{P}_{ie}(s) & \mathbf{P}_{ii}(s) \end{bmatrix}\begin{bmatrix} \mathbf{Y}_e(s) \\ \mathbf{Y}_i(s) \end{bmatrix} = \begin{bmatrix} \mathbf{U}_e(s) \\ \mathbf{U}_i(s) \end{bmatrix} \tag{2.31}$$

where $\mathbf{P}_{ee}(s)$, $\mathbf{P}_{ei}(s)$, $\mathbf{P}_{ie}(s)$ and $\mathbf{P}_{ii}(s)$ are dynamic matrices. Because internal assembly inputs will not be accessible in the condensed model, these inputs are set to zero ($\mathbf{U}_i(s) = \mathbf{0}$) to solve for the external inputs as a function of the external outputs in the form,

$$\mathbf{P}_e(s)\mathbf{Y}_e(s) = \mathbf{U}_e(s) \tag{2.32}$$

where,

$$\mathbf{P}_e(s) = \mathbf{P}_{ee}(s) - \mathbf{P}_{ei}(s)\mathbf{P}_{ii}(s)\mathbf{P}_{ie}^{-1}(s) \tag{2.33}$$

is the condensed dynamic matrix. The condensation process is recursive because the condensed model is again in the standard format (2.5). Internal dynamics matrix inverse $\mathbf{P}_{ii}^{-1}$ is always computable (See Appendix A).

2.5 Example of a Linear MMM Assembly

The assembly of a mechanic transmission linear model and an electric generator linear model is developed in this section. The first component is an electric generator with two ports (Figure 2.5). The generator's first port variables are the input electrical charge $q_{1,1}(t)$ and the output voltage potential $e_{1,1}(t)$. The generator's second port variables are the input rotational torque $\tau_{1,2}(t)$ and the output angular displacement $\theta_{1,2}(t)$. The second component (Figure 2.5) is a mechanical transmission with two mechanical ports. The transmission's first port variables are the input rotational torque $\tau_{2,1}(t)$ and the output angular displacement $\theta_{2,1}(t)$ while the transmission's second port variables are input rotational torque $\tau_{2,2}(t)$ and the output angular displacement $\theta_{2,2}(t)$. In total, the two components have $r = 4$ ports.

The assembly is executed thorough three ($f = 3$) joins that connect three ($l = 3$) assembly ports to the four ($r = 4$) component ports. The left joint connects the external electric charge-output potential port pair $(e_{a,1}(t), q_{a,1}(t))$ to the internal port pair $(e_{1,1}(t), q_{1,1}(t))$. The right joint connects the external torque-angular displacement port pair $(\theta_{a,3}(t), \tau_{a,3}(t))$ to the internal port pair $(\theta_{2,2}(t), \tau_{2,2}(t))$. Finally the middle joint connects the internal port pairs $(\theta_{1,2}(t), \tau_{1,2}(t))$ and $(\theta_{2,1}(t), \tau_{2,1}(t))$ to the external torque-angular displacement port pair $(\theta_{a,2}(t), \tau_{a,2}(t))$.

The dynamic model representation for the generator is

$$\begin{bmatrix} P_{1,11}(s) & P_{1,12}(s) \\ P_{1,21}(s) & P_{1,22}(s) \end{bmatrix} \begin{bmatrix} E_{1,1}(s) \\ \Theta_{1,2}(s) \end{bmatrix} = \begin{bmatrix} Q_{1,1}(s) \\ T_{1,2}(s) \end{bmatrix} \tag{2.34}$$

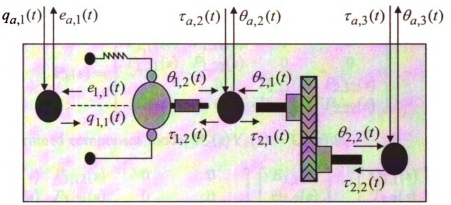

Figure 2.5. Assembly of a Transmission and an Electric Generator

where $E_{1,1}(s)$, $Q_{1,1}(s)$, $\Theta_{1,2}(s)$ and $T_{1,2}(s)$ are respectively the Laplace transform of the time variables $e_{1,1}(t)$, $q_{1,1}(t)$, $\theta_{1,2}(s)$ and $\tau_{1,2}(t)$, the complex variable functions $P_{1,11}(s)$, $P_{1,12}(s)$, $P_{1,21}(s)$ and $P_{1,22}(s)$ are ratio of polynomials respectively representing the relations between the input-output pairs $(Q_{1,1}(s), E_{1,1}(s))$, $(T_{1,2}(s), E_{1,1}(s))$,$(Q_{1,1}(s), \Theta_{1,2}(s))$ and $(T_{1,2}(s), \Theta_{1,2}(s))$.

The transmission dynamic model representation for the is

$$\begin{bmatrix} P_{2,11}(s) & P_{2,12}(s) \\ P_{2,21}(s) & P_{2,22}(s) \end{bmatrix} \begin{bmatrix} \Theta_{2,1}(s) \\ \Theta_{2,2}(s) \end{bmatrix} = \begin{bmatrix} T_{2,1}(s) \\ T_{2,2}(s) \end{bmatrix} \tag{2.35}$$

where $\Theta_{2,1}(s)$, $T_{2,1}(s)$, $\Theta_{2,2}(s)$ and $T_{2,2}(s)$ are respectively the Laplace transform of the time variables $\theta_{2,1}(t)$, $\tau_{2,1}(t)$, $\theta_{2,2}(s)$ and $\tau_{2,2}(t)$, the complex variable functions $P_{2,11}(s)$, $P_{2,12}(s)$, $P_{2,21}(s)$ and $P_{2,22}(s)$ are ratio of polynomials respectively representing the relations between the input-output pairs $(T_{2,1}(s), \Theta_{2,1}(s))$,$(T_{2,2}(s), \Theta_{2,1}(s))$,$(T_{2,1}(s), \Theta_{2,2}(s))$ and $(T_{2,2}(s), \Theta_{2,2}(s))$.

The construction of the unconstrained component model is the first step. The unconstrained matrix is built by inserting in the main diagonal of a square matrix, the dynamic matrices of component models (2.34) and (2.35). The other elements of

the unconstrained matrix are set to zero.

$$\mathbf{P}_c(s) = \begin{bmatrix} P_{1,11}(s) & P_{1,12}(s) & 0 & 0 \\ P_{1,21}(s) & P_{1,22}(s) & 0 & 0 \\ 0 & 0 & P_{2,11}(s) & P_{2,12}(s) \\ 0 & 0 & P_{2,21}(s) & P_{2,22}(s) \end{bmatrix} \tag{2.36}$$

The unconstrained component model $\mathbf{P}_c(s)\mathbf{Y}_c(s) = \mathbf{U}_c(s)$ is,

$$\begin{bmatrix} P_{1,11}(s) & P_{1,12}(s) & 0 & 0 \\ P_{1,21}(s) & P_{1,22}(s) & 0 & 0 \\ 0 & 0 & P_{2,11}(s) & P_{2,12}(s) \\ 0 & 0 & P_{2,21}(s) & P_{2,22}(s) \end{bmatrix} \begin{bmatrix} E_{1,1}(s) \\ \Theta_{1,2}(s) \\ \Theta_{2,1}(s) \\ \Theta_{2,2}(s) \end{bmatrix} = \begin{bmatrix} Q_{1,1}(s) \\ T_{1,2}(s) \\ T_{2,1}(s) \\ T_{2,2}(s) \end{bmatrix} \tag{2.37}$$

The generation of the constraint equations is the second step. Initially the time variable equations that relate component output to assembly outputs are written.

$$\begin{bmatrix} e_{1,1}(t) \\ \theta_{1,2}(t) \\ \theta_{2,1}(t) \\ \theta_{2,2}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_{a,1}(t) \\ \theta_{a,2}(t) \\ \theta_{a,3}(t) \end{bmatrix} \tag{2.38a}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} q_{a,1}(t) \\ \tau_{a,2}(t) \\ \tau_{a,3}(t) \end{bmatrix} = \begin{bmatrix} q_{1,1}(t) \\ \tau_{1,2}(t) \\ \tau_{2,1}(t) \\ \tau_{2,2}(t) \end{bmatrix} \tag{2.38b}$$

Applying the Laplace Transform to (2.38a) and (2.38a),

$$\begin{bmatrix} E_{1,1}(s) \\ \Theta_{1,2}(s) \\ \Theta_{2,1}(s) \\ \Theta_{2,2}(s) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} E_{a,1}(s) \\ \Theta_{a,2}(s) \\ \Theta_{a,3}(s) \end{bmatrix} \tag{2.39a}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} Q_{a,1}(s) \\ T_{a,2}(s) \\ T_{a,3}(s) \end{bmatrix} = \begin{bmatrix} Q_{1,1}(s) \\ T_{1,2}(s) \\ T_{2,1}(s) \\ T_{2,2}(s) \end{bmatrix} \tag{2.39b}$$

The constraint matrix is

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.40}$$

The assembly model generation is the third step. Using (2.39a) and (2.39b) to generate the assembly's dynamic matrix $\mathbf{P}_a(s) = \mathbf{S}^T\mathbf{P}_c\mathbf{S}$,

$$\begin{bmatrix} P_{1,11}(s) & P_{1,12}(s) & 0 \\ P_{1,21}(s) & P_{1,22}(s)+P_{2,11}(s) & P_{2,12}(s) \\ 0 & P_{2,21}(s) & P_{2,22}(s) \end{bmatrix} \begin{bmatrix} E_{a,1}(s) \\ \Theta_{a,2}(s) \\ \Theta_{a,3}(s) \end{bmatrix} = \begin{bmatrix} Q_{a,1} \\ T_{a,2} \\ T_{a,3} \end{bmatrix} \tag{2.41}$$

Equation (2.41) is the dynamic model of the generator-transmission assembly. This model in the standard format (2.5). At this point, any port of (2.41) can be selected as internal or external. Internal ports make internal variables unavailable to the user. Because internal ports are not accessible, internal inputs are set to zero. The condensed model is determined assuming that the internal variables are $\Theta_{a,2}(s)$ and $T_{a,2}(s)$ and the external variables are $E_{a,1}(s)$, $Q_{a,1}(s)$, $\Theta_{a,3}(s)$ and $T_{a,3}(s)$.

The condensed model generation is the fourth step and uses the matrix,

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.42}$$

to reorganize the assembly variables into external and internal vector variables,

$$\mathbf{T} \begin{bmatrix} E_{a,1}(s) \\ \Theta_{a,3}(s) \\ \Theta_{a,2}(s) \end{bmatrix} = \begin{bmatrix} E_{a,1} \\ \Theta_{a,2} \\ \Theta_{a,3} \end{bmatrix} \tag{2.43a}$$

$$\begin{bmatrix} Q_{1,1}(s) \\ T_{a,3}(s) \\ T_{a,2}(s) \end{bmatrix} = \mathbf{T}^T \begin{bmatrix} Q_{1,1} \\ T_{a,2} \\ T_{a,3} \end{bmatrix} \tag{2.43b}$$

Using the definition in (2.30), (2.31), (2.32) and (2.33) yields the condensed model,

$$\begin{bmatrix} P_{ee}(s) & P_{ei}(s) \\ P_{ie}(s) & P_{ii}(s) \end{bmatrix} \begin{bmatrix} E_{a,1}(s) \\ \Theta_{a,3}(s) \end{bmatrix} = \begin{bmatrix} Q_{1,1}(s) \\ T_{a,3}(s) \end{bmatrix} \tag{2.44}$$

where

$$P_{ee}(s) = P_{1,11}(s) - \frac{P_{1,12}(s)P_{1,21}(s)}{P_{1,22}(s)+P_{2,11}(s)}$$

$$P_{ei}(s) = -\frac{P_{1,12}(s)P_{2,12}(s)}{P_{1,22}(s)+P_{2,11}(s)}$$

$$P_{ie}(s) = \frac{P_{2,21}(s)P_{1,21}(s)}{P_{1,22}(s)+P_{2,11}(s)}$$

$$P_{ii}(s) = P_{2,22}(s) - \frac{P_{2,21}(s)P_{2,12}(s)}{P_{1,22}(s)+P_{2,11}(s)}$$

The condensed model is also in the standard format (2.5) and can be assembled to other standard models using the same algorithm. The condensation process is again recursive.

The analysis and simulation of the generator-transmission system assembly, requires the simulation format (2.26). This model is obtained by inverting the condensed model (2.44) to yield,

$$\begin{bmatrix} E_{a,1}(s) \\ \Theta_{a,3}(s) \end{bmatrix} = \begin{bmatrix} G_{ee}(s) & G_{ei}(s) \\ G_{ie}(s) & G_{ii}(s) \end{bmatrix} \begin{bmatrix} Q_{1,1}(s) \\ T_{a,3}(s) \end{bmatrix} \tag{2.45}$$

where,

$$G_{ee}(s) = \left[P_{2,22}(s)P_{1,22}(s) + P_{2,22}(s)P_{2,11}(s) - P_{2,21}(s)P_{2,12}(s)\right]/D(s)$$

$$G_{ei}(s) = \left[P_{1,12}(s)P_{2,12}(s)\right]/D(s)$$

$$G_{ie}(s) = \left[P_{2,21}(s)P_{1,21}(s)\right]/D(s)$$

$$G_{ii}(s) = \left[\left(P_{1,22}(s) + P_{2,11}(s)\right)P_{1,11}(s) - P_{1,12}(s)P_{1,21}(s)\right]/D(s)$$

$$D(s) = G_{ee}(s)P_{1,11} - P_{1,12}P_{1,21}P_{2,22}$$

2.6 Summary

In this chapter, the recognition that two model formats are necessary for physical system model assembly and simulation is addressed for the first time. The use of two model formats decreases the computational time required to assemble and to simulate physical system models. These two model formats are the transfer function representation and the dynamic representation. The transfer function representation is the traditional format used for simulation. This thesis shows, that in a constraint-based model assembly, the Transfer Function representation is not appropriate for MMM assembly because a matrix inversion is required. Matrix inversion demands substantial computational time, a clear disadvantage in a global distributed environment. Additionally, for this inverse to exist, system models are limited to have independent inputs and independent outputs. This is a big limitation on allowable models that

excludes many dynamic system models. In contrast, if the format used for assembly is the dynamic representation, no matrix inversion is required and system models with dependent inputs and dependent outputs can also be assembled.

This chapter also introduces the process of model condensation. Model condensation is the removal of ports that are considered interior to the model. Removing those degrees of freedom reduces the overall size of the model, and additionally, protects the proprietary information of the model. Condensation of models significantly increases the difficulty associated with reverse engineering a model. Invariably the number of internal model parameters exceeds the number of inputs/output pairs making reverse engineering difficult. In the next chapters the MMM process for assembly physical nonlinear models is presented.

CHAPTER 3

Assembly of Affine Physical System Models

3.1 Affine Physical System Models

Affine systems are important in the MMM procedure because the assembly of general differentiable nonlinear physical models performing at a constant operating point can be executed as the assembly of its affine approximations. In an affine system the inputs and outputs exhibit a proportional relationship, but the model outputs are nonzero at zero input. Affine systems are nonlinear because they do not obey the two linear system properties: superposition and homogeneity [23]. Affine systems can be static or dynamic. A static affine system is characterized by an input-output relationship that is independent of the system's input and output time derivatives. Static affine systems are discussed first, then, dynamic affine systems are presented.

A static affine system having p outputs and q inputs is,

$$\mathbf{y} = \mathbf{K}\mathbf{u} + \mathbf{c} \tag{3.1}$$

where $\mathbf{y}$ is a $(p \times 1)$ vector that represents the system's outputs, $\mathbf{u}$ is a $(q \times 1)$ vector that represents the system's inputs, $\mathbf{K}$ is a $(p \times q)$ matrix of constants and $\mathbf{c}$ is a $(p \times 1)$ vector of constants called the bias vector. The static affine model (3.1) is non-linear because it does not obey superposition and homogeneity. Superposition is

not satisfied by (3.1) because given two arbitrary inputs $\mathbf{u}_1$ and $\mathbf{u}_2$ the relation:

$$\mathbf{y}(\mathbf{u}_1 + \mathbf{u}_2) = \mathbf{y}(\mathbf{u}_1) + \mathbf{y}(\mathbf{u}_2) \tag{3.2}$$

does not hold. Homogeneity is not satisfied by (3.1) because given an arbitrary constant λ and any input $\mathbf{u}$, the relation

$$\mathbf{y}(\lambda\mathbf{u}) = \lambda\mathbf{y}(\mathbf{u}) \tag{3.3}$$

does not hold.

Two examples of physical affine systems are an electrical battery model with non-zero open circuit potential (Figure 3.1) and a mechanical system with fixed load (Figure 3.2). Both systems have input-output models with non-zero output at zero input. The static affine electric battery model (Figure 3.1)

$$e = -Ri + V \tag{3.4}$$

has output potential e, input load current i and battery internal resistance R. Here the bias constant V, is the battery potential at zero current.

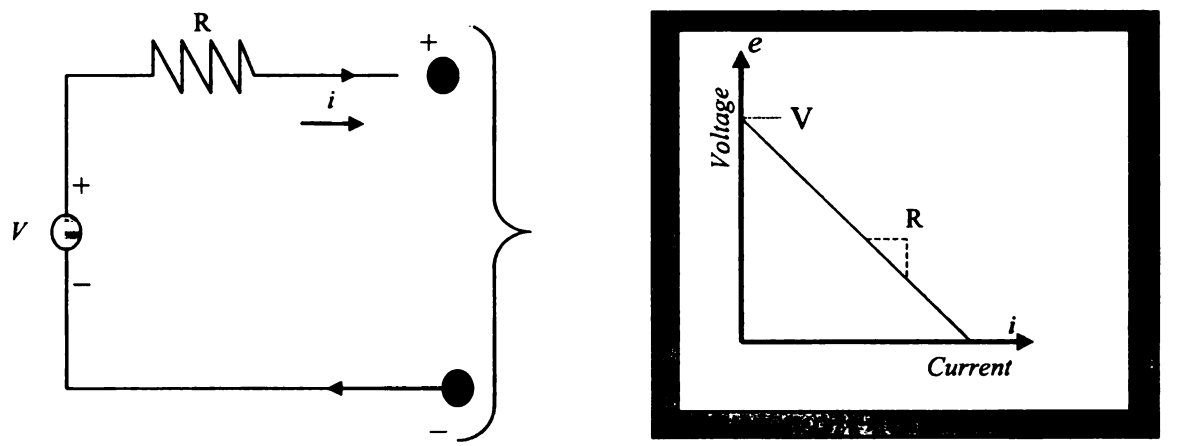

Figure 3.1. Electric Battery

The dynamic affine pulley-load assembly model (Figure 3.2) is described by the nonlinear ordinary differential equation

$$J\ddot{\theta} + k\theta = \tau + WR \tag{3.5}$$

with output angular displacement θ, input torque τ, spring parameter k , mass moment of inertia J, and bias constant WR. The bias constant depends on both the pulley radius R and the load W. At static equilibrium model (3.5) becomes,

$$\theta_{eq} = (1/k)\tau + (WR)/k \tag{3.6}$$

where, θ_{eq} is the angular displacement at equilibrium for a given constant torque τ. The static model (3.6) does not include the point $(\theta_{eq}, \tau) = (0, 0)$. Like the previous example, the static (3.6) and the dynamic (3.5) models are nonlinear because they do not obey superposition and homogeneity.

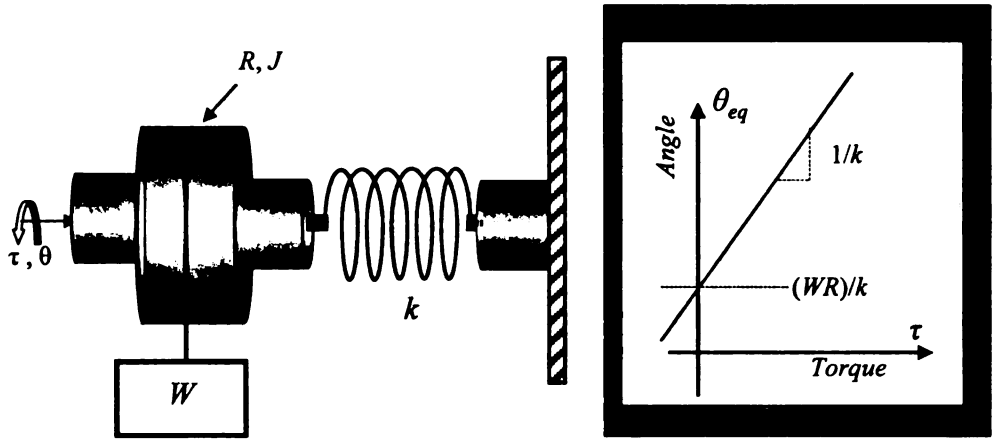

Figure 3.2. Pulley-Load Assembly

An affine system is an intermediate result of any linearization process. A linearization (Figure. 3.3), usually generates a linear model with respect to the operational variables but an affine model with respect to the real physical variables. Physical variables are most appropriate for use in a model's assembly process because when models are connected to form a system model, their behaviors are constrained [6] and these constraints are in function of physical variables.

