The University Of Sheffield.

H. A. Bashar

"Meta-Modelling of Intensive Computational Models"

A Thesis Submitted to the University of Sheffield in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

February 2016

“Meta-Modelling of Intensive Computational Models”

Hasanain A. Bashar

Supervised by:

Prof Robert F. Harrison

A Thesis Submitted to the University of Sheffield in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

Department of Automatic Control and Systems Engineering

February 2016

University of Sheffield

Abstract

Engineering process design for applications that use computationally intensive nonlinear dynamical systems can be expensive in time and resources. The presented work reviews the concept of a meta-model as a way to improve the efficiency of this process. The proposed meta-model will have a computational advantage in implementation over the computationally intensive model therefore reducing the time and resources required to design an engineering process. This work proposes to meta-model a computationally intensive nonlinear dynamical system using reduced-order linear parameter varying system modelling approach with local linear models in velocity based linearization form. The parameters of the linear time-varying meta-model are blended using Gaussian Processes regression models. The meta-model structure is transparent and relates directly to the dynamics of the computationally intensive model while the velocity-based local linear models faithfully reproduce the original system dynamics anywhere in the operating space of the system. The non-parametric blending of the meta-model local linear models by Gaussian Processes regression models is ideal to deal with data sparsity and will provide uncertainty information about the meta-model predictions. The proposed meta-model structure has been applied to second-order nonlinear dynamical systems, a small sized nonlinear transmission line model, medium sized fluid dynamics problem and the computationally intensive nonlinear transmission line model of order 5000.

Acknowledgment

I would like to thank my thesis advisor professor Robert F. Harrison for his continuous support, encouragement, patience and guidance throughout my time at the Department of Automatic Control and Systems Engineering in the University of Sheffield.

I come from a different engineering background to computational data modelling and Professor Harrison guidance and clear vision made the presented study possible.

I would like to extend my thanks to Mr. Andy Mills for his input from an industrial point of view during the formulation of the presented work.

I would like to thank my research mates in the University of Sheffield Technology Centre especially; Dr. Martha Arbayani Bin Zaidan, Dr. Rue Wang, and Mr. Vigneshwaran Venugopalan for their technical and non-technical support especially during the long hours spent on computer codes.

Finally, I would like to thank the research support staff at the Department of Automatic Control and Systems Engineering for different kinds of help provided through my time at the department.

To my parents

Table of Contents

Chapter 1 - Introduction 1

1.1 Computationally Intensive Models 1

1.2 The Concept of a Meta-Model 2

1.3 Motivations and Contributions 6

1.4 Thesis Outline 9

Chapter 2 - Mathematical Modelling of a NDS 11

2.1 Introduction 11

2.2 The Nonlinear Auto Regressive, Moving Average and Exogenous Input NDS Structure 15

2.2.1 The NARMAX Model Classes 16

2.2.2 The NARMAX Model Structure 17

2.2.3 Meta-Modelling a CI-NDS with NARMAX 19

2.3 Divide and Conquer Methods 19

2.4 Velocity-Based Linearization 20

2.5 NDS modelling with Gaussian Processes 25

2.6 Meta-modelling with GP blended VBL-LPV system 28

2.7 Conclusions 29

Chapter 3 - Gaussian Processes Blended VBL-LPV system 31

3.1 Introduction 31

3.2 Meta-Model Structure 32

3.3 VBL-LPV Parameters Blending with Gaussian Processes Regression 32

3.3.1 Gaussian Processes 33

3.3.2 GPM Covariance Function 37

3.3.3 Properties of the GP Regression Model 40

3.3.4 Meta-Model Training 41

3.4 The Meta-Model Solver 50

3.4.1 The Velocity Solution 51

3.4.2 The State Equation Solver 55

3.4.3 The Meta-Model Output 55

3.4.4 Solving the Two-Tanks NDS Meta-Model 55
3.5 Conclusions 64

Chapter 4 - Model Order Reduction 66

4.1 Introduction 66
4.2 Linear Model Order Reduction 66
4.2.1 Proper Orthogonal Decomposition methods 68
4.2.2 Krylov Subspace methods 69
4.2.3 Truncated Balanced Realization 73
4.3 Model Order Reduction of CI-NDSs 76
4.4 Meta- Modelling CI-NDS with Reduced Order Local Linear Models ... 78
4.5 Canonical Transformations 81
4.6 Meta-Modelling of the 10th order Nonlinear Transmission Line 83
4.7 Conclusions 93

Chapter 5 - Uncertainty Propagation 96

5.1 Introduction 96
5.2 The Meta-Model Solver Uncertainty Propagation 96
5.3 Meta-Modelling of a Chemical Reactor 105
5.4 Meta-Modelling of the 1-D Burgers Equations 116
5.5 Conclusions 122

Chapter 6 - Meta-Model Computational Complexity Analysis 124

6.1 Introduction 124
6.2 The Meta-Model Training Cost 126
6.2.1 Collection of Training Data 126
6.2.2 Computing the LLMs 127
6.2.3 GP Model Training 128
6.3 The Meta-Model Solver 131
6.4 Meta-Modelling of a CI-NDS 137
6.4.1 Meta-Model Training Points 137
6.4.2 Meta-Model Parameters Training and Validation 144
6.4.3 Meta-Model Response to Test Inputs 145
6.5 Conclusions 156

Chapter 7 - Conclusions and Recommendations 159

7.1 Conclusions 159

7.2 Recommendations 163

List of References 165

Appendix A 180

A.1 Introduction 180

A.2 Two-Tanks NDS 180

System Description 180

System Equilibrium points 181

System Jacobian Matrices 182

A.3 Continuously-Stirred Tank Reactor (CSTR) NDS 182

System Description 182

System Equilibrium points 184

System Jacobian Matrices 186

A.4 Nonlinear Transmission Line NDS 187

System Description 187

System Equilibrium points 187

System Jacobian Matrices 188

A.5 1-D Burgers Equations 188

System Description 188

System Equilibrium points 189

System Jacobian Matrices 190

Appendix B 191

B.1 Approximation of the Input Derivative 191

B.2 Adapting the Approximation for Use with the Fixed Time Step Solver
194

List of Tables

Table 3-1: Two-Tanks NDS GPM training results for the time-varying entries of meta-model parameter $\boldsymbol{A}(\boldsymbol{t})$ 48

Table 3-2: Computational environment 62

Table 3-3: Meta-Model fitness results for the Two-Tanks NDS simulated with step sequence input using different velocity estimation methods 62

Table 3-4: ETI values for the Two-Tanks NDS meta-model for different test inputs 63

Table 4-1: Meta-Model training results for the 10th Order NTL 87

Table 4-2: ETI values for the 10th order NTL meta-model the for multi-step and frequency sweep test inputs 93

Table 5-1: CSTR meta-model training results 108

Table 6-1 Total number of flops spent in the meta-model velocity vector $\boldsymbol{w}(k+1)$ computation using Forward-Euler method 134

Table 6-2 Total number of flops spent in the meta-model state vector $\boldsymbol{x}(k+1)$ computation 134

Table 6-3 Total number of flops spent in the meta-model output vector $\boldsymbol{y}(k)$ computation 135

Table 6-4: 4th order meta-model training results for the CI-NDS NTL model . 145

Table 6-5: 10th order meta-model training results for the CI-NDS NTL model 145

Table 6-6: Computational time performance of the of the two meta-model scenarios versus the RK4 method for the NTL model of order 5000 (all measurements are in seconds) 153

Table 6-7: Computational time performance of the MATLAB ODE suit of solvers for the NTL model of order 5000 (all measurements are in seconds)............. 155

List of Figures

Figure 2-1: VBL ith LLM in state-space form. 23

Figure 3-1: Two tanks NDS (MATLAB system identification tool box). 45

Figure 3-2: Two-Tanks NDS pump voltage sequence used to excite the model at off-equilibrium dynamics 46

Figure 3-3: $\%RSD$ for the time-varying entries of the $\boldsymbol{A}(t)$ parameter for the Two-Tanks meta-model validation data set 50

Figure 3-4: Meta-model solution of the Two-Tanks NDS to a step function 56

Figure 3-5: Meta-model solution of the Two-Tanks NDS to a ramp test input.. 57

Figure 3-6: Meta-model solution of the Two-Tanks NDS to exponential decay test input 58

Figure 3-7: Two-Tanks NDS step seqiunce test input 59

Figure 3-8: Two-Tanks NDS meta-model solution of the to the step sequence test input 59

Figure 3-9: Plot of $\%RSD$ for the non-constant entries of the $\boldsymbol{A}(t)$ parameter for the Two-Tanks meta-model during simulation with the step sequence input ... 60

Figure 3-10: Composite phase plot for the Two-Tanks NDS training points and the meta-model solution to the step sequence test input 61

Figure 4-1: The Nonlinear Transmission Line circuit. 83

Figure 4-2: Randomly generated step sequence used to excite the 10th order NTL model in off-equilibrium regions. The sequence was generated using Algorithm 3-2 with parameters: $umin = 0, umax = 1, Nticks = 20,\ Ts = 0.01s$ and $stm = 2$ 85

Figure 4-3: Mean percentage of state contributions of the 10th order NTL system balanced training LLMs. 86

Figure 4-4: Multi-level step test input (top plot), the 10th order NTL model response versus the reduced order meta-model response with confidence regions (bottom plot) 88

Figure 4-5: Sinusoidal test input generated from a cosine linear frequency sweep from DC, crossing $25Hz$ at $0.5s$. 89

Figure 4-6: The 10th order NTL model response versus the reduced order meta-model response with confidence regions 90

Figure 4-7: $\%RSDmax$ for the 10th order NTL reduced-order meta-model during the solution of the multi-step test input (Figure 4-4 (Top)). 91

Figure 4-8: $\%RSDmax$ for the 10th order NTL reduced-order meta-model during the solution of the frequency sweep test input (Figure 4-5). 92

Figure 5-1: Schematic of the (Non-Adiabatic) Continuously-Stirred Tank Reactor 105

Figure 5-2: Random Gaussian signal applied to the CSTR model at the reactor coolant temperature input $u3(t)$ to collect meta-model off-equilibrium training data 107

Figure 5-3: CSTR meta-model $u1(t)$ test input component (incoming feed concentration) 109

Figure 5-4: CSTR meta-model $u2(t)$ test input component (incoming feed temperature) 109

Figure 5-5: CSTR meta-model $u3(t)$ test input component (reactor vessel coolant temperature). This is the main control input during the CSTR meta-model simulation 110

Figure 5-6: CSTR meta-model output $y1(t)$ (concentration of incoming feed inside the reactor) 110

Figure 5-7: CSTR meta-model output $y2(t)$ (reactor contents temperature) 111

Figure 5-8: CSTR meta-model repsonses with qualitivate view of uncertainty propgation in the meta-model 112

Figure 5-9: Time-varying $\%RSD$ for CSTR meta-model (trained with 62 training points) outputs, the dashed lines shows $\%RSD$ values computed using quantitative uncertainty propagation and the solid lines represent $\%RSD$ values computed using qualitative uncertainty propagation values 114

Figure 5-10: Qualitative time-varying $\%RSD$ for both meta-model training cases without (default) and with additional training data (improved) 115

Figure 5-11: Mean percentage of state contributions of the 100th order 1-D Burgers Equations system balanced training LLMs 117

Figure 5-12: The RK4 solution of the 100th order 1-D Burgers model versus the 4th order meta-model solution with quantitative confidence intervals.............119

Figure 5-13: 100th order 1D-Burgers equation NDS time-varying parameter $a21(t)$ part of the reduced-order canonical state matrix $\boldsymbol{A}rc$, $a21(t)$ was plotted for the first few seconds during the training and the solution of the meta-model 121

Figure 6-1 Normalized local output errors between the full-order LLMs and the Krylov-reduced LLMs at the collected training data of the NTL CI-NDS.........139

Figure 6-2 Mean percentage of state contributions of the balanced Krylov-reduced LLMs for the NTL meta-model collected training data 141

Figure 6-3 NTL meta-models response to test input $t = H(t - 0.05)$ versus the RK4 response of the CI-NDS 147

Figure 6-4 NTL meta-model multi-step test input 148

Figure 6-5 NTL meta-models response to the multi-step test input versus the RK4 response of the CI-NDS 148

Figure 6-6 NTL meta-models response to cosine test input versus the RK4 response of the CI-NDS 149

Figure 6-7 NTL meta-model response to exponential test input versus the RK4 response of the CI-NDS 150

Figure 6-8 NTL meta-models response to sinusoidal test input versus the RK4 response of the CI-NDS 151

Figure 6-9 NTL meta-models response to sinusoidal linear frequency sweep test input versus the RK4 response of the CI-NDS 152

Figure B. 1 a Step function. 191

Figure B. 2 Step function approximation ($K = 1$). 193

List of Abbreviations

ANN	Artificial Neural Networks
CFD	Computational Fluids Dynamics
CI	Computationally Intensive
CSTR	Continuously-Stirred Tank Reactor
EA	Evolutionary Algorithms
ED	Exact-Discretization
EM	Expectation Maximization
ETI	Execution Time Index
FEA	Finite Element Analysis
FSGP	Fixed Structure Gaussian Process
GP	Gaussian Processes
GPM	Gaussian Processes Model
HSV	Hankel Singular Values
KLD	Karhunen-Loéve Decomposition
LLM	Local Linear Model
LMN	Local Model Networks
LS	Least Square
MEMS	Micro-Electro-Mechanical Systems
MIMO	Multi Input Multi Output
MSE	Mean Squared Error
NARMAX	Nonlinear Auto Regressive, Moving Average and Exogenous Input
NARX	Nonlinear Auto Regressive with Moving Average
NDS	Nonlinear Dynamical System
NTL	Nonlinear Transmission Line
ODE	Ordinary Differential Equations
PCA	Principal Component Analysis
PDE	Partial Differential Equations
PEM	Prediction Error Minimization
POD	Proper Orthogonal Decomposition
PR	Polynomial Regression
RBF	Radial Basis Functions
RK	Runge-Kutta
RSD	Relative Standard Deviation
SEM	Simulation Error Minimization
SISO	Single Input Single Output
SMSE	Standardized Mean Squared Error
TBR	Truncated Balanced Realization
TPWL	Trajectory Piecewise Linear
VBL-LPV	Velocity-Based Linearization-Linear Parameters Varying
ZOH	Zero-Order Hold

Chapter 1 - **Introduction**

1.1 Computationally Intensive Models

To discuss *Computationally Intensive* (CI) models, the concepts of *system*, *model*, and *simulator* are presented (Troch and Breitenecker, 2000):

A system is "*a structured sum of elements with well-defined properties and well-established relations between these elements and with the environment"*.

A model is "*an image or abstraction of reality, a mental, physical or mathematical representation or description of an actual system. Modelling is the development of equations, constraints and logic rules to describe the system*".

An experiment performed on the model is called *simulation;* this is performed using a simulator which is a piece of computer code that describes the model. Simulation is the exercising of a model.

A CI model can be defined as the model which requires an extensive amount of computational resources to run its simulator. Even though computational resources continue to expand and grow in power and speed, CI model simulation codes continue to grow in complexity and remain computationally expensive (Meckesheimer et al., 2002, Castelletti et al., 2012). Most engineering processes require a number of runs of the model simulator in their design, validation testing, optimization etc. stages, when this simulator is CI; it will result in an expensive design cycles. CI simulators are also used in settings where physical system experimentation is impossible such in climate modelling (Levy and Steinberg, 2010).

There are many examples of CI models such as:

- *Finite Element Analysis* models (FEA) of mechanical structures cf.eg. the simulation of bumper-rail assembly of a pickup truck (Farhang-Mehr and Azarm, 2005), this model involve $10^5 - 10^6$ degrees of freedom and requires many hours or days of computer time. Another example is the simulation of *Micro-Electro-Mechanical Systems* (MEMS) models (Cao et al., 2005).

- *Computational Fluid Dynamics* (CFD) models of many systems such as the simulation of gas turbine engines (Reed and Afjeh, 2000) and the simulation of environmental problems such as greenhouse gas emissions (Zhou et al., 2004).

- CI models of electronic circuits such as the simulation of large DC/DC converters for a space station (Karimi et al., 1996), these systems are expensive to simulate due to size and high level complexity. Simulation of nonlinear transmission lines (Chen and White, 2000).

1.2 The Concept of a Meta-Model

A *meta-model* is a "*model of a model*" (Blanning, 1974), also known as a *surrogate* model or an *approximation* model, the meta-model is an approximation model of a CI model, it has the important property of being computationally less expensive than a CI model.

A meta-model is not a problem approximation but a model approximation; an example for this is the *Computational Fluids Dynamics* (CFD) model of a turbine blade (Jin, 2005) modelled using three-dimensional Navier-Stokes equations, this model can be approximated by a CFD model using three-dimensional Euler equations (notice that the approximation is using the same model class). Another example of this in the model of nonlinear thermal dynamics of multi-zone buildings (Goyal and Barooah, 2012) in which the original set of nonlinear of *Partial Differential Equations* (PDE) are replaced by a reduced-order approximation based on Taylor series expansion.

The meta-model concept has its roots in the topic of sensitivity analysis of CI model simulator (Kleijnen, 1975) which have shaped a set of properties a meta-model should have:

- The meta-model is computationally less expensive than a CI model.
- The meta-model may not include the full set of original inputs of the CI model; using the meta-model with a different input set after training may reduce its accuracy.
- The meta-model should provide an error quantification of the predictions.

Villa-Vialaneix et al. (2012) describes meta-model advantages of being:

- Easier integration into other processes and simulation platforms.
- Faster execution and reduced storage needs to simulate one output.
- Easier applicability across different spatial and/or temporal scales.

Meta-models structures are generally divided into two types (Shan and Wang, 2010):

- *Parametric* meta-models such as *Polynomial Regression* (PR) models (also known by *response surfaces*), *splines* (*piece-wise polynomial functions*), *Artificial Neural Networks* (ANN) and *Radial Basis Functions* (RBF).
- *Non-parametric* or *interpolating* meta-models such as *Gaussian Processes* (GP) otherwise known by *Kriging*.

Zhou et al. (2007) distinguishes parametric meta-models as a generalization based on training data, from interpolating models being restricted to a certain training data. Parametric meta-models suffer from the *curse of dimensionality*, or *over-fitting* when the number meta-model training data is sparse compared to the number of meta-model parameters. *Non-parametric* meta-models generally have far less parameters (*hyper-parameters*) and they do not suffer from over-fitting problems with a limited size training data set , they allow online training data addition to improve quality of predictions (El-Beltagy and Keane, 2001).

In terms of computational speed, parametric meta-models are faster to train and to predict the solution when the CI model dimensionality is low (Villa-Vialaneix et al., 2012). Their computational speed is independent of the size of the available training data.

Within the group of parametric meta-models, PR and splines methods are the easiest to train and the fastest to predict solutions, however they are confined to low-dimensionality CI models, RBF meta-models are more robust than PR and splines methods (Barton, 1998, Giunta et al., 1998).

Non-parametric meta-models have the distinct advantage in that they provide a confidence measure (uncertainty) about the predictions, which make them more useful as meta-models in general and especially in the context of sensitivity analysis.

Meta-models have been part of many engineering processes such as the simulation of environmental problems using parametric and non-parametric meta-models (Villa-Vialaneix et al., 2012), simulation of freshwater reservoir described by PDEs (Castelletti et al., 2012), simulation of structural reliability using Kriging meta-models (Sudret, 2012), the evolutionary optimization of FEA and CFD models using GP meta-models (El-Beltagy and Keane, 2001), sensitivity analysis of computer simulations using *Evolutionary Algorithms* (EA)(Storlie et al., 2009), electromagnetic design optimization using Kriging meta-models (Martone et al., 2007), aerodynamic wing design using Kriging meta-models (El-Beltagy and Keane, 1999) and within the general context of multi-disciplinary design optimization (Simpson et al., 2004, Jin, 2005, Egea et al., 2007, Shan and Wang, 2010).

The process of building a certain meta-model is concerned with the following points:

i) Analysis; this includes examining the set of inputs, states and outputs of the CI model. The analysis of the available CI model response data and the design of experiments to detect different dynamical system behaviour such as nonlinearities, discontinuities and randomness[1].

ii) Selection of a suitable meta-model structure; different meta-model structure have different properties it terms of model transparency, suitability for high-dimensional training data, solution error management, amount of time to setup the model etc.. This will be further discussed in the next chapter.

iii) The design and selection of input space parameters; this is a crucial step to formulate the model's dimensionality which often relates to the nature of a computational intensiveness of a simulator.

iv) Meta-Model training and validation; this includes the design of experiments that will generate response data using the CI model to identify the meta-model parameters.

[1]CI stochastic dynamical systems are an important part of systems theory however; this thesis will only focus on the meta-modelling of deterministic CI models.

v) Meta-Model testing; this includes design of experiments to test the constructed meta-model. This will be accomplished by testing the meta-model using new set of inputs unseen during the meta-model training and validation phases. A set of measures will be used to evaluate the results such as model fitness and uncertainty information associated with the predictions.

The trade-off between a meta-model and the CI model is the computational speed improvement versus the reduced accuracy; the CI model is a generalization of the underlying system while the meta-model often focuses of certain parts of the system. Engineering processes designer probably will spend considerable time to train, validate and test a certain meta-modelling strategy, but this is only done one time, then the meta-model can replace the CI model to speed up the engineering process design cycle.

1.3 Motivations and Contributions

Meta-model approach selection depends mainly on the dimensionality of the CI model with parametric meta-models aimed at CI models with moderate dimensionality and non-parametric meta-models usually reserved for high-dimensionality problems with sparse training data.

Jin et al. (2001) identified a number of meta-model performance measures in terms of accuracy, efficiency, robustness, simplicity and transparency. Shan and Wang (2010) after surveying two hundred research papers on meta-modelling techniques within the context of multi-disciplinary optimization have concluded that "*most of the research focuses on the meta-modelling strategy itself but neglect studying and taking advantage of characteristics of the underlying CI model*". This is driven by the fact that all the meta-modelling approaches are in fact a form of black-box system identification method with the distinct difference that they are operating on the CI simulator rather than the underlying system. This raises the issue of meta-model transparency, in that all of the current meta-modelling strategies are not transparent. Castelletti et al. (2012) noted that "*users of meta-models tend to be reluctant to let their CI models be operated upon by some numerical and statistical identification tool and then replaced by a meta-model that is seen to be the result of some obscure black-box procedure*".

Analytical CI *Nonlinear Dynamical Systems* (NDS) are used to model many practical systems such as turbine aero-engines, nonlinear transmission lines, nonlinear electronic circuits and many CFD and FEA methods that can be reduced to a set of nonlinear *Ordinary Differential Equations* (ODE). Analytical CI-NDSs are transparent models and their systems are fully described from first-principle. However, applying current meta-modelling techniques to analytical CI-NDS models does not utilize their internal structure and will obscure the transparency of those models.

Having identified these two shortcomings in the literature of the current meta-modelling strategies, the goal of this thesis is to device, implement and test a new meta-modelling approach specifically tailored for analytical CI-NDSs, this new meta-modelling approach will have the following properties:

- Utilize the internal structure of CI-NDSs to construct an accurate meta-model.
- The meta-model will be transparent, i.e. its structure will relate directly to the underlying dynamics of the original model.
- Provide uncertainty information about the meta-model predictions.

The thesis suggest to replace the analytical CI-NDS with *Velocity-Based Linearization-Linear Parameters Varying* (VBL-LPV) system. VBL-LPV system is constructed from *Jacobians* of the linearization of the CI-NDS around preselected set of training points to cover the operating space of the CI model. VBL-LPV system has the property that it is not an approximation but an exact representation of the CI-NDS dynamics; it is also a transparent realization of the original CI model. After constructing the *Local Linear Models* (LLM) of the VBL-LPV, a suitable linear *model order reduction* (MOR) approach is applied to reduce the dimensionality of the meta-model realization. GP models are used to blend the reduced-order VBL-LPV meta-model parameters; this will give it an advantage to deal with sparse training data, and will provide uncertainty information about the meta-model parameters which can be used to improve the accuracy of those parameters. Propagation of uncertainty from the meta-model parameters to the meta-model solution will be derived and used to improve the accuracy of the meta-model predictions. The new meta-modelling approach is called *Gaussian Processes blended Reduced-Order Velocity-Based Linearization-Linear Parameters Varying* system (GP blended reduced order VBL-LPV system).

The GP blended reduced-order VBL-LPV system has the potential of having a reduced computational cost compared to the CI-NDS, because it involves solving a reduced-order set of time-varying LLMs. It also preserves the transparency of the CI-NDS because it is coupled to the original dynamics of the CI-NDS via linearization and it utilizes the underlying structure of the CI-NDS when computing training the meta-model parameters.

The thesis main contributions are:

- The application of linear MOR projection-based methods to the GP blended VBL-LPV system.

- Propagation of uncertainty of the GP blended VBL-LPV system from the time-varying parameters to the model solution.

1.4 Thesis Outline

Chapter two is to review different modelling techniques of NDSs with emphasis on the model structure suitability for the meta-modelling of analytical CI-NDS. It will be focusing on the modelling structure transparency, its applicability to high-dimensional problems, and its capacity to deal with sparse training data and the global uncertainty bounds (or lack of) on the model predictions.

Chapter three will reintroduce the GP blended VBL-LPV model structure. It will review the math of GP regression model and thoroughly discuss the full-order meta-model training, validation, and testing methodology using a simple NDS.

Chapter four is the first thesis contributions chapter; it will review linear model order reduction techniques and their applicability to the meta-modelling of CI models in general. This will be followed by a detailed implementation of model order reduction for the proposed meta-model structure of Chapter three.

Chapter five is the second thesis contributions chapter; it will present a complete derivation of the uncertainty propagation for the proposed meta-model and how it can be applied to improve the accuracy of the proposed meta-model predictions.

Chapter six discusses the computational complexity of the proposed meta-model structure and wraps up with an example implementation of the meta-model for an analytical CI-NDS.

Chapter seven discusses thesis conclusions and recommendations. Finally, appendices A and B contain support materials for the research.

Chapter 2 - **Mathematical Modelling of a NDS**

2.1 Introduction

This chapter sets out to compare and choose a suitable analytical CI-NDS meta-model structure by exploring different model classes used in modelling NDSs in general. The main focus will be on NDS model classes that are transparent and the ones that can utilize the underlying structure of the model itself. The chapter will be used to advance the preceding thesis arguments about the proposed GP blended VBL-LPV system.

The chapter will outline NDS global function approximation methods in section 2.2, however it will not dwell on these concepts because they are essentially black-box models therefore not transparent and does not fit with argument of this thesis for selecting a transparent meta-model structure and the utilization of the analytical NDS internal structure.

Next, the chapter will review NDS model classes that are based on the divide and conquer (also called operating space decomposition) approach (sections 2.3 and 2.4) and within that the suggested VBL-LPV system. Divide and conquer model structures are more transparent than global function approximation methods, therefore fits nicely with the argument of the thesis. Section 2.5 will discuss the non-parametric GP models of NDS from the control and the applied statics perspectives and their suitability as a meta-model within the selection criteria of the thesis.

Within the context of NDS models, a *physical system* can be defined as the interaction of a group of physical components to perform a certain job or a function. A physical system consists of:

i) *Outputs;* represent the group of observable and measured physical quantities.

ii) *Inputs;* represent the group of physical quantities that can affect the outputs in the physical system. Some inputs can be modified by the observer to produce desired changes in the outputs, these are called *controlled inputs*. Other types of inputs are observable but not controllable and they are called *exogenous inputs*, these inputs sum the environment effects on a physical system.

"*A dynamical system is a mathematical concept that is used to describe a physical system by observing the relation of its components with time*" (Kalman et al., 1969). The *states* of the dynamical system (a function of time) represent the "*minimal set of variables that describe the dynamics of the physical system*" (De Silva, 2009). A dynamical system model consists of inputs and outputs and states.

Depending on the mathematical concept used in describing a physical system, dynamical system models can fall in one of three main categories:

i) *First-Principle* (otherwise called *white-box*) model is a mathematical model where all the dynamics of the physical system can be defined by the use of mathematical expressions which reflect the best knowledge of the mathematical rules that govern the dynamics of the physical system.

ii) *Black-Box* model is a predefined mathematical structure with its parameters identified by the use of datasets from the dynamical system response data (inputs, outputs and states).

iii) *Grey-Box* model is a hybrid model where first-principle model and black-box model are both incorporated in defining certain parts in the structure of the model.

Let S be a finite dimensional, continues time, nonlinear, time-invariant, and smooth dynamical system given by:

$$\begin{aligned}\frac{d\boldsymbol{x}(t)}{dt} &= \dot{\boldsymbol{x}}(t) = \boldsymbol{F}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ \boldsymbol{y}(t) &= \boldsymbol{G}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ \boldsymbol{x}(t_0) &= \boldsymbol{x}_0\end{aligned} \tag{2.1}$$

$\boldsymbol{u}(t) = [u_1(t), u_2(t), \dots, u_m(t)]^T, \boldsymbol{u}(t) \in \mathbb{R}^{m\times 1}$ is a vector of inputs to S.