An affine approximation of the external nonlinear model,

$$\mathbf{f}(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, \cdots, \mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, \cdots) = \mathbf{0} \tag{3.7}$$

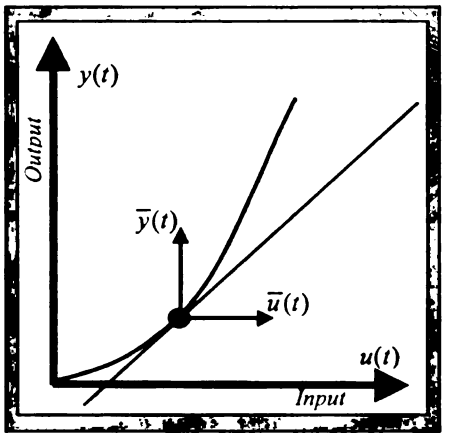

Figure 3.3. Linearization Yielding and Affine System

where $\mathbf{f}(\bullet)$ is a $(r \times 1)$ vector of functions, $\mathbf{y}(t)$ is the $(p \times 1)$ system physical output vector and $\mathbf{u}(t)$ is the $(q \times 1)$ system physical input vector, can be obtained through a Taylor expansion in five steps.

In the first step, all the input and output derivatives of (3.7) are set to zero,

$$\mathbf{f}^*(\mathbf{y}, \mathbf{u}) = \mathbf{f}(\mathbf{y}, \mathbf{0}, \cdots, \mathbf{u}, \mathbf{0}, \cdots) = \mathbf{0} \tag{3.8}$$

where for any $\mathbf{y}$ on the map, at least one $\mathbf{u}$ exists.

In the second step a point $(\mathbf{y}, \mathbf{u})$ on the map (3.8) is selected. Traditionally, a desired value of the output $\mathbf{y} = \mathbf{y}_0$ is defined first to determine one input $\mathbf{u} = \mathbf{u}_0$ that satisfies,

$$\mathbf{f}^*(\mathbf{y}_0, \mathbf{u}_0) = \mathbf{0} \tag{3.9}$$

In the third step, the equilibrium point

$$\begin{aligned} \mathbf{u} = \mathbf{u}_0,\ \dot{\mathbf{u}}_0 = \ddot{\mathbf{u}}_0 = \cdots = \mathbf{0} \\ \mathbf{y} = \mathbf{y}_0,\ \dot{\mathbf{y}}_0 = \ddot{\mathbf{y}}_0 = \cdots = \mathbf{0} \end{aligned} \tag{3.10}$$

is selected as the system's operating point (op). Applying Taylor series expansion to equation (3.7) around the operating point (3.10)

$$\begin{aligned} \mathbf{f}(\mathbf{y}, \mathbf{u}, \cdots) = \mathbf{f}(\mathbf{y}_0, \mathbf{u}_0, \cdots) + \mathbf{D}_{\mathbf{y}}(\mathbf{f})|_{op}(\mathbf{y} - \mathbf{y}_0) + \tfrac{1}{2!}\mathbf{D}^2_{\mathbf{y}}(\mathbf{f})|_{op}(\mathbf{y} - \mathbf{y}_0)^2 \\ + \cdots + \mathbf{D}_{\mathbf{u}}(\mathbf{f})|_{op}(\mathbf{u} - \mathbf{u}_0) + \tfrac{1}{2!}\mathbf{D}^2_{\mathbf{u}}(\mathbf{f})|_{op}(\mathbf{u} - \mathbf{u}_0)^2 + \cdots = \mathbf{0} \end{aligned} \tag{3.11}$$

where each element of the matrix $\mathbf{D}^k_{\mathbf{g}}(\mathbf{h}(\mathbf{g},\dot{\mathbf{g}})),\cdots))$ is the the k^{th} partial derivative of the vector function $\mathbf{h}$ respect the vector $\mathbf{g}$. Since (3.10) is a solution of (3.7) the first right side term of (3.11) is,

$$\mathbf{f}\left(\mathbf{y}_0,\mathbf{u}_0,\mathbf{0},\cdots\right)=\mathbf{0} \tag{3.12}$$

In the fourth step, equation (3.10) and (3.12) are substituted into (3.11) and retaining only the 1^{st} partial derivative terms,

$$\begin{aligned}\mathbf{f}(\mathbf{y},\mathbf{u},\cdots)\cong\ &\mathbf{D}_{\mathbf{y}}(\mathbf{f})|_{op}(\mathbf{y}-\mathbf{y}_0)+\mathbf{D}_{\dot{\mathbf{y}}}(\mathbf{f})|_{op}\dot{\mathbf{y}}+\mathbf{D}_{\ddot{\mathbf{y}}}(\mathbf{f})|_{op}\ddot{\mathbf{y}}+\cdots\\ &+\mathbf{D}_{\mathbf{u}}(\mathbf{f})|_{op}(\mathbf{u}-\mathbf{u}_0)+\mathbf{D}_{\dot{\mathbf{u}}}(\mathbf{f})|_{op}\dot{\mathbf{u}}+\mathbf{D}_{\ddot{\mathbf{u}}}(\mathbf{f})|_{op}\ddot{\mathbf{u}}+\cdots\end{aligned} \tag{3.13}$$

Equation (3.13) can be reorganized in the form

$$\begin{aligned}\mathbf{f}(\mathbf{y},\mathbf{u},\cdots,\dot{\mathbf{y}},\dot{\mathbf{u}})\cong\ &\mathbf{D}_{\mathbf{y}}(\mathbf{f})|_{op}\mathbf{y}+\mathbf{D}_{\dot{\mathbf{y}}}(\mathbf{f})|_{op}\dot{\mathbf{y}}+\mathbf{D}_{\ddot{\mathbf{y}}}(\mathbf{f})|_{op}\ddot{\mathbf{y}}+\cdots\\ &+\mathbf{D}_{\mathbf{u}}(\mathbf{f})|_{op}\mathbf{u}+\mathbf{D}_{\dot{\mathbf{u}}}(\mathbf{f})|_{op}\dot{\mathbf{u}}+\mathbf{D}_{\ddot{\mathbf{u}}}(\mathbf{f})|_{op}\ddot{\mathbf{u}}+\cdots\\ &-\mathbf{D}_{\mathbf{y}}(\mathbf{f})|_{op}\mathbf{y}_0-\mathbf{D}_{\mathbf{u}}(\mathbf{f})|_{op}\mathbf{u}_0\end{aligned} \tag{3.14}$$

All the partial derivatives evaluated at the operating point (op) in (3.14), yield constants matrices. The fifth and final step is to define:

1. n $(r\times p)$ matrices of constants $(1\leq i\leq n)$ $\mathbf{N}_i=\left[\partial\mathbf{f}/\partial(d^i\mathbf{y}/dt^i)\right]|_{op}$

2. m $(r\times q)$ matrices of constants $(1\leq i\leq m)$ $\mathbf{M}_i=\left[\partial\mathbf{f}/\partial(d^i\mathbf{u}/dt^i)\right]|_{op}$

where n and m are respectively the highest derivative of the output and the input of equation (3.7).

3. A $(r\times 1)$ vector of bias constants $\mathbf{c}=-\left[\partial\mathbf{f}/\partial\mathbf{y}\right]|_{op}\mathbf{y}_0-\left[\partial\mathbf{f}/\partial\mathbf{u}\right]|_{op}\mathbf{u}_0$

to write (3.14) in the form,

$$\begin{aligned}\mathbf{f}(\mathbf{y},\mathbf{u},\cdots,\dot{\mathbf{y}},\dot{\mathbf{u}})\cong\ &\mathbf{N}_0\mathbf{y}+\mathbf{N}_1\dot{\mathbf{y}}+\mathbf{N}_2\ddot{\mathbf{y}}+\cdots+\\ &\mathbf{M}_0\mathbf{u}+\mathbf{M}_1\dot{\mathbf{U}}+\mathbf{M}_2\ddot{\mathbf{U}}+\cdots+\mathbf{c}\end{aligned} \tag{3.15}$$

Equation (3.15) evaluated in its physical variables is affine if the vector $\mathbf{c}$ does not vanish. The vector $\mathbf{c}$ will vanish if,

$$\left[\partial\mathbf{f}/\partial\mathbf{y}\right]|_{op}\mathbf{y}_0+\left[\partial\mathbf{f}/\partial\mathbf{u}\right]|_{op}\mathbf{u}_0=\mathbf{0} \tag{3.16}$$

Operating points that satisfy condition (3.16) can be approximated in its physical coordinates by a linear model and are a special case. This fact would imply either

that the operating point of the assembly is the origin $(\mathbf{y}_0, \mathbf{u}_0) = (\mathbf{0}, \mathbf{0})$, what in most cases is not practical, or that even though the operating point is not the origin, the linearized system intersects the origin. Any differentiable nonlinear system that does not satisfy condition (3.16) can be locally approximated in its physical variables by an affine model (3.15).

An affine model describing a physical system with r external ports, requires r equations in the form,

$$\mathbf{N}_0\mathbf{y} + \mathbf{N}_1\dot{\mathbf{y}} + \mathbf{N}_2\ddot{\mathbf{y}} + \cdots + \mathbf{M}_0\mathbf{u} + \mathbf{M}_1\dot{\mathbf{u}} + \mathbf{M}_2\ddot{\mathbf{u}} + \cdots + \mathbf{c} = \mathbf{0} \tag{3.17}$$

where respectively $\forall(0 \leq i \leq n)$ and $\forall(0 \leq j \leq m)$, $\mathbf{N}_i$ and $\mathbf{M}_j$ are $(r \times r)$ matrices of constants and $\mathbf{y}(t)$, $\mathbf{u}(t)$ and $\mathbf{c}$ are $(r \times 1)$ vectors. Physical systems are causal requiring [19]. The bias vector,

$$\mathbf{c} = -\mathbf{N}_0\mathbf{y}_0 - \mathbf{M}_0\mathbf{u}_0 \tag{3.18}$$

substituted into (3.17) yields,

$$\mathbf{N}_0\mathbf{y} + \mathbf{N}_1\dot{\mathbf{y}} + \mathbf{N}_2\ddot{\mathbf{y}} + \cdots + \mathbf{M}_0\mathbf{u} + \mathbf{M}_1\dot{\mathbf{u}} + \mathbf{M}_2\ddot{\mathbf{u}} + \cdots - \mathbf{N}_0\mathbf{y}_0 - \mathbf{M}_0\mathbf{u}_0 = \mathbf{0} \tag{3.19}$$

A procedure to assemble affine physical models in their physical variables is proposed in this chapter. The procedure satisfies the four characteristics required by a global engineering strategy and applies to any differentiable nonlinear physical model. The chapter is organized as follows: Affine systems were defined in the first section. The networked distribution process for affine models is explained in the second section. The standard model format is described in the third section. The constraints required to assemble affine physical models are defined in the fourth section. The MMM algorithm to assemble affine physical models is shown in the fifth section. An example is shown in the sixth section. Finally conclusions are provided.

3.2 Internet Distribution of Affine Models

The networked distribution of affine models requires a two-part query-response format. In the first part, agent clients request from lower-level server agents, models of systems valid around specific output operating conditions. In the second part, server agents provide the models and the input operating conditions required to operate the system at the desired outputs.

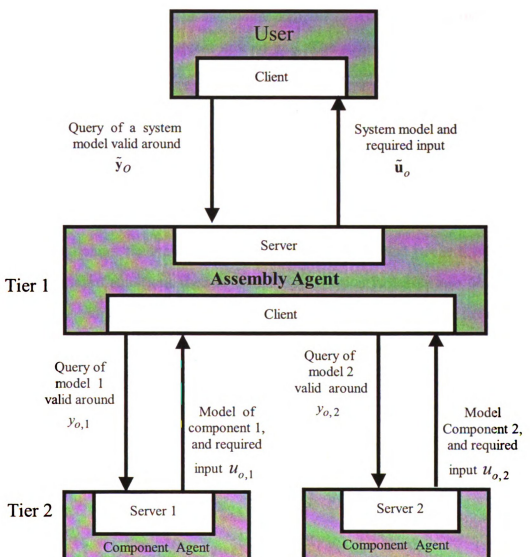

Figure 3.4. Query-Response Model Distribution Format

In a two-level network example (Figure 3.4), the User client makes a request to the Tier 1's server agent for a model of a system valid around the desired output operating conditions $\tilde{\mathbf{y}}_0$. The assembly agent determines that two model components, the first operating around the output $y_{0,1}$ and the second operating around $y_{0,2}$, are required. The Tier 1's client requests a model to each of the two component agents. Two responses are generated. In the first, the server 1 returns the model component 1 and the required input $u_{0,1}$ to operate the first component around $y_{0,1}$. In the second, server 2 returns the model component 2 and the required input $u_{0,2}$ to operate this second component around $y_{0,2}$. The assembly agent uses component model information to execute the assembly. The assembly agent also returns to the user client, the system model and the required $\tilde{\mathbf{u}}_0$ inputs to operate the system around the desired outputs $\tilde{\mathbf{y}}_0$.

3.3 The Standard Model Formats

This section defines the standard dynamic model formats used in the MMM process and the standard equations used to transform deviation variables into physical variables. In the proposed networked environment, component models are distributed as linear models defined in deviation variables but assembled as affine models defined in physical variables. Initially the linear format is presented, then the required variable transform equations are defined.

The standard MMM model format can be obtained by substituting the change of variables

$$\bar{\mathbf{y}} = \mathbf{y}(t) - \mathbf{y}_0, \qquad \frac{d^i \bar{\mathbf{y}}}{dt^i} = \frac{d^i \mathbf{y}}{dt^i}, \ \forall (1 \leq i \leq n) \tag{3.20}$$

$$\bar{\mathbf{u}} = \mathbf{u}(t) - \mathbf{u}_0, \qquad \frac{d^j \bar{\mathbf{u}}}{dt^j} = \frac{d^j \mathbf{u}}{dt^j}, \ \forall (1 \leq j \leq m) \tag{3.21}$$

into the affine model (3.19). This process yields a linear differential equation in the

deviation variables $\bar{\mathbf{y}}(t)$ and $\bar{\mathbf{u}}(t)$,

$$\mathbf{N}_0\bar{\mathbf{y}} + \cdots + \mathbf{N}_n\left(d^n\bar{\mathbf{y}}/dt^n\right) + \mathbf{M}_0\bar{\mathbf{u}} + \cdots + \mathbf{M}_m\left(d^m\bar{\mathbf{u}}/dt^m\right) = \mathbf{0} \tag{3.22}$$

Applying the Laplace Transform to (3.22) yields,

$$\left[\mathbf{N}_0 s^0 + \cdots + \mathbf{N}_n s^n\right]\bar{\mathbf{Y}}(s) + \left[\mathbf{M}_0 s^0 + \cdots + \mathbf{M}_n s^m\right]\bar{\mathbf{U}}(s) = \mathbf{0} \tag{3.23}$$

where $\bar{\mathbf{Y}}(s)$ and $\bar{\mathbf{U}}(s)$ are the Laplace transform of the deviation variables $\bar{\mathbf{y}}(t)$ and $\bar{\mathbf{u}}(t)$. Defining the two polynomial matrices,

$$\mathbf{N}(s) = \left[\mathbf{N}_0 s^0 + \cdots + \mathbf{N}_n s^n\right] \tag{3.24a}$$

$$\mathbf{M}(s) = -\left[\mathbf{M}_0 s^0 + \cdots + \mathbf{M}_n s^m\right] \tag{3.24b}$$

and substituting them into (3.23), yields the general form

$$\mathbf{N}(s)\bar{\mathbf{Y}}(s) = \mathbf{M}(s)\bar{\mathbf{U}}(s) \tag{3.25}$$

where $\mathbf{N}$(s) and $\mathbf{M}$(s) are $(r \times r)$ matrices.

Two external model representations are derived from (3.25). These are the dynamic stiffness [20] and the Transfer Function representation [21]. The Dynamic representation

$$\mathbf{P}(s)\bar{\mathbf{Y}}(s) = \bar{\mathbf{U}}(s) \tag{3.26}$$

uses the $(r \times r)$ matrix

$$\mathbf{P}(s) = [\mathbf{M}(s)]^{-1}\mathbf{N}(s) \tag{3.27}$$

The Dynamic representation

$$\bar{\mathbf{Y}}(s) = \mathbf{G}(s)\bar{\mathbf{U}}(s) \tag{3.28}$$

uses the $(r \times r)$ matrix

$$\mathbf{G}(s) = [\mathbf{N}(s)]^{-1}\mathbf{M}(s) \tag{3.29}$$

Affine model representations (3.26) and (3.28) are respectively analogous to linear model representations (2.5) and (2.7). As it was shown in the last chapter for the linear assembly process, model representation (3.26) is a convenient format for assembly affine physical system models but it is not appropriate for simulation since is not possible to solve directly for the outputs. Model representation (3.28) is used for simulations of affine models because it can be used to compute the output given the inputs.

The required change of variables is obtained by defining the unit step $u(t)$ and substituting the equivalent time functions $\mathbf{y}_0^* = \mathbf{y}_o u(t)$ and $\mathbf{u}_0^* = \mathbf{u}_o u(t)$ into (3.20) and (3.21),

$$\bar{\mathbf{y}}(t) = \mathbf{y}(t) - \mathbf{y}_0^*(t) \tag{3.30a}$$

$$\bar{\mathbf{u}}(t) = \mathbf{u}(t) - \mathbf{u}_0^*(t) \tag{3.30b}$$

The Laplace Transform of (3.30a) and (3.30b) is,

$$\bar{\mathbf{Y}}(s) = \mathbf{Y}(s) - \mathbf{y}_0(1/s) \tag{3.31a}$$

$$\bar{\mathbf{U}}(s) = \mathbf{U}(s) - \mathbf{u}_0(1/s) \tag{3.31b}$$

where $\mathbf{y}_0$ and $\mathbf{u}_0$ are respectively the output and input operating vectors of the component, $\bar{\mathbf{Y}}(s)$ and $\bar{\mathbf{U}}(s)$ are the Laplace transform of the output and input deviation variables, and $\mathbf{Y}$(s) and $\mathbf{U}$(s) are the Laplace transform of the output and input physical variables. Equations (3.31a) and (3.31b) are the component standard equations used to change deviation variables into physical variables.

3.4 Physical Assembly Constraints

Affine component models are specified with respect to the output and the input component operating points $\mathbf{y}_0$ and $\mathbf{u}_0$. The resultant assembled affine model is specified with respect to the output and the input assembly operating point vectors

$\tilde{\mathbf{y}}_0$ and $\tilde{\mathbf{u}}_0$. Since component operating point vectors $\mathbf{y}_0$ and $\mathbf{u}_0$ are a subset of the component variables $\mathbf{y}_c$ and $\mathbf{u}_c$ and the assembly operating point vectors $\tilde{\mathbf{y}}_0$ and $\tilde{\mathbf{u}}_0$ are a subset of the assembly variables $\mathbf{y}_a$ and $\mathbf{u}_a$, equations (2.15) and (2.18) are also satisfied by the component and assembly operating points

$$\mathbf{y}_0 = \mathbf{S}\tilde{\mathbf{y}}_0 \tag{3.32a}$$

$$\mathbf{S}^T \mathbf{u}_0 = \tilde{\mathbf{u}}_0 \tag{3.32b}$$

where $\mathbf{y}_0$ and $\mathbf{u}_0$ are $(r \times 1)$ and $\tilde{\mathbf{y}}_0$ and $\tilde{\mathbf{u}}_0$ are $(l \times 1)$. Equations (3.32a) and (3.32b) provide a method of computing the output operating points of the components from the output operating points of the assembly.

The process of determining system's operating point $(\tilde{\mathbf{y}}_0, \tilde{\mathbf{u}}_0)$ follows a standard network procedure (Figure 3.4). The client specifies the operating point output $\tilde{\mathbf{y}}_0$ in a request to the assembly agent for a model. The assembly agent knows the assembly constraints $\mathbf{S}$ and uses (3.32a) to compute the operating point outputs $\tilde{\mathbf{y}}_0$ for all the assembly's components. These component outputs are sent to component agents as part of a request to each component for a model. At every level, each component responds to a model request with that component's linearized model and that component's operating point inputs $\mathbf{u}_0$. The assembly agent determines the assembly's operating points inputs $\tilde{\mathbf{u}}_0$ using the connection constraint $\mathbf{S}$ and (3.32b).

3.5 Assembly of Affine Physical Models

The MMM uses a systematic process to assemble dynamic matrix-based models. These models have port pairs standardized through the two concepts that characterized physical systems. The assembly process for affine models includes three steps. These steps are: 1) Generation of the unconstrained component model, 2) Generation of constraint equations, and 3) Generation of the assembly model.

The unconstrained component model is the first step. This process formulates a diagonal matrix of component dynamic matrices. An assembly of k component models with a total number of r ports yields,

$$\mathbf{P}_c(s) = \begin{bmatrix} \mathbf{P}_1(s) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \ddots & \mathbf{0} & \mathbf{P}_k(s) \end{bmatrix} \tag{3.33}$$

where $\mathbf{P}_i(s)$ is the $(r_i \times r_i)$ dynamic matrix of the i^{th} component and $\mathbf{P}_c(s)$ is the $(r \times r)$ unconstrained dynamic matrix of components. The unconstrained component model relates the component deviation output vector $\bar{\mathbf{Y}}_c(s)$ to the deviation component input vector $\bar{\mathbf{U}}_c(s)$ in the form,

$$\mathbf{P}_c(s)\bar{\mathbf{Y}}_c(s) = \bar{\mathbf{U}}_c(s) \tag{3.34}$$

The required equations to transform deviation variables into physical variables are also available.

$$\bar{\mathbf{Y}}_c(s) = \mathbf{Y}_c(s) - \mathbf{y}_0(1/s) \tag{3.35a}$$

$$\bar{\mathbf{U}}_c(s) = \mathbf{U}_c(s) - \mathbf{u}_0(1/s) \tag{3.35b}$$

The formulation of the dynamic constraint equations (2.19a) and (2.19b) is the second step. The objective is to generate the constrain matrix $\mathbf{S}$ that relates component output variables and assembly output variables. This relation is established by considering that output variables are constrained to be equal to other outputs variables when ports are connected. The process is illustrated in the next section.