$\boldsymbol{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T, \boldsymbol{x}(t) \in \mathbb{R}^{n\times 1}$ is a vector of states of S and n is the *dynamical system order*, $\dot{\boldsymbol{x}}(t) \in \mathbb{R}^{n\times 1}$.

$\boldsymbol{x}_0 \in \mathbb{R}^{n\times 1}$ is the initial state of the dynamical system at $t = t_0$.

$\boldsymbol{y}(t) = [y_1(t), y_2(t), \dots, y_p(t)]^T, \boldsymbol{y}(t) \in \mathbb{R}^{p\times 1}$ is a vector of outputs of S.

$\boldsymbol{F}: \mathbb{R}^{n\times 1} \times \mathbb{R}^{m\times 1} \to \mathbb{R}^{n\times 1}$ is the system (state) mapping.

$\boldsymbol{G}: \mathbb{R}^{n\times 1} \times \mathbb{R}^{m\times 1} \to \mathbb{R}^{p\times 1}$ is the output mapping.

The mathematical solution to equation (2.1) is called the *trajectory* of the dynamical system and is given by:

$$\boldsymbol{x}(t) = \int_{t_0}^{t} \dot{\boldsymbol{x}}(\tau)d\tau = \int_{t_0}^{t} \boldsymbol{F}\big(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)\big)d\tau \tag{2.2}$$

Equation (2.2) means that if the state of the dynamical system is known at initial time $\tau = t_0$ and the input to the dynamical system is known at both times $\tau = t_0$ and future time $\tau = t$ then the future state of the dynamical system $\boldsymbol{x}(t)$ can be found (De Silva, 2009).

Dynamical system models are compared based on the following properties (De Silva, 2009):

i) *Transparency* of a model is how transparent a model is in describing the underlying physical system that it was formed from. Since first-principle models are directly formed from mathematical relations that describe physical system dynamics, they are the most transparent way to model a physical system.

   The dynamical system transparency is a measure that describes in general how well the model clearly relates to the underlying dynamics of its physical system.

ii) *Reliability* is the model ability to successfully predict physical system behaviour each time new operating conditions are introduced. First-principle models that accurately describe the physical interpretations of a physical system are considered to be the most reliable models.

iii) *Scalability* is the model ability to scale and expand with the introduction of new physical system components while maintaining the same structure of the model itself. Since first-principle models are mathematical concepts based on physical insights of the physical system, scaling and expanding them will also produce first-principle models, therefore; the model structure is preserved.

iv) *Complexity* of a model describes the mathematical complexity of the dynamical system model. Dynamical models can get more complicated with the increase of independent dynamical interpretations of the physical system they are formed from.

v) *Computational burden* of dynamical system models describes the associated computational cost of running a dynamical system model simulator. This computational cost comes from the evaluations of mathematical expressions contained in the model and for complex dynamical system models this can be *a* computationally expensive process.

2.2 The Nonlinear Auto Regressive, Moving Average and Exogenous Input NDS Structure

The *Nonlinear Auto Regressive, Moving Average and Exogenous Input* (NARMAX) is one of the important NDS structures, it was introduced in (Leontaritis and Billings, 1985a, b). NARMAX is concerned with discrete time representations of the NDS, two conditions must be met for it to be used (Chen and Billings, 1989):

i. The dynamical system must have a finite dimensional state space representation.

ii. If the dynamical system is operating near or at a certain equilibrium point then a linearization must exists for it at that point.

A NARMAX of a *Single Input Single Output* (SISO) discrete time dynamical system with input noise is given in as (Chen et al., 1990b):

$$y(k+1) = f\begin{pmatrix} y(k-1), y(k-2), \dots, y(k-n_x), \dots, \\ u(k-1), u(k-2), \dots, u(k-n_u), \dots, \\ e(k-1), e(k-2), \dots, e(k-n_e) \end{pmatrix} + e(k) \tag{2.3}$$

$k \in \mathbb{Z}$ is the discrete time step.

$e(.)$ is the input noise term.

$f(.)$ is a nonlinear function.

$n_y,\ n_u, n_e \in \mathbb{Z}$ are the number of delayed sample terms observed in the outputs and inputs and noise respectively.

A NARMAX system can have infinitely different yet equivalent realizations similar to equation (2.3) (Chen and Billings, 1989); therefore a minimal representations is required i.e. the minimum value for n_y, n_u and n_e that can successfully describe the nonlinear system at hand. A NARMAX can be used to represent *Multi Input Multi Output* (MIMO) NDS as well (Chen et al., 1990b).

2.2.1 The NARMAX Model Classes

The design process of a NARMAX NDS starts by selecting a class for the function $f(.)$ in equation (2.3). Several classes for $f(.)$ that exist in the literature (Aguirre and Letellier, 2009) are:

i) The Polynomial model class (Leontaritis and Billings, 1985b, Chen and Billings, 1989) which is linear in parameters making the process of parameter identification straight forward (Mendez and Billings, 2001) using *Least Squares* (LS) based algorithms.

ii) Rational model class (Billings and Zhu, 1991); where the function $f(.)$ is the ratio of two polynomials; though the polynomials in this model are linear in parameters; the ratio between them will result in a model structure that has nonlinear parameters. (Aguirre and Letellier, 2009) suggest that the identification of parameters in this class can lead to numerical instability issues that are known to be non-trivial.

iii) Artificial Neural Network model class (Narendra and Parthasarathy, 1990); the ANN can consist of one hidden layer with a linear output layer and activation functions similar to those in the hidden layer. Using more than one hidden layer will result in a modelling class where the identified parameters are highly nonlinear (Chen et al., 1990b). There is also the problem of specifying the number of hidden layers and the number of neurons in each layer (Sjöberg et al., 1995).

iv) Radial Basis Function model class (Chen et al., 1990a); is similar to the one layered ANN modelling class except that there are no weights between the input layer and the hidden layer (Aguirre and Letellier, 2009); instead the nonlinear weights of the radial basis functions model class are identified.

v) Wavelet Networks (Liangyue et al., 1995); is similar to the RBF approach except that they are formed using wave-nets class of functions. (Aguirre and Letellier, 2009) suggest that there is no clear guide on how to select these functions in practical implementations. Again the weights associated with this class are nonlinear.

vi) Fuzzy logic model class; where linguistic terms are weighted and used to model the NDS particularly using RBF and ANN. Incorporating linguistic terms in the model can prove useful to some problems in the modelling of dynamical systems.

2.2.2 The NARMAX Model Structure

The practical design and identification of NARMAX models starts by the selection and identification of a starter set of structures belonging to a certain class using an algorithm which analyse the input/output training data. This process can yield a set of structures that are all valid representations of the NDS. To complete the design process, another algorithm is used to select and identify the final parsimonious set of NDS representations. Aguirre and Letellier (2009) suggest that there is no clear methodology to select the most parsimonious structure descriptive of physical system.

Hong et al. (2008) suggest that from the classes of NARMAX system models discussed above, modelling classes which are linear in the parameters have verifiable learning and convergence conditions and can be implemented using parallel processing which may count towards the goal of reducing the computational cost of the model learning and predictions.

Some authors base their model structure selection upon *Prediction Error Minimization* (PEM) metrics while others use *Simulation Error Minimization* (SEM) metrics. The parameters of the chosen structure are identified using *Least Square* (LS) regression algorithms, Farina and Piroddi (2011) suggest to use a fully SEM algorithm to select both the structure and parameters in one go. They argued that this approach has more advantages than the more known and used PEM metric, because PEM metric can produce inaccurate and unstable models especially if the noise model does not describe the system generating the model's training data.

The more recent non-parametric (probabilistic) approach to select and train the parameters of the NDS input-output model is the *Expectation Maximization* (EM) algorithm (Baldacchino et al., 2012). This approach involves the maximization of the likelihood function in the training phase of the models parameters. Since EM approach is probabilistic in nature, it will result in a model with uncertainty information about the predictions.

2.2.3 Meta-Modelling a CI-NDS with NARMAX

All NARMAX model classes are black-box models; they will result in a non-transparent dynamical system realization of the CI model that does not directly relate to the dynamical description of the underlying physical system.

Aside from the NARMAX model transparency issue, the model class itself is not suitable to deal with high-dimensionality problems, because such problems will need a considerable amount of training information to reach any reasonable accuracy.

The EM structure selection and parameter training of the polynomial *Nonlinear Auto Regressive and Exogenous Input* (NARX) system discussed in (Baldacchino et al., 2012) is more suited to deal with high-dimensionality problems with sparse training data.

As for uncertainty information, all the model classes that use parametric regression methods in the structure selection and parameter training will not provide uncertainty information about the predictions with the exception of the EM approach.

At this point, it is clear that none of the preceding NARMAX models agree with the thesis argument regarding meta-model transparency and utilization of the internal structure of analytical NDSs.

2.3 Divide and Conquer Methods

An improvement to the transparency of global function approximation methods was introduced in (Johansen and Foss, 1992) based on the idea of Divide and Conquer. Instead of treating the NDS as a global optimization problem, the operating space of the NDS is divided into several regions where each region can get its local model. This allowed mixing first-principle and black-box modelling techniques to form a grey-box global model that have better transparency than global function approximation models.

Johansen and Foss (1992) suggested to use *Local Model Networks* (LMN) to describe the local dynamics of the divided NDS operating space. After dividing the operating space of the NDS into sub-regions, each local model is assigned a weight that is a function of a scheduling vector; this will determine the contribution of the LLM to the global model response at any point in the operating space of the NDS.

The authors in (Leith and Leithead, 2002, Leith and Leithead, 2003) suggested to use LPV systems. LPV systems in contrast to LMNs produce a global model that is identical to the LLM structure it was formed from. This gave LPV systems a transparency advantage when modelling NDS compared with other NDS identification methods.

2.4 Velocity-Based Linearization

The concept of modelling a NDS with a family of VBL-LLMs was introduced within the context of Divide and Conquer modelling framework for gain-scheduling control of Single-Input Single-Output (SISO) nonlinear wind turbine systems in (Leith and Leithead, 1996), extended to Multiple-Inputs Multiple-Outputs (MIMO) nonlinear wind turbine systems in (Leith and Leithead, 1998a). The framework was set for the modelling of continues-time NDS and it has shown that this model is an exact representation of the original NDS system with proven stability analysis of the resulted VBL-LPV system (Leith and Leithead, 1998b, 1998c, 1999, 1999, 2000). The concept of VBL-LPV system continued to be successfully used in gain-scheduling control of NDS for example in auto-pilot design for agile missile (Leith et al., 2001), internal mode controller for pH-neutralization process (Toivonen et al., 2003) which describes the framework when dealing with discrete-time NDS, control for turbo fan engine (Reberga et al., 2005), nonlinear tracking of NDS (Guang-Bin et al., 2010), modelling and control of an air breathing hypersonic vehicle (Cai et al., 2011) and aero-engine nonlinear model (Yu et al., 2011). The VBL-LPV NDS model has not been as popular as global function approximation methods (black-box models) because it was limited to model analytical NDSs, aside from the cited applications of this system, very little research has gone into exploring it.

To explain the blended VBL-LPV model of a NDS, reconsider the NDS of interest in equation (2.1). If the NDS is following a trajectory passing through a general operating point at time $t = t_i$, then, the system at that point is described as

$$\dot{\boldsymbol{x}}(t_i) = \boldsymbol{F}(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)) \tag{2.4}$$

Assuming the system is continuously differentiable at that point, it can be approximated using first-order Taylor series expansion given as

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{F}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \\ &= \boldsymbol{F}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) + \nabla_{\boldsymbol{x}} \boldsymbol{F}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) \boldsymbol{\delta x}(t) \\ &\qquad + \nabla_{\boldsymbol{u}} \boldsymbol{F}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) \boldsymbol{\delta u}(t) \end{aligned} \tag{2.5}$$

And the small signal perturbations are given as

$$\begin{aligned} \boldsymbol{\delta x}(t) &= \boldsymbol{x}(t) - \boldsymbol{x}(t_i) \\ \boldsymbol{\delta u}(t) &= \boldsymbol{u}(t) - \boldsymbol{u}(t_i) \end{aligned} \tag{2.6}$$

Let $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ be appropriately dimensioned parameters that are given by

$$\begin{aligned} \boldsymbol{A}_i &= \nabla_{\boldsymbol{x}} \boldsymbol{F}(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)) \\ \boldsymbol{B}_i &= \nabla_{\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)) \end{aligned} \tag{2.7}$$

And let,

$$\boldsymbol{\gamma}_i = \boldsymbol{F}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) - \big(\boldsymbol{A}_i \boldsymbol{x}(t_i) + \boldsymbol{B}_i \boldsymbol{u}(t_i)\big) \tag{2.8}$$

The LLM is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_i \boldsymbol{x}(t) + \boldsymbol{B}_i \boldsymbol{u}(t) + \boldsymbol{\gamma}_i \tag{2.9}$$

Examining the LLM in (2.9), it is locally linear and it describes the NDS anywhere in the operating space. When the NDS reaches an operating point that is an equilibrium point, the $\boldsymbol{\gamma}_i$ term becomes zero. Johansen et al. (1998) have shown that by having number of these LLMs and interpolating their outputs, an accurate reconstruction of the NDS can be obtained.

From model transparency point of view, LLMs relate to the underlying dynamics of the physical systems when they are operating near equilibrium points of the NDS, but at transient regions of the NDS operating space, they lack the interpretability as to what do they mean when viewed in terms of the dynamics of the underlying physical system. This issue was pointed out in (Leith and Leithead, 1999, Shorten et al., 1999, Murray-Smith et al., 1999).

Differentiating equation (2.9) w.r.t. time and substituting $\boldsymbol{w}(t)$ for $\dot{\boldsymbol{x}}(t)$ yields

$$\dot{\boldsymbol{w}}(t) = \boldsymbol{A}_i \boldsymbol{w}(t) + \boldsymbol{B}_i \dot{\boldsymbol{u}}(t) \tag{2.10}$$

Equation (2.10) is the VBL form of the LLM (Figure 2-1). It is linear and consists of only two parameters that change with the current operating point (the term $\boldsymbol{\gamma}_i$ is constant and thus becomes zero upon differentiating). This form of LLM overcomes the interpretability problem that first-order Taylor's series expansion LLM suffered from. It provides a transparent model that directly relates to the dynamics of the underlying physical system. VBL-LPV systems faithfully describe the nonlinear dynamics anywhere in the operating space (including points not local to an equilibrium operating point). A LLM in VBL form requires having the derivative of the input signal. In the case that the input signal is not continuously differentiable (e.g. a step function), a continues time mathematical approximation may be available (McLoone et al., 2001) (e.g. a sigmoid function for a step function) that is continuously differentiable. This approach is explained further by Appendix B.

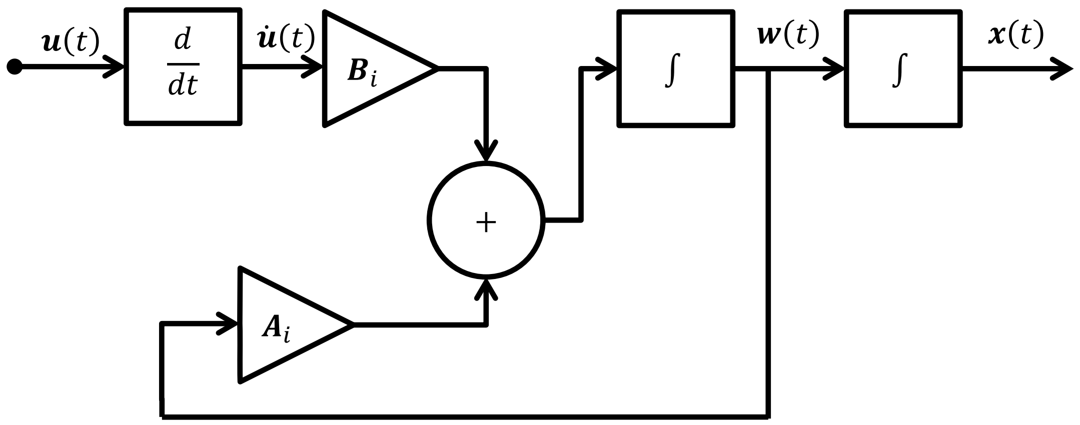


Figure 2-1: VBL ith LLM in state-space form.

There are couple of drawbacks in using the VBL-LPV system, the first being the requirement of access to the NDS internal states which limit the frame work to analytical NDSs, and the second is the requirement of input derivative.

A drawback of using the VBL-LPV modelling framework is the requirement of solving second-order set of linear equations. This framework is only available to model NDS whose states are accessible.

The identification of the VBL-LPV parameters is divided between two regions of the operating space in a way that covers as much as possible of the change in dynamics of interest. At equilibrium points this can be done using many linear system identification methods such as subspace system identification (Ljung, 1998). The VBL-LPV system LLM at a certain operating point is computed using equation (2.7) for the state part and the parameters for the output part are given as:

$$\begin{aligned} \boldsymbol{C}_i &= \nabla_{\boldsymbol{x}} \boldsymbol{G}(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)) \\ \boldsymbol{D}_i &= \nabla_{\boldsymbol{u}} \boldsymbol{G}(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)) \end{aligned} \tag{2.11}$$

However, in the case of off-equilibrium points of the operating space, it is difficult to determine the parameters of the VBL-LLM using any linear systems identification technique and therefore it must be calculated analytically using first-principle NDS Jacobians. This is why VBL-LPV model class is restricted to analytical NDS only.

The NDS spends so little time at a transient point which makes the process of collecting enough amounts of data of sufficient quality for parameter identification very hard. The identification data sparsity problem relative to off-equilibrium points is all also called the problem of *off-equilibrium dynamics*. Murray-Smith et al. (1999) were the first to point out this problem. This problem is not specific to the divide and conquer approach and was pointed out throughout the literature of NDS identification.

2.5 NDS modelling with Gaussian Processes

GP regression models are non-parametric, probabilistic models trained from empirical data. They provide predictions that are random variables with means and variances which can be used to better solve many engineering problems. GP models provide a dependency measure between predictions and their regressors and may lead to reductions in computational load (this will be explained further in section 3.3). GP models can be used to model NDS directly as black-box models or to blend parameters of grey-box representations of the NDS (such as LMN and LPV systems).

GP models were first used to model a NDS that came in an autoregressive form in (Murray-Smith et al., 1999). GP model is a non-parametric black-box model that is identified from response data of the system; therefore it suffers from model transparency issue. In addition to this problem, GP model training and predictions requires the inversion of a covariance matrix whose size depend on the size of the training data set, this pose an additional computational cost in comparison to parametric models of NDSs.

In order to decrease the training data set size, Leith et al. (2002) suggested to use a "hybrid local/global" GP model which incorporated local linearization of the NDS (derivative information) with response data from off-equilibrium regions of the system. Instead of using a group of points to describe the NDS behaviour in equilibrium, local linearization can describe the NDS by one training data point in terms of the linearization Jacobian at that point. This led to a reduction in the size of the covariance matrix which in turn improved the computational speed of a GP model. However, this model still suffered from model transparency issue.

GP regression models were used in creating LMN structures (Gregorčič and Lightbody, 2007). A global GP model was used to select the centres of the validity function of the LMN (by observing the variance). The local models were selected to be local GP models. The boundaries of the validity function were identified by observing the variance of the corresponding local GP models. Gregorčič and Lightbody (2007) have shown that by using local GP models based on a linear covariance function, an analytical expression can be reached for both the local GP model and the global LMN. However, GP identified LMN suffer just like the parametric version of a LMN when it comes to model transparency.

Azman and Kocijan (2006) proposed to use GP models to blend the parameters of VBL-LPV model. They used the GP blended VBL-LPV system to model continues-time NDS and later to model discrete-time NDS (Ažman and Kocijan, 2009), their method was called *Fixed Structure Gaussian Process* (FSGP). Since the FSGP model structure is based on the transparent VBL-LPV model structure, it inherited its transparency.

Under the study of simulators in computer experiments, it was proposed that non-parametric (probabilistic) meta-models of the CI-NDS simulator can be a valuable tool to "*analyse simulator discrepancies, quantify its uncertainties, sensitivity analysis, parameter uncertainty, residual variability, observation errors, model inadequacy and code uncertainty*" (Conti et al., 2004).

The study of computer emulators started with the research in (Sacks et al., 1989, Kennedy and O'Hagan, 2001). Two main approaches where suggested at the beginning to meta-model complex computer experiments of deterministic systems.

The first approach called the response surface (Sacks et al., 1989, Simpson et al., 2001) approach which assumes a class of low order polynomial regression (up to third order). It is noted that this approach when applied to simulators of dynamical systems resembles the polynomial class of NARMAX under a restrictive low order model structure.

The second more favourable approach is Kriging which is the Gaussian Processes model. The Gaussian Processes model is an exact interpolator (Kleijnen, 2009) in a sense that the model will exactly predict outputs for the same simulator input-output pairs it was trained with. The Kriging model is called non-parametric model because it does not force the function to have predetermined class or structure (Kennedy and O'Hagan, 2001).

There exits three main types of Kriging (Deng et al., 2012) depending on the choice of the mean function:

i) *Ordinary Kriging*; the mean function is constant, it is considered to be a wide spread choice (Baldi Antognini and Zagoraiou, 2010).

ii) *Universal Kriging*; the assumption of a fixed regression mean model.

iii) *Bayesian Kriging*; the assumption of Bayesian (probabilistic) means.

The Ordinary Kriging is 100 times faster than the Monte-Carlo emulation methods (Conti and O’Hagan, 2010).

The work of emulating complex computer codes started by designing emulators to deal with static simulators; The emulators of dynamic simulators first used a single step static emulator in a recursive approach (Conti and O’Hagan, 2010). This approach has to deal with drawbacks like the need to extend the theory of static emulators to dynamic emulators; higher accuracy requirements are needed at each time step and the fading of computational advantage over the Monte-Carlo emulation methods for large time spans. In addition to this; the assumption that a recursive emulator having a Gaussian distribution is not true if the second time step was fed with a Gaussian distribution input rather than the mean of the prediction distribution.

Depending on the number of outputs a Kriging-based dynamic emulator has to represent; three approaches where discussed (Conti and O'Hagan, 2010):

i) Multi-Outputs Kriging; this approach consists of using a multi-output emulator and considered the simplest approach.

ii) Many Single-Output Kriging; in this approach each output is modelled using a single emulator.

iii) Time Input Kriging; this approach involves building a single-output simulator with the simulation time vector included as part of the input space. This approach will demand more computational load due to the inclusion of the simulation time as direct input component.

The detailed implementation of the GP regression model will discussed further in chapter three.

2.6 Meta-modelling with GP blended VBL-LPV system

The goal of this thesis was to propose a meta-modelling technique tailored for analytical CI-NDS in a way that preserve the transparency of the meta-model and the underlying structure of the analytical CI-NDS is utilized in the meta-model parameters identification process. As stated before; there cannot be any model more transparent than a model that was constructed from first-principle. Linear first-principle models may be more transparent and useful than equivalent nonlinear descriptions of the same system depending on the final goal of creating the meta-model such as in the design of controllers. Not all CI-NDS models are well understood from first-principle therefore, black box global function approximation methods such as NARMAX and its different classes are bound to be used to describe such systems.

For the rest of CI-NDS constructed from first-principle; the GP blended VBL-LPV system can faithfully describe them without the loss of transparency of the original model and perhaps improving that transparency further. Examples of analytical CI-NDS are systems encountered in FEA models, CFD models, MEMS and nonlinear electronic circuits etc. Solving a set of linear differential equations may consume less time than solving a nonlinear set of differential equations of similar dimensionality (Berkooz et al., 1993).

2.7 Conclusions

Parametric global function approximation methods lack global model transparency; they are not suitable to meta-model high-dimensional CI-NDS and they generally do not provide uncertainty information about their predictions with a notable exception of non-parametric global function approximation methods such as GP models but still suffering from model transparency issue; they however are the first choice to model a NDS not well understood from first-principle and they do not require accesses to the NDS internal states.

NDS models based on the idea of Divide and Conquer provide more transparent models than global function approximation methods, however, blending their parameters using parametric methods might not be wise when dealing high-dimensional sparse training data. This can be alleviated by using non-parametric blending methods such as GP models.

VBL-LPV systems have the most transparent model after the analytical CI-NDS model; they are not mathematical approximations but an exact description of the local model dynamics everywhere in the operating space. When mated with non-parametric blending methods such as GP models; they will be more suited to deal with high-dimensional sparse training data and will provide global uncertainty information about their predictions. VBL-LPV systems being locally (and globally) linear in nature, there is a room for the reduction of the computational cost of the proposed meta-model by employing reduced order modelling for the VBL-LLMs, this is one of the two main contributions of the thesis which will be studied in depth in chapter four.

The obvious problem with VBL-LPV system is that they require access to the CI-NDS states therefore restricting them to analytical CI-NDS models; however; there still exists a wide class of CI-NDS models well understood from first-principle such as FEA models, CFD problems, MEMS and nonlinear electronic circuits etc. that will benefit from the proposed meta-modelling technique.

Chapter 3 - Gaussian Processes Blended VBL-LPV system

3.1 Introduction

The idea of using VBL-LPV system to model a NDS (section 2.3) is an appealing method to choose for a meta-model because it provides a transparent global model that resembles a linear time-varying dynamical system. LPV systems require a model for scheduling the parameters according to the current operating point of the NDS, and from the parametric point of view, these models perform badly in high dimensions added to the problem of the training data sparsity in off-equilibrium regions of the NDS.

Azman and Kocijan (2006) suggested to use GP regression model to model the change in the LPV parameters over the operating manifold (section 2.5). The appealing properties of the GP regression model made it the logical choice to schedule the parameters of the VBL-LPV system.

The smoothing property of GP models meant it can achieve high accuracy compared to parametric methods using small number of identification data therefore reducing the effects of high dimensionality and off-equilibrium training data sparsity.

GP regression models provide dependency measures which relate the training data (the parameter) to the regressors (the operating point) which meant the ability to reduce the size of the scheduling vector which may lead in turn to increased computational speed.

This chapter will introduce the GP blended VBL-LPV meta-modelling structure in full order, it will examine global function approximation based on GP model and show how to apply it to compute the proposed meta-model time-varying parameters, it will also go through the details of designing the meta-model for a simple nonlinear dynamical system.

3.2 Meta-Model Structure

A GP blended VBL-LPV system global model is given as

$$\begin{aligned}\dot{\boldsymbol{w}}(t) &= \boldsymbol{A}(t)\boldsymbol{w}(t) + \boldsymbol{B}(t)\dot{\boldsymbol{u}}(t)\\ \boldsymbol{w}(t) &= \dot{\boldsymbol{x}}(t)\\ \boldsymbol{y}(t) &= \boldsymbol{C}(t)\boldsymbol{x}(t) + \boldsymbol{D}(t)\boldsymbol{u}(t)\\ \boldsymbol{w}(0) &= \boldsymbol{w}_0, \boldsymbol{x}(0) = \boldsymbol{x}_0\end{aligned} \tag{3.1}$$

The parameters of the VBL-LPV system are the predictions of the GPMs to the time-varying elements of each parameter matrix given as

$$\begin{aligned}\boldsymbol{A}(t) &= \boldsymbol{GPM_A}(\boldsymbol{x}(t))\\ \boldsymbol{B}(t) &= \boldsymbol{GPM_B}(\boldsymbol{x}(t))\\ \boldsymbol{C}(t) &= \boldsymbol{GPM_C}(\boldsymbol{x}(t))\\ \boldsymbol{D}(t) &= \boldsymbol{GPM_D}(\boldsymbol{x}(t))\end{aligned} \tag{3.2}$$

The numerical solution of equation (3.1) requires a suitable discretization method and numerical solvers which will be discussed later in this chapter. The VBL-LPV parameters blending using GPM will be discussed in the following sections after some background information about GP regression models.

3.3 VBL-LPV Parameters Blending with Gaussian Processes Regression

In the previous chapter; the review pointed out to the poor performance of parametric recursive input-output NDS models and divide and conquer NDS models when they use sparse data to train their parameters (the data scarcity comes from the difficulty to gather quality training data in off-equilibrium regions of the NDS to identify the parameters of the LLMs). In addition to this problem, the number of LLMs increases with the increase in the dimensionality of the NDS (this increase the number of weights to train) leading to high computational loads.

In the context of Bayesian curve fitting, O'Hagan and Kingman (1978) defended GP model for regression. Williams and Rasmussen (1996) in an effort to replace parametric function approximation methods like ANN, they used GP to model a nonlinear function. GP curve fitting models are probabilistic models with a number of *hyper-parameters* to train, in contrast to parametric curve fitting models; training data are used to train the hyper-parameters and to predict new function values. GP model provides a measure of confidence in their predictions which can be used to measure the quality of the predictions and the choice of suitable training data sets; it can also be used to increase the accuracy of the predictions.

3.3.1 Gaussian Processes

In the context of universal function approximation, following a similar derivation in (Rasmussen and Williams, 2006), consider a training data set $\boldsymbol{R} \in \mathbb{R}^{nt \times Nt}$ given below that consists from Nt training input vectors $\boldsymbol{r}_i \in \mathbb{R}^{nt \times 1}, i \in \{1,2,\dots,Nt\}$ and their corresponding noisy function observations $z_i \in \mathbb{R}$.

$$\boldsymbol{R} = \begin{bmatrix} r_{1,1} & \cdots & r_{1,Nt} \\ \vdots & \ddots & \vdots \\ r_{nt,1} & \cdots & r_{nt,Nt} \end{bmatrix} \tag{3.3}$$

The assumed relationship between the inputs and the noisy function observations are given as

$$z_i = f(\boldsymbol{r}_i) + \varepsilon_i \tag{3.4}$$

The noise component ε_i is a random variable with a Gaussian distribution of zero mean, and σ_n^2 variance given by

$$\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2) \tag{3.5}$$

The noise is assumed to be independent and identical across all the training data samples; in the case of noise-free samples, it can still be introduced in very low magnitude.

In GP regression, a Gaussian Process is a *random field* that maps the training data set $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \dots, \boldsymbol{r}_{Nt}\}$ to a set of random variables given by the noise-free function observations $\{f(\boldsymbol{r}_1), f(\boldsymbol{r}_2), \dots, f(\boldsymbol{r}_{Nt})\}$. A Gaussian process is specified by its mean function $m(\boldsymbol{r})$ and its covariance function $c(\boldsymbol{r}_i, \boldsymbol{r}_j)$.

$$f(\boldsymbol{r}) \sim \mathcal{GP}\big(m(\boldsymbol{r}), c(\boldsymbol{r}_i, \boldsymbol{r}_j)\big) \tag{3.6}$$

The probability distribution of the individual noise-free function observations is a Gaussian distribution specified by its mean vector $\boldsymbol{m} \in \mathbb{R}^{Nt\times 1}$, and a positive-definite covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{Nt\times Nt}$, it is given by

$$p(f(\boldsymbol{R})|\boldsymbol{R}) = \mathcal{N}(\boldsymbol{m}, \boldsymbol{\Sigma}) \tag{3.7}$$

The mean function $\boldsymbol{m}$ is usually set to zero. The covariance matrix elements $\Sigma_{i,j},\ i\,, j \in \{1,2, \dots, Nt\}$ are the covariates between individual noise-free function observations $f(\boldsymbol{r}_i)$ which are a function of the individual training inputs $\boldsymbol{r}_i$ specified by the covariance function $c(\boldsymbol{r}_i, \boldsymbol{r}_j)$. A choice for the covariance function with a distance measure θ_d , $d \in \{1,2, \dots nt\}$ is given by

$$\Sigma_{i,j} = c(\boldsymbol{r}_i, \boldsymbol{r}_j) = \theta_0 \exp\left(-\frac{1}{2}\sum_{d=1}^{nt} \theta_d \left(r_{d,i} - r_{d,j}\right)^2\right) \qquad (3.8)$$

θ_0 is the process variance parameter, θ_d can be considered as a dependency measure parameter that is used to determine the dependency of the training data targets on their corresponding inputs, the higher the value for θ_d the lower is the dependence. The use of a squared exponential covariance function like the one in equation (3.8) implies that the translations in the input space are smooth, similar inputs will have a higher correlation and using an exponential function will lead this correlation to increase or decrease faster.

Going back to the noisy regression model given earlier by equation (3.4), the probability distribution of the noisy function observations given the function inputs $p(\boldsymbol{z}|\boldsymbol{R})$ is also Gaussian with zero mean and a covariance matrix $\boldsymbol{K}_{tt} = \boldsymbol{\Sigma} + \sigma_n^2 \boldsymbol{I}$, $\{\boldsymbol{K}_{tt}, \boldsymbol{I} \in \mathbb{R}^{Nt \times Nt}\}$. Parameters of the covariance function with the additive noise variance are grouped together in a vector $\boldsymbol{\theta}$ called the hyper-parameters vector given by

$$\boldsymbol{\theta} = [\theta_0 \quad \theta_1 \quad \cdots \quad \theta_n \quad \sigma_n^2]^T \qquad (3.9)$$

The covariance function hyper-parameters must be determined before it can make any prediction with the GP model. Williams and Rasmussen (1996) suggested that the hyper-parameters to be inferred by minimizing a cost function of the hyper-parameters $J(\boldsymbol{\theta})$ given by the negative log of the probability distribution function of the *marginal likelihood* $p(\boldsymbol{z}|\boldsymbol{R})$. This cost function is given by

$$J(\boldsymbol{\theta}) = -\frac{1}{2}(log(det(\boldsymbol{K}_{tt}^{-1})) + \boldsymbol{z}^T\boldsymbol{K}_{tt}^{-1}\boldsymbol{z} + Nt\,log(2\pi)) \qquad (3.10)$$

Equation (3.10) is called the *Maximum Likelihood Estimator* (Martin and Simpson, 2005). The hyper-parameters optimization process starts by assigning initial values for the hyper-parameters then minimising by evaluating the partial derivative of equation (3.10) w.r.t. ith hyper-parameter θ_i as

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_i} = \frac{1}{2}\left(tr\left(\boldsymbol{K}_{tt}^{-1}\frac{\partial \boldsymbol{K}_{tt}}{\partial \theta_i}\right) - \boldsymbol{z}^T\boldsymbol{K}_{tt}^{-1}\frac{\partial \boldsymbol{K}_{tt}}{\partial \theta_i}\boldsymbol{K}_{tt}^{-1}\boldsymbol{z}\right) \qquad (3.11)$$

The goal is to predict the function value $z_p \in \mathbb{R}$ for a new input $\boldsymbol{r}_p \in \mathbb{R}^{nt\times 1}$. The prediction has a *prior* Gaussian probability distribution with a zero mean, variance k_{pp} and is given by

$$p(z_p|\boldsymbol{r}_p) = \mathcal{N}(0, k_{pp}) \qquad (3.12)$$

The prior variance k represents the auto-covariance of the prediction target and if calculated using the covariance function in equation (3.8) then $k_{pp} = \theta_0$.

The *joint probability* of the training targets $\boldsymbol{z}$ and the prediction target z_p is a multivariate Gaussian distribution function given by

$$p(\boldsymbol{z}, z_p) = \mathcal{N}\left(0, \begin{bmatrix} \boldsymbol{K}_{tt} & \boldsymbol{k}_{tp} \\ \boldsymbol{k}_{tp}^T & k_{pp} \end{bmatrix}\right) \qquad (3.13)$$

$\boldsymbol{k}_{tp} \in \mathbb{R}^{Nt\times 1}$ is a covariance vector which describes the covariates between the noisy function observations $\boldsymbol{z}$ and prediction target z_p, the elements of $\boldsymbol{k}_{tp}$ vector are constructed using the covariance function given earlier by equation (3.8). To make a prediction we calculate the *posterior* probability given by

$$p(z_p|\boldsymbol{z}, \boldsymbol{R}, \boldsymbol{r}_p) = \frac{p(\boldsymbol{z}, z_p)}{p(\boldsymbol{z}|\boldsymbol{R})} \tag{3.14}$$

The mean and the variance of the posterior probability distribution (a Gaussian distribution) are given as (Williams and Rasmussen, 1996).

$$\begin{aligned} m_p &= \boldsymbol{k}_{tp}^T \boldsymbol{K}_{tt}^{-1} \boldsymbol{z} \\ \sigma_p^2 &= k_{pp} - \boldsymbol{k}_{tp}^T \boldsymbol{K}_{tt}^{-1} \boldsymbol{k}_{tp} + \sigma_n^2 \end{aligned} \tag{3.15}$$

It is clear from equation (3.15) that the prediction mean and variance of a GP regression model is a function of the covariance matrix inverse. The computational cost of evaluating equation (3.15) is proportional to the size of the training data set N_t. While the covariance matrix inverse can be computed beforehand and stored aside before making any predictions; for very large training data sets this might be infeasible.

3.3.2 GPM Covariance Function

The GPM covariance function (equation (3.8)) has a central role in capturing the covariance between different input space samples. The choice of the covariance function affects the shape of the functions drawn under the Gaussian probability prior therefore care must be taken when specifying a certain covariance function to avoid model misspecification.

Not any function can be used as a GP covariance function because the function has to be positive semi-definite (Rasmussen and Williams, 2006). A Covariance function can be stationary (invariant to translations in input space) or non-stationary. Different covariance functions can be combined together to produce one big covariance function that gives higher likelihood the drawn functions from the Gaussian probability prior match the true underlying function described by the data multiplication of two covariance functions resembles a logical AND operator and addition of them resembles a logical OR operator.

The squared-exponential covariance function of equation (3.8) is an example of a stationary covariance function. This covariance function can provide what is called *automatic relevance determination* (ARD) known in ANN with Bayesian treatment (Neal, 1996). The covariance function hyper-parameters play the role of characteristic length scale which indicates the relevance of a certain input component. If the optimized hyper-parameter has a relatively high value for a certain input component then it can be eliminated from future training of the GPM.

The squared exponential function can be called *isotropic* if it depends only on $|\boldsymbol{r}_i - \boldsymbol{r}_j|$ with a single length scale hyper-parameter for all input dimensions**.** The squared exponential covariance function is a smooth function (infinitely differentiable) and considered the most widely used in regression problems (Rasmussen and Williams, 2006).

A covariance function is called *dot-product* if it depends on $(\boldsymbol{r}_i.\boldsymbol{r}_j)$**,** a dot-product covariance function is invariant to rotation around origin but not to translations in inputs space thus it is considered to be a non-stationary covariance function.

Another important covariance function is the *Matèrn-class* given below for the case of one-dimensional input vector $(\boldsymbol{r})$ (Rasmussen and Williams, 2006).

$$c(r_i, r_j) = \frac{2^{1-v}}{\Gamma(v)} \left(\frac{\sqrt{2v(r_i - r_j)}}{l} \right)^{v} K_v \left(\frac{\sqrt{2v(r_i - r_j)}}{l} \right) \tag{3.16}$$

With , $l > 0$, K_v is modified Bessel function.

The Matèrn-class covariance function is often employed where discontinuity is present in the function modelled by the GPM which can be a true discontinuity or due to poor description of the underlying function by the training data (due to noise or limited number of training data samples).

Another important covariance function is the neural network covariance function (Rasmussen and Williams, 2006) given by

$$c(\boldsymbol{r}_i, \boldsymbol{r}_j) = \theta_0 \frac{2}{\pi} sin^{-1} \left(\frac{2\tilde{\boldsymbol{r}}_i^{\,T} \boldsymbol{\Lambda} \tilde{\boldsymbol{r}}_j}{\sqrt{\left(1 + 2\tilde{\boldsymbol{r}}_i^{\,T} \boldsymbol{\Lambda} \tilde{\boldsymbol{r}}_i\right)\left(1 + 2\tilde{\boldsymbol{r}}_j^{\,T} \boldsymbol{\Lambda} \tilde{\boldsymbol{r}}_j\right)}} \right) \tag{3.17}$$

Where $\tilde{\boldsymbol{r}} = [1, r_1, \dots, r_{nt}]^T$ is an augmented input vector and $\boldsymbol{\Lambda} = \boldsymbol{I}\boldsymbol{l}$ with $\boldsymbol{l} = [l_1, \dots, l_{nt}]^T$ is the covariance function vector of length scales**.**

The neural network covariance function is a non-stationary covariance function; it allows the saturation of the process parameter θ_0 in both negative and positive directions therefore allowing it to generate rapidly changing functions under the GP prior. This is important when modelling functions that are rapidly changing or discontinues.

There are other types of covariance functions out of the scope of the presented research and for more in depth review the reader is referred to (Rasmussen and Williams, 2006) and the references therein.

3.3.3 Properties of the GP Regression Model

GP regression models in comparison with parametric regression models can provide function predictions with a confidence measure that can be used to improve the selection of the training data sets to further improve the accuracy of those predictions, this is accomplished by isolating high uncertainty predictions and examining the amount of GPM training data available in the vicinity of those predictions, increasing the number of the training data at these locations may improve the accuracy of the predication (reduce the uncertainty) provided that the covariance function model is specified correctly.

The training of a GP model provides a way to determine the relevance between an input component and its prediction, this is done through the training of the hyper parameter θ_d in the dth dimension of the individual input vector $\boldsymbol{r}_i$. This knowledge helps to choose input components that will be used to make the predictions later. This may result in regression dimensionality reduction of the input component which leads to computational savings when making the predictions.

The smoothing property of the GP regression helps the model to perform relatively better than a parametric regression models when it deals with sparse function observations.

On the other hand, Regression with GP models involves the inversion of the $Nt \times Nt$ covariance matrix. This inversion is a computationally intensive process that increases with the increase in the size of the training data set.

All the practical results for the GP regression models are obtained using the open source Gaussian Processes for Machine Learning (GPML) toolbox developed in (Rasmussen and Nickisch, 2010) for MATLAB.

3.3.4 Meta-Model Training

Recalling equation (3.2) which defines the parameters of the VBL-LPV meta-model, each non-constant element in every parameter is described by a GPM function of the current operating point of the CI-NDS of which the meta-model is trained for.

The first step in meta-model training is the collection of training data. The meta-model has to be trained ideally in equilibrium and off-equilibrium points of the required trajectory starting from some initial conditions $(\boldsymbol{u}_0, \boldsymbol{x}_0, \boldsymbol{w}_0)$. The number of training points should be enough to describe the underlying parameter function along this trajectory (the minimum number of training points to achieve high accuracy prediction and reduce the computational cost).

The equilibrium training $(\boldsymbol{u}_e, \boldsymbol{x}_e, \boldsymbol{w}_e)$ points can be analytically evaluated in most cases, or by allowing the NDS velocity to approach zero and measuring the corresponding equilibrium states and inputs.

The off-equilibrium training points are a collection of points selected along the trajectory starting at some initial condition leading to some final-states and velocities denoted by $(\boldsymbol{u}_f, \boldsymbol{x}_f, \boldsymbol{w}_f)$.

One approach to secure a finite number of training data in the case of CI-NDS modelled by a piecewise-linear model was proposed in (Vasilyev et al., 2003). The first training data point is set to the initial condition of the system; then the CI-NDS following a certain input $\boldsymbol{u}(t)$ is simulated while a suitable distance measure between the current simulation state and its initial state is less than some predefined maximum threshold. If this maximum threshold is passed the simulator state passing that threshold is registered as a training point along with its corresponding input at that time. This procedure is given by Algorithm 3-1

```
1  input: $\boldsymbol{x}_0$ (initial state), $\boldsymbol{u}_m$ (trajectory input), $\delta$: $(0 < \delta < 1)$, $it_{max} \in \mathbb{N}^+$
2  compute $\boldsymbol{x}_e$ at $\boldsymbol{u}_m$ from NDS first-principle model.
3  if $||\boldsymbol{x}_0|| = 0$, set $dist_{max} := \delta||\boldsymbol{x}_e||$, else $dist_{max} := \delta\frac{||\boldsymbol{x}_e-\boldsymbol{x}_0||}{||\boldsymbol{x}_0||}$
4  $i := 1$, set $\boldsymbol{R}(:,i) := [\boldsymbol{u}_m; \boldsymbol{x}_0]$, $\boldsymbol{X}(:,i) \coloneqq \boldsymbol{x}_0$
5  while $i < it_{max}$
6  while $\frac{||x-\boldsymbol{X}(:,i)||}{||\boldsymbol{X}(:,i)||} < dist_{max}$ simulate NDS with $\boldsymbol{u}_m$
7  $i = i + 1$, set $\boldsymbol{R}(:,i) = [\boldsymbol{u}_m; \boldsymbol{x}]$, $\boldsymbol{X}(:,i) = \boldsymbol{x}$
8  end while loop, end while loop
9  return: $\boldsymbol{R} = [\boldsymbol{R}, [\boldsymbol{u}_m; \boldsymbol{x}_e]]$ (Set of meta-model training data points)
```

Algorithm 3-1: Meta-model training data

The maximum threshold the simulator state should pass to identify an off-equilibrium training point is given by the normalized distance between the final equilibrium state $\boldsymbol{x}_e$ and the initial state $\boldsymbol{x}_0$ multiplied by a positive constant δ. Algorithm 3-1is suitable for one-dimensional training inputs transitioning to a single operating point from some initial state i.e. single trajectories.

Other authors have proposed to excite the system with large randomly varying magnitude step inputs to try and capture the system at various off-equilibrium operating points. This approach will result in a larger number of operating points covering wider area of the NDS state-space. Algorithm 3-2 was used to generate a random sequence of pulses with varying magnitudes and widths to excite the NDS in off-equilibrium regions.

```
1  **input:** $u_{min}$(minimum input), $u_{max}$ (maximum input), $N_{ticks} \geq 2, N_{ticks} \in \mathbb{N}^{+}$(number of identification pulses), $Ts$ (sampling time) and $stm \geq 1, stm \in \mathbb{R}^{+}$(sampling time multiplier)
2  **initialize** pulse amplitude vector $\boldsymbol{pa} \in \mathbb{R}^{1\times N_{ticks}}$
3  **initialize** pulse width vector $\boldsymbol{pw} \in \mathbb{R}^{1\times N_{ticks}}$
4  **initialize** pulse position index vector $\boldsymbol{pidx} = [1 \quad \ldots \quad N_{ticks}], \boldsymbol{pidx} \in \mathbb{R}^{1\times N_{ticks}}$
5  **initialize** individual pulse sequence vector $\boldsymbol{pseq}$
6  **initialize** final off- equilibrium input pulse sequence vector $\boldsymbol{u}_{off}$
7  $i := 1, \boldsymbol{pa}(i) = u_{min},\ \boldsymbol{pw}(i) = stm \times Ts, i = i + 1$
8  **while** $i < N_{ticks}$
9  $\boldsymbol{pa}(i) = \boldsymbol{pa}(i-1) + \frac{u_{max} - u_{min}}{N_{ticks}}$
10 $\boldsymbol{pw}(i) = i \times stm \times Ts$
11 $i = i + 1$
12 **end while** loop
13 **compute** random permutation of $\boldsymbol{pidx}$ entries
14 $i = 1,$
15 **while** $i < N_{ticks}$
16 $j := 1$
17 **while** $j < \boldsymbol{pw}(\boldsymbol{pidx}(i))$
18 $\boldsymbol{pseq}(j) = \boldsymbol{pa}(\boldsymbol{pidx}(i))$
19 $j = j + 1$
20 **end while** loop
21 $\boldsymbol{u}_{off} = [\boldsymbol{u}_{off}, \boldsymbol{pseq}]$
22 $i = i + 1$
23 **end while** loop
24 **compute** time-series object with $\boldsymbol{u}_{off}$ uniformly re-sampled at $Ts$
25 **return:** $\boldsymbol{u}_{off}$ (off-equilibrium input step sequence)
```

Algorithm 3-2: Meta-model off-equilibrium training input using randomly generated pulse sequence

The inputs to Algorithm 3-2 are the minimum and maximum pulse sequence magnitudes (u_{min}, u_{max}), the number of individual levels in the sequence (N_{ticks}), the default sampling time (Ts) of the NDS and a sampling time multiplier (stm) to specify base pulse widths larger than Ts. stm sets the length of the generated pulse sequence, relatively large stm value (compared to ($stm \times Ts$)) will generate longer minimum pulse width therefore allowing to the NDS to approach equilibrium state at the end of the an individual pulse (depends on the NDS characteristics).

Algorithm 3-2 will return $\boldsymbol{u}_{off}$ an identification input pulse sequence which is applied to the NDS to compute the meta-model off-equilibrium training data (states).

Due to the computational complexity associated with CI-NDSs, a short length excitation signal is desirable to reduce the time needed to collect the meta-model training and validation data. A small training data set for the meta-model is important to reduce the computational overhead associated with making prediction using the GP regression model (This will be discussed further in Chapter 6).

The Two-Tanks NDS (Figure 3-1) and appendix A.2 is a simple second order NDS that can be found part of MATLAB system identification tool box. The model being of 2nd order, its phase response can be visualized and will be used as an example of how the meta-model is created based on the proposed GP blended VBL-LPV meta-modelling structure.

The Two-Tanks NDS has one input described by the voltage applied to a water pump $u(t)$ that creates an inflow to the upper tank which has a small hole in the bottom that creates an outflow to the lower tank, the lower tank has a small hole in the bottom that generate an outflow. The water levels (meters) in both tanks at any time are the two states of the system $x_1(t) and\ x_2(t)$ and the model output is specified by $y(t) = x_2(t)$. The minimum pump voltage was inferred from an example of the system in the MATALB documentation to be greater than one, this will allow the incoming water flow speed to be greater than system water outflow speed to prevent a singularity in the solution.

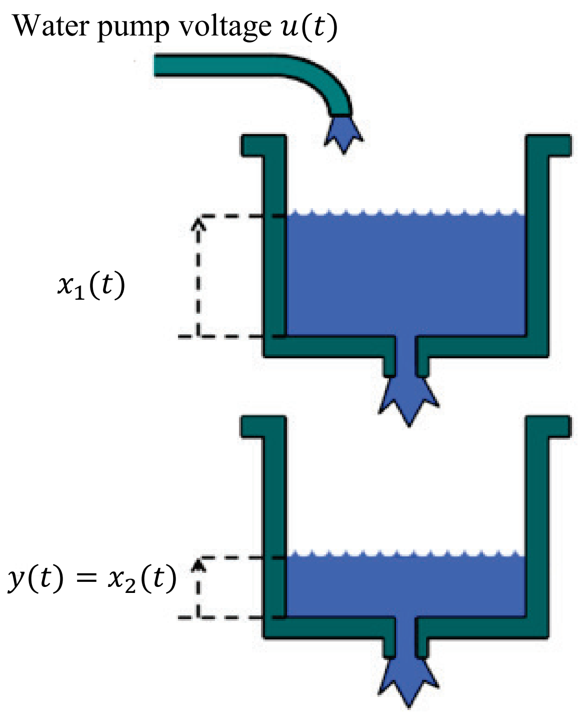


Figure 3-1: Two tanks NDS (MATLAB system identification tool box).

The upper range of the pump voltage was set arbitrarily at 10v. The initial water level in both tanks at the minimum pump voltage must be greater than zero to prevent singularity in the model solution, for this experiment the initial conditions was set at $x_0 = (0.05, 0.1)$. The set of the above initial conditions will result in a well behaved and predictable response across the forcing input range. The default sampling time of the system is $0.2s$.

The equilibrium points of the system were analytically evaluated uniformly at 100 points across the pump voltage range ($1 \geq u(t) \geq 10$), this resulted in 100 equilibrium data points. The off-equilibrium training data are taken of the system response to a randomly generated sequence of varying magnitude steps (Algorithm 3-2). This was accomplished by randomly sampling the pump voltage range $1 \geq u(t) \geq 10$ at 100 points and setting $stm = 2$. The generated pulse sequence was uniformly sampled every 0.2s resulted in 588 data points that includes off-equilibrium points of the system, the identification pulse sequence is given by Figure 3-2.

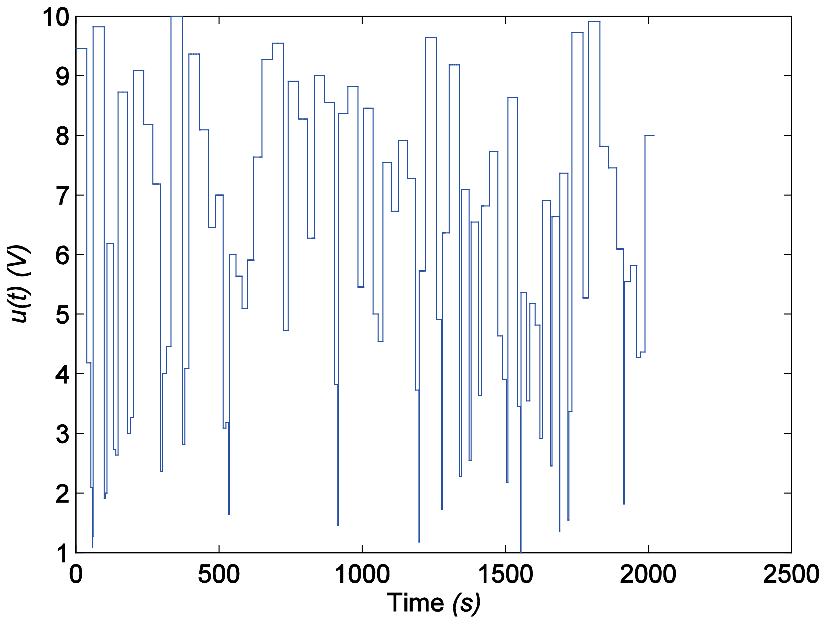


Figure 3-2: Two-Tanks NDS pump voltage sequence used to excite the model at off-equilibrium dynamics

After the collection of the meta-model training points (a total of 688), the GP models of the model's time-varying parameters must be trained; in the case of the Two-Tanks NDS, only three entries in parameter $\boldsymbol{A}(t)$ needs to be trained with GPM, parameters $\boldsymbol{B}(t), \boldsymbol{C}(t)$ and $\boldsymbol{D}(t)$ are constants (appendix A.2). The meta-model training points function targets are computed using the true values of the parameters calculated from the Jacobians of the NDS first-principle model given by equations (2.7) and (2.11) at each training point.

A covariance function has to be selected for each GP model of the meta-model time-varying parameters, initial guesses has to be assigned as to what the hyper-parameters values are before training of the GP model.

In the case of the Two-Tanks NDS, a squared-exponential with ARD has been chosen and the GP model's means were set to zero. This choice was repeated during the training of the GPM for all three time-varying entries in the $\boldsymbol{A}(t)$ parameter. During the training of each GPM, cross-validation (Rasmussen and Williams, 2006) was used to choose a best GPM model that agrees with validation data. This was implemented by splitting the collected training points into two disjoint sets one for training and the other for validation using uniform sampling with odd-even indices, this resulted in 344 points used for the training of the GPMs and the rest are used for validation. Although the training data are noise free, very small (1×10^{-5}) zero-mean Gaussian noise was added to all training points' targets to stabilize GPM calculations.

The GPM covariance function hyper-parameters are trained by minimizing the cost function $\boldsymbol{J}(\boldsymbol{\theta})$ in equation (3.10) using the conjugate gradient optimization method (part of the GPML toolbox). The chosen covariance function for the Two-Tanks NDS time-varying entries of parameter $\boldsymbol{A}$(t) has a total of four parameters including the GPM noise parameter.

A measure for the trained GPM fitness can be computed using the *Mean Square Error* (MSE) which is the average squared error between the GPM predictions and true targets but is sensitive to the magnitude of the data so *Standardized Mean Squared Error* (SMSE) (Rasmussen and Williams, 2006) (which is the MSE divided by the variance of the true targets variance) was computed instead. MSE and SMSE metrics are only useful when having results from different experiments (for the same parameter model) such as when performing cross-validation, examining the lowest achievable SMSE value indicates a good agreement between the trained GPM and the validation data set with minimum uncertainty. Another way to test how the validation data agrees with the trained GPM in any single experiment is given by

$$\%fitness = 100\left(1 - \frac{||\boldsymbol{f}_{true} - \boldsymbol{f}_{prediction}||}{||\boldsymbol{f}_{true} - mean(\boldsymbol{f}_{true})||}\right) \qquad (3.18)$$

Equation (3.18) fitness measure will also be used to evaluate the meta-model solution as well.

The GPMs training result for the Two-Tanks NDS meta-model parameters are given in Table 3-1.

Table 3-1: Two-Tanks NDS GPM training results for the time-varying entries of meta-model parameter $\boldsymbol{A}(\boldsymbol{t})$

	$GPM(a_{11})$			$GPM(a_{21})$		$GPM(a_{22})$	
	$\boldsymbol{\theta}_{intial}$	$\boldsymbol{\theta}_{trained}$	$\%fitness$	$\boldsymbol{\theta}_{trained}$	$\%fitness$	$\boldsymbol{\theta}_{trained}$	$\%fitness$
θ_1	1	1.28×10^{-2}	97.28	1.88×10^{-2}	97.79	1.86×10^{3}	97.52
θ_2	1	8.62×10^{2}	$SMSE$	3.23×10^{5}	$SMSE$	3.27×10^{-2}	$SMSE$
θ_3	1	1.86		3.44×10^{1}		1.03×10^{1}	
θ_4	1×10^{-5}	2.3×10^{-3}	7.37×10^{-4}	2.6×10^{-3}	4.89×10^{-4}	2.39×10^{-3}	6.12×10^{-4}

Table 3-1 shows that across all the three meta-model parameters, the GPMs successfully captured the underlying function with model finesses greater than 97%. Among things to observe in the trained hyper-parameters, the chosen squared-exponential covariance function with ARD assigned the first two hyper-parameters $\{\theta_1, \theta_2\}$ to each one of the possible regression inputs and in the case of the Two-Tanks NDS they are $\{x_1, x_2\}$. The covariance function ARD feature automatically revealed the relevant input to model the underlying parameter function by assigning a relatively small value to the relevant input. This can be verified by observing the analytical Two-Tanks NDS Jacobians (appendix A.2).