The generation of the assembled model is the third step. Initially the equation (3.35a) and (3.35b) are substituted into (3.34)

$$\mathbf{P}_c(s)\left[\mathbf{Y}_c(s) - \mathbf{y}_0(1/s)\right] = \mathbf{U}_c(s) - \mathbf{u}_0(1/s) \tag{3.36}$$

Multiplying both sides of (3.36) by $\mathbf{S}^T$ yields,

$$\mathbf{S}^T\mathbf{P}_c(s)\left[\mathbf{Y}_c(s) - \mathbf{y}_0(1/s)\right] = \mathbf{S}^T\mathbf{U}_c(s) - \mathbf{S}^T\mathbf{u}_0(1/s) \tag{3.37}$$

Substituting (2.19a),(2.19b), (3.32a) and (3.32b) into (3.37) yields,

$$\mathbf{S}^T\mathbf{P}_c(s)\left[\mathbf{S}\mathbf{Y}_a(s) - \mathbf{S}\tilde{\mathbf{y}}_0(1/s)\right] = \mathbf{U}_a(s) - \tilde{\mathbf{u}}_0(1/s) \tag{3.38}$$

Factoring matrix $\mathbf{S}$ in the right side and rewriting (3.38)

$$\mathbf{P}_a(s)\left[\mathbf{Y}_a(s) - \tilde{\mathbf{y}}_0(1/s)\right] = \mathbf{U}_a(s) - \tilde{\mathbf{u}}_0(1/s) \tag{3.39}$$

where the assembly dynamic stiffness matrix is

$$\mathbf{P}_a(s) = \mathbf{S}^T\mathbf{P}_c(s)\mathbf{S} \tag{3.40}$$

Defining a new set of $(l \times 1)$ deviation variable vectors,

$$\bar{\mathbf{Y}}_a(s) = \mathbf{Y}_c(s) - \tilde{\mathbf{y}}_0(1/s) \tag{3.41a}$$

$$\bar{\mathbf{U}}_a(s) = \mathbf{U}_c(s) - \tilde{\mathbf{u}}_0(1/s) \tag{3.41b}$$

and finally, substituting (3.41a) and (3.41b) into (3.39) yields the assembly model in deviation assembly variables,

$$\mathbf{P}_a(s)\bar{\mathbf{Y}}_a(s) = \bar{\mathbf{U}}_a(s) \tag{3.42}$$

The assembled model (3.42) is in the MMM assembly standard format (3.26). The process is recursive. Model (3.42) can be assembled to other standard higher order models using the same algorithm. As in the linear case, model (3.42) must be inverted to obtain the transfer function model in deviation variables. The transfer function model is used in the simulation procedure

3.6 Example

The assembly of a mechanic transmission model and an electric generator model is developed in this section (Figure 3.5). The electric generator has two ports. The generator first port variables are the input electrical charge $q(t)$ and the output voltage

potential $e(t)$. The generator second port variables are the input rotational torque $\tau_1(t)$ and the output angular displacement $\theta_1(t)$. The mechanical transmission has two ports. The transmission first port variables are the input rotational torque $\tau_2(t)$ and the output angular displacement $\theta_2(t)$. The transmission second port variables are the rotational torque $\tau_3(t)$ and the output angular displacement $\theta_3(t)$.

The assembly uses seven ports and three joins. Three of these seven ports are assembly ports. The other four are component ports. The left joint connects the charge-potential assembly port $(q^*(t), e^*(t))$ to the component port pair $(q(t), e(t))$. The right joint connects the torque-angular displacement assembly port $(\tau_3^*(t), \theta_3^*(t))$ to the component port $(\tau_3(t), \theta_3(t))$. The middle joint connects the component ports $(\tau_1(t), \theta_1(t))$ and $(\tau_2(t), \theta_2(t))$ to the torque-angular displacement assembly port $(\tau_a(t), \theta_a(t))$.

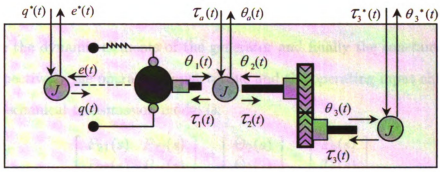

Figure 3.5. Transmission-Electric Generator Assembly

The external nonlinear models for the generator and the transmission have respectively the general form,

$$\mathbf{f}_g = \left(e, \theta_1, \dot{e}, \dot{\theta}_1, \cdots, q_1, \tau_1, \dot{q}_1, \dot{\tau}_1, \cdots\right) = \mathbf{0} \tag{3.43a}$$

$$\mathbf{f}_\tau = \left(\theta_2, \theta_3, \dot{\theta}_2, \dot{\theta}_3, \cdots, \tau_2, \tau_3, \dot{\tau}_2, \dot{\tau}_3, \cdots\right) = \mathbf{0} \tag{3.43b}$$

where $\mathbf{f}_g(t)$ and $\mathbf{f}_\tau(t)$ are vectors of functions. The user requests a model of the assembly by defining its output operating points. These are the operating voltage e_0^*, and the operating angular displacements $\theta_{0,a}^*$, and $\theta_{0,3}^*$. Using this information,

the assembly agent computes the generator output operating points $e_0 = e_0^*$ and $\theta_{0,1} = \theta_{0,a}$, and the transmission output operating points $\theta_{0,2} = \theta_{0,a}$ and $\theta_{0,3} = \theta_{0,3}^*$.

Component agents have at this point all the information to provide the component models and the physical variable transformation equations. For the generator,

$$\begin{bmatrix} P_{11}(s) & P_{12}(s) \\ P_{13}(s) & P_{14}(s) \end{bmatrix} = \begin{bmatrix} \bar{\Theta}_1(s) \\ \bar{E}(s) \end{bmatrix} = \begin{bmatrix} \bar{T}_1(s) \\ \bar{Q}(s) \end{bmatrix} \tag{3.44a}$$

$$\begin{bmatrix} \bar{\Theta}_1(s) \\ \bar{E}(s) \end{bmatrix} = \begin{bmatrix} \Theta_1(s) \\ E(s) \end{bmatrix} - \begin{bmatrix} \theta_{0,1} \\ e_0 \end{bmatrix} \frac{1}{s} \tag{3.44b}$$

$$\begin{bmatrix} \bar{T}_1(s) \\ \bar{Q}(s) \end{bmatrix} = \begin{bmatrix} T_1(s) \\ Q(s) \end{bmatrix} - \begin{bmatrix} \tau_{0,1} \\ q_0 \end{bmatrix} \frac{1}{s} \tag{3.44c}$$

where $\bar{Q}(s)$, $\bar{T}_1(s)$, $\bar{E}(s)$ and $\bar{\Theta}_1(s)$ are respectively the Laplace Transform of the deviation time variables $\bar{q}(t)$, $\bar{\tau}_1(t)$, $\bar{e}(t)$ and $\bar{\theta}_1(t)$. The complex variables $Q(s)$, $T_1(s)$, $E(s)$ and $\theta_1(s)$ are respectively the Laplace Transform of the physical variables $q(t)$, $\tau_1(t)$, $e(t)$ and $\theta_1(t)$. The complex variable functions $P_{11}(s)$, $P_{12}(s)$, $P_{13}(s)$ and $P_{14}(s)$ are the dynamic elements of the generator and finally the constants $\tau_{0,1}$ and q_0 are respectivelly the operating input torque and the operating input charge.

The mechanical transmission model is,

$$\begin{bmatrix} P_{21}(s) & P_{22}(s) \\ P_{23}(s) & P_{24}(s) \end{bmatrix} = \begin{bmatrix} \bar{\Theta}_2(s) \\ \bar{\Theta}_3(s) \end{bmatrix} = \begin{bmatrix} \bar{T}_2(s) \\ \bar{T}_3(s) \end{bmatrix} \tag{3.45a}$$

$$\begin{bmatrix} \bar{\Theta}_2(s) \\ \bar{\Theta}_3(s) \end{bmatrix} = \begin{bmatrix} \Theta_2(s) \\ \Theta_3(s) \end{bmatrix} - \begin{bmatrix} \theta_{0,2} \\ \theta_{0,3} \end{bmatrix} \frac{1}{s} \tag{3.45b}$$

$$\begin{bmatrix} \bar{T}_2(s) \\ \bar{T}_3(s) \end{bmatrix} = \begin{bmatrix} T_2(s) \\ T_3(s) \end{bmatrix} - \begin{bmatrix} \tau_{0,2} \\ \tau_{0,3} \end{bmatrix} \frac{1}{s} \tag{3.45c}$$

where $\bar{\Theta}_2(s)$, $\bar{T}_2(s)$, $\bar{\Theta}_3(s)$ and $\bar{T}_3(s)$ are respectively the Laplace Transform of the deviation time variables $\bar{\theta}_2(t)$, $\bar{\tau}_2(t)$, $\bar{\theta}_3(t)$ and $\bar{\tau}_3(s)$. The complex variables $Q(s)$, $T_1(s)$, $E(s)$ and $\theta_1(s)$ are respectively the Laplace Transform of the physical variables $q(t)$, $\tau_1(t)$, $e(t)$ and $\theta_1(t)$. The complex variable functions $P_{21}(s)$, $P_{22}(s)$, $P_{23}(s)$ and $P_{24}(s)$ are the dynamic elements of the transmission and finally the constants $\tau_{0,2}$ and $\tau_{0,3}$ are the operating torque inputs.

The first step is to generate the unconstrained component model. Initially the unconstrained component matrix is generated. This matrix is,

$$\mathbf{P}_c(s) = \begin{bmatrix} P_{11}(s) & P_{12}(s) & 0 & 0 \\ P_{13}(s) & P_{14}(s) & 0 & 0 \\ 0 & 0 & P_{21}(s) & P_{22}(s) \\ 0 & 0 & P_{23}(s) & P_{24}(s) \end{bmatrix} \tag{3.46}$$

The unconstrained component model is,

$$\begin{bmatrix} P_{11}(s) & P_{12}(s) & 0 & 0 \\ P_{13}(s) & P_{14}(s) & 0 & 0 \\ 0 & 0 & P_{21}(s) & P_{22}(s) \\ 0 & 0 & P_{23}(s) & P_{24}(s) \end{bmatrix} \begin{bmatrix} \bar{\Theta}_1(s) \\ \bar{E}(s) \\ \bar{\Theta}_2(s) \\ \bar{\Theta}_3(s) \end{bmatrix} = \begin{bmatrix} \bar{T}_1(s) \\ \bar{Q}(s) \\ \bar{T}_2(s) \\ \bar{T}_3(s) \end{bmatrix} \tag{3.47}$$

The second step is to generate the constraint assembly equations. Initially the time dependent equations that relate component output to assembly outputs are written. These equations are,

$$\begin{gathered} e(t) = e^*(t) \\ \theta_1(t) = \theta_2(t) = \theta_a(t) \\ \theta_3(t) = \theta_3^*(t) \end{gathered} \tag{3.48}$$

Additionally, conservation of energy at each joint requires

$$\begin{gathered} q(t)e(t) = q^*(t)e^*(t) \\ \tau_a(t)\theta_a(t) = \tau_1(t)\theta_1(t) + \tau_2(t)\theta_2(t) \\ \tau_3(t)\theta_3(t) = \tau_3^*(t)\theta_3^*(t) \end{gathered} \tag{3.49}$$

From (3.48) and (3.49), follows that

$$\begin{gathered} q(t) = q^*(t) \\ \tau_a(t) = \tau_1(t) + \tau_2(t) \\ \tau_3(t) = \tau_3^*(t) \end{gathered} \tag{3.50}$$

Applying the Laplace Transform to (3.48) and (3.50) results,

$$\begin{bmatrix} \Theta_1(s) \\ E(s) \\ \Theta_2(s) \\ \Theta_3(s) \end{bmatrix} = \mathbf{S} \begin{bmatrix} E^*(s) \\ \Theta_a(s) \\ \Theta_3^*(s) \end{bmatrix} \tag{3.51a}$$

$$\mathbf{S}^T \begin{bmatrix} T_1(s) \\ Q(s) \\ T_2(s) \\ T_3(s) \end{bmatrix} = \begin{bmatrix} Q^*(s) \\ T_a(s) \\ T_3^*(s) \end{bmatrix} \tag{3.51b}$$

where the constraint matrix $\mathbf{S} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. Similarly, for the operating points,

$$\begin{bmatrix} \theta_{0,1} \\ e_0 \\ \theta_{0,2} \\ \theta_{0,3} \end{bmatrix} = \mathbf{S} \begin{bmatrix} e_0^* \\ \theta_{0,a}^* \\ \theta_{0,3}^* \end{bmatrix} \tag{3.52a}$$

$$\mathbf{S}^T \begin{bmatrix} \tau_{0,1} \\ q_0 \\ \tau_{0,2} \\ \tau_{0,3} \end{bmatrix} = \begin{bmatrix} q_0^* \\ \tau_{0,a} \\ \tau_{0,3}^* \end{bmatrix} \tag{3.52b}$$

where q_0^*, $\tau_{0,a}$ and $\tau_{0,3}^*$ are the operating inputs of the assembly.

In the third step, the assembly model is generated. Substituting (3.44b), (3.44c), (3.45b) and (3.45c) into (3.47) yields,

$$\mathbf{P}_c(s) \left\{ \begin{bmatrix} \Theta_1(s) \\ E(s) \\ \Theta_2(s) \\ \Theta_3(s) \end{bmatrix} - \begin{bmatrix} \theta_{0,1} \\ e_0 \\ \theta_{0,2} \\ \theta_{0,3} \end{bmatrix} \frac{1}{s} \right\} = \left\{ \begin{bmatrix} T_1(s) \\ Q(s) \\ T_2(s) \\ T_3(s) \end{bmatrix} - \begin{bmatrix} \tau_{0,1} \\ q_0 \\ \tau_{0,2} \\ \tau_{0,3} \end{bmatrix} \frac{1}{s} \right\} \tag{3.53}$$

Multiplying both sides of (3.53) by $\mathbf{S}^T$ yields,

$$\mathbf{S}^T\mathbf{P}_c(s) \left\{ \begin{bmatrix} \Theta_1(s) \\ E(s) \\ \Theta_2(s) \\ \Theta_3(s) \end{bmatrix} - \begin{bmatrix} \theta_{0,1} \\ e_0 \\ \theta_{0,2} \\ \theta_{0,3} \end{bmatrix} \frac{1}{s} \right\} = \left\{ \mathbf{S}^T \begin{bmatrix} T_1(s) \\ Q(s) \\ T_2(s) \\ T_3(s) \end{bmatrix} - \mathbf{S}^T \begin{bmatrix} \tau_{0,1} \\ q_0 \\ \tau_{0,2} \\ \tau_{0,3} \end{bmatrix} \frac{1}{s} \right\} \tag{3.54}$$

Replacing (3.51a), (3.51b), (3.52a) and (3.52b) into (3.54) yields,

$$\mathbf{S}^T\mathbf{P}_c(s) \left\{ \mathbf{S} \begin{bmatrix} E^*(s) \\ \Theta_a(s) \\ \Theta_3^*(s) \end{bmatrix} - \mathbf{S} \begin{bmatrix} e_0^* \\ \theta_{0,a} \\ \theta_{0,3}^* \end{bmatrix} \frac{1}{s} \right\} = \left\{ \begin{bmatrix} Q^*(s) \\ T_a(s) \\ T_3^*(s) \end{bmatrix} - \begin{bmatrix} q_0^* \\ \tau_{0,a} \\ \tau_{0,3}^* \end{bmatrix} \frac{1}{s} \right\} \tag{3.55}$$

Factorizing the matrix $\mathbf{S}$ in the right side of (3.55) yields,

$$\mathbf{P}_a(s) \left\{ \begin{bmatrix} E^*(s) \\ \Theta_a(s) \\ \Theta_3^*(s) \end{bmatrix} - \begin{bmatrix} e_0^* \\ \theta_{0,a} \\ \theta_{0,3}^* \end{bmatrix} \frac{1}{s} \right\} = \left\{ \begin{bmatrix} Q^*(s) \\ T_a(s) \\ T_3^*(s) \end{bmatrix} - \begin{bmatrix} q_0^* \\ \tau_{0,a} \\ \tau_{0,3}^* \end{bmatrix} \frac{1}{s} \right\} \tag{3.56}$$

where the assembly matrix $\mathbf{P}_a(s) = \mathbf{S}^T\mathbf{P}_c(s)\mathbf{S}$. Defining a new set of variables ,

$$\begin{bmatrix} \bar{E}^*(s) \\ \bar{\Theta}_a(s) \\ \bar{\Theta}_3^*(s) \end{bmatrix} = \begin{bmatrix} E^*(s) \\ \Theta_a(s) \\ \Theta_3^*(s) \end{bmatrix} - \begin{bmatrix} e_0^* \\ \theta_{0,a} \\ \theta_{0,3}^* \end{bmatrix} \frac{1}{s} \tag{3.57a}$$

$$\begin{bmatrix} \bar{Q}^*(s) \\ \bar{T}_a(s) \\ \bar{T}_3^*(s) \end{bmatrix} = \begin{bmatrix} Q^*(s) \\ T_a(s) \\ T_3^*(s) \end{bmatrix} - \begin{bmatrix} q_0^* \\ \tau_{0,a} \\ \tau_{0,3}^* \end{bmatrix} \frac{1}{s} \tag{3.57b}$$

and replacing them into equation (3.56) yields,

$$\begin{bmatrix} P_{11}(s) & P_{12}(s) & 0 \\ P_{13}(s) & P_{14}(s) + P_{21}(s) & P_{22}(s) \\ 0 & P_{23}(s) & P_{24}(s) \end{bmatrix} \begin{bmatrix} \bar{E}^*(s) \\ \bar{\Theta}_a(s) \\ \bar{\Theta}_3^*(s) \end{bmatrix} = \begin{bmatrix} \bar{Q}^*(s) \\ \bar{T}_a(s) \\ \bar{T}_3^*(s) \end{bmatrix} \tag{3.58}$$

Equation (3.58) is the the assembly of the generator model and the mechanical transmission model. This resultant model is in the standard format defined in (3.26). The variable transformation equations (3.57a) and (3.57b) are in the standard format (3.31a) and (3.31b). The process is recursive and the model (3.58) can be assembled to other standard higher order models using the same algorithm.

3.7 Summary

In this chapter, two important contributions to mechanical engineering are presented. The first contribution is a method to assemble physical port-based affine ODEs around an equilibrium operating point. Affine systems often result of local linearization about an operating point. In this case, the local system model is linear in deviation variables and non-linear in physical variables. Because assembly constraints are always given in terms of physical variables, an explicit method to apply physical assembly constraints for affine physical models was developed. This method addresses one of the most common nonlinear systems in mechanical engineering. Using this method many practical physical nonlinear system models can be obtained by assembling the affine approximations of their subsystems models.

The second contribution is an explicit method to obtain subsystems operating points from the system assembly operating point outputs. This method for the first time provides an explicit, closed form solution to the general operating point problem. In a recursive closed form, the method specifies information required to roll-down an operating point specification through every subsystem to all the lowest level components of a system. For each lowest level component, the method provides the exact port output values where the operating point representation of the nonlinear component model should be developed. The method then provides an explicit method for computing operating point inputs of the component level and use these inputs to compute the operating point inputs of the assembled system model.

CHAPTER 4

Volterra Model Formats

4.1 Background

Physical nonlinear dynamic systems have traditionally been modeled using nonlinear ordinary differential equations (ODEs). One solution technique for single-input, single-output (SISO) nonlinear ODEs is obtained through a Volterra transfer function. Applying this solution technique for multi-input, multi-output (MIMO) nonlinear ODEs is a complicated issue due to the difficulty of analytically determining MIMO Volterra models.

MIMO Volterra transfer function models are multi input-output representations used to describe nonlinear behavior. A Volterra model is nonlinear with respect to its input but linear with respect to its parameters [24]. Volterra models have been used to obtain solutions for SISO nonlinear systems in areas such as dynamic behavior of offshore structures [25], power amplifier behavior [14], rheology [15], nonlinear model reduction [16] and others. Areas such as sub-harmonic nonlinear behavior [26] and noise characterization [27] use Volterra models for describing the behavior of multi-input single-output nonlinear dynamic systems.

Although an analytical procedure to obtain Volterra transfer functions for SISO nonlinear ODEs is available [28], [10], no general analytical procedure to obtain a MIMO Volterra transfer functions from port-based, external nonlinear ODEs has

been published. This is a complicated issue when the number of system's inputs and system's outputs are not equal. The orientation of existing works in MIMO Volterra models is to experimentally identify the model kernels [29], to approximate weakly nonlinear MIMO systems [30] and to determine error bounds [31].

In this chapter, a procedure to analytically obtain MIMO Volterra models from port-based nonlinear ODEs is derived. This procedure provides a solution to this class of MIMO nonlinear ODEs. First, the standard nonlinear port-based ODEs used in this procedure are described. Second, the Volterra MIMO models are presented. Third, a nonlinear operator required to obtain the MIMO Volterra models is defined. Fourth, the procedure to obtain MIMO Volterra model expansions from port-based nonlinear ODEs is presented. Fifth, an example of a two-port nonlinear ODE is shown. The Volterra model response is compared to the nonlinear ODE model response using a Simulink simulation.