The covariance function with ARD feature is useful to detect the relevant regression inputs when the meta-model order has been reduced (chapter four).

The variance of the validation data provided by the GPM validation stage can be used to calculate the percentage of the *Relative Standard Deviation* ($\%RSD$) given below to provide an insight of where the GP regression model was uncertain when it made the prediction.

$$\%RSD = 100\left|\frac{\sigma_p}{m_p}\right| \tag{3.19}$$

σ_p is the standard deviation of the prediction.

m_p is the mean of the prediction.

This fitness measure cannot be used with data where the function targets approaches zero value. A plot of the GPMs validation data $\%RSD$ for the Two-Tanks NDS parameter $\boldsymbol{A}(t)$ is given by Figure 3-3.

The plot in Figure 3-3 demonstrates that most of the GPM validation points achieved $\%RSD$ below 5% indicating high confidence predictions. This is a good indication that the meta-model will be able to provide a low variance solution with unseen test inputs as long as these inputs stayed close to the training data space. There are few outliers above the 5% , this is normal because an ideal fit is not expected between the trained meta-model parameters and the validation data. However, the effect of outliers on the final meta-model solution can be dramatic since this a model for nonlinear dynamical system and an anomaly at any point of time could affect the rest of the solution.

The $\%RSD$ plot is an important tool in the meta-model validation phase because it is easy to plot (two-dimensional) and a single glance at this plot reveals the GPM confidence in predictions.

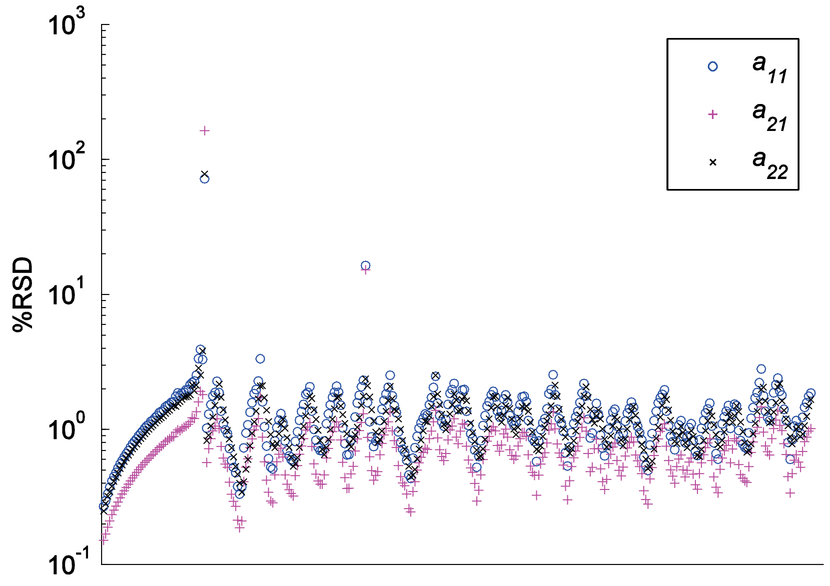


Figure 3-3: $\%RSD$ for the time-varying entries of the $\boldsymbol{A}(t)$ parameter for the Two-Tanks meta-model validation data set

This concludes the training procedure of the GP blended VBL-LPV meta-model.

3.4 The Meta-Model Solver

The thesis has explored a set of numerical methods to solve the meta-model equations (3.1) and (3.2). These methods deploy a discretization approach to the set of differential equations to be solved given by a *Zero-Order Hold* (ZOH) on the meta-model inputs followed by fixed-time step sampling at $t = kh, \{k = 0,1,2 \dots\}$ where h is the fixed-time step. The sampled version of the meta-model equations is given by

$$\begin{aligned}
\dot{\boldsymbol{w}}(kh) &= \boldsymbol{A}(kh)\boldsymbol{w}(kh) + \boldsymbol{B}(kh)\dot{\boldsymbol{u}}(kh) \\
\boldsymbol{w}(kh) &= \dot{\boldsymbol{x}}(kh) \\
\boldsymbol{y}(kh) &= \boldsymbol{C}(kh)\boldsymbol{x}(kh) + \boldsymbol{D}(kh)\boldsymbol{u}(kh) \\
\boldsymbol{w}(0) &= \boldsymbol{w}_0, \boldsymbol{x}(0) = \boldsymbol{x}_0
\end{aligned} \tag{3.20}$$

And the sampled meta-model parameters are given by

$$\begin{aligned}
\boldsymbol{A}(kh) &= \boldsymbol{GPM_A}(\boldsymbol{x}(kh)) \\
\boldsymbol{B}(kh) &= \boldsymbol{GPM_B}(\boldsymbol{x}(kh)) \\
\boldsymbol{C}(kh) &= \boldsymbol{GPM_C}(\boldsymbol{x}(kh)) \\
\boldsymbol{D}(kh) &= \boldsymbol{GPM_D}(\boldsymbol{x}(kh))
\end{aligned} \tag{3.21}$$

At each discrete time step h taken by the solver, two solution estimates are needed; a solution for the velocity part, then the state part later. The proposed meta-model discrete-time step solver is making an assumption that because of the ZOH, the LLM parameters are constant during the fixed-time stepping of the solver which may not be valid for a CI-NDS with very high speed dynamics.

3.4.1 The Velocity Solution

The velocity solution of the VBL-LPV system at any time step is given by

$$\begin{aligned}
\boldsymbol{w}((k+1)h) = {} & e^{\boldsymbol{A}(k)(k+1)h}\boldsymbol{w}(0) \\
& + \int_0^{(k+1)h} e^{\boldsymbol{A}(k)(k+1)h-\tau}\boldsymbol{B}(k)\dot{\boldsymbol{u}}(\tau)d\tau
\end{aligned} \tag{3.22}$$

Equation (3.22) can be decomposed to

$$\boldsymbol{w}\big((k+1)h\big) = e^{\boldsymbol{A}(k)h}\left(e^{\boldsymbol{A}(k)kh}\boldsymbol{w}(0) + \int_{0}^{kh} e^{\boldsymbol{A}(k)(kh-\tau)}\boldsymbol{B}(k)\dot{\boldsymbol{u}}(\tau)d\tau\right) + \int_{kh}^{(k+1)h} e^{\boldsymbol{A}(k)(k+1)h-\tau}\boldsymbol{B}(k)\dot{\boldsymbol{u}}(\tau)d\tau \tag{3.23}$$

Setting the term between large brackets to $\boldsymbol{w}(k)$ and introducing the term $\beta = (k+1)h - \tau$ into equation (3.23) to get

$$\boldsymbol{w}(k+1) = e^{\boldsymbol{A}(k)h}\boldsymbol{w}(k) + \int_{0}^{h} e^{\boldsymbol{A}(k)\beta}d\beta\boldsymbol{B}(k)\dot{\boldsymbol{u}}(k) \tag{3.24}$$

Further rearrangement of equation (3.24) yield

$$\begin{aligned} \boldsymbol{w}(k+1) &= \boldsymbol{A}_d(k)\boldsymbol{w}(k) + \boldsymbol{B}_d(k)\dot{\boldsymbol{u}}(k) \\ \boldsymbol{A}_d(k) &= e^{\boldsymbol{A}(k)h} \\ \boldsymbol{B}_d(k) &= \int_{0}^{h} e^{\boldsymbol{A}(k)\beta}d\beta \end{aligned} \tag{3.25}$$

The meta-model velocity solution in equation (3.25) is called *Exact-Discretization* method (ED). The discrete parameter $\boldsymbol{B}_d(k)$ of the forcing part in the ED velocity solution can be approximated using the equality $\boldsymbol{A}(k)\int_0^h e^{\boldsymbol{A}(k)\beta}d\beta = e^{\boldsymbol{A}(k)\beta} - \boldsymbol{I}$ as given by

$$\boldsymbol{B}_d(k) = \boldsymbol{A}(k)^{-1}(\boldsymbol{A}_d(k) - \boldsymbol{I})\boldsymbol{B}(k) \tag{3.26}$$

Provided that $\boldsymbol{A}(k)$ is not singular**.**

The ED velocity solution method will provide a solution that exactly matches the continues-time solution at the discrete time samples. The rest of the proposed velocity solutions in the following paragraphs can be viewed as numerical approximation to the ED velocity solution.

The velocity solution at any discrete time k can be approximated using *Forward-Euler* method by

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + h\big(\boldsymbol{A}(k)\boldsymbol{w}(k) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k)\big) \tag{3.27}$$

The Forward-Euler method require small time stepping in general for any reasonable accuracy, as the time step h approaches smaller values; rounding errors start accumulating throughout the solution. The Forward-Euler method can produce unstable solutions for larger time steps especially or when meta-modelling stiff NDS. The local truncation error of the Forward-Euler method is $\mathcal{O}(h^2)$ while the total accumulated error is $\mathcal{O}(h)$.

Heun's method (Ascher and Petzold, 1998) can be applied to solve for the velocity, this method is given by

$$\begin{aligned}
\acute{\boldsymbol{w}}(k+1) &= \boldsymbol{w}(k) + h\big(\boldsymbol{A}(k)\boldsymbol{w}(k) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k)\big) \\
\boldsymbol{w}(k+1) &= \boldsymbol{w}(k) \\
&\quad + \frac{h}{2}\Big(\big(\boldsymbol{A}(k)\boldsymbol{w}(k) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k)\big) \\
&\quad + \big(\boldsymbol{A}(k)\acute{\boldsymbol{w}}(k+1) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k+1)\big)\Big)
\end{aligned} \tag{3.28}$$

Heun's method can be viewed as a correction to the Forward-Euler method solution estimate by averaging the velocity estimate at both ends of the fixed time step. Heun's method is a *predictor-corrector* method with Forward-Euler method as the predictor and the *Trapezoidal* method of integration as the corrector. Therefore; Heun's method is more accurate than the Forward-Euler method but still have the same drawbacks of the Forward-Euler method concerning the stability of the obtained solutions. The local truncation error of the Heun's method is $\mathcal{O}(h^2)$ similar to the Forward-Euler method while the total accumulated error is $\mathcal{O}(h^2)$ an improvement on the Forward-Euler method.

To improve on the predictions accuracy of the previous methods, an explicit *classical Runge-Kutta* method of fourth order (RK4) is proposed to solve the velocity equation, this method is given below. The RK4 method is a fourth order method which means the local truncation error is $\mathcal{O}(h^5)$ and the total accumulated error is $\mathcal{O}(h^4)$. Therefore; the RK4 method should provide more accurate solutions to the meta-model velocity equation but will require extra computations. RK4 method fairs better than Forward-Euler method especially with large time steps.

$$\begin{aligned}
d\boldsymbol{w}_1 &= \boldsymbol{A}(k)\boldsymbol{w}(k) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k) \\
d\boldsymbol{w}_2 &= \boldsymbol{A}(k)\left(\boldsymbol{w}(k) + \frac{h}{2}d\boldsymbol{w}_1\right) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k) \\
d\boldsymbol{w}_3 &= \boldsymbol{A}(k)\left(\boldsymbol{w}(k) + \frac{h}{2}d\boldsymbol{w}_2\right) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k) \\
d\boldsymbol{w}_4 &= \boldsymbol{A}(k)(\boldsymbol{w}(k) + hd\boldsymbol{w}_3) + \boldsymbol{B}(k)\dot{\boldsymbol{u}}(k+1) \\
\boldsymbol{w}(k+1) &= \boldsymbol{w}(k) + \frac{h}{6}(d\boldsymbol{w}_1 + 2d\boldsymbol{w}_2 + 2d\boldsymbol{w}_3 + d\boldsymbol{w}_4)
\end{aligned} \tag{3.29}$$

3.4.2 The State Equation Solver

There is not much to solve the state part of the meta-model equation (3.20) other than using a Forward-Euler method to estimate the state solution from the velocity solution. The state solution is given by

$$\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + h\boldsymbol{w}(k+1) \tag{3.30}$$

3.4.3 The Meta-Model Output

The meta-model output only requires the discrete state solution of equation (3.30) to be substituted in the output part of equation (3.20).

3.4.4 Solving the Two-Tanks NDS Meta-Model

To test the Two-Tanks meta-model, a number of test inputs have been applied. The original NDS model response was simulated for each test input using fixed-time step $(h = 0.2s)$ classical RK4 method to provide a reference response.

The first test input is a step function given by

$$u(t) = 5, t < 1.8$$
$$u(t) = 10, t \geq 1.8 \quad (3.31)$$

Since this input is discontinues and the meta-model structure requires a continues-time input with a valid first-order derivative, it was approximated using a sigmoid like function as described in Appendix B.

Exact-Discretization velocity solver was used to obtain the meta-model solution to this input. The response of the meta-model is shown in Figure 3-4.

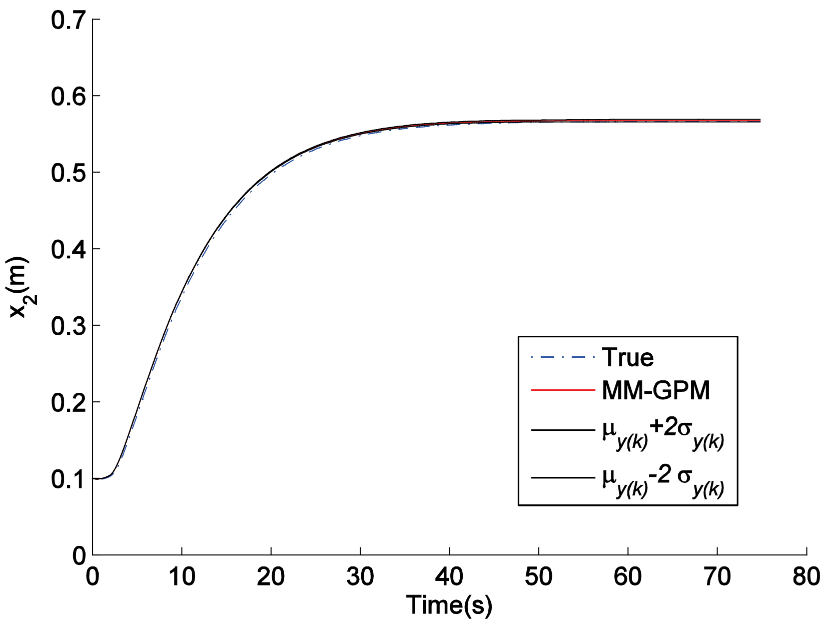


Figure 3-4: Meta-model solution of the Two-Tanks NDS to a step function

Figure 3-4 shows that the meta-model successfully predicted the output with model fitness of 97.14%. The plot contains meta-model confidence regions around the solution and it can be seen that they are virtually indistinguishable from the meta-model response due to the small time step of the solver (more on this will be discussed in chapter 5). This excellent meta-model response to the step test input was expected considering the meta-model training data was from a randomly generated sequence of step inputs.

The next test input is a ramp function with slop of one between 1 and 8.5 sampled every $0.2s$, the meta-model response is shown in Figure 3-5.

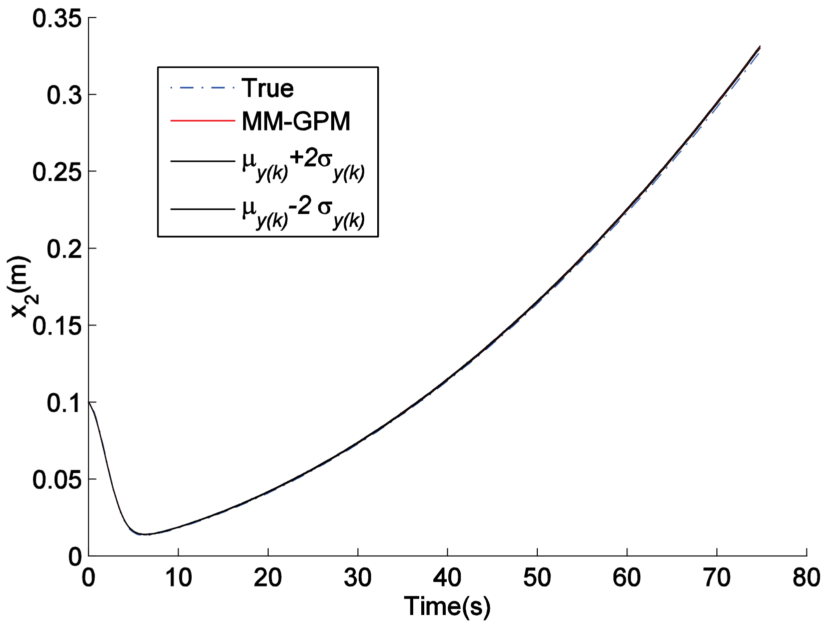


Figure 3-5: Meta-model solution of the Two-Tanks NDS to a ramp test input

The meta-model was successful ($\%fitness = 98.4\%$) to predict the above input.

The next test input is an exponential decay given by

$$u(t) = 4 + e^{-0.1t}, (0 \leq t \leq 75) \qquad (3.32)$$

This input was sampled every $0.2s$, the meta-model response to this input is shown in Figure 3-6.

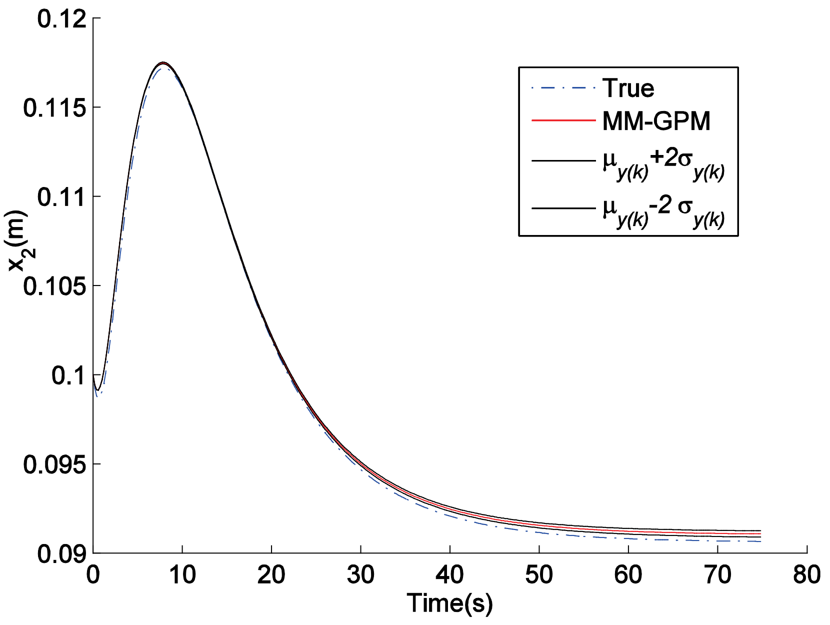


Figure 3-6: Meta-model solution of the Two-Tanks NDS to exponential decay test input

The meta-model response to the exponential decay test input was similar ($\%fitness = 95.38\%$) to the true model response.

The final test input was taken from the Two-Tanks NDS example in the MATLAB system identification tool box. It's a series of step signals controlling the Two-Tanks pump voltage. The original sequence (being discontinues) has been approximated using the method described in Appendix B. The meta-model solver time is $0.2s$ and the resultant test input is shown in Figure 3-7.

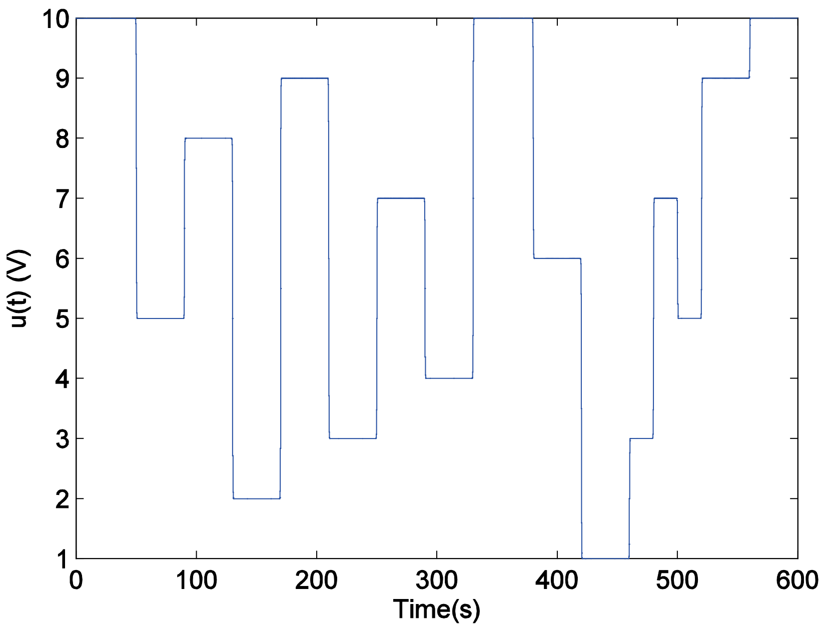


Figure 3-7: Two-Tanks NDS step seqiunce test input

The response of the meta-model to the above test input is given by Figure 3-8.

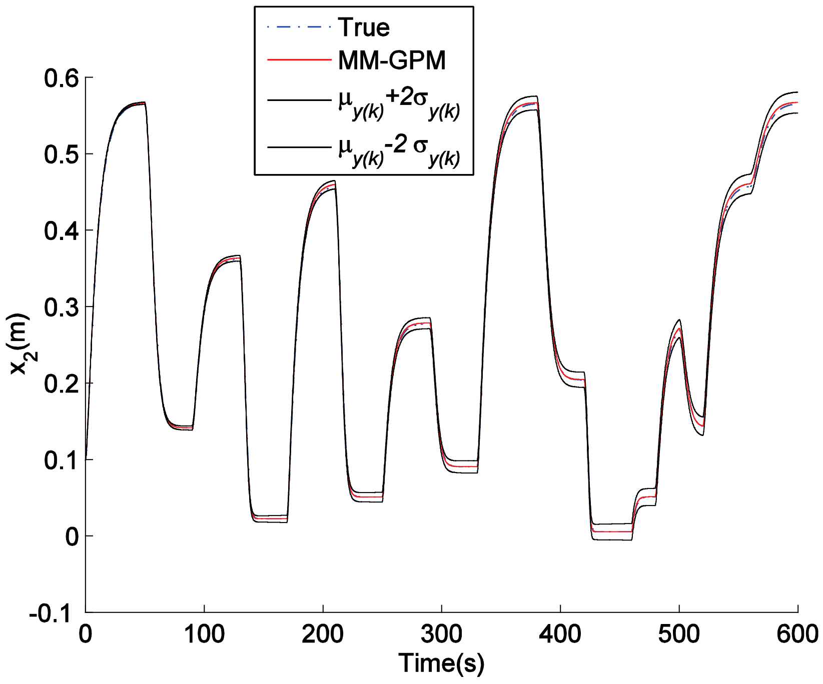


Figure 3-8: Two-Tanks NDS meta-model solution of the to the step sequence test input

Figure 3-8 shows that the meta-model successfully predicted the response of the step sequence and achieved $\%fitness = 97.54\%$.

$\%RSD$ can be computed for the time-varying elements of the $\boldsymbol{A}(t)$ state matrix during the meta-model solution in all the above examples at each step of the solution, this will result in a time-varying confidence measure for the meta-model parameters solution during the simulation and can be useful in identifying problematic regions where the GPM of the parameter is uncertain about the prediction. The time-varying $\%RSD$ plot for the last test input has been produced below as a proof of the concept.

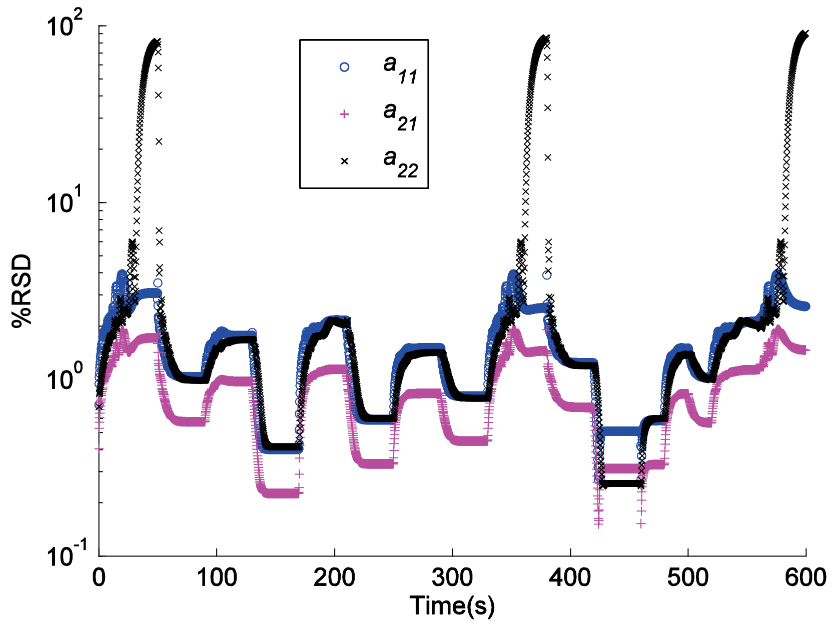


Figure 3-9: Plot of $\%RSD$ for the non-constant entries of the $\boldsymbol{A}(t)$ parameter for the Two-Tanks meta-model during simulation with the step sequence input

Figure 3-9 shows that $\%RSD$ for all three paramters kept below 5% most of the time (high confidence) with the exception of a_{22} parameter peaking at 83%, this indicates that at these operating points, the GPM of the parameter was not confident. Before jumping to conclusions, the problematic parameter magnitude must be tested for approaching zero value. This was not the problem in the case of the a_{22} parameter, therefore the GPM of the parameter struggled to produce confident predictions for some parts of the solution. Since the Two-Tanks NDS is a simple second-order system, the phase plot of the model states can be easily visualized. A composite phase plot containing the states at which the a_{22} GPM was trained, and the meta-model states solution to the step sequence test input (Figure 3-7) highlighting in red the states at which the parameter a_{22} $\%RSD$ was over 5% is shown in Figure 3-10.

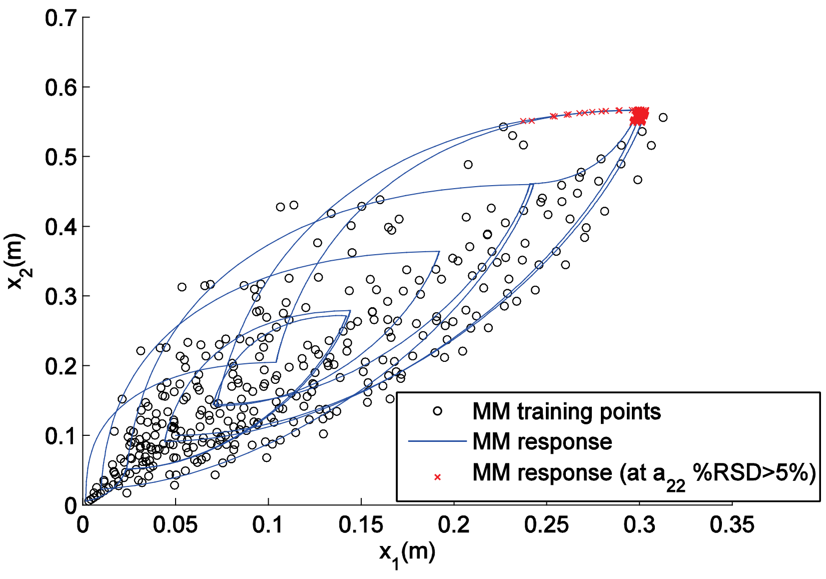


Figure 3-10: Composite phase plot for the Two-Tanks NDS training points and the meta-model solution to the step sequence test input

It can be seen why the a_{22} time-varying $\%RSD$ has peaked during the meta-model solution by simply looking at the red marks in Figure 3-10 and observing the sparsity of meta-model training points around them. Despite this high uncertainty in one of the meta-model parameters, it produced a good approximation to the solution of this test input probably because of the smoothness of the true parameter function.

To present a numerical comparison of the accuracy of the different meta-model solvers proposed earlier, the Two-Tanks NDS was simulated using a fixed-time step ($h = 0.2s$) classical RK4 method; then the meta-model was solved using different velocity solvers for the step-sequence test input (Figure 3-7). All computational work in this thesis was conducted using the computational environment given in Table 3-2.

Table 3-2: Computational environment

Processer	Intel 820QM i7 Quad-Core
Processer maximum speed	$3.06GHz$
RAM	$8GB$ DDR3
RAM speed	$667MHz$
Operating System	Microsoft Windows 7 Ultimate x64
MATLAB version	7.12.0.635 (R2011a) x64
Machine Epsilon	2.2204×10^{-016}
GPML version	3.4

The meta-modelling results are given in Table 3-3.

Table 3-3: Meta-Model fitness results for the Two-Tanks NDS simulated with step sequence input using different velocity estimation methods

Velocity estimation method	$\%fitness$
Forward-Euler	85.45
Heun's	96.95
RK4	97.46
Exact-Discretization	97.54

Table 3-3 shows that Exact- Discretization velocity estimation method performed the best (as predicted by section 3.4.1) and the rest three methods being a numerical approximation to the Exact-Discretization method performed lower with Forward-Euler being the worst considering the large time step this NDS was solved at ($h = 0.2s$). Repeating the Forward-Euler simulation using the time step $h = 0.1s$ improved the meta-model $\%fintess$ to 93.23%.

For the rest of meta-model simulations in this thesis, the default velocity estimation method will be Exact- Discretization unless otherwise stated.

To shed some light on the meta-model solver execution time relative to the true NDS model solver time, the *Execution Time Index* (ETI) is defined as

$$ETI = \frac{TSIM_{NDS} - TSIM_{MM}}{TSIM_{NDS}}$$

$TSIM_{NDS}$ is the NDS simulation time (s). (3.33)

$TSIM_{MM}$ is the meta-model simulation time (s).

Negative ETI value means the meta-model solver is taking longer time than the true NDS solver and positive values indicates computational time saving over the true NDS model solver. The implemented fixed-time step meta-model solver performs many more things in a single solution iteration compared to the RK4 fixed-time step solver, these include predicting meta-model parameters, their variances, predicting the meta-model solution, and performing uncertainty propagation (chapter 5). The meta-model computational complexity will be discussed in depth in chapter 6.

Table 3-4 shows the ETI values for the Two-Tanks NDS four test inputs previously presented in this section.

Table 3-4: ETI values for the Two-Tanks NDS meta-model for different test inputs

Test Inputs	Step-like	Ramp-like	Exponential decay	Step-sequence
ETI	-2.44	-2.04	-2.08	-2.62

ETI values between different runs of the same experiment have variability that depends on many factors such as code optimization, math libraries used, memory storage operations (size of data sets involved and available cache) and amount of computational resources available at the time of execution, therefore the values in Table 3-4 are indicative of meta-model computational speed rather being precise.

ETI values given in Table 3-4 clearly show that the Two-Tanks NDS meta-model execution time was more than twice the execution time of the RK4 fixed time solver using the true NDS model, this is expected because the meta-model order matches the true model order and the meta-model parameters estimation using GP regression model adds a computational overhead (this will be discussed in chapter six).

3.5 Conclusions

This chapter presented the analytical framework for the full-order GP blended VBL-LPV meta-model. GP regression models can deal with high-dimensional data problems; they do not suffer from model overfitting with a limited training data set and provide uncertainty information for their predictions.

Meta-model equilibrium training data can either be computed analytically using the first-principle structure of the CI-NDS or numerically by exciting the CI-NDS with a set of constant inputs until the model outputs reaches equilibrium (or the states derivative approaches zero). Off-equilibrium training data can be gathered by exciting the NDS with randomly generated pulse sequence with variable magnitude and duty cycles.

Training the meta-model time-varying parameters involves using cross-validation through splitting the set of collected training data into disjoints sets, one used for training of the parameters and the other used for validation.

A suitable GP model covariance function should be selected to reflect the properties of the meta-model time-varying parameter underlying function such as smoothness, discontinuities. $SMSE$ and $\%fitness$ metrics was used to compare across different sets of training data.

The GP blended VBL-LPV meta-model was applied to a 2nd order Two-Tanks NDS, the meta-model accurately managed to reproduce the NDS dynamics, although that the Two-Tanks NDS was not a CI model, the computational speed was drastically worse than the original model due to the following:

- The meta-model order is the same order as the tested NDS, the hypothesis is that the meta-model can improve the computational speed by reducing the order of the LLMs, this will be the subject of the next chapter.
- The GP model of the meta-model time-varying parameters adds an additional cost to the meta-model; this cost depends on the size of the meta-model training data set. This will be explained in depth in Chapter six.

The full-order GP blended full-order VBL-LPV meta-model is not yet suitable to improve the computational speed of a CI-NDS model, because they share the same model order, chapter four will introduce reduced-order GP blended VBL-LPV meta-model.

Chapter 4 - **Model Order Reduction**

4.1 Introduction

The GP blended VBL-LPV system has an order equal to that of the analytical CI-NDS, therefore it will not provide any computational speed saving over the analytical CI-NDS. Since the local velocity models are linear, reducing their order before blending by GPMs can decrease the meta-model computational cost.

This chapter is one of the main contribution chapters in the thesis, because linear MOR of VBL LLMs was never attempted before. The chapter will review and present an analytical description of some of the popular linear MOR methods and explain how they can be integrated within the proposed meta-modelling approach of chapter three. Finally, the reduced-order meta-model will tested using a simple NTL model of order 10. The results of this chapter will be used in meta-modelling of a medium CFD problem in chapter five and on the CI NTL model of order 5000 in chapter six.

4.2 Linear Model Order Reduction

The literature concerned with linear model order reduction is well established and a number of schemes are used to achieve reduced order linear models (Antoulas, 2005).

Linear model order methods can be divided into two main classes (Antoulas, 2005):

i) *Projection-based methods*.

ii) *Non-projection based methods*.

Non-projection based MOR methods is concerned with realizations of the liner models in frequency domain rather than time domain (Vasilyev, 2007), therefore the thesis will be focusing on the widely used projection-based methods because they can fit neatly into the developed structure (state space model) of the proposed GP blended VBL-LPV meta-model, Projection-based methods can be divided into three main sub classes:

- *Proper Orthogonal Decomposition* (POD) methods.
- *Krylov Subspace* methods.
- *Truncated Balanced Realization* (TBR) method.

Consider the linear time-invariant and casual system of order n

$$\begin{aligned}
&\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) \\
&\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t) \\
&\boldsymbol{x}(0) = \boldsymbol{x}_0 \\
&\boldsymbol{u}(t) \in \mathbb{R}^{m\times 1}, \boldsymbol{x}(t) \in \mathbb{R}^{n\times 1}, \boldsymbol{y}(t) \in \mathbb{R}^{p\times 1} \\
&\quad \boldsymbol{A} \in \mathbb{R}^{n\times n}, \boldsymbol{B} \in \mathbb{R}^{n\times m}, \boldsymbol{C} \in \mathbb{R}^{p\times n}, \boldsymbol{D} \in \mathbb{R}^{p\times m}
\end{aligned} \tag{4.1}$$

Assume another version of the system in equation (4.1) with model order q such that $q < n$ exists as

$$\begin{aligned}
&\dot{\boldsymbol{x}}_r(t) = \boldsymbol{A}_r\boldsymbol{x}_r(t) + \boldsymbol{B}_r\boldsymbol{u}(t) \\
&\boldsymbol{y}_r(t) = \boldsymbol{C}_r\boldsymbol{x}_r(t) + \boldsymbol{D}\boldsymbol{u}(t) \\
&\boldsymbol{x}_r(0) = \boldsymbol{x}_{r0} \\
&\boldsymbol{u}(t) \in \mathbb{R}^{m\times 1}, \boldsymbol{x}_r(t) \in \mathbb{R}^{q\times 1}, \boldsymbol{y}_r(t) \in \mathbb{R}^{p\times 1} \\
&\quad \boldsymbol{A}_r \in \mathbb{R}^{q\times q}, \boldsymbol{B}_r \in \mathbb{R}^{q\times m}, \boldsymbol{C}_r \in \mathbb{R}^{p\times q}, \boldsymbol{D}_r \in \mathbb{R}^{p\times m}
\end{aligned} \tag{4.2}$$

The MOR of the system in equation (4.1) is achieved by applying state transformation in

$$\boldsymbol{x}(t) = \boldsymbol{U}\boldsymbol{x}_r(t), \boldsymbol{U} \in \mathbb{R}^{n\times q} \tag{4.3}$$

And define the projection matrix $\boldsymbol{V} \in \mathbb{R}^{n\times q}$ such that the following

$$\begin{aligned} \boldsymbol{A}_r &= \boldsymbol{V}^T\boldsymbol{A}\boldsymbol{U} \\ \boldsymbol{B}_r &= \boldsymbol{V}^T\boldsymbol{B} \\ \boldsymbol{C}_r &= \boldsymbol{C}\boldsymbol{U} \end{aligned} \tag{4.4}$$

All projection-based methods determine $\boldsymbol{U}$ and $\boldsymbol{V}$ projection matrices according to constraints that examine the relationship between the original system output and the reduced system output to ensure close approximation to the original system. This is accomplished by computing Euclidian or Infinity norms cost functions.

4.2.1 Proper Orthogonal Decomposition methods

The Proper Orthogonal Decomposition (Moore, 1981) otherwise known by *Principal Component Analysis* (PCA), or *Karhunen-Loéve Decomposition* (KLD), have been used in many scientific fields such as modelling of fluids dynamics (Rowley, 2005, Hinze and Volkwein, 2005, Efe and Ozbay, 2003, Berkooz et al., 1993, Aubry, 1991), modelling of mechanical systems (Sifakis and Barbic, 2012, Kerschen et al., 2005, Lenaerts et al., 2001), *Micro Electromechanical Systems* (MEMS) (Liang et al., 2002a, Liang et al., 2002b), image processing and data compression among many others fields.

The main idea of POD is to construct a matrix of snapshots of the dynamical state of the system over time and reducing the dimensionality of the system via the Singular Value Decomposition (SVD) through elimination of weak states contribution. The POD is considered to be optimal in capturing the most dominant components of system dynamics, when modelled using empirical response data of a dynamical system whose formulation from first-principle is unknown, it reveals hidden structures in the data and helps to give some insight on the original system dynamics. Authors in (Efe and Ozbay, 2003) argue that capturing of all dominant components in the dynamics of a system response may still lead to qualitatively wrong dynamics of the system because of the loss of the weak yet important dynamics (Rowley, 2005). Also the reduction may not preserve the stability characteristics of the original system (Vasilyev, 2007).

4.2.2 Krylov Subspace methods

The Krylov Subspace methods are also known by*, Arnoldi, Lancoz Moment Matching,* methods (Antoulas, 2005) and *Padé Approximation* via Lancoz (Gallivan et al., 1994).

Krylov Subspace methods for linear model order reduction are associated with transfer function description of the linear dynamical system (Druskin and Simoncini, 2011) and moment matching methods (Gugercin et al., 2008, Boley, 1994).

The transfer function of the full-order LTI system in equation (4.1) which relates the system output and its input in frequency domain (through $\boldsymbol{y}(s) = \boldsymbol{H}(s)\boldsymbol{u}(s)$) is given by

$$\boldsymbol{H}(s) = \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D} \qquad (4.5)$$

Krylov subspace methods involve transfer function moments which are given by the Taylor series expansion of the high dimensional transfer function (4.5) around zero to yield low frequency moments, or around infinity to yield high frequency moments (*Markov Parameters*) or any frequencies of interest (Lohmann and Salimbahrami, 2000).

The Krylov subspace is defined by

$$K_q(\widetilde{\boldsymbol{A}},\widetilde{\boldsymbol{b}}) = span(\widetilde{\boldsymbol{b}},\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{b}},\dots,\widetilde{\boldsymbol{A}}^{q-1}\widetilde{\boldsymbol{b}})$$
$$\widetilde{\boldsymbol{A}} \in \mathbb{R}^{n\times n}, \widetilde{\boldsymbol{b}} \in \mathbb{R}^{n\times 1}, \tag{4.6}$$
$$K_q(\widetilde{\boldsymbol{A}},\widetilde{\boldsymbol{b}}) \in \mathbb{R}^{n\times q}$$

$\widetilde{\boldsymbol{b}}$ is called the *starting vector* of the Krylov subspace. The MOR projection matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are any basis of the Krylov subspace in

$$\begin{aligned}\boldsymbol{V} &\subset K_{q_1}(\boldsymbol{A}^{-1},\boldsymbol{A}^{-1}\boldsymbol{B}) \\ &= span\left(\boldsymbol{A}^{-1}\boldsymbol{B},\dots,(\boldsymbol{A}^{-1})^{q_1-1}\boldsymbol{A}^{-1}\boldsymbol{B}\right) \\ \boldsymbol{U} &\subset K_{q_2}((\boldsymbol{A}^{-1})^T,(\boldsymbol{A}^{-1})^T\boldsymbol{C}) \\ &= span\left((\boldsymbol{A}^{-1})^T\boldsymbol{C},\dots,((\boldsymbol{A}^{-1})^T)^{q_2-1}(\boldsymbol{A}^{-1})^T\boldsymbol{C}\right)\end{aligned} \tag{4.7}$$

q_1 and q_2 are chosen such that both projection matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ have a rank of q and the reduced state matrix $\boldsymbol{A}_r$ (equation (4.4)) is not singular. This method is called the two-sided Krylov MOR. If only the $\boldsymbol{V}$ projection matrix is constructed from a Krylov subspace and $\boldsymbol{U}$ chosen arbitrarily the MOR method is called one-sided Krylov (Lohmann and Salimbahrami, 2000). A common choice of the $\boldsymbol{U}$ projection marix in one-sided Krylov MOR is $\boldsymbol{U} = \boldsymbol{V}$.

Krylov subspace methods are iterative methods, for linear systems of order thousand or above; they are generally considered the most preferred way to deal with such systems because of its iterative nature in constructing the Krylov subspace in an increasing order through the projection of the original system matrices to the low dimensional Krylov subspace (Boley, 1994). Krylov methods often converge fast enough and produce low order approximations much lower than that of the original high dimensional system.

Building the basis vectors for Krylov subspaces is accomplished through iterative numerical algorithms that ensure independency of these vectors via orthogonality and normalization of the basis, often done with either Arnoldi Processes, Lanczos Algorithms or modified versions of them (Grimme et al., 1996).

The thesis will consider using one-sided Krylov subspace MOR (Algorithm 4-1) generated by Arnoldi Processes with choice of projection matrix $\boldsymbol{U} = \boldsymbol{V}$ as an example (Lohmann and Salimbahrami, 2000).

```
1   **input:** $\boldsymbol{x}_0$ (initial state), $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ state-space matrices for LTI system of order $n$
2   **initialize** projection matrix $\boldsymbol{V} \in \mathbb{R}^{n\times q}$
3   **compute** $\boldsymbol{A}^{-1}$
4   $\boldsymbol{V}(:,1) \coloneqq \frac{\boldsymbol{A}^{-1}\boldsymbol{B}}{||\boldsymbol{B}||}$ (Krylov subspace starting vector).
5   **for** $i \coloneqq 2$ to $q$
6   $\boldsymbol{V}(:,i) = \boldsymbol{A}^{-1}\boldsymbol{V}(:,i-1)$
7   **for** $j \coloneqq 1$ to $i-1$
8   $ov := \boldsymbol{V}(:,i)^T\boldsymbol{V}(:,j)$
9   $\boldsymbol{V}(:,i) = \boldsymbol{V}(:,i) - ov\boldsymbol{V}(:,j)$ (orthogonalize the $i$th projection basis)
10  **end for** loop
11  $\boldsymbol{V}(:,i) = \frac{\boldsymbol{V}(:,i)}{||\boldsymbol{V}(:,i)||}$ (normalize the $i$th projection basis)
12  **end for** loop
13  $\boldsymbol{A_r} := \boldsymbol{V}^T\boldsymbol{AV}$
14  $\boldsymbol{B_r} := \boldsymbol{V}^T\boldsymbol{B}$
15  $\boldsymbol{C_r} = \boldsymbol{CV}$
16  $\boldsymbol{x_{0r}} = \boldsymbol{V}^T\boldsymbol{x_0}$
17  **return:** $\boldsymbol{x_{0r}}$ (reduced order initial state), $\boldsymbol{A}_r, \boldsymbol{B}_r$ and $\boldsymbol{C}_r$ state-space matrices for LTI system of reduced order $q$
```

Algorithm 4-1: Arnoldi one-sided Krylov subspace MOR

The Arnoldi Algorithm 4-1 does two important things while constructing the projection matrix $\boldsymbol{V}$, it ensures uniqueness of any projection basis by ensuring its orthogonality to the previous ones, then it performs the normalization of all projection basis vectors; therefore; $\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}$. The reduced order linear time invariant state-space model will match the first q moments of the transfer function of the full order system. A final note on Algorithm 4-1, $\boldsymbol{A}^{-1}$ has to be calculated at the start of the algorithm which is inaccurate for high order systems and the practical way is to solve a linear equation given by $\boldsymbol{V}(:,i-1)=\boldsymbol{A}\boldsymbol{V}(:,i)$ during the construction of the projection matrix $\boldsymbol{V}$, there are many ways of solving linear system of equations and for those with sparse and structured matrices the computation runs faster than dense state matrices.

Krylov subspace methods do not always preserve the stability of the original system (Bai, 2002, Lohmann and Salimbahrami, 2000, Grimme et al., 1996, Gallivan et al., 1996), the initial guess of the starting basis vectors is a trial and error procedure and no information regarding the observability matrix is used in the reduction in one-sided methods.

Many solutions have been proposed to tackle these problems, but again they are not global solutions and often tailored to certain system.

Despite its limitations; the low computational complexity (Lohmann and Salimbahrami, 2000), its ability to implement with Parallel Processing techniques and low computational storage needs makes it leading in model order reduction tools of highly complex dynamical systems. The iterative nature of the Krylov subspace MOR methods means once a reduced order model was obtained for a certain choice of order; the calculations for the Krylov projection matrices can be repeated for a higher choice of order starting from the last projection matrices.

Krylov subspace methods have been applied to many complex engineering problems like Structural dynamics (Yue and Meerbergen, 2012), Electronic Circuit Simulation (Freund, 2008, Freund, 2003, Bai, 2002, Freund, 2000), Mechanical Systems (Fischer and Eberhard, 2014), and MEMS (Rewienski, 2003), all did achieve computational time savings in comparison with system's full order models.

4.2.3 Truncated Balanced Realization

The method of model order reduction for a balanced realization of linear time invariant systems was introduced in (Moore, 1981) and later developed in (Pernebo and Silverman, 1982, Laub et al., 1987). A balanced realization of a linear time invariant system is achieved through the examination of the controllability and observability notions in such system, then by applying a state transformation such that the associated controllability $\boldsymbol{\mathcal{P}}$ and observability $\boldsymbol{\mathcal{Q}}$ Gramians are diagonal and equal i.e. are balanced.

The balanced controllability $\boldsymbol{\mathcal{P}}_b$ and observability $\boldsymbol{\mathcal{Q}}_b$ Gramians are the solution to two Lyapunov equations

$$\begin{aligned} \boldsymbol{A}\boldsymbol{\mathcal{P}}_b + \boldsymbol{\mathcal{P}}_b\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{B}^T &= 0 \\ \boldsymbol{A}^T\boldsymbol{\mathcal{Q}}_b + \boldsymbol{\mathcal{Q}}_b\boldsymbol{A} + \boldsymbol{C}\boldsymbol{C}^T &= 0 \end{aligned} \tag{4.8}$$

The square roots of the eigenvalues λ_j of the balanced controllability $\boldsymbol{\mathcal{P}}_b$ and observability $\boldsymbol{\mathcal{Q}}_b$ Gramians ordered decreasingly are called the *Hankel Singular Values* (HSV) σ_j given by

$$\sigma_j = \sqrt[2]{\lambda_j(\boldsymbol{\mathcal{P}}_b = \boldsymbol{\mathcal{Q}}_b)}\,, \qquad \sigma_1 > \sigma_2 > \cdots > \sigma_n \tag{4.9}$$

The balanced realization of the LTI system from equation (4.1) is computed with state transformation $\boldsymbol{x}_b(t) = \boldsymbol{T}\boldsymbol{x}(t), \boldsymbol{T} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{n}}$ and is given by

$$\begin{aligned}\dot{\boldsymbol{x}}_b(t) &= \boldsymbol{TAT}^{-1}\boldsymbol{x}_b(t) + \boldsymbol{TBu}(t)\\ \boldsymbol{y}_b(t) &= \boldsymbol{CTx}_b(t) + \boldsymbol{Du}(t)\\ \boldsymbol{x}_b(0) &= \boldsymbol{Tx}(0)\end{aligned} \tag{4.10}$$

The balanced transformation matrix $\boldsymbol{T}$ relates the unbalanced controllability $\boldsymbol{\mathcal{P}}$ and observability $\boldsymbol{\mathcal{Q}}$ Gramians to the balanced versions through

$$\begin{aligned}\boldsymbol{\mathcal{P}}_b &= \boldsymbol{T}\,\boldsymbol{\mathcal{P}}\boldsymbol{T}^T\\ \boldsymbol{\mathcal{Q}}_b &= (\boldsymbol{T}^{-1})^T\boldsymbol{\mathcal{Q}}\boldsymbol{T}^{-1}\end{aligned} \tag{4.11}$$

Therefor; the balanced state transformation matrix $\boldsymbol{T}$ is found by plugging (4.11) into equation (4.8) and solving.

Model order reduction is accomplished by removing the least controllable and observable states of the balanced realization equation (4.10) which is called *state truncation* and the reduced order modelling technique is called *Truncated Balanced Realization* (TBR). This process is done after the transformation of the original system to balanced realization and observing the HSVs of this system to decide the order of the reduction q, a relatively small singular value means the associated states contribute little to the system response and therefore can be eliminated to reduce the model order.

The balanced realization of the system is given by

$$\begin{aligned}\boldsymbol{A}_b &= \boldsymbol{TAT}^{-1}\\ \boldsymbol{B}_b &= \boldsymbol{TB}\\ \boldsymbol{C}_b &= \boldsymbol{CT}\end{aligned} \tag{4.12}$$

And decide the order of the reduced system q to decompose equation (4.12) to

$$\begin{aligned} \boldsymbol{A}_b &= \begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix} \\ \boldsymbol{B}_b &= \begin{pmatrix} \boldsymbol{B}_1 & \boldsymbol{B}_2 \end{pmatrix} \\ \boldsymbol{C}_b &= \begin{pmatrix} \boldsymbol{C}_1 \\ \boldsymbol{C}_2 \end{pmatrix} \end{aligned} \tag{4.13}$$

The parameters of the reduced order system q are given by

$$\begin{aligned} \boldsymbol{A}_r &= \boldsymbol{A}_{11} \\ \boldsymbol{B}_r &= \boldsymbol{B}_1 \\ \boldsymbol{C}_r &= \boldsymbol{C}_1 \end{aligned} \tag{4.14}$$

In all the above state-space transformations the $\boldsymbol{D}$ matrix was invariant to the projection. Another truncated realization to the reduced order model called *singular-perturbation* TBR, it provide better accuracy in the steady-state response of the TBR MOR system and is given as

$$\begin{aligned} \boldsymbol{A}_r &= \left(\boldsymbol{A}_{11} - \boldsymbol{A}_{12}\boldsymbol{A}_{22}{}^{-1}\boldsymbol{A}_{21}\right) \\ \boldsymbol{B}_r &= \left(\boldsymbol{B}_1 - \boldsymbol{A}_{12}\boldsymbol{A}_{22}{}^{-1}\boldsymbol{B}_2\right) \\ \boldsymbol{C}_r &= \left(\boldsymbol{C}_1 - \boldsymbol{C}_2\boldsymbol{A}_{22}{}^{-1}\boldsymbol{A}_{21}\right) \\ \boldsymbol{D}_r &= \left(\boldsymbol{D} - \boldsymbol{C}_2\boldsymbol{A}_{22}{}^{-1}\boldsymbol{B}_{\boldsymbol{2}}\right) \end{aligned} \tag{4.15}$$

Important properties of the TBR are:

- The optimality of the reduced order model through the informed choice of q based on HSVs of the balanced realization.

- The preservation of the original system stability and provable error bounds.

- The drawback of the TBR is that it is not suitable for large model orders ($n > 1000$) due to computational complexity and storage requirement of the solution to equations (4.8) and (4.11).

Other methods have been proposed to extend the TBR to high order systems (Gugercin and Antoulas, 2004, Willcox and Peraire, 2002, Li, 2000) and also to different linear structures (Sandberg, 2008), for more in-depth review please refer to (Antoulas, 2005) and the references therein.

4.3 Model Order Reduction of CI-NDSs

To date, MOR for CI-NDSs is a premature subject. Nonlinear MOR methods remodel the system nonlinearity with a local linearization or a global trajectory piecewise linear or polynomial approaches obtained from series expansion around the NDS operating points. The end result is always a model with linear parameters that their dimensionality can be reduced through linear MOR projection techniques or a combination of them (Phillips and Silveira, 2005).

POD combined with balanced realization have been applied to MOR of nonlinear systems empirical controllability and observability Gramians constructed from input-output data (Lall et al., 1999, Hahn and Edgar, 2002). Hahn et al. (2003) proposed to use POD with empirical controllability and observability covariance matrices as an improvement over the previous method. The authors have shown that this method can capture the nonlinear behaviour better and is suitable for wide range of inputs.

A *Trajectory Piecewise Linear* (TPWL) nonlinear MOR has been proposed in (Vasilyev et al., 2003). TPWL generated local models from the Taylor's series expansion around the NDS operating point along its simulation trajectory as model training points. The order of the generated local models has been reduced using either POD MOR, or Krylov subspace MOR and or TBR MOR or a combination of them (depending on the computational complexity of the original NDS and the end goal). The reduced order linear models have been assigned weights in terms of the Euclidian distance of the CI-NDS current solution state to the linearization point in a fashion resembling radial basis networks.

TPWL nonlinear MOR approaches have been applied successfully to the modelling of MEMS models, nonlinear transmission line circuits, nonlinear analogue circuits and nonlinear fluid dynamics problems (Bond and Daniel, 2005, Rewieński and White, 2006, Bechtold et al., 2008, Cardoso and Durlofsky, 2010). Nahvi et al. (2013) argued that TPWL weights that are function of the Euclidian distance are constant for the nonlinearity in the trajectory of the NDS and therefore they do not preserve the nonlinear field curvature and superposition principle. They have proposed Nonlinearity-aware TPWL MOR approach (NTPWL). In contrast to TPWL approach, NTPWL places error bounds on the weighting procedure by incorporating the state velocity along with the state; therefore improving the quality of the reduced order NDS predictions. This method was applied successfully to nonlinear transmission line model and nonlinear RC ladder circuit (Nahvi et al., 2013).

Authors in (Dong and Roychowdhury, 2003, Dong and Roychowdhury, 2008) proposed Piecewise Polynomial (PWP) MOR to remodel the nonlinear system. The PWP MOR was proposed because TPWL methods have bad small input signal performance as in the case of nonlinear circuits, TPWL models fail to replicate harmonic distortion and the noise generated in such circuits. PWP MOR was successfully applied to nonlinear transmission line circuit model with small magnitude input signals.

4.4 Meta- Modelling CI-NDS with Reduced Order Local Linear Models

The most popular approach out of the nonlinear MOR schemes explored in the previous section is the TWPL MOR. TWPL local models are linear but their global model is not which creates a model transparency issue from the viewpoint of this thesis. All nonlinear MOR techniques explored by this thesis use parametric regression approach in the blending of the LLMs during the simulation, therefore they lack any uncertainty information to their predictions of the LLMs parameters and by extension the global model predictions. TPWL MOR requires the computation of LLMs at a number of operating points on the trajectory of the NDS; the research has found no information on how these Taylor's series expansion parameters are computed and therefore assumed they are found from the Jacobian (partial derivatives) derived from the CI-NDS first-principle model.

The GP blended VBL-LPV meta-model introduced in chapter three is a globally linear and transparent model, the VBL-LLMs describe the behaviour of the NDS exactly at any operating point along the trajectory. The non-parametric blending of the VBL-LLMs through GPMs provides confidence measure on their predictions and by extension the meta-model predictions.

Since the VBL-LLMs used by the meta-model are linear, their model order can be reduced using suitable projection-based linear MOR techniques.

The meta-model training points described in chapter three can undergo model order reduction prior to the training of the GPMs of those parameters.

The rest of the thesis will explore the one-sided Krylov subspace MOR with Arnoldi procedure (Algorithm 4-1) as well as the TBR MOR (section 4.2.3) or a combination of both methods when modelling a CI-NDS.

The proposed reduced order meta-model is similar to the meta-model structure in chapter three except that after the generation of the VBL-LLMs, MOR technique of choice is applied and new reduced order VBL-LLMs are computed and their associated operating point state vector is transformed according to the computed MOR projection basis.