4.2 Port Based Nonlinear ODE Models

A port-based nonlinear ODE model has equal number of inputs and outputs because an input-output pair constitutes each port. This section considers nonlinear port-based models in the form,

$$\begin{aligned} f_1\left(y_1, \dot{y}_1, \ldots \frac{d^m y_1}{dt^m}, \ldots, y_r, \dot{y}_r, \ldots \frac{d^m y_r}{dt^m}\right) &= u_1 \\ &\vdots \\ f_r\left(y_1, \dot{y}_1, \ldots \frac{d^m y_1}{dt^m}, \ldots, y_r, \dot{y}_r, \ldots \frac{d^m y_r}{dt^m}\right) &= u_r \end{aligned} \tag{4.1}$$

where the number of ports $1 \leq i \leq r$; $f_i(\cdot)$ is the i^{th} port's nonlinear operator; u_i is the i^{th} input and y_i is the i^{th} output of the system. Many physical models of nonlinear engineering systems take this form when modeled using the energy-based Lagrange approach [32].

An example is a two-port system (Figure 4.1) composed of two masses, one nonlinear spring, one linear spring and one linear damper. The mass m_1 is connected to a

fixed point through the nonlinear spring with force $f_{NL} = k_1(y_1 + y_1^3)$. The mass m_2 is connected to the mass m_1 with a linear damper having damping coefficient b_1 and a linear spring with constant k_2. The first port uses the input-output pair (u_1, y_1) where u_1 is the input force applied to the mass and y_1 is its output displacement. The second port uses the input-output pair (u_2, y_2) where u_2 is the input force applied to the mass m_2 and y_2 is its output displacement.

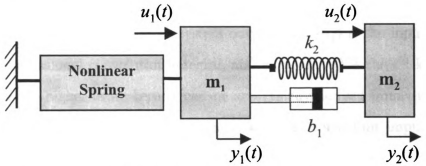

Figure 4.1. Double-Mass Spring Damper-Nonlinear System

The nonlinear, port-based, ODE model for the system (Figure 5.3) in the required format (4.1) is

$$\begin{aligned} m_1\ddot{y}_1 + b_1(\dot{y}_1 - \dot{y}_2) + k_1 y_1 + k_1 {y_1}^3 + k_2(y_1 - y_2) &= u_1 \\ m_2\ddot{y}_2 + b_1(\dot{y}_2 - \dot{y}_1) + k_2(y_2 - y_1) &= u_2 \end{aligned} \tag{4.2}$$

4.3 Volterra Models

In this section two nonlinear model formats are presented. The first model format is the MIMO Volterra transfer function representation. This model representation is used in the simulation of port based nonlinear dynamic models. The second model format is the MIMO Volterra dynamic model. This model representation is used in the nonlinear MMM assembly of port based dynamic physical systems.

A Volterra transfer function model is an extension of the linear convolution integral approach for time-invariant systems. Any nonlinear time-invariant system can be

represented as an infinite sum of multidimensional convolution integrals of increasing order [10]. For a SISO system this is

$$\begin{aligned} y(t) = & \int_0^\infty g_1(\tau_1)u(t-\tau_1)d\tau_1 + \\ & \int_0^\infty \int_0^\infty g_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)d\tau_1 d\tau_2 + \\ & \int_0^\infty \int_0^\infty \int_0^\infty g_3(\tau_1,\tau_2,\tau_3)u(t-\tau_1)u(t-\tau_2)u(t-\tau_3)d\tau_1 d\tau_2 d\tau_3 + \cdots \\ & \vdots \end{aligned} \tag{4.3}$$

where $y(t)$ represents the nonlinear system output, $u(t-\tau_i)$ is the input associated to the i^{th} dimensional convolution integral, and $g_i(\tau_1,\ldots,\tau_i)$ is the i^{th} kernel of the system. All the higher order kernels can be analytically derived from the first order linear kernel $g_1(\tau_1)$ [28]. This kernel is the traditional linear unit impulse response of the system. The second order kernel, $g_2(\tau_1,\tau_2)$, is a two-dimensional function of time and is the response of the system to two independent unit impulses applied at two different points in time. The kernel $g_2(\tau_1,\tau_2)$ is a function of both time and one time lag $(\tau_1-\tau_2)$. The kernel $g_3(\tau_1,\tau_2,\tau_3)$ is a three-dimensional function of time and is the response of the system to three independent unit impulses applied at three different points in time. The kernel $g_3(\tau_1,\tau_2,\tau_3)$ is a function of both time and the two time lags $(\tau_1-\tau_2)$ and $(\tau_2-\tau_3)$ [33].

The i^{th} integral in (4.3) is called a "degree-i homogeneous system" and its respective output

$$y_i = \int_o^\infty \cdots \int_o^\infty g_i(\tau_1,\cdots,\tau_i)u(t-\tau_1)\cdots u(t-\tau_i)d\tau_1\cdots d\tau_i \tag{4.4}$$

is called the "degree-i homogeneous output" (4.4). Notice that if a new input αu where α is a scalar, is applied, the resultant output is $\tilde{y}_i = \alpha^n y_i$ [10]. The SISO response $y(t)$ in (4.4) is then a sum of partial outputs provided by different "degree-i homogeneous systems" in the form,

$$y(t) = y_1(t_1) + y_2(t_1,t_2) + y_3(t_1,t_2,t_3) + \cdots \tag{4.5}$$

where,

$$
\begin{aligned}
y_1 &= \int_0^\infty g_1(\tau_1)u(t-\tau_1)d\tau_1 \\
y_2 &= \int_0^\infty\int_0^\infty g_2(\tau_1,\tau_2)u(t-\tau_1)u(t-\tau_2)d\tau_1 d\tau_2 \\
y_3 &= \int_0^\infty\int_0^\infty\int_0^\infty g_3(\tau_1,\tau_2,\tau_3)u(t-\tau_1)u(t-\tau_2)u(t-\tau_3)d\tau_1 d\tau_2 d\tau_3 \\
&\vdots
\end{aligned}
$$

and $g_i(\tau_1,\cdots\tau_i)$ is the i^{th} kernel.

The Laplace transform of a multi-time variable system is an extension of the single-time Laplace transform. Given an i^{th} dimensional function of time $g_i(\tau_1,\cdots\tau_i)$ that is one-sided in each variable $(0 \leq \tau_i \leq \infty)$, its Laplace transform is defined as [10]

$$
G_i(s_1,\cdots,s_i) = \int_0^\infty \cdots \int_0^\infty g_i(\tau_1,\cdots,\tau_i)e^{-s_1\tau_1},\cdots,e^{-s_i\tau_i}d\tau_1,\cdots,d\tau_i \tag{4.6}
$$

where $(1 \leq k \leq i)$ the set of complex variables $s_k = \sigma_k + \omega_k j$. Applying the Laplace transform to each term in (4.5) yields,

$$
\begin{aligned}
Y_1(s_1) &= G_1(s_1)U(s_1) \\
Y_2(s_1,s_2) &= G_2(s_1,s_2)U(s_1)U(s_2) \\
Y_3(s_1,s_2,s_3) &= G_3(s_1,s_2,s_3)U(s_1)U(s_2)U(s_3) \\
&\vdots
\end{aligned} \tag{4.7}
$$

where $U(s_i)$ is the Laplace transform of the input $u(t_i)$ and $G_i(s_1,\cdots,s_i)$ is the multi-complex variable Laplace transform of the i^{th} order kernel. The SISO Volterra transfer function is obtained by summing all the rows in (4.7)

$$
\begin{aligned}
Y(s_1,s_2,\cdots) = G_1(s_1)U(s_1) &+ G_2(s_1,s_2)U(s_1)U(s_2) \\
&+ G_3(s_1,s_2,s_3)U(s_1)U(s_2)U(s_3) + \cdots
\end{aligned} \tag{4.8}
$$

The model representation (4.8) is used to simulate the response of SISO nonlinear systems [10].

The MIMO extension of (4.7) for a system with r ports is,

$$\begin{aligned} \mathbf{Y}_1(s_1) &= \mathbf{G}_1\left(\mathbf{U}(s_1)\right) \\ \mathbf{Y}_2(s_1, s_2) &= \mathbf{G}_2\left(\mathbf{U}(s_1)\mathbf{U}(s_2)\right) \\ \mathbf{Y}_3(s_1, s_2, s_3) &= \mathbf{G}_3\left(\mathbf{U}(s_1)\mathbf{U}(s_2)\mathbf{U}(s_3)\right) \\ &\vdots \end{aligned} \tag{4.9}$$

Each element $\mathbf{G}_i(\bullet)$ in (4.9) is a functional that operates on a set of $(r \times 1)$ input vectors, yielding $\forall(\leq i \leq \infty)$ the $(r \times 1)$ "degree-i homogeneous output" vector,

$$\mathbf{Y}_i(s_1, \cdots, s_i) = \left[Y_{i,1}(s_1, \cdots, s_i), \cdots, Y_{i,r}(s_1, \cdots, s_i)\right]^T \tag{4.10}$$

The Volterra transfer function for a port-based system with r ports is an extension of (4.8) and is obtained by adding all the $\mathbf{G}_i(\bullet)$ functionals in (4.9). The response is a vector in the form,

$$\mathbf{Y}(s_1, \cdots) = \mathbf{G}_1\left(\mathbf{U}(s_1)\right) + \mathbf{G}_2\left(\mathbf{U}(s_1), \mathbf{U}(s_2)\right) + \mathbf{G}_3\left(\mathbf{U}(s_1), \mathbf{U}(s_2), \mathbf{U}(s_3)\right) + \cdots \tag{4.11}$$

where $\mathbf{U}(s_1), \mathbf{U}(s_2), \cdots$, are $(r \times 1)$ input vectors and $\mathbf{Y}(s_1, \cdots)$ is the resultant $(r \times 1)$ output vector.

The MIMO notation in (4.11) can be simplified if the following input argument list is defined,

$$\begin{aligned} \mathbf{U}_1 &\triangleq \mathbf{U}(s_1) \\ \mathbf{U}_2 &\triangleq \{\mathbf{U}(s_1), \mathbf{U}(s_2)\} \\ \mathbf{U}_3 &\triangleq \{\mathbf{U}(s_1), \mathbf{U}(s_2), \mathbf{U}(s_3)\} \\ &\vdots \end{aligned} \tag{4.12}$$

Using (4.12) into (4.11) yields,

$$\mathbf{Y}(s_1, s_2, \cdots) = \mathbf{G}_1(\mathbf{U}_1) + \mathbf{G}_2(\mathbf{U}_2) + \mathbf{G}_3(\mathbf{U}_3) + \cdots \tag{4.13}$$

In the same form, the output notation is further simplified using

$$\begin{aligned} \mathbf{Y}_{(1)} &\triangleq \mathbf{Y}_1(s_1) = \mathbf{G}_1(\mathbf{U}_1) \\ \mathbf{Y}_{(2)} &\triangleq \mathbf{Y}_2(s_1, s_2) = \mathbf{G}_2(\mathbf{U}_2) \\ \mathbf{Y}_{(3)} &\triangleq \mathbf{Y}_3(s_1, s_2, s_3) = \mathbf{G}_3(\mathbf{U}_3) \\ &\vdots \end{aligned} \tag{4.14}$$

Using (4.14), the MIMO Volterra transfer function model in (4.13) can be written as,

$$\mathbf{Y}(s_1, \cdots) = \mathbf{Y}_{(1)} + \mathbf{Y}_{(2)} + \mathbf{Y}_{(3)} + \cdots \tag{4.15}$$

This model format is used in the simulation of nonlinear port based physical systems.

A procedure generates each of the $\mathbf{Y}_{(i)}$ [34]. Each $\mathbf{Y}_{(i)}$ has an unique standard format that can be represented in function of the first order linear operator $\mathbf{G_1(U_1)} = \mathbf{G}_{1,(1)}\mathbf{U}_1$. The term $\mathbf{G}_{1,(1)}$ is the $(r \times r)$ transfer function matrix obtained from the linearization of the system nonlinear ODE in (4.1) around an equilibrium operating point *op*. The matrix $\mathbf{G}_{1,(1)}$ can also be obtained by inverting the dynamic matrix

$$\mathbf{P}_{1,(1)} = \left[\left(\frac{\partial f_i}{\partial y_j}\right)\right]|_{op} + \left[\left(\frac{\partial f_i}{\partial \dot{y}_j}\right)\right]|_{op} s_1 + \left[\left(\frac{\partial f_i}{\partial \ddot{y}_j}\right)\right]|_{op} s_1^2 + \left[\left(\frac{\partial f_i}{\partial \dddot{y}_j}\right)\right]|_{op} s_1^3 + \cdots \tag{4.16}$$

where $(1 \leq i, j \leq r)$ and each element in the brackets is a $(r \times r)$ matrix of constants. The sub-index (1) in $\mathbf{G}_{1,(1)}$ or $\mathbf{P}_{1,(1)}$ indicates that these matrices are function of a single complex variable s_1 For any sufficiently differentiable ODE, the first three standard Volterra operators are [34],

$$\mathbf{Y}_{(1)} = \mathbf{G}_{1,(1)}\mathbf{U}_1 \tag{4.17a}$$

$$\mathbf{Y}_{(2)} = -\mathbf{G}_{1,(2)}\mathbf{P}_{2,(2)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1\right) \tag{4.17b}$$

$$\mathbf{Y}_{(3)} = -\mathbf{G}_{1,(3)}\begin{bmatrix} \mathbf{P}_{3,(3)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1\right) + \\ \mathbf{P}_{2,(3)}\psi\left[\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(2)}\mathbf{P}_{2,(2)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1\right)\right] \end{bmatrix} \tag{4.17c}$$

where $\mathbf{G}_{1,(v)}$ is the result of inverting the dynamic matrix $\mathbf{P}_{1,(v)}$, and $\psi(\bullet)$ is a nonlinear operator. Any dynamic matrix $\mathbf{P}_{u,(v)}$, includes v complex variables, $s_1, \cdots, s_v$ and can be derived from (4.1) using,

$$\mathbf{P}_{u,(v)} = \frac{1}{u!}\left\{ \begin{array}{l} \left[\left(\frac{\partial^u f_i}{\partial y_j^u}\right)\right]|_{op} + \left[\left(\frac{\partial^u f_i}{\partial \dot{y}_j^u}\right)\right]|_{op} s_{(v)} + \\ \qquad \left[\left(\frac{\partial^u f_i}{\partial \ddot{y}_j^u}\right)\right]|_{op} s_{(v)}^2 + \left[\left(\frac{\partial^u f_i}{\partial \dddot{y}_j^u}\right)\right]|_{op} s_{(v)}^3 + \cdots \end{array} \right\} \tag{4.18}$$

where $s_{(v)} = s_1 + \cdots + s_v$.

The *Volterra Dynamic Model* is other Volterra format. The Volterra dynamic model is composed by a set of multi-complex variable dynamic matrices. For the i^{th} component model that includes q expansion terms, this model has the form,

$$\left\{\mathbf{P}_{i,1,(v)}, \mathbf{P}_{i,2,(v)}, \cdots, \mathbf{P}_{i,q,(v)}\right\} \tag{4.19}$$

Model (4.19) is used for the MMM to assemble nonlinear physical models. For any $\mathbf{P}_{i,u,(v)}$ in (4.19), the index i, represents the component, the index u represent the order of the expansion and the index v represents the number of complex independent variables used. Each $\mathbf{P}_{i,u,(v)}$ matrix satisfies,

$$\mathbf{P}_{i,u+1,(v)} = \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{i,u,(v)}\right) \tag{4.20}$$

where each element of the $(r \times r)$ matrix $\mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{i,u,(v)}\right)$ is the first partial derivative respect the vectors $\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}, \cdots$ of its corresponding elements in the $(r \times r)$ matrix $\mathbf{P}_{i,u,(v)}$. Equation (4.20) is derived from (4.18)by rewritten it in general form $\forall(1 \leq u)$. In example, for the i^{th} component, the dynamic matrices,

$$\begin{aligned}
\mathbf{P}_{i,2,(v)} &= \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{i,1,(v)}\right) \\
\mathbf{P}_{i,3,(v)} &= \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{i,2,(v)}\right) \\
\mathbf{P}_{i,4,(v)} &= \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{i,3,(v)}\right)
\end{aligned}$$

The Volterra series is an infinite power series with memory and suffers from the problem of a limited zone of convergence. For the Volterra representation to properly describe a subsystem system model, the Volterra series must converge over the input/output variable range used in the assembled system. This issue is recognized in multiple works: Frank [39], Brilliant [11], Christensen [40], Boyd [13], Czarniak [41], Tomilson [43], Chatterjee, [42] and others. The series convergence depends on system parameters [42], BIBO stability [40], amplitude of the input applied [13], truncation point of the series and other factors. One necessary condition for existence of the series and its convergence is that the linear term cannot be zero [39]. Boyd [13]

showed that the radius of convergence of a Volterra series is directly related to the amplitude of the input applied. This radius of convergence can be presented in the form $|\mathbf{u}(t)| < \rho$, where $\mathbf{u}(t)$ is the input vector applied to the system and ρ is the radius of convergence. Sandberg [12] has also shown that a truncated Volterra series provides a uniform approximation to the in finite Volterra series on a ball of bounded inputs for a large class of systems. SISO Volterra series have been recognized as a useful approach to engineering modeling. This work assumes a useful convergent Volterra model exist and converges over the domain of model application.

4.4 The Nonlinear Multiplicative Operator

A nonlinear operator $\psi(\bullet)$ is used to build the Volterra functionals in (4.13). This operator uses as argument an arbitrary set of vectors $(r \times 1)$ in the form,

$$\mathbf{R} = \left[\mathbf{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_r \end{pmatrix}, \mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_r \end{pmatrix} \mathbf{w} = \begin{pmatrix} w_1 \\ \vdots \\ w_r \end{pmatrix}, \cdots, \mathbf{z} = \begin{pmatrix} z_1 \\ \vdots \\ z_r \end{pmatrix} \right] \tag{4.21}$$

to perform the operation $\psi(\mathbf{R})$ that yields the $(r \times 1)$ vector

$$\tilde{\mathbf{p}} = \psi(\mathbf{R}) = \begin{bmatrix} (u_1) \cdot (v_1) \cdot (w_1) \cdots \cdot (z_1) \\ \vdots \\ (u_r) \cdot (v_r) \cdot (w_r) \cdots \cdot (z_r) \end{bmatrix} \tag{4.22}$$

Each element in the vector $\tilde{\mathbf{p}}$ is a scalar formed by the product of corresponding elements of the argument vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}, \cdots, \mathbf{z}$. Restated, the i^{th} element of the vector $\tilde{\mathbf{p}}$ results of multiplying the i^{th}elements of the vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}, \cdots, \mathbf{z}$, that is $\tilde{p_i} = (u_i) \cdot (v_i) \cdot (w_i) \cdots (v_i)$.

The result of operating on a set of $(r \times 1)$ degree-i homogeneous output vectors by the multiplicative operator $\psi(\bullet)$ is defined to have the standard notation $\tilde{\mathbf{Y}}_{(a,b,\cdots,w)}$. This process also satisfies (4.22), but the resultant vector notation includes the additional sub-index $(a, b, \cdots, w)$. This sub-index is used to identify the number of argument vectors with equal degree-i homogeneous outputs. In the notation, a is

the number of degree-1 homogeneous output argument vectors, b is the number of degree-2 homogeneous output argument vectors and so on. Finally w is the number of argument vectors having the maximum homogeneous degree.

Two examples are presented to illustrate the multiplicative operator $\psi(\bullet)$ notation. In the first example $\psi(\bullet)$ operates on the combination of two $(r \times 1)$ degree-1 homogeneous output vectors. In the second example $\psi(\bullet)$ operates on the combination of one $(r \times 1)$ degree-1 homogeneous output vector and one $(r \times 1)$ degree-2 homogeneous output vector. These examples will illustrate the use of the required sub-index when different combination of degree are present in the operator's argument list.

In the first example, the $(r \times 1)$ vector

$$\tilde{\mathbf{Y}}_{(2)} = \psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}) = [(Y_{(1),1}) \cdot (Y_{(1),1}), \cdots, (Y_{(1),r}) \cdot (Y_{(1),r})]^T \tag{4.23}$$

is the result of operating two degree-1 homogeneous $(r \times 1)$ output vectors $\mathbf{Y}_{(1)} = [(Y_{(1),1}), \cdots, (Y_{(1),r})]^T]$ (Figure 4.2). Each element of the vector $\tilde{\mathbf{Y}}_{(2)}$ is obtained by performing $\forall(1 \leq i \leq r)$ the operation $(Y_{(1),i}) \cdot (Y_{(1),i})$. The sub-index (2) in $\tilde{\mathbf{Y}}_{(2)}$ indicates that it is the result of operating on (2) degree-1 homogeneous output vectors. Because the indices in parentheses end after the first entry, no higher degree terms are indicated.

The resultant output $\tilde{\mathbf{Y}}_{(2)}$ can be also presented as a function of the external input $\mathbf{U}_1$. Substituting the first row of equation (4.14) into (4.23) yields,

$$\tilde{\mathbf{Y}}_{(2)} = \psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1)\right) \tag{4.24}$$

Using (4.24), Volterra functionals (4.14) that includes $\tilde{\mathbf{Y}}_{(2)}$, can be written as a function of the system external inputs.