For the linear MOR technique to work, an informed choice for the reduced model order q has to be made. Since all the collected LLMs in the training phase of the meta-model were obtained at different parts of the CI-NDS trajectory, a choice of reduced LLM order q is not necessarily be globally valid for all the collected LLMs i.e. some LLM will need higher reduced model order than others. Therefore, an error measure has to be constructed to quantify how much the reduced order LLM deviated from the full order one. In the case of the proposed meta-model structure; this error is calculated by observing the error between the outputs of the full and the reduced order model at its associated operating point $\{\boldsymbol{x}_i, \boldsymbol{u}_i\}$. For the ith LLM; this error is given by

$$\begin{aligned} err_{qi} &= \frac{\left||\boldsymbol{y}_i - \boldsymbol{y}_{ri}|\right|}{\left||\boldsymbol{y}_i|\right|}, \qquad \forall\left||\boldsymbol{y}_i|\right| \neq 0 \\ \boldsymbol{y}_i &= \boldsymbol{C}_i\boldsymbol{x}_i + \boldsymbol{D}_i\boldsymbol{u}_i \\ \boldsymbol{y}_{ri} &= \boldsymbol{C}_{ri}\boldsymbol{x}_{ri} + \boldsymbol{D}_i\boldsymbol{u}_i \end{aligned} \tag{4.16}$$

Where $\boldsymbol{y}_i$ are the full-order function targets and $\boldsymbol{y}_{ri}$ are the reduced-order function targets, If $\left||\boldsymbol{y}_i|\right| = 0$ then $err_{qi} = \left||\boldsymbol{y}_{ri}|\right|$.

With the error criteria specified, the reduced order LLMs in the meta-model training phase can be tested for any choice of q and a good choice of q can be selected if all the computed errors are below some specified error tolerance.

This error estimation is of great importance when using Krylov subspace MOR. As stated before, Krylov subspace MOR lack error bounds and the choice of the reduced model order q is as good as any without some error quantification. The error estimation in equation (4.16) can be incorporated into the Krylov subspace MOR algorithm when constructing the Krylov projection basis $\boldsymbol{V}$ as a stopping criteria for the algorithm to reach some optimal solution at order q. The application of this stopping criteria based on some error tolerance to all the collected full order LLMs may result in different values of q and since all the reduced order LLMs must be of the same order for the meta-model GPMs blending to work; the highest reduced order among all the reduced LLMs orders can be selected and the MOR repeated at that order to insure all local errors are below the specified error tolerance.

In the case of TBR MOR method; an informed choice for the reduced model order at any LLM can be selected through the observation of the HSVs (equation (4.9)) of the balanced full order LLM. Each HSV reflects how much energy each state of the balanced full-order LLM is contributing to the output, therefore a plot of these HSVs can show where the states vector can be truncated.

The percentage of the jth state contribution $(psc)_j$ in the ith balanced LLM of order n is given in terms of the jth HSV (σ_j) computed in the balancing of the LLM, as

$$(psc)_j = 100 \times \left(\frac{\sigma_j}{\sum_1^n \sigma_j} \right) \tag{4.17}$$

For all the collected full order LLMs in the meta-model training, all state contribution percentages for each balanced LLM can be added up and averaged to reflect the percentages of the average contribution of states across the collected LLMs. This will require the storage of all the balanced realization for all the LLMs prior to deciding the reduced order model q which may consume great amounts of memory and might not be feasible for large order systems, in this case; selecting the reduced order model q can be done in a similar fashion to Krylov subspace MOR method.

4.5 Canonical Transformations

The reduced order VBL-LLMs parameters obtained using Krylov subspace MOR and/ or TBR MOR are fully parameterized even when constructed from sparse full order matrices. If left in their dense form; they will require large number of GPMs for each time-varying element of the reduced order parameter matrix thus contributing heavily to the training and simulation times of the meta-model. If the feed-through matrix $\boldsymbol{D}$ of the full order LLM was zero; then the maximum number of GPMs that needs training and used during the simulation of the reduced-order meta-model can reach $(q^2 + qm + pq)$.

The thesis is proposing to transform reduced order LLMs to a canonical form that has more constant elements in the parameters of the reduced order LLMs.

One suitable form is the *modal canonical form* also known as the *diagonal canonical form*.

The modal canonical form is achieved by computing a projection matrix $\boldsymbol{T}_c \in \mathbb{R}^{q\times q}$ with the state transformation $\boldsymbol{x}_{rc} = \boldsymbol{T}_c\boldsymbol{x}_r$. The modal canonical form of the reduced order model is given by

$$\begin{aligned}\dot{\boldsymbol{x}}_{rc}(t) &= \boldsymbol{A}_{rc}\boldsymbol{x}_{rc}(t) + \boldsymbol{B}_{rc}\boldsymbol{u}(t)\\ \boldsymbol{y}_{rc}(t) &= \boldsymbol{C}_{rc}\boldsymbol{x}_{rc}(t) + \boldsymbol{D}\boldsymbol{u}(t)\\ \boldsymbol{x}_{rc}(0) &= \boldsymbol{x}_{rc0}\end{aligned} \tag{4.18}$$

The parameters of the modal form of the reduced-order model are given by

$$\begin{aligned}\boldsymbol{A}_{rc} &= \boldsymbol{T}_c\boldsymbol{A}_r{\boldsymbol{T}_c}^{-1}\\ \boldsymbol{B}_{rc} &= \boldsymbol{T}_c\boldsymbol{B}_r\\ \boldsymbol{C}_{rc} &= \boldsymbol{C}_r{\boldsymbol{T}_c}^{-1}\end{aligned} \tag{4.19}$$

The modal form state matrix $\boldsymbol{A}_{rc}$ is a block-diagonal matrix of the eigenvalues of the original reduced order $\boldsymbol{A}_r$ matrix. In the case that the reduce order model only contained real eigenvalues; the number of the non-constant elements that need to be trained and simulated by the meta-model is $(q + qm + pq)$ therefore greatly reducing the meta-model training and simulation times.

Another form is the *controllability staircase* canonical form which is basically a decomposition of the reduced order LLMs to controllable and uncontrollable parts.

There is generally no way to know beforehand which of the two forms, if any, will work for a particular reduced-order meta-model. In addition to the requirement that a certain canonical form have the highest number of constants during the blending of the reduced-order meta-model parameters to reduce the computational load from GPM predictions, they should also provide smooth parameter functions if possible.

4.6 Meta-Modelling of the 10th order Nonlinear Transmission Line

The Nonlinear Transmission Line (NTL) circuit (Chen and White, 2000) given in Figure 4-1 is a CI-NDS, the NTL system has stiff ordinary differential equations contributing to the long simulation time especially in high model orders (even with dedicated solvers).

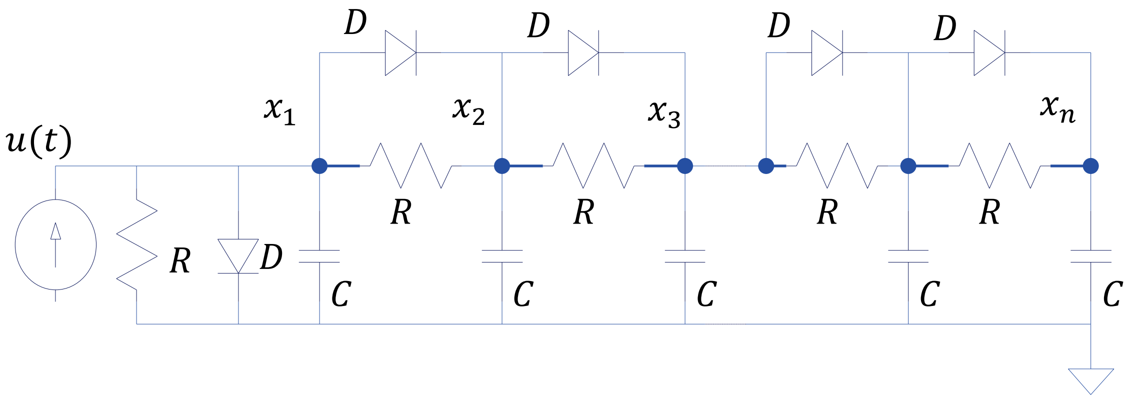


Figure 4-1: The Nonlinear Transmission Line circuit.

The NTL in Figure 4-1is a nonlinear dynamical system of order n, it consists of one input $u(t)$ which is a current source feeding node x_1, a cascading number of electrical elements of unit-resistors ($R = 1Ohm$), unit capacitors ($C = 1Farad$) and diodes (D). The states of the NDS are the voltages at nodes $\{x_1, x_2, \dots, x_n\}$. The nonlinear part of the NTL is the diode current given by $i_D(v) = e^{40v-1}$ in terms of the voltage (v) across each diode/resistor combination (Chen and White, 2000), this lead to the set of nonlinear ordinary differential equations given by appendix A.4. The range of the input current $u(t)$ is between zero and one amperes. The default initial condition $\boldsymbol{x}_0$ of the NTL model is set to zero.

In this section, a 10th order NTL model with output set to $y = x_5$ (node five) will be used to showcase the proposed reduced-order meta-modelling (TBR-MOR applied to the full-order LLMs), the 10th NTL is not a CI-NDS, and the thesis will present the reduced-order meta-modelling for a far more complex 5000 order NTL in chapter six to highlight the prospective savings in computational time. The 10th order NTL meta-model equilibrium training data was generated by uniformly selecting 20 points from model input range between zero and one, and then analytically computing the corresponding equilibrium states (appendix A.4).

The step sequence in Figure 4-2 generated using Algorithm 3-2 and used to excite the 10th order NTL model to collect the meta-model off-equilibrium training data (210 points). The total meta-model training data is 230 points. VBL-LLMs for each collected training point are computed with the NTL model Jacobians obtained from first-principle (appendix A.4). The order of the collected LLMs is given by $n = 10$.

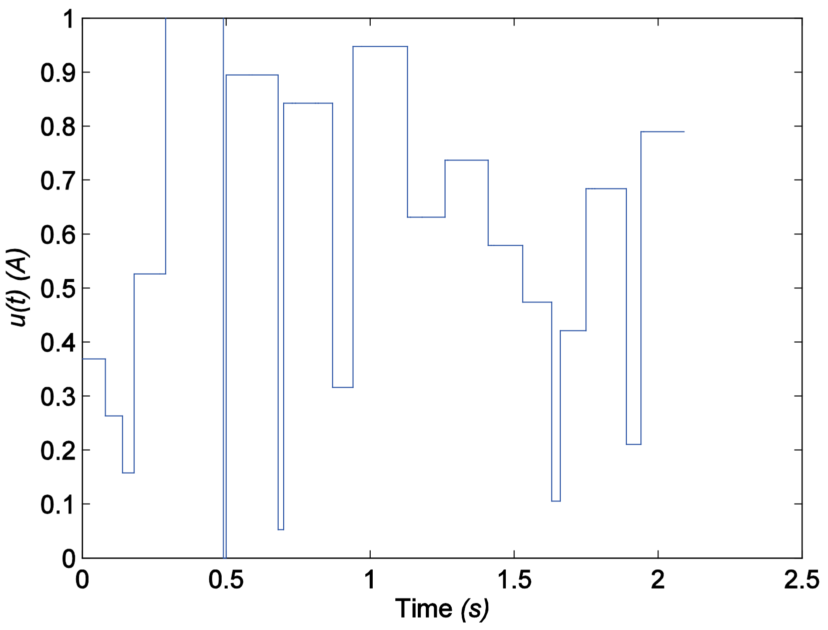


Figure 4-2: Randomly generated step sequence used to excite the 10th order NTL model in off-equilibrium regions. The sequence was generated using Algorithm 3-2 with parameters: $u_{min} = 0, u_{max} = 1, N_{ticks} = 20,\ Ts = 0.01s$ and $stm = 2$

TBR MOR approach was selected to compute a reduced order LLMs because the problem of order reduction is small. The mean percentages of states contribution to the balanced LLMs was computed with the aid of the balanced LLMs HSVs and equation (4.17) and is given by Figure 4-3.

Figure 4-3 helps to make an informed decision on the order of the reduction for the LLMs by observing that state contribution to the output energy falls dramatically after the 4th balanced state component, therefore the reduced order of LLMs was set to $q = 4$ in the case of the 10th order NTL system.

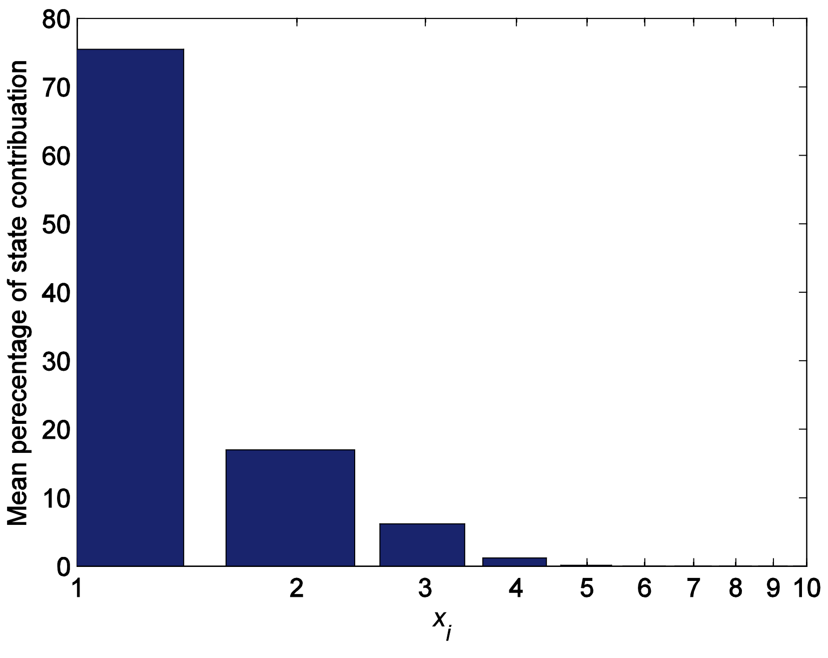


Figure 4-3: Mean percentage of state contributions of the 10th order NTL system balanced training LLMs.

Controllability canonical form (section 4.5) was applied to the reduced order LLMs therefore reducing the non-constant elements of $\boldsymbol{A}_r$ from sixteen to ten in $\boldsymbol{A}_{rc}$, therefore the total number of GPMs that needs training is ten ($(\boldsymbol{b}_{rc}, \boldsymbol{c}_{rc})$ vectors are cosntant).

A squared-exponential with ARD covariance function was chosen and the GPM means are assumed to be zero for all the reduced order LLM time-varying entries.

During the training of each GPM, cross-validation was used to test the accuracy resulted GPM model. This is done through splitting the collected training points into two disjoint sets (115points each) using uniform sampling with odd-even indices.

The GPM covariance function hyper-parameters are trained by minimizing the cost function $\boldsymbol{J(\theta)}$ in equation (3.10). The chosen covariance function for the 10th order NTL model has a total of six parameters (four for each state, one for process variance and one for noise parameter). The initial values of the six GPM parameters prior to optimization were set to one (parameters one to five) and the noise parameter was set to 1×10^{-7} after adding the same amount of zero-mean Gaussian noise to the collected training data targets. The GPMs optimization results for the 10th order NTL NDS meta-model are given by Table 4-1.

Table 4-1: Meta-Model training results for the 10th Order NTL

$GPM(\boldsymbol{A}_{rc})$	$SMSE$	$\%fitness$	$\%RSD_{max}$
a_{11}	4.07×10^{-4}	97.97	6.47×10^{-2}
a_{12}	6.96×10^{-5}	99.16	7.57×10^{-2}
a_{21}	6.96×10^{-5}	99.16	7.57×10^{-2}
a_{22}	1.37×10^{-3}	96.28	1.58×10^{-2}
a_{23}	1.05×10^{-3}	96.74	1.5×10^{-1}
a_{32}	1.05×10^{-3}	96.74	1.5×10^{-1}
a_{33}	1.05×10^{-4}	98.97	5.92×10^{-2}
a_{34}	3.75×10^{-4}	98.06	1.36×10^{-1}
a_{43}	3.75×10^{-4}	98.06	1.36×10^{-1}
a_{44}	1.93×10^{-5}	99.56	5.61×10^{-2}

Table 4-1 shows the high values of GPM fitness when tested with the validation data set coupled with low values of $SMSE$, $\%RSD$ values are all below 0.2% indicating high confidence in the predictions. There are similar values of $SMSE, \%fitness$ and $\%RSD_{max}$ among some of the trained meta-model parameters; it indicates that those parameters share the same underlying model for the considered training data and this property can used to significantly boost the meta-model solver performance during simulation by predicting only one parameter at each time step and using this prediction directly for all the other similar parameters, this is called parameters similarity detection. With the cross-validation phase finished; the reduced order meta-model of the 10th order NTL model is ready to be tested. The simulation of the 10th order NTL was conducted using classical fixed-time RK4 solver.

The first test input is given by simple multi-level step input approximated by a sigmoid function (appendix B) and its reduced-order meta-model simulation result is given by Figure 4-4.

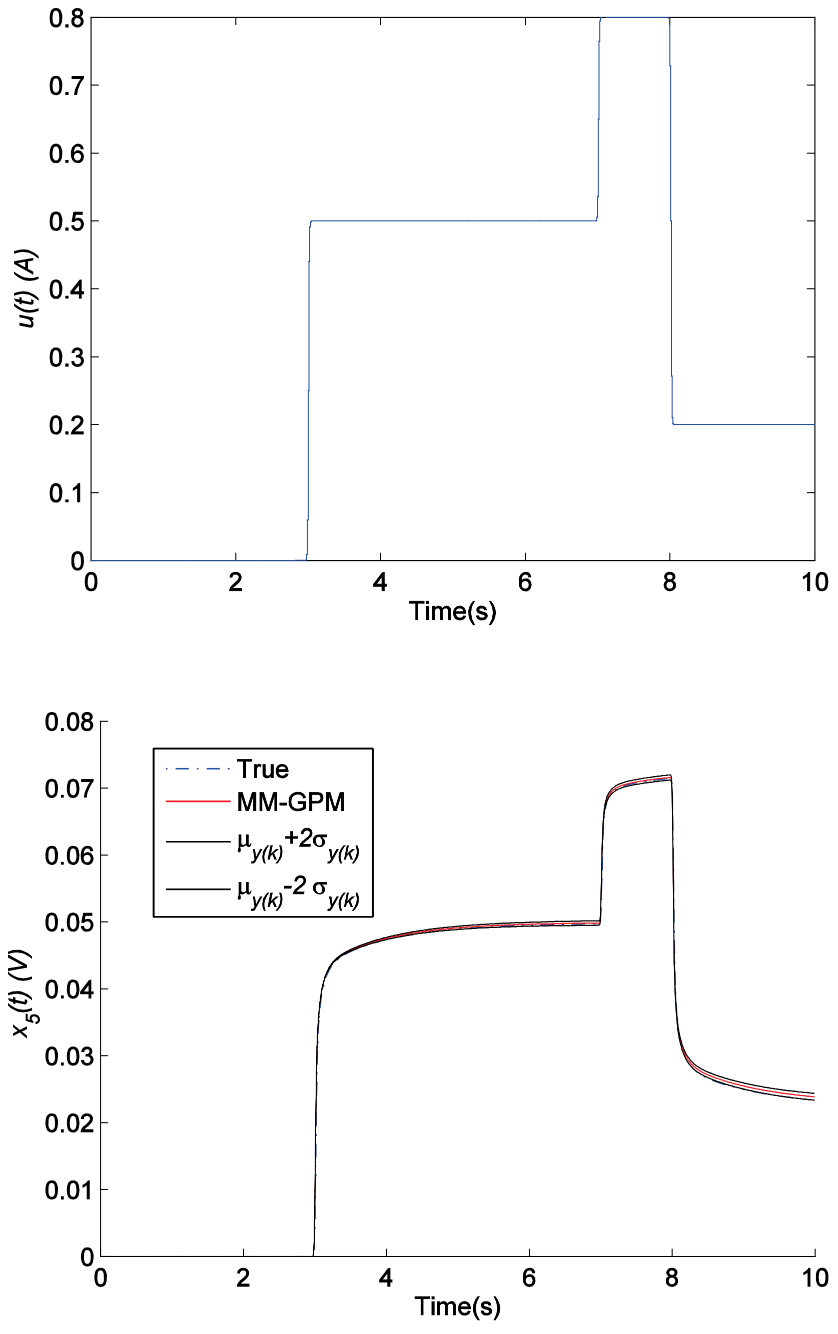


Figure 4-4: Multi-level step test input (top plot), the 10th order NTL model response versus the reduced order meta-model response with confidence regions (bottom plot).

The multi-level step test input in Figure 4-4 was sampled every $0.01s$, therefore the NTL full order model and the reduced order meta-model were both solved with a fixed time step of $h = 0.01s$. Exact-Discretization velocity solver was used in the reduced-order meta-model because it provided the highest numerical accuracy (section 3.4.1). The reduced-order meta-model achieved 97.38% fitness and prediction has tight confidence regions (subject of Chapter five).

To push the limits of the trained meta-model, a more challenging frequency sweep sinusoidal test input was used (Figure 4-5), this signal was generated with the aid of 'chirp' command in MATLAB which generates a cosine upward linear frequency sweep using time frame of one second, starting from DC, crossing $25Hz$ at $0.5s$ and sampled at $10kHz$ (this high sampling rate was chosen to account for the nature of fixed-time step meta-model solver).

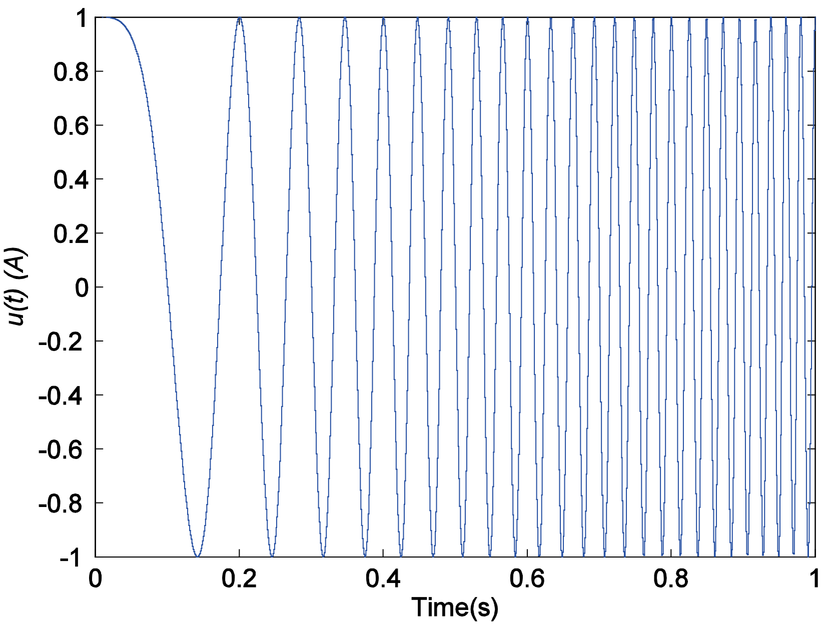


Figure 4-5: Sinusoidal test input generated from a cosine linear frequency sweep from DC, crossing $25Hz$ at $0.5s$.

The generated sinusoidal test input of Figure 4-5 has a top frequency of $50Hz$, the choice of the maximum frequency of the sinusoidal test input was due to the selection of the NTL circuit passive components having a high time-constant($RC = 1s$, with $-3dB$ cut-off frequency of $\frac{1}{2\pi RC} = 0.16Hz$) will be highly attenuated, if the NTL model was used with small time constants, higher frequency test inputs can be used but this will require variable-time meta-model solvers which are beyond the scope of this work. The meta-model solver is of fixed-time step and it cannot produce a good response accuracy for anything less than a fixed-time step of $h = 1 \times 10^{-4}s$. The meta-model response for the sinusoidal test input in Figure 4-5 is given by Figure 4-6.

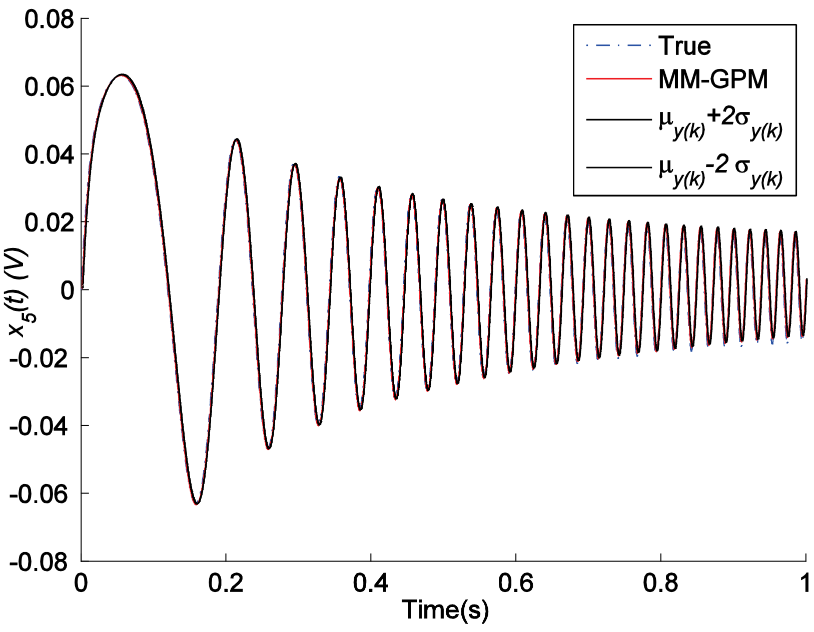


Figure 4-6: The 10th order NTL model response versus the reduced order meta-model response with confidence regions.

Although the frequency sweep test input of Figure 4-5 has amplitude which extends below zero well outside the range of the collected meta-model training inputs(between zero and one), the reduced-order meta-model achieved 97.37% fitness and the predictions has indistinguishable confidence regions due to the very small solver time step (subject of Chapter five).

The $\%RDS_{max}$ can give insight about the uncertainty of the reduced order meta-model solver time-varying parameters during the solution of the previous test inputs. For the 10th order NTL system, these parameters are the time-varying entries in $\boldsymbol{A}_{rc}$ matrix (10 of them). $\%RDS_{max}$ values for the multi-step test input in Figure 4-4 are given by Figure 4-7.

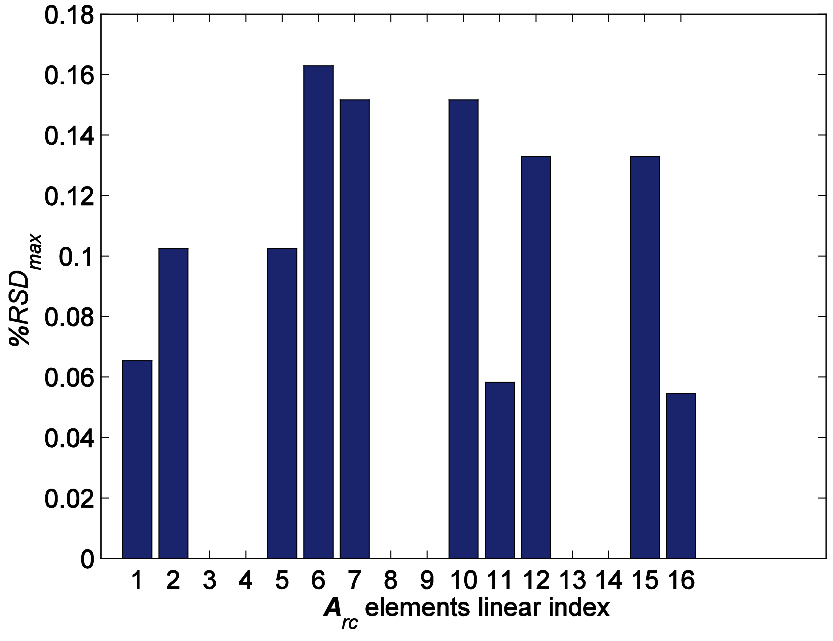


Figure 4-7: $\%RSD_{max}$ for the 10th order NTL reduced-order meta-model during the solution of the multi-step test input (Figure 4-4 (Top)).

Figure 4-7 show that $\%RSD_{max}$ kept very small values druing the solution of the multi-step test input therefore indicating high confidence in the reduced-order meta-model parameters predictions. Constant-entries in the $\boldsymbol{A}_{rc}$ matrix have no uncertainty associated with them and are denoted with white spaces in Figure 4-7, while equal time-varying entries in the $\boldsymbol{A}_{rc}$ state matrix have equal values of uncertainty and therefore equal values of $\%RDS_{max}$.

$\%RDS_{max}$ values for the frequency sweep test input in Figure 4-5 are given by Figure 4-8.