In the second example, the $(r \times 1)$ vector

$$\tilde{\mathbf{Y}}_{(1,1)} = \psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(2)}) = [(Y_{(1),1}) \cdot (Y_{(2),1}), \cdots, (Y_{(1),r}) \cdot (Y_{(2),r})]^T \tag{4.25}$$

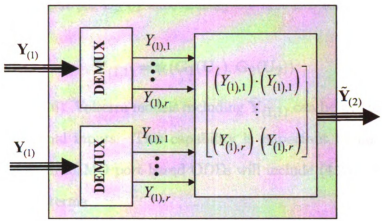

Figure 4.2. Block Diagram Representing the Operator $\psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)})$

is the result of operating on a combination of one degree-1 homogeneous ($r \times 1$) output vector, $\mathbf{Y}_{(1)} = [(Y_{(1),1}), \cdots, (Y_{(1),r})]^T$ with one degree-2 homogeneous ($r \times 1$) output vector, $\mathbf{Y}_{(2)} = [(Y_{(2),1}), \cdots, (Y_{(2),r})]^T]$ (Figure 4.3). Each element of the vector $\tilde{\mathbf{Y}}_{(1,1)}$ is obtained by performing $\forall(1 \leq i \leq r)$ the operation $(Y_{(1),i}) \cdot (Y_{(2),i})$. The sub-index (1,1) in $\tilde{\mathbf{Y}}_{(1,1)}$ indicates that it is the result of operating on one degree-1 homogeneous vector and one degree-2 homogeneous vector. Because no additional indices are included beyond the first two, no higher degree arguments are present.

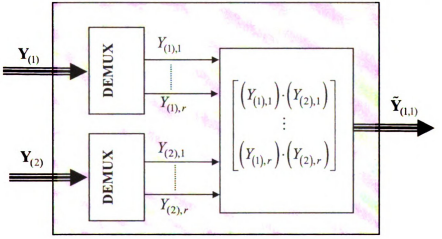

Figure 4.3. Block Diagram Representing the Operator $\psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(2)})$

The resultant output $\tilde{\mathbf{Y}}_{(1,1)}$ can be also presented as a function of the external

input $\mathbf{U}_1$ and the input set $\mathbf{U}_2$ by substituting the first two rows of equation (4.14) into (4.25) to yield,

$$\tilde{\mathbf{Y}}_{(1,1)} = \psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_2(\mathbf{U}_2)\right) \tag{4.26}$$

Using (4.24) and (4.26), Volterra models including $\tilde{\mathbf{Y}}_{(1,1)}$ can be written as a function of the system external inputs. This capability is important because the Volterra models obtained from MIMO port-based ODEs will include (4.23), (4.24) and other similar higher order terms.

4.5 Generating Volterra Models From Port-Based ODEs

The Volterra model of a port-based MIMO nonlinear ODE model in the form (4.1), is obtained in this section. Assuming that this system is operating around an equilibrium point (op) located at the origin $(\mathbf{y}_0, \mathbf{u}_0) = (\mathbf{0}, \mathbf{0})$ and using the vector notation

$$\mathbf{a}^n = \left[a_1^n, \cdots, a_r^n\right]^T \tag{4.27}$$

a Taylor series expansion is applied around this equilibrium point $\forall(1 \leq i, j \leq r)$ and $\forall(1 \leq k \leq m)$, (m is the maximum time derivative of the outputs) to yield,

$$\sum_{n=1}^{\infty}\left[\frac{1}{n!}\left(\left[\left(\frac{\partial^n f_i}{\partial y_j^n}\right)|_{op}\right]\mathbf{y}^n + \left[\left(\frac{\partial^n f_i}{\partial \dot{y}_j^n}\right)|_{op}\right]\dot{\mathbf{y}}^n + \left[\left(\frac{\partial^n f_i}{\partial \ddot{y}_j^n}\right)|_{op}\right]\ddot{\mathbf{y}}^n + \cdots\right)\right] = \mathbf{u} \tag{4.28}$$

Defining the $(r \times r)$ matrix of constants,

$$\left[b_{i,j}\right]_{(n,k)} = \left[\left(\frac{\partial^n f_i}{\partial\left[d^k y_j^n / dt^k\right]}|_{op}\right)\right] \tag{4.29}$$

For an specific k and n, each constant $b_{i,j}$ in the matrix, is the n^{th} partial derivative of the scalar function $f_i(\mathbf{y}, \dot{\mathbf{y}}, \cdots)$ respect the variable $[d^k y_j^n / dt^k]$ evaluated at (op).

Using a constant α, a new input in the form

$$\tilde{\mathbf{u}} = \alpha\mathbf{u} = \alpha\left[u_1(t_1), \cdots, u_r(t_1)\right]^T \tag{4.30}$$

is defined. If this new input is applied to (4.28), the resultant output $\tilde{\mathbf{y}}$ is a MIMO extention of the SISO form shown in [28],

$$\tilde{\mathbf{y}} = \alpha \mathbf{y}_1(t_1) + \alpha^2 \mathbf{y}_2(t_1, t_2) + \alpha^3 \mathbf{y}_3(t_1, t_2, t_3) + \cdots \tag{4.31}$$

Equation (4.31) is a direct result of the "degree-i homogeneous system" properties (4.5) and is extended here to the vector case. Defining $\mathbf{y}_{(i)} = \mathbf{y}_i(t_1, \cdots, t_i)$, (4.31) can be written in the form,

$$\tilde{\mathbf{y}} = \alpha \mathbf{y}_{(1)} + \alpha^2 \mathbf{y}_{(2)} + \alpha^3 \mathbf{y}_{(3)} + \cdots \tag{4.32}$$

Substituting (4.30) and (4.32) $\forall (1 \leq i, j \leq r)$ into (4.28) yields,

$$\begin{gathered}
\left[b_{i,j}\right]_{(1,0)}\left(\alpha \mathbf{y}_{(1)} + \alpha^2 \mathbf{y}_{(2)} + \cdots\right) + \left[b_{i,j}\right]_{(1,1)}\left(\alpha \dot{\mathbf{y}}_{(1)} + \alpha^2 \dot{\mathbf{y}}_{(2)} + \cdots\right) + \cdots + \\
\tfrac{1}{2!}\left[b_{i,j}\right]_{(2,0)}\left(\alpha \mathbf{y}_{(1)} + \alpha^2 \mathbf{y}_{(2)} + \cdots\right)^2 + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,1)}\left(\alpha \dot{\mathbf{y}}_{(1)} + \alpha^2 \dot{\mathbf{y}}_{(2)} + \cdots\right)^2 + \cdots + \\
\tfrac{1}{3!}\left[b_{i,j}\right]_{(3,0)}\left(\alpha \mathbf{y}_{(1)} + \alpha^2 \mathbf{y}_{(2)} + \cdots\right)^3 + \tfrac{1}{3!}\left[b_{i,j}\right]_{(3,1)}\left(\alpha \dot{\mathbf{y}}_{(1)} + \alpha^3 \dot{\mathbf{y}}_{(2)} + \cdots\right)^3 + \cdots = \alpha \mathbf{u}
\end{gathered} \tag{4.33}$$

Terms multiplied by a particular $\alpha^p, \forall (1 \leq p \leq \infty)$ in (4.33), are independent from all others because α is an arbitrary constant. This property is used by Schetzen [28] to determine all the SISO Volterra transfer functions in (4.8). Extending this procedure to the MIMO case, all the Volterra operator $\mathbf{G}_p(\bullet)$ in (4.11) $\forall (1 \leq p \leq \infty)$ can be derived from the terms that are multiplied by the constant α^p.

The first term in the expansion (4.11) is the linear Volterra operator $\mathbf{G}_1(\bullet)$. This term is derived from the elements multiplied by α^1,

$$\left[b_{i,j}\right]_{(1,0)}\mathbf{y}_{(1)} + \left[b_{i,j}\right]_{(1,1)}\dot{\mathbf{y}}_{(1)} + \left[b_{i,j}\right]_{(1,2)}\ddot{\mathbf{y}}_{(1)} + \cdots + \left[b_{i,j}\right]_{(1,m)}\frac{d^m \mathbf{y}_{(1)}}{dt^m} = \mathbf{u} \tag{4.34}$$

where m is the maximum output derivative. Define the time dependent 1^{st} order vector operator,

$$\mathbf{p}_1(\mathbf{x}) = \left[b_{i,j}\right]_{(1,0)}\mathbf{x} + \left[b_{i,j}\right]_{(1,1)}\dot{\mathbf{x}} + \left[b_{i,j}\right]_{(1,2)}\ddot{\mathbf{x}} + \cdots + \left[b_{i,j}\right]_{(1,m)}\frac{d^m \mathbf{x}}{dt^m} \tag{4.35}$$

where $\mathbf{x}$ is an arbitrary $(r \times 1)$ vector, to write (4.34) in the form,

$$\mathbf{p}_1\left(\mathbf{y}_{(1)}\right) = \mathbf{u} \tag{4.36}$$

Applying the Laplace transform to (4.36) with respect to t_1 variable yields,

$$\mathbf{P}_{1,(1)}\mathbf{Y}_{(1)} = \mathbf{U}_1 \tag{4.37}$$

where $\mathbf{Y}_{(1)}$ and $\mathbf{U}_1$ are respectively the Laplace transform of $\mathbf{y}_{(1)}$ and $\mathbf{u}$ and,

$$\mathbf{P}_{1,(1)} = \left[b_{i,j}\right]_{(1,m)} s_1^m + \cdots + \left[b_{i,j}\right]_{(1,2)} s_1^2 + \left[b_{i,j}\right]_{(1,1)} s_1 + \left[b_{i,j}\right]_{(1,0)}$$

is a $(r \times r)$ dynamic matrix of polynomials in the complex variable s_1. The elements between parenthesis in the sub-index of the matrix $\mathbf{P}_{1,(1)}$ indicates that this matrix is function of one independent complex variable. All the above elements of $\mathbf{P}_{1,(1)}$ in brackets are $(r \times r)$ matrices of constants. Inverting $\mathbf{P}_{1,(1)}$ yields,

$$\mathbf{Y}_{(1)} = \mathbf{G}_{1,(1)}\mathbf{U}_1 \tag{4.38}$$

where

$$\mathbf{G}_{1,(1)} = \left[\left[b_{i,j}\right]_{(1,m)} s_1^m + \cdots + \left[b_{i,j}\right]_{(1,2)} s_1^2 + \left[b_{i,j}\right]_{(1,1)} s_1 + \left[b_{i,j}\right]_{(1,0)}\right]^{-1} \tag{4.39}$$

is the transfer function matrix. Equation (4.38) is equivalent to the linearized model of the nonlinear system (4.28).

The second term in the expansion (4.11) is the nonlinear operator $\mathbf{G}_2(\mathbf{U}_2)$. This term is found by equating the elements that are multiplied by the coefficient α^2 in both sides of (4.33),

$$\begin{aligned} &\left[b_{i,j}\right]_{(1,0)}\mathbf{y}_{(2)} + \left[b_{i,j}\right]_{(1,1)}\dot{\mathbf{y}}_{(2)} + \cdots + \left[b_{i,j}\right]_{(1,m)}\frac{d^m\mathbf{y}_{(2)}}{dt^m} + \\ &\frac{1}{2!}\left[b_{i,j}\right]_{(2,0)}\mathbf{y}_{(1)}^2 + \frac{1}{2!}\left[b_{i,j}\right]_{(2,1)}\dot{\mathbf{y}}_{(1)}^2 + \cdots + \frac{1}{2!}\left[b_{i,j}\right]_{(2,m)}\frac{d^m\mathbf{y}_{(1)}^2}{dt^m} = \mathbf{0} \end{aligned} \tag{4.40}$$

Using the vector notation (4.27) and the operator (4.22), the $(r \times 1)$ vector

$$\tilde{\mathbf{y}}_{(2)} = \mathbf{y}_{(1)}^2 \tag{4.41}$$

The definition (4.35) of operator $\mathbf{p}_1(\bullet)$ and (4.41) are applied to (4.40) to yields,

$$\mathbf{p}_1\left(\mathbf{y}_{(2)}\right) + \mathbf{p}_2\left(\tilde{\mathbf{y}}_{(2)}\right) = \mathbf{0} \tag{4.42}$$

where the second order functional

$$\mathbf{p}_2(\mathbf{x}) = \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,0)}\mathbf{x}^2 + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,1)}\dot{\mathbf{x}}^2 + \cdots + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,m)}\frac{d^m\,\mathbf{x}^2}{dt^m} \tag{4.43}$$

Applying the Laplace transform to (4.42) with respect to two independent time variables and using the Theorem 2.3 in Rugh [10] yields,

$$\mathbf{P}_{1,(2)}\mathbf{Y}_{(2)} + \mathbf{P}_{2,(2)}\tilde{\mathbf{Y}}_{(2)} = \mathbf{0} \tag{4.44}$$

where the $(r \times r)$ matrices

$$\mathbf{P}_{1,(2)} = \left[b_{i,j}\right]_{(1,m)}\left(s_1+s_2\right)^m + \cdots + \left[b_{i,j}\right]_{(1,1)}\left(s_1+s_2\right) + \left[b_{i,j}\right]_{(1,0)}$$

$$\mathbf{P}_{2,(2)} = \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,m)}\left(s_1+s_2\right)^m + \cdots + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,1)}\left(s_1+s_2\right) + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,0)}$$

The vector $\tilde{\mathbf{Y}}_{(2)}$ in (4.45) is defined in (4.24). Substituting this result into (4.45) yields

$$\mathbf{P}_{1,(2)}\mathbf{Y}_{(2)} + \mathbf{P}_{2,(2)}\psi\left(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}\right) = \mathbf{0} \tag{4.45}$$

Solving for the $(r \times 1)$ vector $\mathbf{Y}_{(2)}$ by inverting the matrix $\mathbf{P}_{1,(2)}$,

$$\mathbf{Y}_{(2)} = -\mathbf{G}_{1,(2)}\mathbf{P}_{2,(2)}\psi\left(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}\right) \tag{4.46}$$

where $\mathbf{G}_{1,(2)} = \left[\mathbf{P}_{1,(2)}\right]^{-1}$. Finally substituting (4.38) into (4.46) yields,

$$\mathbf{Y}_{(2)} = -\mathbf{G}_{1,(2)}\mathbf{P}_{2,(2)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1\right) = \mathbf{G}_2(\mathbf{U}_2) \tag{4.47}$$

The output vector $\mathbf{Y}_{(2)}$ is obtained (Figure 4.4) by multiplying each of the two inputs $\mathbf{U}_1$ vectors by the $\mathbf{G}_{1,(1)}$ matrices. The two $\mathbf{Y}_{(1)}$ resultant outputs, are operated on by $\psi(\bullet)$ to yield the vector $\tilde{\mathbf{Y}}_{(2)}$. This vector is then operated by the matrix $\mathbf{P}_{2,(2)}$. The resultant output is finally operated by the matrix $-\mathbf{G}_{1,(2)}$.

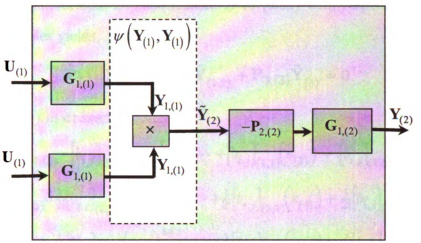

Figure 4.4. Nonlinear Operator $\mathbf{G}_2(\mathbf{U}_2)$

The third term in the expansion (4.13) is the nonlinear operator $\mathbf{G}_3(\mathbf{U}_3)$. This term is found by equating the coefficients of α^3 in both sides of (4.33),

$$\begin{aligned}
&\left[b_{i,j}\right]_{(1,0)}\mathbf{y}_{(3)}+\cdots+\left[b_{i,j}\right]_{(1,m)}\frac{d^m\mathbf{y}_{(3)}}{dt^m}+\\
&+\tfrac{1}{2!}\left[b_{i,j}\right]_{(2,0)}\begin{Bmatrix} y_{(1),1}y_{(2),1} \\ \vdots \\ y_{(1),r}y_{(2),r}\end{Bmatrix}+\cdots+\tfrac{1}{2!}\left[b_{i,j}\right]_{(2,m)}\frac{d^m}{dt^m}\begin{Bmatrix} y_{(1),1}y_{(2),1} \\ \vdots \\ y_{(1),r}y_{(2),r}\end{Bmatrix}^2 \qquad (4.48)\\
&\tfrac{1}{3!}\left[b_{i,j}\right]_{(3,0)}\mathbf{y}^3_{(1)}+\cdots+\tfrac{1}{3!}\left[b_{i,j}\right]_{(3,m)}\frac{d^m\mathbf{y}^3_{(1)}}{dt^m}=\mathbf{0}
\end{aligned}$$

The vectors

$$\tilde{\mathbf{y}}_{(1,1)}=\begin{Bmatrix} y_{(1),1}y_{(2),1} \\ \vdots \\ y_{(1),r}y_{(2),r}\end{Bmatrix} \qquad (4.49)$$

$$\tilde{\mathbf{y}}_{(3)}=\mathrm{y}^3_{(1)} \qquad (4.50)$$

and the functional (4.35) are substituted into (4.48) and,

$$\mathbf{p}_1\left(\mathbf{y}_{(3)}\right)+\mathbf{p}_2\left(\tilde{\mathbf{y}}_{(1,1)}\right)+\mathbf{p}_3\left(\tilde{\mathbf{y}}_{(3)}\right)=\mathbf{0} \qquad (4.51)$$

where the functional

$$\mathbf{p}_3(\mathbf{x})=\tfrac{1}{3!}\left[b_{i,j}\right]_{(3,0)}\mathbf{x}^3+\tfrac{1}{3!}\left[b_{i,j}\right]_{(3,1)}\dot{\mathbf{x}}^3+\cdots+\tfrac{1}{3!}\left[b_{i,j}\right]_{(3,m)}\frac{d^m\mathbf{x}^3}{dt^m} \qquad (4.52)$$

Applying the multivariable Laplace transform to (4.51) with respect three independent time variables yields,

$$\mathbf{P}_{1,(3)}\mathbf{Y}_{(3)} + \mathbf{P}_{2,(3)}\tilde{\mathbf{Y}}_{(1,1)} + \mathbf{P}_{3,(3)}\tilde{\mathbf{Y}}_{(3)} = \mathbf{0} \tag{4.53}$$

where the $(r \times r)$ matrices,

$$\mathbf{P}_{1,(3)} = \left[b_{i,j}\right]_{(1,m)}\left(s_{(3)}\right)^m + \cdots + \left[b_{i,j}\right]_{(1,1)}\left(s_{(3)}\right) + \left[b_{i,j}\right]_{(1,0)}$$

$$\mathbf{P}_{2,(3)} = \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,m)}\left(s_{(3)}\right)^m + \cdots + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,1)}\left(s_{(3)}\right) + \tfrac{1}{2!}\left[b_{i,j}\right]_{(2,0)}$$

$$\mathbf{P}_{3,(3)} = \tfrac{1}{3!}\left[b_{i,j}\right]_{(3,m)}\left(s_{(3)}\right)^m + \cdots + \tfrac{1}{3!}\left[b_{i,j}\right]_{(3,1)}\left(s_{(3)}\right) + \tfrac{1}{3!}\left[b_{i,j}\right]_{(3,0)}$$

and $s_{(3)} = s_1 + s_2 + s_3$. The nonlinear operator (4.22) is used to evaluate the vector,

$$\tilde{\mathbf{Y}}_3 = \psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1)\right) \tag{4.54}$$

Substituting (4.54) and (4.32) into (4.53) yields,

$$\begin{aligned}\mathbf{P}_{1,(3)}\mathbf{Y}_{(3)} + \mathbf{P}_{2,(3)}\psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_2(\mathbf{U}_2)\right) + \\ \mathbf{P}_{3,(3)}\psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1)\right) = \mathbf{0}\end{aligned} \tag{4.55}$$

Solving for the vector $\mathbf{Y}_{(3)}$ by inverting the matrix $\mathbf{P}_{1,(3)}$ yields,

$$\begin{aligned}\mathbf{Y}_{(3)} = -\mathbf{G}_{1,(3)}\mathbf{P}_{3,(3)}\psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1)\right) - \\ \mathbf{G}_{1,(3)}\mathbf{P}_{2,(3)}\psi\left(\mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_2(\mathbf{U}_2)\right)\end{aligned} \tag{4.56}$$

where $\mathbf{G}_{1,(3)} = \left[\mathbf{P}_{1,(3)}\right]^{-1}$. Finally by substituting (4.47) and (4.38) into (4.56) yields,

$$\begin{aligned}\mathbf{Y}_{(3)} = \mathbf{G}_3(\mathbf{U}_3) = -\mathbf{G}_{1,(3)}\mathbf{P}_{3,(3)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1\right) - \\ \mathbf{G}_{1,(3)}\mathbf{P}_{2,(3)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(2)}\mathbf{P}_{2,(2)}\psi\left(\mathbf{G}_{1,(1)}\mathbf{U}_1, \mathbf{G}_{1,(1)}\mathbf{U}_1\right)\right)\end{aligned} \tag{4.57}$$

The addition of two third order components (Figure 4.5) is required to compute the output vector $\mathbf{Y}_{(3)}$. Using a similar procedure, the i^{th} Volterra functional $\mathbf{G}_i(\mathbf{U}_i)$ can also be calculated. Any $\mathbf{P}_{u,(v)}$ matrix of this Volterra operator can be calculated using the equation,

$$\mathbf{P}_{u,(v)} = \tfrac{1}{u!}\left[b_{i,j}\right]_{(u,m)}\left(s_{(v)}\right)^m + \cdots + \tfrac{1}{u!}\left[b_{i,j}\right]_{(u,1)}\left(s_{(v)}\right) + \tfrac{1}{u!}\left[b_{i,j}\right]_{(u,0)} \tag{4.58}$$

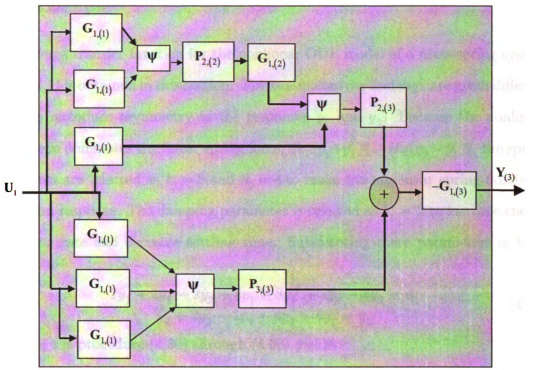

Figure 4.5. Nonlinear Operator $\mathbf{G}_3(\mathbf{U}_3)$

where $s_{(v)} = (s_1 + \cdots + s_v)$.