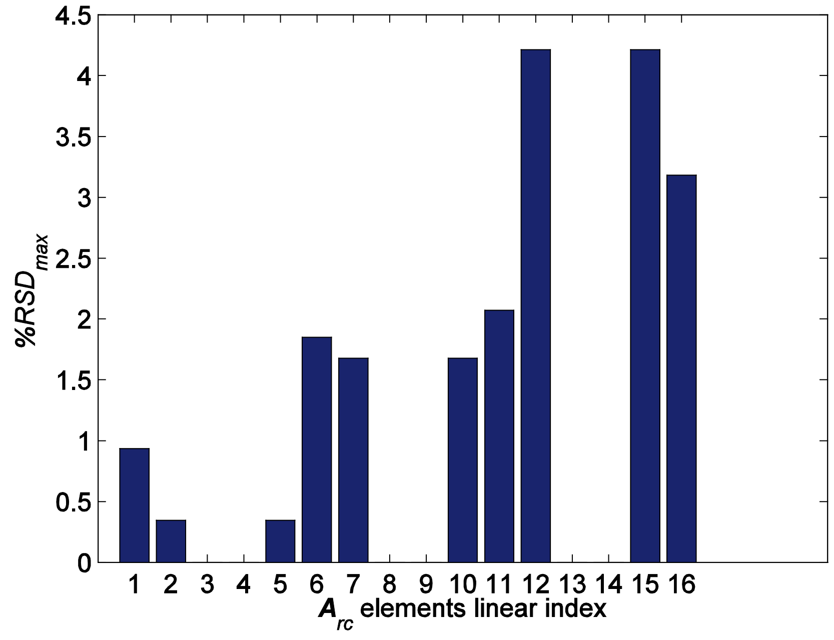


Figure 4-8: $\%RSD_{max}$ for the 10th order NTL reduced-order meta-model during the solution of the frequency sweep test input (Figure 4-5).

Figure 4-8 show that $\%RSD_{max}$ values (for the time-varying entries in $\boldsymbol{A}_{rc}$ state matrix) during the simulation of the frequency sweep test input are higher than those values obtained during the meta-model simulation of the multi-step input, this is because of the frequency sweep test input range (Figure 4-5) being outside the collected training data of the meta-model.

Finally, the execution time of the fourth-order meta-model to the 10th order NTL model was compared using ETI (equation (3.33)) but this time for two cases of the meta-model solver(with or without the meta-model parameters similarity detection feature). ETI values for the two previously discussed test inputs are given by Table 4-2.

Table 4-2: ETI values for the 10th order NTL meta-model the for multi-step and frequency sweep test inputs.

Test Inputs	Multi-step test input	Frequency sweep test input
ETI	-2.83	-2.89
ETI (parameter similarity detection)	-2.1	-2.35

Table 4-2 shows that all the computed ETI values are negative meaning that the reduced-order meta-model was slower than the full-order model of the 10th order NTL system. This result was expected because the of the nature of the GPM in the sense it adds a computational cost overhead which depends on the size of the included training data set. There is a certain combination of reduced model order and size of the meta-model training data at which the proposed meta-model will become faster in execution compared to the CI-NDS full order model, this will be the subject of chapter six. The meta-model solver's parameter similarity detection feature produced lower execution times in both test inputs because of the computational saving accomplished by not making redundant predictions for similar meta-model parameters (highlighted entries in Table 4-1) during simulation.

4.7 Conclusions

This chapter reviewed linear MOR projection methods and demonstrated their integration within the proposed GP blended VBL-LPV meta-model structure of chapter three. This chapter also reviewed some of nonlinear MOR methods implemented by others based on parametric blending and shown that our method provides a transparent local and global meta-model, exact representation of the system dynamics everywhere in the operating space and deals well with training data sparsity almost guaranteed to be a problem for parametric blending methods in high dimensions.

The amount of meta-model time-varying parameters that needs training depends on the linearization of the original full-order model in question, and most of those time-varying parameters are contained in the state-matrix of the LLM in the case of SISO models, which can be reduced by applying a suitable canonical method. However, when dealing with MIMO models, the amount of time-varying parameters in need of training will increase (through the additional time-varying parameters contained in the input and output matrices), therefore putting an additional computational cost to the training of the meta-model and when making predictions. This is the cost that must be paid if the goal was to provide a transparent meta-model, compared to nonlinear MOR methods with parametric blinding. There is the issue of the computational overhead associated with GPM predictions (it depends on the size of the training data set) which parametric blending methods does not suffer from, its justified since GPM ability to deal with sparse meta-model training data in higher dimensions better than parametric global function approximation methods, and they have the advantage of providing uncertainty information about their predictions which can be helpful when collecting CI-NDS training data (which involves multiple runs of the CI model itself) to select where additional training data are needed to improve the model accuracy.

The reduced-order meta-model of the 10th order NTL model has been successfully applied as an example. TBR-MOR method was applied to reduce the LLM order from ten to four and controllability canonical form was applied to reduce the number of reduced-order meta-model time varying parameters from sixteen to ten.

The 10th order NTL meta-model performance has been evaluated for two test inputs (one of them drove the model outside the range of the collected training data); both cases scored high model fitness percentages. The reduced-order meta-model solver had higher computational cost compared to the full order model which is expected because of the GPM compositional overhead.

The 10th NTL model served its purpose as an preliminary example to showcase the proposed reduced-order meta-modelling method, in chapter five, it will be applied to a 100th order CFD model in which LLM order is reduced using Krylov subspace MOR method and in chapter six it will be applied to a 5000th order NTL model in which LLM order is reduced using a combination of Krylov and TBR-MOR methods.

Chapter 5 - **Uncertainty Propagation**

5.1 Introduction

This chapter discusses the idea of uncertainty propagation in the GP blended reduced-order VBL-LPV meta-model (section 4.4). It is also the second contribution chapter in the thesis. The meta-model time-varying parameters are blended using GP models, and for each one of these parameters, uncertainty information (variance) is computed during the meta-model solver iteration. This chapter aims to answer the following questions:

- How local parameter uncertainties can propagates through the meta-model solver to reach its outputs? Can the variance be quantified in the meta-model outputs?

- What can the meta-model solution variances tell about the meta-model solution confidence? Can it help to select better training data for the meta-model to improve the accuracy and confidence of predictions?

To answer the above, the meta-model solver equations (chapter three) will be re-examined and an analytical derivation of the meta-model solution variances will be presented.

The resultant meta-model with uncertainty propagation will be tested on a second-order MIMO NDS of nonlinear chemical reactor as a meta-modelling case without MOR, then on a 100th order SISO fluid dynamics model with MOR.

5.2 The Meta-Model Solver Uncertainty Propagation

The solution to the discrete time version of the proposed meta-model is given by (section 3.4).

$$\begin{aligned}\boldsymbol{w}(k+1) &= \boldsymbol{A}_d(k)\boldsymbol{w}(k) + \boldsymbol{B}_d(k)\dot{\boldsymbol{u}}(k)\\ \boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + h\boldsymbol{w}(k+1)\\ \boldsymbol{y}(k) &= \boldsymbol{C}(k)\boldsymbol{x}(k) + \boldsymbol{D}(k)\boldsymbol{u}(k)\\ \boldsymbol{w}(0) &= \boldsymbol{w}_0, \boldsymbol{x}(0) = \boldsymbol{x}_0\end{aligned} \tag{5.1}$$

The parameters of the meta-model velocity and output solutions in equation (5.1) will have their non-constant entries modelled by their respective GPMs to be normally distributed random variables specified by their statistical first (their means) and second moments (their variances). The goal is to quantify this uncertainty in terms of the statistical second moments in the meta-model velocity, state and output equations. There is a need to quantify the uncertainty at every time step $t = kh$ taken by the solver and to identify what uncertainty propagates to the subsequent time step $(k+1)h$.

At each time step kh, the meta-model will provide a prediction of the parameters for the VBL-LLMs at the current operating point of the system (given by the mean value of the current state).

The GPMs predictions of the non-constant entries in the parameters for the VBL-LLMs are normally distributed variables given by

$$\begin{aligned}A_{ij} &\sim \mathcal{N}\left(\mu_{A_{ij}}, \sigma^2_{A_{ij}}\right), B_{ij} \sim \mathcal{N}\left(\mu_{B_{ij}}, \sigma^2_{B_{ij}}\right)\\ C_{ij} &\sim \mathcal{N}\left(\mu_{C_{ij}}, \sigma^2_{C_{ij}}\right), D_{ij} \sim \mathcal{N}\left(\mu_{D_{ij}}, \sigma^2_{D_{ij}}\right)\end{aligned} \tag{5.2}$$

Therefore, the computed VBL-LLM parameter matrices at any discrete time step k will have elements as a combination of deterministic (constant entries) and normally distributed random variables.

The computed $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ paramters at any time step k are of the continues time version of the VBL-LLMs and therefore the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrcies will have an equlivent sampled versions of them $\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ which in turn depends on the method the discrete meta-model velocity equation is being solved (section 3.4.1).

$\boldsymbol{A}_d$ and $\boldsymbol{B}_d$ matrices obtained by the Exact-Discretization method used in section 3.4.1 are given as

$$\begin{aligned} \boldsymbol{A}_d(k) &= e^{\boldsymbol{A}(k)h} \\ \boldsymbol{B}_d(k) &= \boldsymbol{A}(k)^{-1}\left(e^{\boldsymbol{A}(k)h} - \boldsymbol{I}\right)\boldsymbol{B}(k) \end{aligned} \tag{5.3}$$

The matrix exponential in equation (5.3) can be approximated in terms of the power series given by

$$e^{\boldsymbol{A}(k)} = \sum_{j=0}^{\infty} \frac{1}{j!}\left(\boldsymbol{A}(k)\right)^j \tag{5.4}$$

The random entries of the non-constant elements of the $\boldsymbol{A}$ matrix will have their probability distribution function change to something that is difficult to quantify and express analytically if computations of matrix exponential are involved.

The research suggests to approximate matrix exponential in equation (5.3) with something more analytically tractable expressed by the first two terms in the power series equation (5.4). The approximation is given by

$$e^{\boldsymbol{A}(k)} \cong \left(\boldsymbol{I} + \boldsymbol{A}(k)\right) \tag{5.5}$$

Substituting the matrix exponential approximation of equation (5.5) in equation (5.3) will result in approximate discrete time parameters given by equation (5.6).

$$\begin{aligned} \boldsymbol{A}_d(k) &= (\boldsymbol{I} + \boldsymbol{A}(k)h) \\ \boldsymbol{B}_d(k) &= h\boldsymbol{B}(k) \end{aligned} \tag{5.6}$$

The simplified discrete time meta-model solution in equation (5.1) is now given by

$$\begin{aligned} \boldsymbol{w}(k+1) &= \boldsymbol{w}(k) + h\boldsymbol{A}(k)\boldsymbol{w}(k) + h\boldsymbol{B}(k)\dot{\boldsymbol{u}}(k) \\ \boldsymbol{x}(k+1) &= \boldsymbol{x}(k) + h\boldsymbol{w}(k+1) \\ \boldsymbol{y}(k) &= \boldsymbol{C}(k)\boldsymbol{x}(k) + \boldsymbol{D}(k)\boldsymbol{u}(k) \\ \boldsymbol{w}(0) &= \boldsymbol{w}_0, \boldsymbol{x}(0) = \boldsymbol{x}_0 \end{aligned} \tag{5.7}$$

The approximated velocity solution in equation (5.7) is the solution velocity estimation method of Forward-Euler previously discussed in section 3.4.1.

In the following derivation, we will assume statistical independence among all the GPMs predicted random non-constant entries of the parameters of the VBL-LLMs.

For the real valued function $f_R(.)$ of two independent random variables R_1, $R_2 \in \mathbb{R}$ with means μ_{R_1} and μ_{R_2} and variances $\sigma_{R_1}^2$ and $\sigma_{R_2}^2$ the following set of variance identities holds true

$$\begin{aligned} f_R(R_1) &= aR_1 \rightarrow \sigma_{f_R}^2 = a^2\sigma_{R_1}^2, a \in \mathbb{R} \\ f_R(R_1, R_2) &= aR_1 + bR_2 \rightarrow \sigma_{f_R}^2 = a^2\sigma_{R_1}^2 + b^2\sigma_{R_2}^2, a, b \in \mathbb{R} \\ f_R(R_1, R_2) &= R_1R_2 \rightarrow \sigma_{f_R}^2 = \sigma_{R_1}^2\sigma_{R_2}^2 + \sigma_{R_1}^2\mu_{R_2}^2 + \sigma_{R_2}^2\mu_{R_1}^2 \end{aligned} \tag{5.8}$$

We will make use of the above variance estimation identities to compute the uncertainty in the meta-model solver.

At discrete time $t = (0h) = 0$; the meta-models solution is given by

$$\begin{aligned} \boldsymbol{w}(1) &= \boldsymbol{w}_0 + h\boldsymbol{A}(0)\boldsymbol{w}_0 + h\boldsymbol{B}(0)\dot{\boldsymbol{u}}(0) \\ \boldsymbol{x}(1) &= \boldsymbol{x}_0 + h\boldsymbol{w}(1) \\ \boldsymbol{y}(0) &= \boldsymbol{C}(0)\boldsymbol{x}_0 + \boldsymbol{D}(0)\boldsymbol{u}(0) \end{aligned} \tag{5.9}$$

The second moments of equation (5.9) by random numbers algebra written with slight abuse of notation (all the vector/matrix square operation are on their individual elements) are given by

$$\begin{aligned} \boldsymbol{\sigma}^2_{\boldsymbol{w}(1)} &= h^2\left(\boldsymbol{\sigma}^2_{\boldsymbol{A}(0)}\boldsymbol{w}_{\boldsymbol{0}}{}^2 + \boldsymbol{\sigma}^2_{\boldsymbol{B}(0)}\dot{\boldsymbol{u}}(0)^2\right) \\ \boldsymbol{\sigma}^2_{\boldsymbol{x}(1)} &= h^2\boldsymbol{\sigma}^2_{\boldsymbol{w}(1)} \\ \boldsymbol{\sigma}^2_{\boldsymbol{y}(0)} &= \boldsymbol{\sigma}^2_{\boldsymbol{C}(0)}\boldsymbol{x}_0{}^2 + \boldsymbol{\sigma}^2_{\boldsymbol{D}(0)}\boldsymbol{u}(0)^2 \end{aligned} \tag{5.10}$$

$\boldsymbol{\sigma}^2_{(.)}$ is a vector or matrix of the variance of the random entries in the predicted velocity, state and solution vectors computed in terms of the variances of the normally distributed non-constant entries of the VBL-LLMs parameter matrices. The variance for the deterministic part is simply zero. The variance enters equation (5.10) through the GPMs predicted meta-model parameters variance in the velocity and output parts. At this step, the probability distribution of the velocity, state and output is Gaussian.

When the solver progress to the discrete time step $t = 1h$, the newly estimated meta-model parameters will be a function of the current deterministic operating point of the system $\{\boldsymbol{\mu}_{\boldsymbol{x}(1)}, \boldsymbol{u}(1)\}$. The meta-model solution at this time step is given by

$$\begin{aligned} \boldsymbol{w}(2) &= \boldsymbol{w}(1) + h\boldsymbol{A}(1)\boldsymbol{w}(1) + h\boldsymbol{B}(1)\dot{\boldsymbol{u}}(1) \\ \boldsymbol{x}(2) &= \boldsymbol{x}(1) + h\boldsymbol{w}(2) \\ \boldsymbol{y}(1) &= \boldsymbol{C}(1)\boldsymbol{x}(1) + \boldsymbol{D}(1)\boldsymbol{u}(1) \end{aligned} \tag{5.11}$$

And the second moments of the meta-model solution at this time step is given by

$$
\begin{aligned}
\boldsymbol{\sigma}^2_{\boldsymbol{w}(2)} &= \boldsymbol{\sigma}^2_{\boldsymbol{w}(1)} + h^2\left(\boldsymbol{\sigma}^2_{\boldsymbol{A}(1)}{\boldsymbol{\mu}_{\boldsymbol{w}(1)}}^2 + \boldsymbol{\sigma}^2_{\boldsymbol{B}(1)}\dot{\boldsymbol{u}}(1)^2\right) \\
\boldsymbol{\sigma}^2_{\boldsymbol{x}(2)} &= \boldsymbol{\sigma}^2_{\boldsymbol{x}(1)} + h^2\boldsymbol{\sigma}^2_{\boldsymbol{w}(2)} \\
\boldsymbol{\sigma}^2_{\boldsymbol{y}(1)} &= \boldsymbol{\sigma}^2_{\boldsymbol{C}(1)}\boldsymbol{\sigma}^2_{\boldsymbol{x}(1)} + \boldsymbol{\sigma}^2_{\boldsymbol{C}(1)}{\boldsymbol{\mu}_{\boldsymbol{x}(1)}}^2 + \boldsymbol{\sigma}^2_{\boldsymbol{x}(1)}{\boldsymbol{\mu}_{\boldsymbol{C}(1)}}^2 \\
&\quad + \boldsymbol{\sigma}^2_{\boldsymbol{D}(1)}\boldsymbol{u}(1)^2
\end{aligned}
\tag{5.12}
$$

For the velocity and state parts of the solution at this time step, the variance is the sum of the velocity and state variances from the previous time step added to the current time step variances. Since probability distributions of the previous time step velocity and state predictions are Gaussian, and the added uncertainty probability distributions are also Gaussians, then the probability distribution of the velocity and state solution at this time step will also be Gaussian.

The variance of the output part in equation (5.11) is the variance of the product of two random normally distributed quantities (the output matrix multiplied by the state vector) added to the variance incurred by the normally distributed input feed-through matrix of the system). The product of two random Gaussians is a Gaussian distribution scaled by a Gaussian shaped function but is not a valid probability distribution (Bromiley, 2013) i.e. the integration of the of the resulted probability distribution function for this product from minus infinity to infinity is not one. The meta-model output part of equation (5.11) has no known probability distribution but its variance can still be estimated using the identities in equation (5.8). In the special case where the meta-model output matrix is deterministic, the meta-model output will be a normally distributed random variable.

Progressing further in the meta-model solver time and according to the above reasoning, the first moment can be computed by applying the expected value operator $\mathbb{E}(.)$ (the mean) for the meta-model solver equation (5.7) at any time step $t = kh$ is

$$\begin{aligned}
\boldsymbol{\mu}_{\boldsymbol{w}(k+1)} &= \boldsymbol{\mu}_{\boldsymbol{w}(k)} + h\boldsymbol{\mu}_{\boldsymbol{A}(k)}\boldsymbol{\mu}_{\boldsymbol{w}(k)} + h\boldsymbol{\mu}_{\boldsymbol{B}(k)}\dot{\boldsymbol{u}}(k) \\
\boldsymbol{\mu}_{\boldsymbol{x}(k+1)} &= \boldsymbol{\mu}_{\boldsymbol{x}(k)} + h\boldsymbol{\mu}_{\boldsymbol{w}(k+1)} \\
\boldsymbol{\mu}_{\boldsymbol{y}(k)} &= \boldsymbol{\mu}_{\boldsymbol{C}(k)}\boldsymbol{\mu}_{\boldsymbol{x}(k)} + \boldsymbol{\mu}_{\boldsymbol{D}(k)}\boldsymbol{u}(k)
\end{aligned} \tag{5.13}$$

And the second moment at any time step $t = kh$ is

$$\begin{aligned}
\boldsymbol{\sigma}^2_{\boldsymbol{w}(k+1)} &= \boldsymbol{\sigma}^2_{\boldsymbol{w}(k)} + h^2\left(\boldsymbol{\sigma}^2_{\boldsymbol{A}(k)}{\boldsymbol{\mu}_{\boldsymbol{w}(k)}}^2 + \boldsymbol{\sigma}^2_{\boldsymbol{B}(k)}\dot{\boldsymbol{u}}(k)^2\right) \\
\boldsymbol{\sigma}^2_{\boldsymbol{x}(k+1)} &= \boldsymbol{\sigma}^2_{\boldsymbol{x}(k)} + h^2\boldsymbol{\sigma}^2_{\boldsymbol{w}(k+1)} \\
\boldsymbol{\sigma}^2_{\boldsymbol{y}(k)} &= \boldsymbol{\sigma}^2_{\boldsymbol{C}(k)}\boldsymbol{\sigma}^2_{\boldsymbol{x}(k)} + \boldsymbol{\sigma}^2_{\boldsymbol{C}(k)}{\boldsymbol{\mu}_{\boldsymbol{x}(k)}}^2 + {\boldsymbol{\mu}_{\boldsymbol{C}(k)}}^2\boldsymbol{\sigma}^2_{\boldsymbol{x}(k)} \\
&\quad + \boldsymbol{\sigma}^2_{\boldsymbol{D}(k)}\boldsymbol{u}(k)^2
\end{aligned} \tag{5.14}$$

Examining equations (5.13) and (5.14) for the meta-mode solution derived first and second moments; the following important points are observed:

- The expected value of the meta-model solution in equation (5.13) is simply the deterministic version of the original meta-model solver in equation (5.7).
- The variance of the meta-model solver in equation (5.14) will monotonically increase as the solver progresses in time due to all variance contributions by the past time samples resulting in an expanding cone of uncertainty around the meta-model solution.
- The local uncertainty contributions at any time step to the velocity and state parts of the meta-model solution are heavily scaled down by a factor of the square of the fixed time step h. This reflects that the uncertainty about the velocity and state parts of the solution decreases with $O(h^2)$ as the meta-model solver make smaller jumps in time which matches the local truncation error expected by using the Forward-Euler method in computing velocity and state

solutions previously described in section 3.4. This heavy scale down of local uncertainty estimation in the meta-model velocity and state solutions is simply saying that local uncertainty in the velocity solution parameters does not contribute much to the global uncertainty of the solution and the final deciding factor in the meta-model accuracy is the time step h. In a setting where the meta-model solution is computed using variable step size solver; the impact of local uncertainty estimation of the velocity and state solutions will be further sensed as the time step increase or decrease in size.

- The global uncertainty of the meta-model solution output is a combination of the weak local uncertainty of the state solution and the strong uncertainty of the meta-model output and feed-through matrices. In the case the meta-model solution had deterministic output (and feed-through) matrices (which is the case in any meta-model with full order parameters), the global uncertainty estimation of the meta-model solution will not reveal much about the uncertainty especially for small solver time steps. In the case the meta-model had reduced order parameters, the local contribution by the output matrix uncertainty will have the greatest impact on the global uncertainty of the meta-model output solution.

- The assumption of statistical independence between the meta-model solutions computed at any two subsequent time steps stems from the fact that the GPMs predictions of the meta-model parameters are functions of the mean of the state solution $\boldsymbol{\mu}_{\boldsymbol{x}(k)}$ therefore transferring no uncertainty information to the GPMs regression inputs.

- The derived first and second moments of the meta-model solution are valid for Forward-Euler meta-model solver; however, they can only be considered as an approximation when used with other meta-model velocity solvers (section 3.4). While it is difficult to quantify the first and second moments for the Exact-Discretization method as we have demonstrated earlier in this chapter, similar reasoning can be applied to the proposed other numerical methods of the meta-model velocity solution (section 3.4.1) given by Heuns and RK4 methods with the expectation that these two methods will result in lower value of variance for the meta-model velocity solution and by extension to the meta-model state and output solutions.

Despite the heavy scale down of uncertainty in the velocity and state parts of the meta-model solution, the derived global uncertainty propagation measure of equation (5.14) can still play a significant rule in relating meta-model uncertainties in the output solution contributed by the meta-model output matrix (and the input feed-through matrix) uncertainties in the case of reduced order meta-models which sees this matrix to be fully parameterized by GPMs predictions.

The uncertainty propagation in general can serve as a starting point to analyse the meta-model solution but more in depth picture is best to be observed in the individual uncertainties of the meta-model parameters.

The uncertainty in GPMs of meta-model parameters do not necessarily give a statement about the correctness/incorrectness of the meta-model solution, but are more about the confidence of the meta-model in the predictions based on the prior knowledge provided in terms of the selected meta-model training data. In simple words, the meta-model might still be confident in its predictions even if it has provided the wrong solution to the original model.

5.3 Meta-Modelling of a Chemical Reactor

The (Non-Adiabatic) Continuously-Stirred Tank Reactor (CSTR) model (Bequette and Bequette, 1998) (also part of MTALAB system identification tool box) is a second-order nonlinear dynamical system model of a chemical reactor, the schematic of the reactor is given by Figure 5-1.

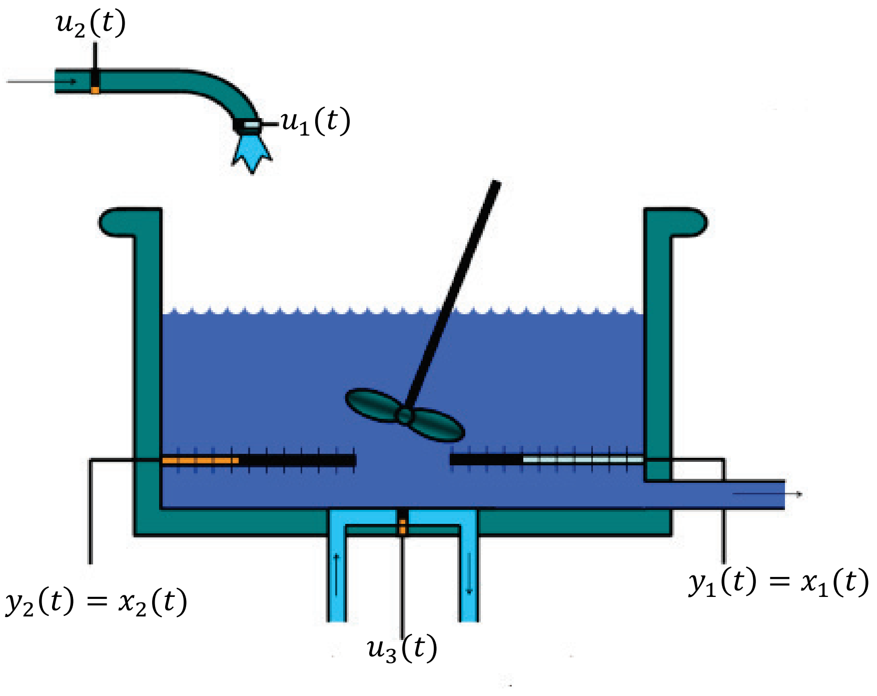


Figure 5-1: Schematic of the (Non-Adiabatic) Continuously-Stirred Tank Reactor

The chemical reactor in Figure 5-1mixes incoming stream with the contents, the vessel of the reactor is thermally isolated and cooled with coolant, inputs of the model are the sensors readings given in terms of incoming feed concentration $u_1(t)$ with units of $(kgmol/m^3)$, incoming stream temperature $u_2(t)$ and reactor vessel coolant temperature $u_3(t)$ both in degree Kelvins. During the reaction, the incoming stream concentration $u_1(t)$ and its temperature $u_2(t)$ are kept constant at $10 kgmol/m^3$ and $298K$. The reactor coolant temperature is varied between $273K$ and $325K$.

After the initial mixing of the chemical reactor contents, their volume is kept constant throughout the reaction by allowing the reactor output stream flow rate to be equal to the flow rate of incoming stream. The chemical reactor model outputs are the sensors readings for the incoming feed concentration inside the reactor vessel $y_1(t) = x_1(t)$ and the reactor contents temperature $y_2(t) = x_2(t)$. The nonlinear model of the chemical reactor is given by appendix A.3, for more information consult the cited reference at the beginning of this section. The CSTR model initial conditions are given by $\boldsymbol{x}_0 = [8.5695, 311.267]^T$.

The CSTR meta-model equilibrium training data were analytically estimated (appendix A.3) using model input values of $u_{e1} = 10, u_{e2} = 298$ and u_{e3} uniformly sampled between 273 and 325 at 52 points, this will result in 52 meta-model equilibrium training points.

The CSTR meta-model off-equilibrium points were collected on the trajectory of the system response to a randomly generated Gaussian signal (Figure 5-2) (299 mean and 26 standard deviation) applied to $u_3(t)$, $u_1(t)$ and $u_2(t)$ are both constants at 10 and 325. The generated random sequence is sampled every $0.1 hour$.