The Volterra functional in (4.13) requires an infinite number of expansions. A realizable Volterra functional uses only a finite number of expansion terms and is a truncated version of the general Volterra model (4.13). For example, a Volterra functional truncated at the third expansion is obtained by adding the Volterra models (4.38), (4.47) and (4.57)

$$\mathbf{Y}_{1-3} = \mathbf{G}_1(\mathrm{U}_1) + \mathbf{G}_2(\mathbf{U}_2) + \mathbf{G}_3(\mathbf{U}_3) \tag{4.59}$$

To obtain a truncated time solution of (4.59), the inverse Laplace transform is applied sequentially with respect to the time variables t_1, t_2 and then t_3. The result is the multi-time variable truncated solution $\mathbf{y}_{1-3}(t_1, t_2, t_3)$. Finally the condition $t = t_1 = t_2 = t_3$ is applied to obtain the single variable solution $\mathbf{y}_{1-3}(t)$ [28]-[10].

4.6 Example

The Volterra transfer function for the nonlinear ODE model of a mass-spring system (Figure 4.1) is obtained in this section. The two masses m_1 and m_2 , are given different values to introduce asymmetry in the responses y_1 and y_2. Because the nonlinear spring force dominates the linear spring force, $\|k_1(y_1+y_1^3)\| \geq \|k_2(y_2-y_1)\|$, the spring parameters are selected as $k_1=8$ and $k_2=4$ to make the nonlinear spring dominate the system response. The damping parameter is selected as $b_1=2$ to increase energy dissipation rate and decrease settling time. Substituting these parameters in (4.2) yields,

$$\begin{aligned} 2\ddot{y}_1 + 2\dot{y}_1 - 2\dot{y}_2 + 8y_1 + 8y_1^3 + 4y_1 - 4y_2 &= u_1 \\ \ddot{y}_2 + 2\dot{y}_2 - 2\dot{y}_1 + 4y_2 - 4y_1 &= u_2 \end{aligned} \tag{4.60}$$

Following the procedure (4.30) through (4.33) yields,

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \left(\alpha\ddot{\mathbf{y}}_{(1)} + \alpha^2\ddot{\mathbf{y}}_{(2)} + \cdots\right) + \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \left(\alpha\dot{\mathbf{y}}_{(1)} + \alpha^2\dot{\mathbf{y}}_{(2)} + \cdots\right) +$$
$$\begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix} \left(\alpha\mathbf{y}_{(1)} + \alpha^2\mathbf{y}_{(2)} + \cdots\right) + \begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \left(\alpha\mathbf{y}_{(1)} + \alpha^2\mathbf{y}_{(2)} + \cdots\right)^3 = \alpha\mathbf{u} \tag{4.61}$$

The linearized transfer function matrix $\mathbf{G}_{1,(1)}$ is found by equating the elements multiplied by α^1 in (4.61),

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \ddot{\mathbf{y}}_{(1)} + \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \dot{\mathbf{y}}_{(1)} + \begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix} \mathbf{y}_{(1)} = \mathbf{u} \tag{4.62}$$

Defining the operator

$$\mathbf{p}_1(\mathbf{x}) = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \ddot{\mathbf{x}} + \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \dot{\mathbf{x}} + \begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix} \mathbf{x} \tag{4.63}$$

where ($\mathbf{x} \epsilon \Re^2$), the equation (4.62) can be written as,

$$\mathbf{p}_1\left(\mathbf{y}_{(1)}\right) = \mathbf{u} \tag{4.64}$$

Applying the Laplace transform with respect to one time independent variable yields and zero initial conditions yields,

$$\mathbf{P}_{1,(1)}\mathbf{Y}_{(1)} = \mathbf{U}_1 \tag{4.65}$$

where $\mathbf{Y}_{(1)}$ and $\mathbf{U}_{(1)}$ are the Laplace transform respect t_1 of the vectors $\mathbf{y}_{(1)}$ and $\mathbf{u}$ and the dynamic matrix

$$\mathbf{P}_{1,(1)} = \begin{bmatrix} 2s_1^2 + 2s_1 + 12 & -2s_1 - 4 \\ -2s_1 - 4 & s_1^2 + 2s_1 + 4 \end{bmatrix}$$

Solving for $\mathbf{Y}_{(1)}$ yields,

$$\mathbf{Y}_{(1)} = \mathbf{G}_{1,(1)}\mathbf{U}_1 \tag{4.66}$$

where

$$\mathbf{G}_{1,(1)} = \begin{bmatrix} \frac{s_1^2+2s_1+4}{2s_1^4+6s_1^3+20s_1^2+16s_1+32} & \frac{2s_1+4}{2s_1^4+6s_1^3+20s_1^2+16s_1+32} \\ \frac{2s_1+4}{2s_1^4+6s_1^3+20s_1^2+16s_1+32} & \frac{2s_1^2+2s_1+12}{2s_1^4+6s_1^3+20s_1^2+16s_1+32} \end{bmatrix}$$

is the linearized matrix transfer function and is equal to the inverse of the linearized dynamic matrix $\mathbf{P}_{1,(1)}$.

The term $\mathbf{G}_2(\mathbf{U}_2)$, is found by equating the elements multiplied by the coefficient α^2 in (4.61),

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\ddot{\mathbf{y}}_{(2)} + \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}\dot{\mathbf{y}}_{(2)} + \begin{bmatrix} 12 & 4 \\ -4 & 4 \end{bmatrix}\mathbf{y}_{(2)} = \mathbf{0} \tag{4.67}$$

Using the functional (4.63), equation (4.67) can be written in the form,

$$\mathbf{p}_1\left(\mathbf{y}_{(2)}\right) = \mathbf{0} \tag{4.68}$$

Taking the laplace transform of (4.68) for zero initial conditions yields,

$$\mathbf{P}_{1,(2)}\mathbf{Y}_{(2)} = \mathbf{0} \tag{4.69}$$

For nonsingular $\mathbf{P}_{1,(2)}$, $\mathbf{Y}_{(2)} = \mathbf{0}$. This result, $\mathbf{Y}_{(2)} = \mathbf{0}$, is expected because the nonlinear system is an odd function. This result shows the second Volterra operator,

$$\mathbf{G}_2(\mathbf{U}_2) = \mathbf{Y}_{(2)} = \mathbf{0} \tag{4.70}$$

The term $\mathbf{G}_3(\mathbf{U}_3)$, is found by equating the elements multiplied by the coefficient α^3 in (4.61),

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \ddot{\mathbf{y}}_{(3)} + \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \dot{\mathbf{y}}_{(3)} + \begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix} \mathbf{y}_{(3)} + \begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{y}^3_{(3)} = \mathbf{0} \tag{4.71}$$

Using the functional (4.63) and substituting (4.50) into (4.71) yields,

$$\mathbf{p}_1\left(\mathbf{y}_{(3)}\right) = -\begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \tilde{\mathbf{y}}_{(3)} \tag{4.72}$$

Applying the Laplace transform to (4.72) with respect to three time independent variable yields,

$$\mathbf{P}_{1,(3)} \mathbf{Y}_{(3)} = -\begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \tilde{\mathbf{Y}}_{(3)} \tag{4.73}$$

Substituting the multiplicative operator (4.54) into (4.73),

$$\mathbf{P}_{1,(3)} \mathbf{Y}_{(3)} = -\mathbf{P}_3 \psi\left(\mathbf{G}_{(1)}(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1), \mathbf{G}_1(\mathbf{U}_1)\right) \tag{4.74}$$

where $\mathbf{P}_3 = \begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix}$ and

$$\mathbf{P}_{1,(3)} = \begin{bmatrix} 2s^2_{(3)} + 2s_{(3)} + 12 & -2s_{(3)} - 4 \\ -2s_{(3)} - 4 & s^2_{(3)} + 2s_{(3)} + 4 \end{bmatrix}$$

Solving for $\mathbf{Y}_{(3)}$ by inverting $\mathbf{P}_{1,(3)}$ and substituting (4.66) in (4.74) yields,

$$\mathbf{Y}_{(3)} = -\mathbf{G}_{1,(3)} \mathbf{P}_3 \psi\left(\mathbf{G}_{1,(1)} \mathbf{U}_1, \mathbf{G}_{1,(1)} \mathbf{U}_1, \mathbf{G}_{1,1} \mathbf{U}_1\right) \tag{4.75}$$

where $\mathbf{G}_{1,(3)} = \left[\mathbf{P}_{1,(3)}\right]^{-1}$. Equation (4.75) is the third term in the Volterra model. The Volterra model truncated at the third expansion includes equations (4.66) and (4.75) (Figure 4.6),

$$\mathbf{Y}_{1-3} = \mathbf{G}_1(\mathbf{U}_1) + \mathbf{G}_3(\mathbf{U}_3) \tag{4.76}$$

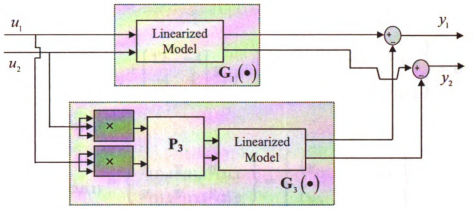

Figure 4.6. Simulink Diagram for the Volterra Model Including the First and the Third Expansions

The term $\mathbf{G_4(U_4)}$, is found by equating the elements multiplied by α^4 in (4.61),

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \ddot{\mathbf{y}}_{(4)} + \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix} \dot{\mathbf{y}}_{(4)} + \begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix} \mathbf{y}_{(4)} + \begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} 3\left(y_{(1),1}\right)^2 y_{(2),1} \\ 3\left(y_{(1),2}\right)^2 y_{(2),2} \end{Bmatrix} = \mathbf{0} \tag{4.77}$$

Using the operator (4.64), equation (4.77) can be written in the form,

$$\mathbf{p_1}\left(\mathbf{y}_{(4)}\right) = -\begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} 3\left(y_{(1),1}\right)^2 y_{(2),1} \\ 3\left(y_{(1),2}\right)^2 y_{(2),2} \end{Bmatrix} \tag{4.78}$$

Using (4.70)

$$\mathbf{y}_{(2)} \begin{bmatrix} y_{(2),1} \\ y_{(2),2} \end{bmatrix} = \mathbf{0} \tag{4.79}$$

into (4.78),

$$\mathbf{p_1}\left(\mathbf{y}_{(4)}\right) = \mathbf{0} \Longrightarrow \mathbf{G_4(U_4)} = \mathbf{0} \tag{4.80}$$

Result (4.80) is expected because the nonlinearity is an odd function.

The term $\mathbf{G_5(U_5)}$, is found by equating the elements multiplied by the coefficient

α^5 in (4.61),

$$\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \ddot{\mathbf{y}}_{(5)} + \begin{bmatrix} 2 & 2 \\ -2 & 2 \end{bmatrix} \dot{\mathbf{y}}_{(5)} + \begin{bmatrix} 12 & -4 \\ -4 & 4 \end{bmatrix} \mathbf{y}_{(5)} + \begin{bmatrix} 8 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} 3\left(y_{(1),1}\right)^2 y_{(3),1} \\ 3\left(y_{(1),2}\right)^2 y_{(3),2} \end{Bmatrix} = \mathbf{0} \quad (4.81)$$

Defining the vector,

$$\tilde{\mathbf{y}}_{(2,0,1)} = \begin{Bmatrix} y_{(1),1} y_{(1),1} y_{(3),1} \\ y_{(1),2} y_{(1),2} y_{(3),2} \end{Bmatrix} = \begin{Bmatrix} \left(y_{(1),1}\right)^2 y_{(3),1} \\ \left(y_{(1),2}\right)^2 y_{(3),2} \end{Bmatrix} \quad (4.82)$$

Substituting (4.82) into (4.81) and using the operator in (4.63) yields,

$$\mathbf{p}_1\left(\mathbf{y}_{(5)}\right) = -\begin{bmatrix} 24 & 0 \\ 0 & 0 \end{bmatrix} \tilde{\mathbf{y}}_{(2,0,1)} \quad (4.83)$$

Applying the Laplace Transform to (4.83) respect five independent time variables yields,

$$\mathbf{P}_{1,(5)} \mathbf{Y}_{(5)} = -3\mathbf{P}_3 \tilde{\mathbf{Y}}_{(2,0,1)} \quad (4.84)$$

where

$$\mathbf{P}_{1,(5)} = \begin{bmatrix} 2s_{(5)}^2 + 2s_{(5)} + 12 & -2s_{(5)} - 4 \\ -2s_{(5)} - 4 & s_{(5)}^2 + 2s_{(5)} + 4 \end{bmatrix} \quad (4.85)$$

Substituting the vector

$$\tilde{\mathbf{Y}}_{(2,0,1)} = \begin{bmatrix} \left(Y_{(1),1}\right)\left(Y_{(1),1}\right)\left(Y_{(3),1}\right) \\ \left(Y_{(1),2}\right)\left(Y_{(1),2}\right)\left(Y_{(3),2}\right) \end{bmatrix} = \psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}, \mathbf{Y}_{(3)}) \quad (4.86)$$

and inverting the matrix $\mathbf{P}_{1,(5)}$ into (4.84) yields,

$$\mathbf{Y}_{(5)} = -\mathbf{G}_{1,(5)} \mathbf{P}_5 \psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}, \mathbf{Y}_{(3)}) \quad (4.87)$$

where $\mathbf{G}_{1,(5)} = \left[\mathbf{P}_{1,(5)}\right]^{-1}$ and $\mathbf{P}_5 = 3\mathbf{P}_3$ and previous results (4.66), and (4.75) yield,

$$\mathbf{Y}_{(1)} = \mathbf{G}_{1,(1)} \mathbf{U}_1$$

$$\mathbf{Y}_{(3)} = -\mathbf{G}_{1,(3)} \mathbf{P}_3 \psi\left(\mathbf{G}_{1,(1)} \mathbf{U}_1, \mathbf{G}_{1,(1)} \mathbf{U}_1, \mathbf{G}_{1,(1)} \mathbf{U}_1\right)$$

all terms of the fifth order expansion can now be computed from the linearized transfer function $\mathbf{G}_{1,(\upsilon)}$

The Volterra model truncated at the fifth expansion, includes equations (4.67), (4.77) and (4.87) (Figure 4.7)

$$\mathbf{Y}_{1-5} = \mathbf{G}_1(\mathbf{U}_1) + \mathbf{G}_3(\mathbf{U}_3) + \mathbf{G}_5(\mathbf{U}_5) \tag{4.88}$$

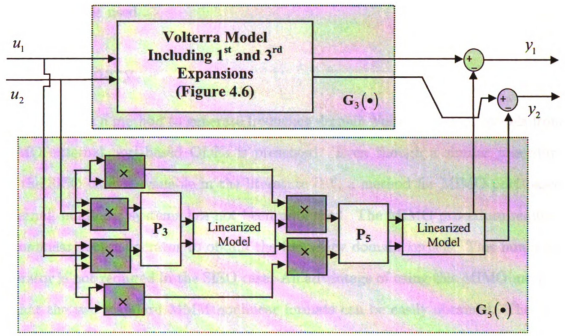

Figure 4.7. Simulink Diagram for the Volterra Model Including the First, the Third and the Fifth Expansions

The time responses of the truncated Volterra models in (4.66), (4.75) and (4.87) are compared to the response of the nonlinear ODE model (4.60). The two inputs used to excite the system are sine waves with amplitudes -0.3 and -0.15 at frequencies of 2.27 rad/sec and 1.5 rad/sec respectively. These frequencies are the two natural frequencies of the linearized system and are determined by calculating the roots of the characteristic equation of $\mathbf{G}_{1,(1)}$ in (4.66). The linearized model exhibits the greatest error between all the three responses. The error shown by the truncated Volterra

model that includes the linearized and the third expansion terms is substantially smaller than the error exhibit by the linearized system (Figures 4.8 and 4.9). The error exhibit by the truncated Volterra model that includes the linearized, the third and the fifth order expansion term is the smallest of all the three responses. For a specific bound at the input, the more expansion the Volterra model includes, the smaller is its error. Theoretically, if the Volterra series is convergent, the error between the Volterra model and the nonlinear ODE will tend to zero when an infinite number of expansions is used .

4.7 Summary

In this chapter a method to generate frequency domain Volterra system models from MIMO external port-based ODEs is presented. Even though a similar procedure for the SISO case is available in the literature [10], a method for MIMO port-based external, nonlinear systems has not been published. The MIMO procedure requires a nonlinear vector operator to obtain the frequency domain kernels. This nonlinear operator is not required in the SISO case. An advantage of using this MIMO method is that the two standard MMM nonlinear formats can be easily obtained. The first format is the Volterra transfer function representation. This format is traditionally used in model analysis and simulation. The second format is the Volterra dynamic representation. This format is used in the assembly of nonlinear system models. The Volterra dynamic representation was recognized and defined for first time in this work. The Volterra dynamic representation is important in the MMM because this external port-based model format protects internal design details. This procedure generates the Volterra dynamic port-based system models in an explicit form.

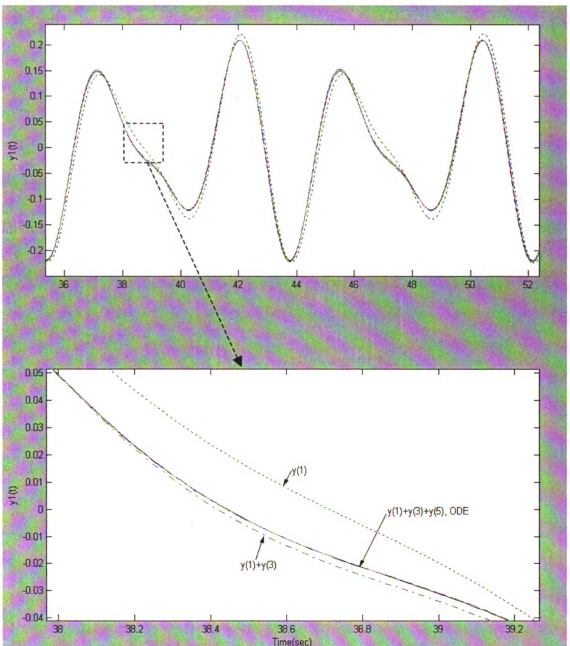

Figure 4.8. Nonlinear ODE response $y_1(t)$ compared with different truncated Volterra transfer function responses. $y(1)$ is the linearized response, $y(3)$ is the third expansion and $y(5)$ is the fifth expansion.

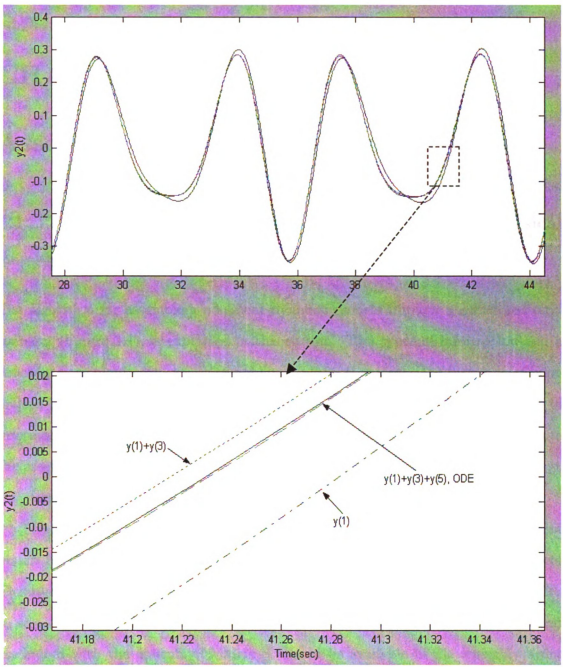

Figure 4.9. Nonlinear ODE response $y_2(t)$ compared with different truncated Volterra transfer function responses. y(1) is the linearized response, y(3) is the third expansion and y(5) is the fifth expansion.

CHAPTER 5

Assembly of Nonlinear Physical Systems

5.1 The MMM Algorithm to Assemble Nonlinear Physical Models

The algorithm for assembling nonlinear dynamic models through a networked environment (Figure 2.1), is explained in this section. Two types of information and a procedure are required to assemble nonlinear physical models. The first type of information is the component connectivity structure of the assembly. The second type of information is the Volterra dynamic model for each of the components that constitute the assembly. Once these two sets of information are available at the assembly agent, the standard procedure to assemble the component models is initiated.

The component connectivity structure of an assembly is enclosed in its constraint matrix **S**. The matrix **S** is dependent on the order of connection between components. This matrix equates each component output in the unconstrained output vector $\mathbf{Y}_c$ to an assembly output in the assembly vector $\mathbf{Y}_a$ based on the physical constrained associated with the assembly. Matrix **S** should be available in the assembly agent before the assemble procedure is initiated .

If the i^{th} component model includes q expansion terms, the component Volterra dynamic model has the form,

$$\left\{\mathbf{P}_{i,1,(v)}, \mathbf{P}_{i,2,(v)}, \cdots, \mathbf{P}_{i,q,(v)}\right\} \tag{5.1}$$

For any dynamic matrix $\mathbf{P}_{i,j,(v)}$, the index i, j and v respectively indicates the component number, the expansion term and the number of independent complex variables

The standard procedure to assembly Volterra dynamic models include 2 steps. In the *first* step, the Assembly Agent uses all the Volterra dynamic component dynamic models to build the unconstrained Volterra dynamic model. This model has the form,

$$\left\{\mathbf{P}_{c,1,(v)}, \mathbf{P}_{c,2,(v)}, \cdots, \mathbf{P}_{c,q,(v)}\right\} \tag{5.2}$$

where (v) is the number of independent complex variables. In an assembly constituted by k components with a total of r ports, $\forall(1 \leq i \leq q)$, each $(r \times r)$ unconstrained component matrix in (5.3) has the form,

$$\mathbf{P}_{c,i,(v)} = \begin{bmatrix} \mathbf{P}_{1,i,(v)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{2,i,(v)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{P}_{k,i,(v)} \end{bmatrix} \tag{5.3}$$

In the *second* step, the assembly agent generates the Assembly Volterra Dynamic model. This model is,

$$\left\{\mathbf{P}_{a,1,(v)}, \mathbf{P}_{a,2,(v)}, \cdots, \mathbf{P}_{a,q,(v)}\right\} \tag{5.4}$$

Each assembly Volterra Dynamic Matrix in (5.4) is determined using the transformation,

$$\mathbf{P}_{a,i,(v)} = \mathbf{S}^T \mathbf{P}_{c,i,(v)} \mathbf{S} \tag{5.5}$$

The reason of using (5.5) to determine the Assembly model is explained in detail at the Appendix B.

The assembly model (5.4) is in the same format than the component model (5.1) and can be recursively used as a component model to built higher order physical assembly models. The Volterra dynamic model representation is particularly effective as the standard assembly nonlinear MMM format because constraints (2.19a) and (2.19b) can be easily applied [38]. In contrast, the Volterra transfer function model

(4.15) cannot be used as the assembly nonlinear MMM format because (2.19a) and (2.19b) cannot be substituted directly [38]. Even though the Volterra dynamic model format is effective for the nonlinear model assembly process, it is not appropriate for simulations because is not possible to use it to solve directly for the outputs given the inputs. A different model format is required for simulation.

5.2 Simulation of the Assembled Nonlinear Physical Model

The Volterra transfer function model representation (4.15) is chosen as the nonlinear MMM simulation format because it can be directly used to solve for the outputs given the inputs. The simulation of the assembled nonlinear physical model is done in two steps:

In the *first* step, the Dynamic Assembly Matrix $\mathbf{P}_{a,1,(v)}$ is inverted to generate the Volterra Transfer Function Matrix of the assembly $\mathbf{G}_{a,1,(v)}$.

In the *second* step, the assembly Volterra Transfer Function Matrix $\mathbf{G}_{a,1,(v)}$ and each of the Volterra Dynamic Assembly Matrices $\mathbf{P}_{a,2,(v)}, \cdots, \mathbf{P}_{a,q,(v)}$ are substituted with the appropriate index (v) into the standard Volterra Transfer Function formats (4.17a), (4.17b), (4.17c),$\cdots$). For the first three standard formats,

$$\mathbf{Y}_{a,(1)} = \mathbf{G}_{a,1,(1)}\mathbf{U}_1 \tag{5.6a}$$

$$\mathbf{Y}_{a,(2)} = -\mathbf{G}_{a,1,(2)}\mathbf{P}_{a,2,(2)}\psi\left(\mathbf{Y}_{a,(1)}, \mathbf{Y}_{a,(1)}\right) \tag{5.6b}$$

$$\mathbf{Y}_{a,(3)} = -\mathbf{G}_{a,1,(3)}\left[\ \mathbf{P}_{a,3,(3)}\psi\left(\mathbf{Y}_{a,(1)}, \mathbf{Y}_{a,(1)}, \mathbf{Y}_{a,(1)}\right) + \mathbf{P}_{a,2,(3)}\psi\left(\mathbf{Y}_{a,(1)}, \mathbf{Y}_{a,(2)}\right)\ \right] \tag{5.6c}$$

The Volterra transfer function of the assembly is,

$$\mathbf{Y}_a = \mathbf{Y}_{a,(1)} + \mathbf{Y}_{a,(2)} + \mathbf{Y}_{a,(3)} + \cdots \tag{5.7}$$

A simulation of a Volterra model truncated up to the second expansion [10] uses recursively equations (5.6a) and (5.6b). Initially the matrix $\mathbf{G}_{a,1,(1)}$ is multiplied by

the input vector $\mathbf{U}(s_1)$. The resultant output is the vector $\mathbf{Y}_{a,(1)}$ in function of the complex variable s_1. The inverse Laplace transform respect s_1 is applied to $\mathbf{Y}_{a,(1)}$ what yields the the degree-1 time homogeneous output $\mathbf{y}_{a,1}(t_1)$. The time variable t_1 is redefined as $t = t_1$ what finally yields the degree-1 time homogeneous output $\mathbf{y}_{a,1}(t)$.

Equation (5.6b) operates the previously calculated vector $\mathbf{Y}_{a,(1)}$ into $\psi(\mathbf{Y}_{a,(1)}, \mathbf{Y}_{a,(1)})$. The resultant vector output of this operation is multiplied by the matrix $\mathbf{P}_{a,2,(2)}$, and then by the matrix $-\mathbf{G}_{a,1,(2)}$ what yields the vector $\mathbf{Y}_{a,(2)}$ which is a function of the complex variables s_1 and s_2. The inverse Laplace transform is applied sequentially to $\mathbf{Y}_{a,(2)}$ two times, first respect s_1 and then respect to s_2 what yields the the degree-2 time homogeneous output $\mathbf{y}_{a,1}(t_1, t_2)$. The time variable t_1 and t_2 are redefined as $t = t_1 = t_2$ what finally yields the degree-2 time homogeneous output $\mathbf{y}_{a,2}(t)$.

The time response of the Volterra model truncated up to the second expansion $\mathbf{y}_{a,1-2}(t)$ is obtained by adding the degree-1 time homogeneous output to the the degree-2 time homogeneous output what yields,

$$\mathbf{y}_{a_{1-2}}(t) = \mathbf{y}_{a,1}(t) + \mathbf{y}_{a,2}(t) \tag{5.8}$$

Simulation of higher order terms are obtain following a similar procedure.

5.3 Example

The assembly of two component models using the nonlinear MMM algorithm is described in this section. The first component is a linear model with one external port. This port is the input-output pair (u_3, y_3). The second component is a nonlinear model with two external ports. These ports are the input-output pairs (u_1, y_1) and (u_2, y_2). Two joins are used in the assembly. The first joint connectes the port (u_3, y_3) of the linear component and the port (u_2, y_2) of the nonlinear component to an as-

sembly port $(\tilde{u}_2, \tilde{y}_2)$. The second join connects the component port (u_1, y_1) to the assembly port $(\tilde{u}_1, \tilde{y}_1)$. The assembly model has two ports, these are the input-output pairs $(\tilde{u}_1, \tilde{y}_1)$ and $(\tilde{u}_2, \tilde{y}_2)$ (Figure 5.1).

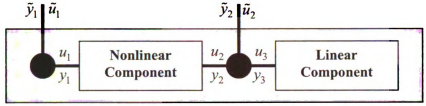

Figure 5.1. Port Connection Diagram of the Assembly

The linear model describes a one-port linear mass-spring system (Figure 5.2), with mass m_3 and spring constant k_3. The port uses the input-output pair (u_3, y_3) where u_3 is the input force applied to the mass m_3 and y_3 is its output displacement. The ordinary differential equation that represents this linear mass-spring-system is,

$$m_3 \ddot{y}_3 + k_3 y_3 = u_3 \tag{5.9}$$

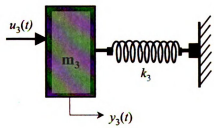

Figure 5.2. Linear Mass-Spring System

The nonlinear model describes a two-port mass-spring-damper system (Figure 5.3) composed of two masses, one nonlinear spring, one linear spring and one linear damper. The mass m_1 is connected to a fixed point through the nonlinear spring with force $f_{NL} = k_1(y_1 + y_1^3)$. The mass m_2 is connected to the mass m_1 with a linear damper having damping coefficient b_1 and a linear spring with constant k_2. The first port uses the input-output pair (u_1, y_1) where u_1 is the input force applied to the

mass m_1 and y_1 is its output displacement. The second port uses the input-output pair (u_2, y_2) where u_2 is the input force applied to the mass m_2 and y_2 is its output displacement.

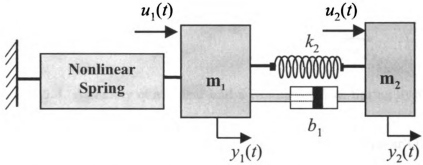

Figure 5.3. Double-Mass Spring Damper-Nonlinear System

The ODE that represents this nonlinear mass-spring-damper system is,

$$\begin{aligned} m_1\ddot{y}_1 + b_1(\dot{y}_1 - \dot{y}_2) + k_1 y_1 + k_1 {y_1}^3 + k_2(y_1 - y_2) &= u_1 \\ m_2\ddot{y}_2 + b_1(\dot{y}_2 - \dot{y}_1) + k_2(y_2 - y_1) &= u_2 \end{aligned} \tag{5.10}$$

The assembly is performed by attaching the mass m_3 of the linear system to the mass m_2 of the nonlinear system (Figure 5.4). The resultant assembled system has two ports. The first port uses the input-output pair $(\tilde{u}_1, \tilde{y}_1) = (u_1, y_1)$. The second port uses the pair $(\tilde{u}_2, \tilde{y}_2)$ where $\tilde{u}_2$ is the input force applied to the mass $m_a = m_2 + m_3$ and $\tilde{y}_2$ is its output displacement. Traditionally and ad-hoc procedure is used to obtained the ordinary differential equation of the assembly. This procedure is difficult to automatize because it usually requires a new reformulation using physical laws or energy methods. The new ODE for the assembled system is,

$$\begin{aligned} m_1\ddot{\tilde{y}}_1 + b_1(\dot{\tilde{y}}_1 - \dot{\tilde{y}}_2) + k_1\tilde{y}_1 + k_1{\tilde{y}_1}^3 + k_2(\tilde{y}_1 - \tilde{y}_2) &= \tilde{u}_1 \\ (m_2 + m_3)\ddot{\tilde{y}}_2 + b_1(\dot{\tilde{y}}_2 - \dot{\tilde{y}}_1) + k_2(\tilde{y}_2 - \tilde{y}_1) + k_3\tilde{y}_2 &= \tilde{u}_2 \end{aligned} \tag{5.11}$$

The nonlinear MMM algorithm requires the Volterra dynamic model of each assembly component. A standard procedure is used to derive the corresponding Volterra dynamic models of the linear and nonlinear ODEs in (5.9) and (5.10). Each Volterra dynamic model provided has a finite number of expansion terms.

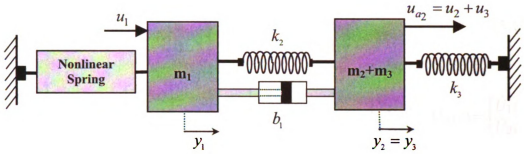

Figure 5.4. Assembly of a Linear and a Nonlinear Mass Spring System

The third order truncated Volterra dynamic model for the nonlinear ODE in (5.10) has the form,

$$\left\{\mathbf{P}_{1,1,(v)}, \mathbf{P}_{1,2,(v)}, \mathbf{P}_{1,3,(v)}\right\} \tag{5.12}$$

where

$$\mathbf{P}_{1,1,(v)} = \begin{bmatrix} m_1 s_{(v)}^2 + b_1 s_{(v)} + k_1 + k_2 & -b_1 s_{(v)} - k_2 \\ -b_1 s_{(v)} - k_2 & m_2 s_{(v)}^2 + b_1 s_{(v)} + k_2 \end{bmatrix} \tag{5.13a}$$

$$\mathbf{P}_{1,2,(v)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.13b}$$

$$\mathbf{P}_{1,3,(v)} = \begin{bmatrix} k_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.13c}$$

The third order truncated Volterra dynamic model for the linear ODE in (5.10) has the form

$$P_{2,1,(v)} = m_3 s_{(v)}^2 + k_3 \tag{5.14a}$$

$$P_{2,2,(v)} = P_{2,3,(v)} = 0 \tag{5.14b}$$

The matrix $\mathbf{S}$ is generated by relating the component and assembly variables. The component variables are $y_1(t)$, $y_2(t)$, $y_3(t)$, $u_1(t)$, $u_2(t)$ and $u_3(t)$. The assembly variables are $\tilde{y}_1(t)$, $\tilde{y}_2(t)$, $\tilde{u}_1(t)$ and $\tilde{u}_2(t)$. The two constraint equations for this example are:

$$\begin{bmatrix} y_1(t) \\ y_2(t) \\ y_3(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{y}_1(t) \\ \tilde{y}_2(t) \end{bmatrix} \tag{5.15a}$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} \tilde{u}_1(t) \\ \tilde{u}_2(t) \end{bmatrix} = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \end{bmatrix} \tag{5.15b}$$

where,

$$\mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \mathbf{Y}_c(s) = \begin{bmatrix} Y_1(s) \\ Y_2(s) \\ Y_3(s) \end{bmatrix}, \mathbf{U}_c(s) = \begin{bmatrix} U_1(s) \\ U_2(s) \\ U_3(s) \end{bmatrix}, \mathbf{Y}_a(s) = \begin{bmatrix} \tilde{Y}_1(s) \\ \tilde{Y}_2(s) \end{bmatrix}, \mathbf{U}_a(s) = \begin{bmatrix} \tilde{U}_1(s) \\ \tilde{U}_2(s) \end{bmatrix}$$

In the first step the Assembly agent generate the unconstrained Volterra dynamic model. This model have five matrices which are,

$$\left\{ \begin{bmatrix} \mathbf{P}_{1,1,(v)} & \mathbf{0} \\ \mathbf{0} & P_{2,1,(v)} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{1,2,(v)} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{1,3,(v)} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \right\} \tag{5.16}$$

In the second step, the Assembly Agent determines the five dynamic assembly matrices. These are:

$$\mathbf{P}_{a,1,(v)} = \mathbf{S}^T \begin{bmatrix} \mathbf{P}_{1,1,(v)} & \mathbf{0} \\ \mathbf{0} & P_{2,1,(v)} \end{bmatrix} \mathbf{S}$$

$$\mathbf{P}_{a,1,(v)} = \begin{bmatrix} m_1 s_{(v)}^2 + b_1 s_{(v)} + k_1 + k_2 & -b_1 s_{(v)} - k_2 \\ -b_1 s_{(v)} - k_2 & (m_2 + m_3) s_{(v)}^2 + b_1 s_{(v)} + k_2 + k_3 \end{bmatrix} \tag{5.17a}$$

$$\mathbf{P}_{a,2,(v)} = \mathbf{S}^T \begin{bmatrix} \mathbf{P}_{1,2,(v)} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \mathbf{S} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.17b}$$

$$\mathbf{P}_{a,3,(v)} = \mathbf{S}^T \begin{bmatrix} \mathbf{P}_{1,3,(v)} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \mathbf{S} = \begin{bmatrix} k_1 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.17c}$$

Finally the Volterra dynamic model is,

$$\left\{ \mathbf{P}_{a,1,(v)}, \mathbf{P}_{a,2,(v)}, \mathbf{P}_{a,3,(v)} \right\} \tag{5.18}$$

The simulation is done in two parts

1. The dynamic matrix $\mathbf{P}_{a,1,(v)}$ is inverted to generate the assembly transfer function matrix,

$$\mathbf{G}_{a,1,(v)} = \left[\mathbf{P}_{a,1,(v)} \right]^{-1} = \begin{bmatrix} \frac{(m_2+m_3)s_{(v)}^2 + b_1 s_{(v)} + k_2 + k_3}{as^4 + bs^3 + cs^2 + ds + e} & \frac{b_1 s_{(v)} + k_2}{as^4 + bs^3 + cs^2 + ds + e} \\ \frac{b_1 s_{(v)} + k_2}{as^4 + bs^3 + cs^2 + ds + e} & \frac{m_1 s_{(v)}^2 + b_1 s_{(v)} + k_1 + k_2}{as^4 + bs^3 + cs^2 + ds + e} \end{bmatrix} \tag{5.19}$$

where

$$a = m_1(m_2 + m_3)$$
$$b = b_1(m_1 + m_2 + m_3)$$
$$c = (k_2 + k_3)m_1 + (k_1 + k_2)m_2 + (k_1 + k_2)m_3$$
$$d = b_1(k_1 + k_3)$$
$$e = k_1k_2 + k_1k_3 + k_2k_3$$

2. The assembly transfer function matrix $\mathbf{G}_{a,1,(v)}$ and the four dynamic assembly matrices $\mathbf{P}_{a,2,(v)}, \cdots, \mathbf{P}_{a,5,(v)}$ are substituted into the standard Volterra transfer function formats. The first five Volterra formats for this system are,

$$\mathbf{Y}_{(1)} = \mathbf{G}_{a,1,(1)}\mathbf{U}_1 \tag{5.20a}$$

$$\mathbf{Y}_{(2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.20b}$$

$$\mathbf{Y}_{(3)} = -\mathbf{G}_{a,1,(3)}\mathbf{P}_{a,3,(3)}\psi\left(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}\right) \tag{5.20c}$$

$$\mathbf{Y}_{(4)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.20d}$$

$$\mathbf{Y}_{(5)} = 3\mathbf{G}_{a,1,(5)}\mathbf{P}_{a,3,(5)}\psi(\mathbf{Y}_{(1)}, \mathbf{Y}_{(1)}, \mathbf{Y}_{(3)}) \tag{5.20e}$$

The truncated Volterra model up to the fifth expansion is,

$$\mathbf{Y}_{1-5} = \mathbf{Y}_{(1)} + \mathbf{Y}_{(3)} + \mathbf{Y}_{(5)} \tag{5.21}$$

One way to simulate the Volterra transfer function model is using a Simulink diagram. The Simulink diagram for the truncated model in (5.21) is shown in Figure 5.5.

The time responses of the linear, third and fifth truncated Volterra models are compared to the nonlinear ODE of the assembly in (5.11) using the system parameters, $m_1 = 2$, $m_2 = 1$, $m_3 = 1$, $b_1 = 2$, $k_1 = 8$, $k_2 = 4$ and $k_3 = 1$. The two inputs used to excite the system are sine waves with amplitudes -0.3 and -0.15 at frequencies of 2.36 rad/sec and 1.31 rad/sec respectively. These frequencies are the two natural frequencies of the linearized system of the assembly and are determined by calculating

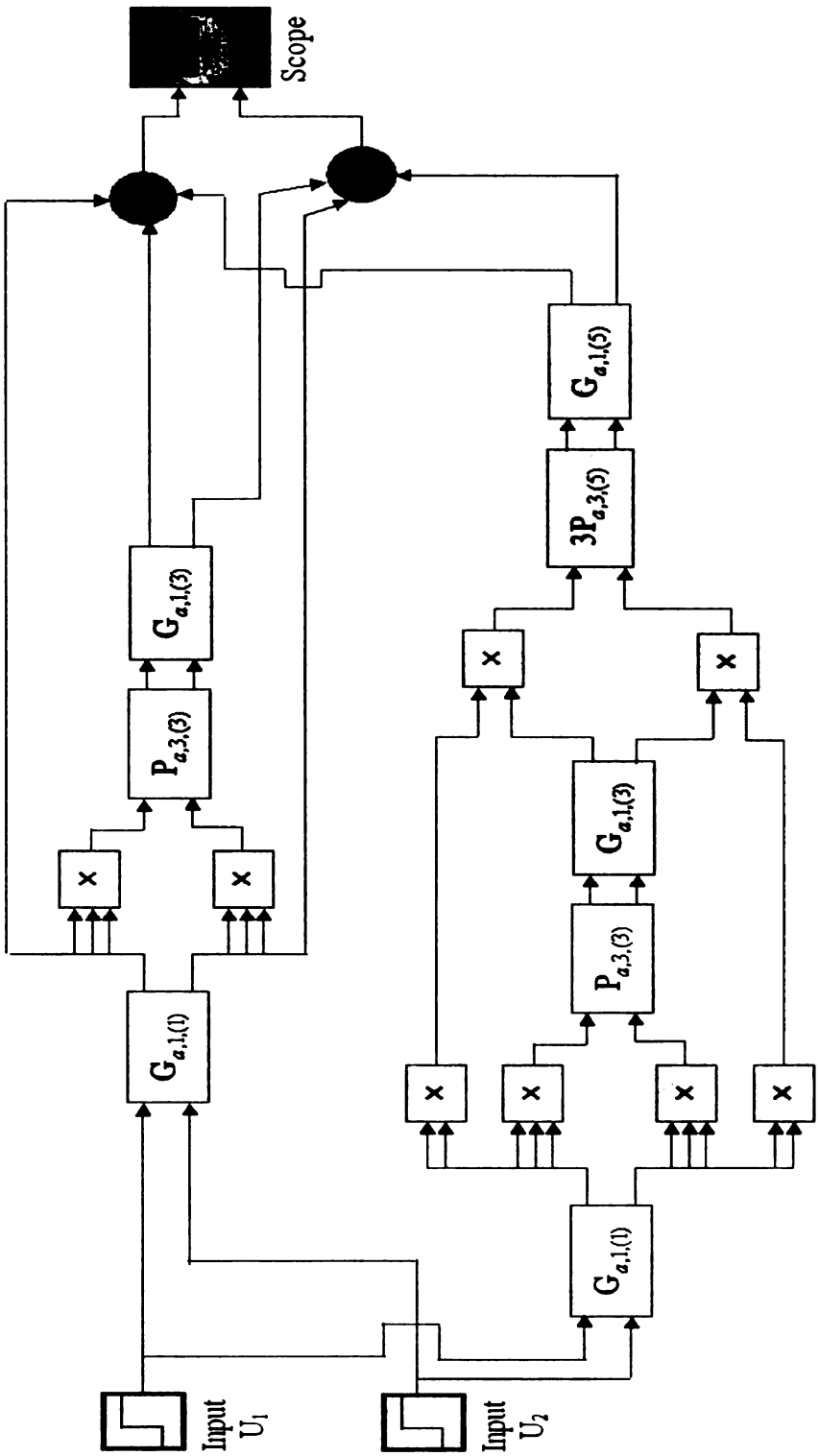

Figure 5.5. Simulink Program Used to Simulate the Volterra Model

the roots of the characteristic equation of $\mathbf{G}_{1,(1)}$ in (5.19). The linearized model exhibits the greatest error between all the three responses. The error shown by the truncated Volterra model that includes the linearized and the third expansion terms is substantially smaller than the error exhibit by the linearized system (Figures 5.6 and 5.7). The error exhibit by the truncated Volterra model that includes the linearized, the third and the fifth order expansion term is the smallest of all the three responses. For a specific bound at the input, the more expansion the Volterra model includes, the smaller is its error. Theoretically the error between the Volterra model and the nonlinear ODE will tend to zero when an infinite number of expansions is used.

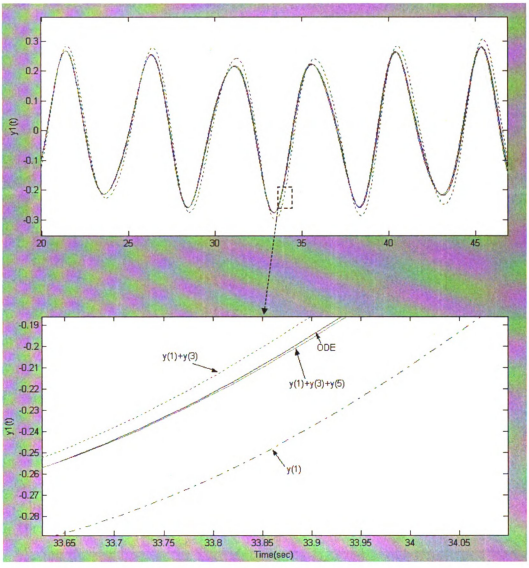

Figure 5.6. Nonlinear ODE assembled response $y_1(t)$ compared with different truncated Volterra transfer function responses. $\mathbf{y(1)}$ is the linearized response, $\mathbf{y(3)}$ is the third expansion and $\mathbf{y(5)}$ is the fifth expansion.

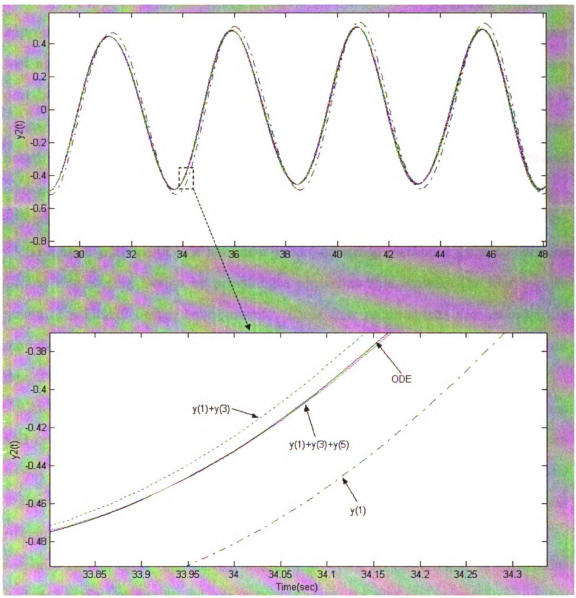

Figure 5.7. Nonlinear ODE assembled response $y_2(t)$ compared with different truncated Volterra transfer function responses. y(1) is the linearized response, y(3) is the third expansion and y(5) is the fifth expansion.

5.4 Summary

In this section an original method to assemble physical MIMO physical external port-based Volterra system models is presented for first time. Volterra representations are appropriate to the MMM approach to distributed model assembly because they protect internal design details. This method extends the MMM approach to nonlinear port-based models through a method that is computationally efficient because it is recursive and does not require matrix inverses. These properties make this Volterra-based assembly procedure ideal for distribution and assembly of nonlinear physical engineering system models. This method does not present a new modeling methodology but rather a new strategy to distribute and assemble existent nonlinear models.

The main purpose of this nonlinear model assembly methodology is to enhance Global Engineering Design by providing a procedure that satisfies the four requirements of a Global design environment. These four requirements are: standardize model components, single query exchange of model information, external input-output model formats and a recursive assembly.

An example of an assembly of one, nonlinear 2-port mass-spring-damper system and one, linear 1-port mass-spring system was presented. The resultant Volterra model of the assembly includes the first five expansions. Different truncated MIMO Volterra models for the assembly where also simulated using Simulink, and their responses were compared with the nonlinear ODE of the assembled system. The response results show increasing simulation accuracy as the number of expansions included in the Volterra model is increased. Experience with these simulations models shows convergence a requires bound on input amplitudes. Convergence radii are available for SISO Volterra models [13]. Future work will be needed to find these limits for the MIMO models derived from the MMM assembly methodology.

CHAPTER 6

Conclusions

The nonlinear Modular Model distribution and assembly methodology presented in this work, provides six (6) significant, original contributions to mechanical engineering. These contributions are:

1. An extension of the Modular Modeling Methodology (MMM) to nonlinear systems.

2. The recognition that two model formats are necessary for physical system model assembly and simulation.

3. A method to assemble physical port-based affine ODEs around an equilibrium operating point.

4. A method to obtain subsystems operating points from the system assembly operating point outputs.

5. A method to obtain frequency domain Volterra system models from MIMO port-based ODEs.

6. A method to assemble physical MIMO port-based Volterra system models. Each of these contributions is individually important to the development of a Global Engineering Design (GED) strategy.

This methodology has two (2) limitations. These limitations are:

1. The nonlinear subsystems must be modeled using port-based Volterra representations.

2. The assembled system model is valid if its subsystem models are valid. These

limitation still allows the model assembly of a high variety of nonlinear port-based physical system models.

The nonlinear model distribution and assembly methodology developed in this work enhances Global Engineering Design by providing a procedure that satisfies the four requirements of a Global design environment: a standard model format, single query exchange of model information, external input-output models and a recursive assembly method.

APPENDIX A

Singular Internal Stiffness Matrix

A singular internal dynamic matrix $\mathbf{P}_{ii}(s)$ occurs when the assembled system includes internal disconnected subsystems or non-observable modes. When $\mathbf{P}_{ii}(s)$ is singular, a modification of the approach taken in (2.44) it is required. The objective is to remove the internal singular degrees of freedom associated with the disconnected subsystems. The internal singular degrees of freedom are removed using a null-space transformation. Once these degrees of freedom are removed, the modified internal dynamic matrix $\hat{\mathbf{P}}_{ii}(s)$ is non-singular and the condensation continues.

Initially the rank ρ orthonormal null space $\mathbf{T}(s)$ $(\upsilon \times \rho)$ matrix of $\mathbf{P}_{ii}(s)$ is determined. The columns of this matrix are the ρ eigenvectors of $\mathbf{P}_{ii}(s)$ whose eigenvalues are zero. The null space $\mathbf{T}(s)$ defines the $(\upsilon \times \upsilon)$ transformation matrix

$$\mathbf{Q}(s) = \left[\begin{array}{c|c} \mathbf{I} & \\ \mathbf{0} & \mathbf{T}(s) \end{array}\right] \tag{A.1}$$

constructed with the $(\upsilon \times \upsilon)$ null space $\mathbf{T}(s)$ and an arbitrary set of independent vectors. The independent set vectors constructed here are an $(\upsilon - \rho) \times (\upsilon - \rho)$ identity matrix filled below with a $(\rho \times \rho)$ zero matrix.

The above null space transformation is used to define a new set of internal output variables

$$\mathbf{Y}_i(s) = \mathbf{Q}(s)\tilde{\mathbf{Y}}(s) \tag{A.2}$$

where the new internal output variables $\tilde{\mathbf{Y}}(s) = \left[\begin{array}{cc} \tilde{\mathbf{Y}}_i(s) & \mathbf{Y}_0(s) \end{array}\right]^T$ are the $(\upsilon-\rho)$ non-

singular output $\tilde{\mathbf{Y}}_i(s)$ combined with the ρ singular outputs $\mathbf{Y}_0(s)$. Work must be conserved under this coordinate transformation. Defining the transformed generalized input $\tilde{\mathbf{U}}(s)$ and applying a work analysis parallel to (2.16) above,

$$\mathbf{Y}_i(s)^T\mathbf{U}_i(s) = \tilde{\mathbf{Y}}(s)^T\tilde{\mathbf{U}}(s) \tag{A.3}$$

Substituting (A.2) into (A.3) yields

$$\mathbf{Q}(s)^T\mathbf{U}_i(s) = \tilde{\mathbf{U}}(s) \tag{A.4}$$

Applying this transformation symmetrically to (2-5-11) yields

$$\begin{bmatrix} \mathbf{P}_{ee}(s) & \mathbf{P}_{ei}(s)\mathbf{Q}(s) \\ \mathbf{Q}^T(s)\mathbf{P}_{ie}(s) & \mathbf{Q}^T(s)\mathbf{P}_{ii}(s)\mathbf{Q}(s) \end{bmatrix} \begin{bmatrix} \mathbf{Y}_e(s) \\ \hline \tilde{\mathbf{Y}}_i(s) \\ \mathbf{Y}_0(s) \end{bmatrix} = \begin{bmatrix} \mathbf{U}_e(s) \\ \mathbf{Q}^T(s)\mathbf{U}_i(s) \end{bmatrix} \tag{A.5}$$

Transformation $\mathbf{Q}(s)$ contains the null space transformation with $\mathbf{P}_{ii}(s)\mathbf{T}(s)$ so that (A.3) becomes

$$\begin{bmatrix} \mathbf{P}_{ee}(s) & \hat{\mathbf{P}}_{ei}(s) & \mathbf{0} \\ \hat{\mathbf{P}}_{ie}(s) & \hat{\mathbf{P}}_{ii}(s) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{Y}_e(s) \\ \tilde{\mathbf{Y}}_i(s) \\ \mathbf{Y}_0(s) \end{bmatrix} = \begin{bmatrix} \mathbf{U}_e(s) \\ \tilde{\mathbf{U}}_i(s) \\ \mathbf{U}_0(s) \end{bmatrix} \tag{A.6}$$

None of the system equations involve $\mathbf{Y}_0(s)$ and this vector can be removed from the system of equations. The last ρ rows and columns of the transformed system are now removed and the analysis continued with $\tilde{\mathbf{U}}_i(s) = \mathbf{0}$ applied on the transformed matrices to find

$$\hat{\mathbf{P}}_e(s)\mathbf{Y}_e(s) = \mathbf{U}_e(s) \tag{A.7}$$

where

$$\hat{\mathbf{P}}_e(s) = \left[\mathbf{P}_{ee}(s) - \hat{\mathbf{P}}_{ei}(s)\hat{\mathbf{P}}_{ii}^{-1}(s)\hat{\mathbf{P}}_{ie}(s)\right]$$

APPENDIX B

The Assembly Operator

The reason of using (5.5) to obtain all the assembly Volterra dynamic matrices is explained here. Since the matrix $\mathbf{G}_{a,1,(v)}$ is the linearized transfer function of the assembly, its inverse, $\mathbf{P}_{a,1,(v)}$, is the linearized dynamic matrix of the assembly. It was shown previously that $\mathbf{P}_{a,1,(v)}$ can be obtained using,

$$\mathbf{P}_{a,1,(v)} = \mathbf{S}^T \mathbf{P}_{c,1,(v)} \mathbf{S} \tag{B.1}$$

The dynamic matrix $\mathbf{P}_{a,1,(v)}$ satisfies (4.20). Applying the operator $\mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}(\bullet)$ to both sides of (B.1) yields,

$$\mathbf{P}_{a,2,(v)} = \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{a,1,(v)}\right) = \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{S}^T \mathbf{P}_{c,1,(v)} \mathbf{S}\right) \tag{B.2}$$

The matrix of constants $\mathbf{S}$ can be taken out of the operator to yield,

$$\mathbf{P}_{a,2,(v)} = \mathbf{S}^T \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{c,1,(v)}\right) \mathbf{S} \tag{B.3}$$

Using (4.20) in the left side of (B.3) yields

$$\mathbf{P}_{a,2,(v)} = \mathbf{S}^T \mathbf{P}_{c,2,(v)} \mathbf{S} \tag{B.4}$$

Applying again the operator $\mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}(\bullet)$ to both sides of (B.4) yields

$$\mathbf{P}_{a,3,(v)} = \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{a,2,(v)}\right) = \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{S}^T \mathbf{P}_{c,2,(v)} \mathbf{S}\right) \tag{B.5}$$

The matrix of constants $\mathbf{S}$, can be again taken out of the derivative operator to yield,

$$\mathbf{P}_{a,3,(v)} = \mathbf{S}^T \mathbf{D}_{\mathbf{y},\dot{\mathbf{y}},\ddot{\mathbf{y}},\cdots}\left(\mathbf{P}_{c,2,(v)}\right) \mathbf{S} \tag{B.6}$$

Using again (4.20) in the left side of (A.6) yields

$$\mathbf{P}_{a,3,(v)} = \mathbf{S}^T \mathbf{P}_{c,3,(v)} \mathbf{S} \tag{B.7}$$

Applying sequentially a similar procedure for any $i = 3, 4, 5, \cdots$ yields the assembly operator (5.5) ,

BIBLIOGRAPHY

[1] Elmqvist, H., 1999, "Modelica: A language for Physical System Modeling, Visualization and Interaction", CACSD'99 IEEE Symposium on Computer Control System Design, Hawaii, August 22-27.

[2] Reichenbach, D., Radcliffe, C., and Sticklen, J., 2004, "Modular Distributed Models of Structural Dynamics", Automated Modeling 2004 ASME IMECE, Anaheim, CA., November 13-19.

[3] Tian, G., Yin, G., and Taylor, D., 2002, "Internet-Based Manufacturing: A review and a new Infrastructure for distributed Intelligent Manufacturing", Journal of Intelligent Manufacturing, Vol. 13, pp. 323-333.

[4] Gu, B., and Asada, H., 2004, "Co-Simulation of Algebraic Coupled Dynamic Systems Without Disclosure of Proprietary Subsystem Models", Journal of Dynamic Systems Measurement and Control, Vol. 126, pp.1-13.

[5] Zienkiewicz, O., 1989,*The Finite Element Method*, McGraw-Hill, London.

[6] Mattsson, E., 1993, "Index Reduction in Differential Algebraic Equations Using Dummy Derivatives", SIAM Journal on Scientific Computing, Vol. 14(3), pp. 677-692.

[7] Karnopp, Dean C., Margolis, D., and Rosenberg, R., 2000, *System Dynamics: Modeling and Simulation of Mechatronic Systems*, John Wiley Sons, Inc., New York.

[8] Byam, B., 1999, *Modular Modeling of Engineering Systems Using Fixed I-O Structure* , PhD. Dissertation, Michigan State University, East Lansing, MI.

[9] Byam, B., and Radcliffe, C., 2000, "Direct-Insertion Realization of Linear Modular Models of Engineering Systems Using Fixed Input-Output Structure", ASME, Proceedings of DET2000: 26th Design Automation Conference, Baltimore, MD, September 10-13.

[10] Rugh, W., 1981, *Nonlinear System Theory*, The Johns Hopkins University Press.

[11] Brilliant M, March 1958, "Theory of the Analysis of Nonlinear Systems", Technical report 345, Massachussetts Institute of Technology, Research Laboratory of Electronics.

[12] Sandberg W. I, June 1983, "Series Expansions for Nonlinear Systems" Circuits, Systems, and Signal Processing, vol. 2, no. 1, pp. 77-87.

[13] Boyd, S. and Chua L, November 1985, "Fading Memory and The problem of Approximating Nonlinear Operators with Volterra Series", IEEE Transactions on Circutis and Systems, Vol cas-32, No 11.

[14] Wang, T., and Brazil, Thomas., 2000, "The Estimation of Volterra Transfer Functions With Applications to RF Power Amplifier Behavior Evaluation for CDMA Digital Communication, IEEE Microwave Symposium Digest., Vol. 1, June 2000 11-16, pp 425 - 428.

[15] Draganescu, E., and Ercuta A., 2003, "Identification of Nonlinearities in Anaelastic Polycrystalline Material Using Volterra-Fourier Transform", Journal of Optoelectronics and Advanced Materials, Vol. 5(1), pp. 301-304.

[16] Li P., and Pileggi T. L., 2003, "NORM: Compact Model Order Reduction of Weakly Nonlinear Systems", DAC 2003, Anaheim, California, June2-6.

[17] Giret, A., and Botti, V., 2004, "Holons and Agents", Journal of Intelligent Manufacturing, Vol. 15, pp. 645-659.

[18] Takahashi Y., Rabins, M., and Auslander D., 1970, *Control and Dynamic Systems*, Addison-Wesley Publishing Company.

[19] Morari, M., and Zafiriou, E., 1989, *Robust Process Control*, Prentice Hall, Englewood Cliffs, New Jersey

[20] Genta, G., 1999, *Vibration of Structures and Machines*, Springer-Verlag, New York.

[21] Skogestad, S., and Postlethwaite, I., 2000, *Multivariable Feedback Control*, John Wiley and Sons, England.

[22] D'Souza, A., 1988, *Design of Control Systems*, Prentice-Hall, New Jersey.

[23] Buck, R., Willcox, A. 1971, "Calculus of Several Variables", Houghton Mifflin Company, Boston

[24] Favier, G., Kibangou, A., and Y., Khouaja, 2004, "Nonlinear System Modelling by Means of Volterra Models. Aproaches for Parametric Complexity Reduction", Simposium Techniques Avancees et Strategies Innovantes en Modelisation et Commandes Robustes des Processus Industriels, Martigues, September 21-22.

[25] Bikerlund, Y., and Hanssen, A., 2003, "On the Estimation of Nonlinear Volterra Models in Offshore Engineering", International Journal of Offshore and Polar Engineering, Vol. 13(1).

[26] Boaghe, O., and Billings S., 2003, "Sub-harmonic Oscillation Modeling and MISO Volterra Series", IEEE Transactions on Circuits and Systems, Vol. 50(7), pp 877-884.

[27] Storrs, Samuel M., 1999, "Noise Chacterization of devices for Optical Computing", Ph.D. thesis, Texast Techn. University.

[28] Schetzen, M., 1980, *The Volterra and Wiener Theories of Nonlinear Systems*, John Wiley

[29] Yangwang F., and Jiao, L, 2001, "MIMO Volterra Filter Equalization Using Pth-Order Inverse Approach" Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 1, pp 177-180.

[30] Dobrowiecki, T., P., and Schoukens, J., 2004, "Linear Approximation of Weakly Nonlinear MIMO Systems", Proc. IMTC 2004, Como, Italy, May 18-20.

[31] Yangwang F., and Jiao, L, Ruixuan, W., Pan. J. 2000, "Identification of MIMO Nonlinear Systems Based on Volterra Kernel Matrices" Proceedings. of the 3rd World Congress on Intelligent Control and Automation, June 28-July 2, Hefei, P.R. China.

[32] Greenwood, D, 2003, *Advanced Dynamics*, Cambridge University Press.

[33] Nam,C., 2006, Aeroelastic Analysis Using Matlab, at *http://ctas.poly.asu.edu/chnam/ASE Book/*

[34] Motato, E., Radcliffe, C. and Reichenbach, D., 2006, "A Procedure for Obtaining Multi Input Multi Output Volterra Models From Port-Based Nonlinear Differential Equations". Submitted to the 2007 American Control Conference, New York, NY.

[35] Chua, L., and Lin, Pen-Min , 1975, *Computer-Aided Analysis of Electronic Circuits*, Prentice-Hal, Inc., Engewood Cliffs, New Jersey

[36] Iserman, R., 1992, Adaptive Control Systems, Prentice Hall.

[37] Kerr, Brad., 2000, "Redesigning work Processes and Computing Environments", Automotive Engineering International, Vol. 108(7), pp 147-149.

[38] Motato, E., Radcliffe, C. and Reichenbach, D., 2006, "Networked Assembly of Linear Physical Models". Submitted to the ASME Journal of Dynamic Systems, Measurement , and Control.

[39] Frank P, and McFEE, June 1963, "Determining Input-Output Relationship of Nonlinear Systems by Inversions" IEEE Transactions on Circuit Theory, pp 169-180.

[40] Christensen G, December 1969, "On the convergence of Volterra Series" IEEE transactions on Automatic Control, 13, 736-737.

[41] Czarniak A and Kudrewicz, August 1984, "The Convergence of Nonlinear Series for Nonlinear Networks" IEEE transactions on Circuit and Systems, 31, 751-752.

[42] Chatterjee A., and Vyas N., February 2000, "Convergence analysis of Volterra Series Response of Nonlinear Systems Subjected to Harmonic Exitation", Journal of Sound and Vibrations. 236(2), pp 339-358

[43] Tomilson G. R. and Manson G., February 1996, "A simple Criterion for Establishing an Upper Limit to the Harmonic Excitation Level of the Duffing Oscillator Using the Volterra Series", Journal of Sound and Vibrations. 190(2), pp 751-762