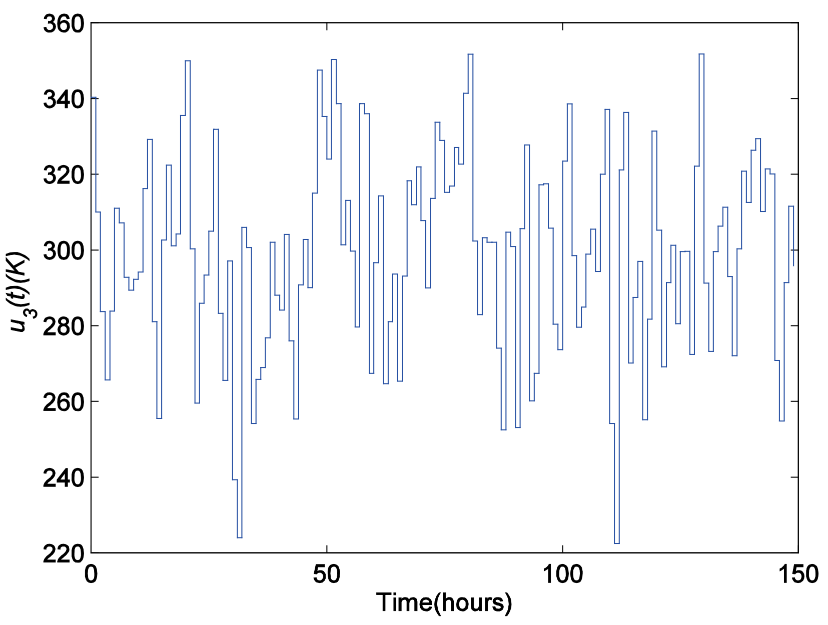


Figure 5-2: Random Gaussian signal applied to the CSTR model at the reactor coolant temperature input $u_3(t)$ to collect meta-model off-equilibrium training data

Exciting the CSTR model with the signal contained in Figure 5-2 will generate 544 meta-model off-equilibrium training data points. The meta-model training time-varying parameters were analytically computed (appendix A.3) using the collected meta-model training data points in equilibrium and off-equilibrium regions (596 points), a small zero-mean Gaussian noise (1×10^{-5}) was added. The resultant meta-model training data were randomly shuffled and equally-split into two disjoint training and validation sets.

A squared exponential with ARD covariance function has been selected to for each of the GPMs of the four time-varying parameters in the $\boldsymbol{A}$ matrix Jacobian of the CSTR model (appendix A.3). The results of the CSTR meta-model training are given in Table 5-1.

Table 5-1: CSTR meta-model training results

$GPM(\boldsymbol{A})$	$SMSE$	$\%fitness$
a_{11}	2.56×10^{-6}	99.84
a_{12}	6.23×10^{-4}	97.5
a_{21}	1.84×10^{-8}	99.99
a_{22}	4.19×10^{-6}	99.8

Table 5-1 shows that the CTSR meta-model parameters achieved satisfactory model fitness when verified against the collected validation data set, $SMSE$ values are given for reference only, and they are useful when training the meta-model using multi-partition cross-validation (more than two partitions), relatively lower $SMSE$ values means better fit.

The CSTR meta-model was tested using a real world example input to the model taken from the CSTR model documentation in MATLAB system identification toolbox. The inputs are discontinues, therefore they were approximated using a sigmoid like function (appendix B). The CSTR three approximated test input components are given by Figure 5-3, Figure 5-4 and Figure 5-5. All test input components were resampled at $h = 0.01hour$ to match the meta-model solver fixed-time step.

The main control input of the CSTR model is $u_3(t)$ *the reactor vessel coolant temperature) given by Figure 5-5, the other two input components (Figure 5-3and Figure 5-4) have little variation around their nominal levels during this experiment.

The CSTR meta-model response to the above test input was computed using Exact-Discretization velocity solver, the CSTR model response was computed using RK4 fixed-time step solver, the CSTR model outputs were plotted against those of the meta-model in Figure 5-6 and Figure 5-7.

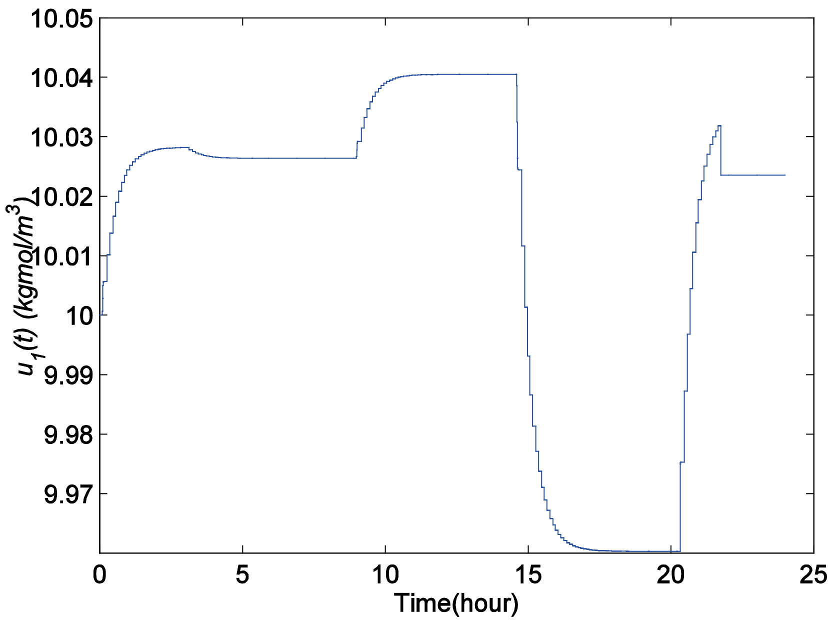


Figure 5-3: CSTR meta-model $u_1(t)$ test input component (incoming feed concentration)

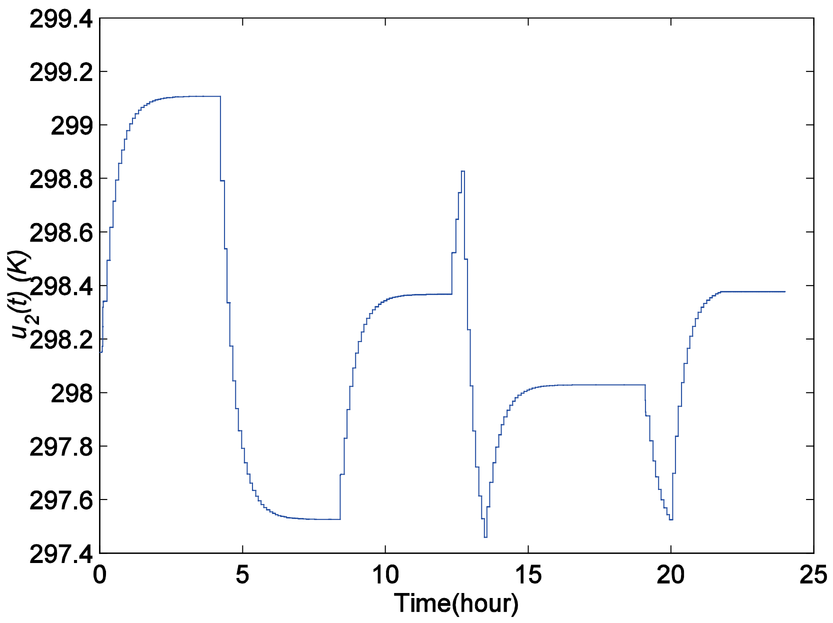


Figure 5-4: CSTR meta-model $u_2(t)$ test input component (incoming feed temperature)

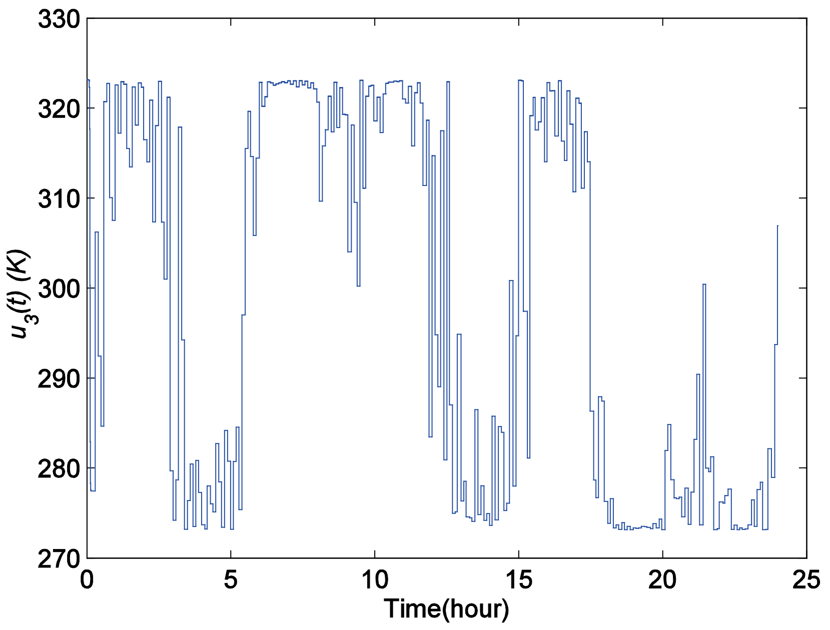


Figure 5-5: CSTR meta-model $u_3(t)$ test input component (reactor vessel coolant temperature). This is the main control input during the CSTR meta-model simulation

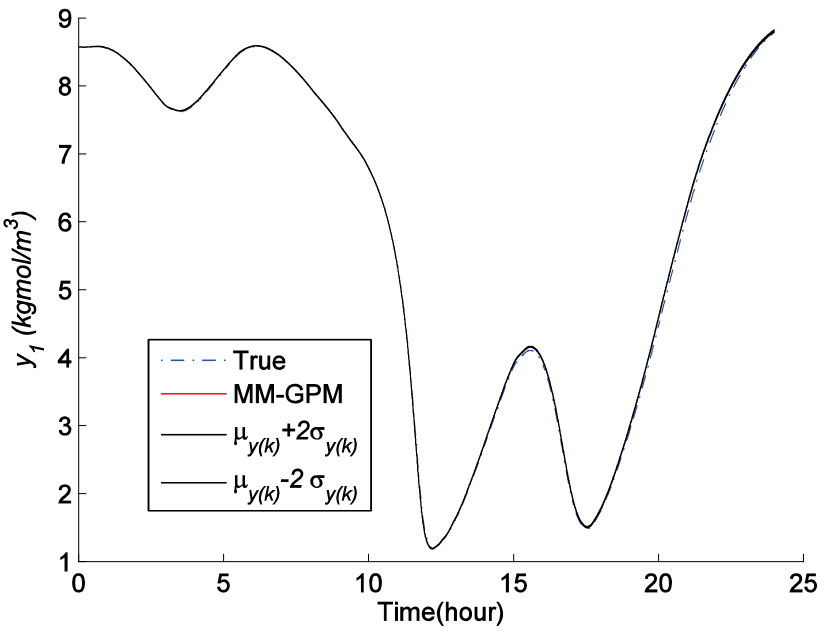


Figure 5-6: CSTR meta-model output $y_1(t)$ (concentration of incoming feed inside the reactor)

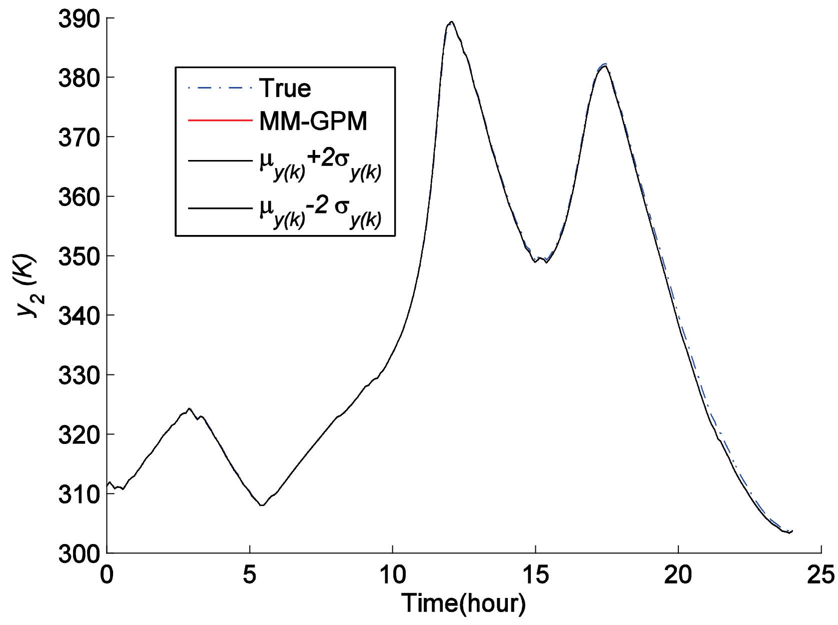
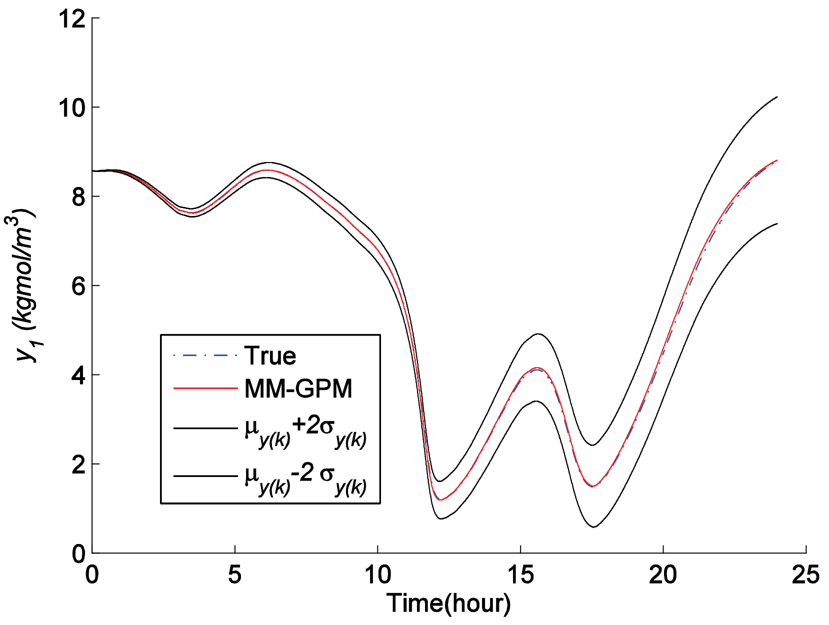


Figure 5-7: CSTR meta-model output $y_2(t)$ (reactor contents temperature)

The CSTR meta-model did well to describe the original model with model fitness of 98.07% for y_1 and 97.45% for y_2.Uncertainty propagation was computed using equation (5.14) and the two standard deviations confidence regions of the meta-model simulation were plotted around the meta-model response in Figure 5-6 and Figure 5-7. The confidence regions are extremely narrow and practically indistinguishable from the meta-model response plot (unless by zooming in the y-axis of the plots), this is expected due to the heavy scale down of the meta-model velocity and state solutions local uncertainties (equation (5.14)) by a factor of $O(h^2)$.

A qualitative view of the meta-model velocity and state solution uncertainties can be obtained if the solver time step h was set to unity in equation (5.14), therefore rendering the local uncertainty of the meta-model solutions independent of the solver time step. A plot of CSTR meta-model responses with scaled uncertainties versus the CSTR model RK4 responses are given by Figure 5-8.



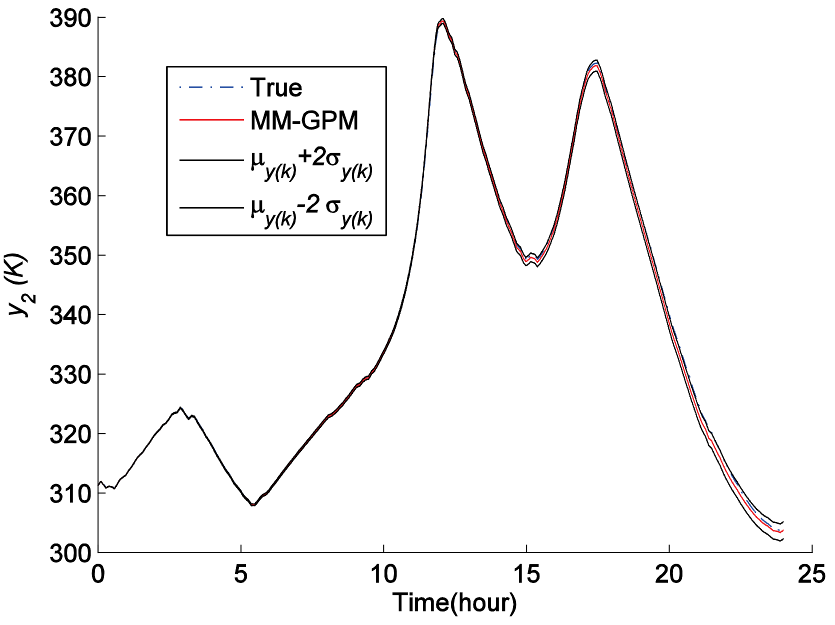


Figure 5-8: CSTR meta-model repsonses with qualitivate view of uncertainty propgation in the meta-model

The qualitative view of the meta-model solution uncertainty in Figure 5-8 is more apparent than the quantitative view of uncertainty in Figure 5-7. The meta-model output in Figure 5-8 is contained inside the qualitative confidence regions therefore indicating the trained meta-model is a good approximation to the underlying CSTR model, this is supported by the obtained $\%fitness$ for meta-model outputs. The computed meta-model outputs qualitative uncertainties in Figure 5-8 are characterized by expanding cones as the meta-model solver progresses in time; this is due to local uncertainty propagation adding up in each solver time step, the same observation can be made on a smaller scale in Figure 5-7 due to the heavy scale down of the meta-model quantitative uncertainties sustained by the solver time step.

To demonstrate how uncertainty propagation information can be used to improve the meta-model predictions, the CSTR meta-model was trained with a smaller training data set of 62 points randomly selected from the training data set obtained earlier in this section. The meta-model training follows the same procedure describe earlier in this section just with smaller training data set (as opposed to 298 points in the first case). The meta-model response to the test inputs in Figure 5-3, Figure 5-4 and Figure 5-5, achieved model faintness of 93.73% for $y_1(t)$ and 92.1%, lower than what was achieved when bigger size training data set was used.

The time-varying $\%RSD$ for this meta-model outputs was computed twice using the quantitative uncertainty and the qualitative uncertainty and is shown in Figure 5-9. The plot show the effect of the meta-model solver time step greatly influencing the magnitude of the quantitative uncertainty propagation, therefore giving false confidence in the meta-model predictions, using the qualitative uncertainty propagation to compute the time-varying $\%RSD$ removes the effects of the solver time step, therefore gives better view of the underlying uncertainty in the meta-model predictions. Uncertainty around meta-model output $y_1(t)$ peaks at solver time $12.41 hour$ and $17.71 hour$ (circled in Figure 5-9), while uncertainty around meta-model output $y_2(t)$ is relatively much lower throughout the meta-model solution.

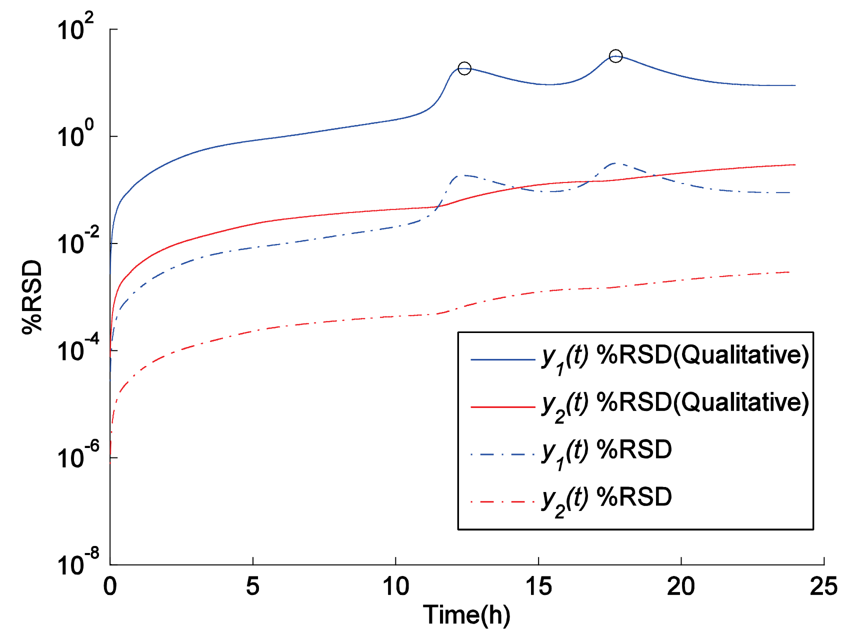


Figure 5-9: Time-varying $\%RSD$ for CSTR meta-model (trained with 62 training points) outputs, the dashed lines shows $\%RSD$ values computed using quantitative uncertainty propagation and the solid lines represent $\%RSD$ values computed using qualitative uncertainty propagation values

Those two peaks in uncertainty around meta-model output $y_1(t)$ can be used to improve the meta-model predictions by constraining the GPM of the meta-model parameters to have a prior knowledge of these operating points. A single operating point can be added just before the one where the peaking in global uncertainty occurs or add a number of them leading and/or trailing this point (more is better until no useful improvement in meta-model predictions is detected). The newly added meta-model training points are obtained from the real model response around the location the uncertainty peak in time. In this case of the CSTR meta-model, 10 continuous operating points time indices was selected before and after each uncertainty peak, then augmented with the meta-model training data set (now consists of 104 training points), these state-space points were used to compute the meta-model parameters using the Jacobians of the CSTR model (appendix A.3), then the meta-model was retrained and the solution obtained again. This procedure improved the meta-model $\%fitness$ to 97.44% for $y_1(t)$ and 96.89% for $y_2(t)$. Qualitative time-varying $\%RSD$ for both meta-model training cases (without/with additional training data) was computed and plotted in Figure 5-10.

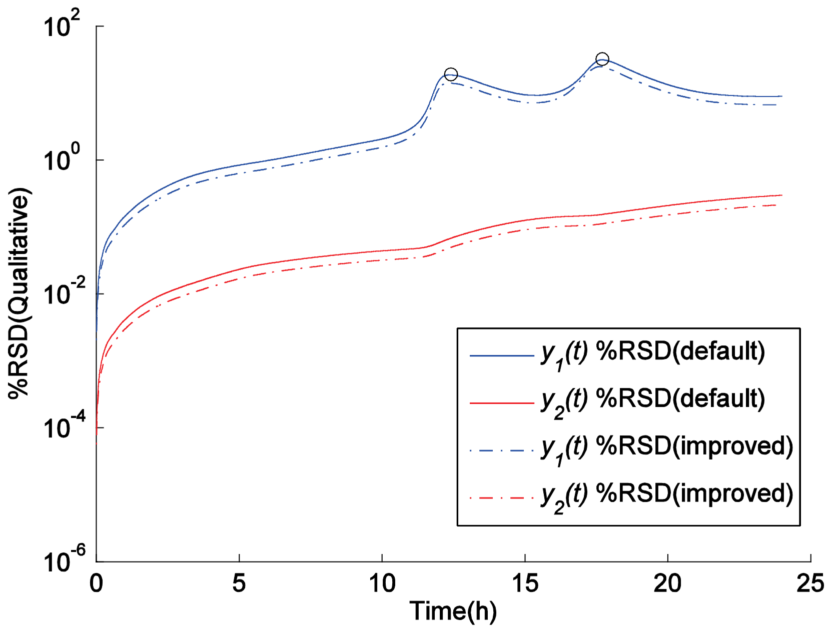


Figure 5-10: Qualitative time-varying $\%RSD$ for both meta-model training cases without (default) and with additional training data (improved)

Figure 5-10 shows an overall improvement in the meta-model qualitative time-varying $\%RSD$ after adding more training points to the meta-model training data, the meta-model uncertainty improvement is globally sensed across the meta-model solution.

Therefore, it has been demonstrated that uncertainty propagation information can be used to improve the quality of meta-model predictions in this example, the results here are directly applicable to meta-models of higher orders and to those with reduced order LLMs. The time-varying $\%RSD$ for individual meta-model parameters can also be used to improve meta-model predictions in similar manner, however, for high order meta-models with hundred or even thousands parameters, this task becomes intractable and this is where having a global uncertainty measure coupled with the meta-model outputs is advantageous.

5.4 Meta-Modelling of the 1-D Burgers Equations

The 1-D Burgers equations (Rewienski, 2003) are nonlinear partial differential equations describing the movement of shock in a fluid or a medium. The 1-D Burgers equations are an example of nonlinear boundary value problem, the exciting input is part of the boundary of the problem. Spatial discretization of these equations in one dimension will result a set of nonlinear ordinary differential equations (appendix A.5). The default fixed-time step for the equations solver is $0.1s$. This system was considered to be a numerical challenge, because it has high speed dynamics and requires extended computational time in high dimensions. This section will consider the meta-modelling of the 100th order 1-D Burgers equations. The reduced-order meta-model of this system will result in a parameterized output matrix in which the effects of the uncertainty propagation can be examined. Each state in the full order 1-D Burgers equations is a prospective output of the system, for sake of simplicity, only one output of the system at node 5 was considered, higher output nodes will require higher order meta-model and therefore will require to have more meta-model parameters. The considered system is a mathematical abstract and therefore does not have any units assigned to its inputs, states or outputs in its present form.

The Meta-model training points were collected along the trajectory of the system starting from the initial condition $(x_1 \dots x_{100} = 1)$ to a constant simulation input $u(t) = \sqrt{5}$ sampled every $h = 0.1s$ with simulation time of $50s$, this amount of simulation time allows the system to enter equilibrium state therefore provides meta-model training points that cover both transient and equilibrium states of the system. 501 meta-model training points were collected and full order LLMs were constructed using the Jacobians of the 1-D Burgers equations (appendix A.5).

TBR MOR was applied to the collected full-order LLMs; the balanced realizations of the LLMs were computed at first and the mean percentages of states contribution to the balanced LLMs were computed with the aid of the balanced LLMs HSVs, equation (4.17) and are given by Figure 5-11.

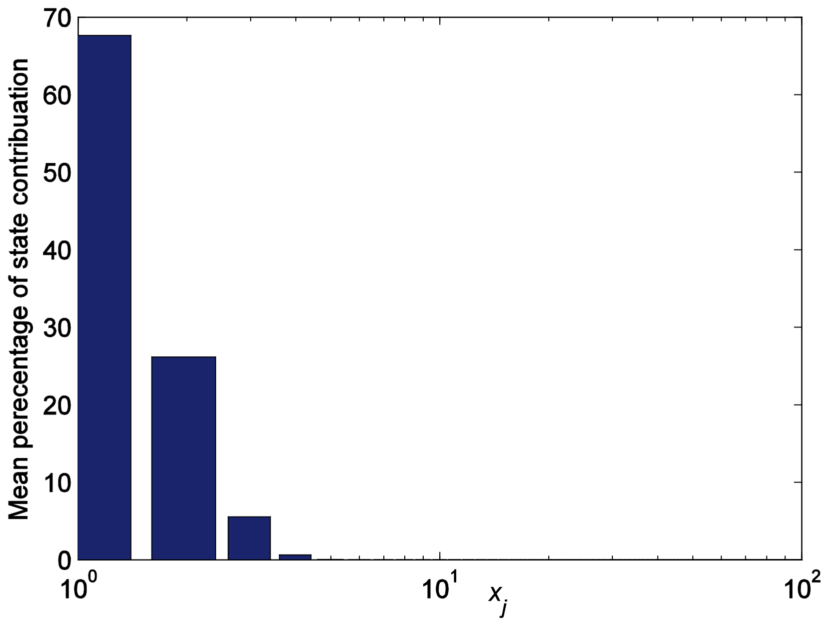


Figure 5-11: Mean percentage of state contributions of the 100th order 1-D Burgers Equations system balanced training LLMs

Figure 5-11 was used to make an informed decision on the order of the reduction for the LLMs by observing that states contribution to the output energy falls dramatically after the 4th balanced state component, therefore the reduced order of LLMs was set to $q = 4$ in the case of the 100th order 1-D Burgers reduced-order meta-model.

Controllability canonical form (section 4.5) was applied to the reduced order LLMs therefore reducing the non-constant elements of $\boldsymbol{A}_r$ from sixteen to thirteen and input reduced-order vector $\boldsymbol{b}_r$ from four to one. The total number of reduced-order meta-model parameters that needs training is eighteen (thirteen for the state matrix, one for the input vector and four for the output vector).

A squared-exponential with ARD covariance function was chosen to train the reduced-order meta-model parameters. The chosen covariance function for the 100th order 1-D Burgers Equations NDS has a total of six parameters (one process parameter, one added noise parameter and four regression input parameters corresponding to the four states of the reduced order meta-model).

During the training of each GPMs, cross-validation was used to test the accuracy of the resulted GPM model by splitting the collected training points into two equal disjoint sets one for training and the other for validation. A very small (between 1×10^{-8} and 1×10^{-13}) zero-mean Gaussian noise was added to all training points' targets. The GPM covariance function hyper-parameters are trained by minimizing the cost function $\boldsymbol{J}(\boldsymbol{\theta})$ in equation (3.10). All trained meta-model parameters achieved model $\%fitness$ over 99% for the given validation data-set.

The reduced order meta-model was tested using a simulation input $u(t) = \sqrt{5}$ sampled every $h = 0.1s$ with simulation time of $50s$ and Exact-Discretization meta-model velocity solver (section 3.4.1). The results of the 4th order meta-model simulation is plotted against the result of the RK4 simulation of the 100th order 1-D Burgers equations in Figure 5-12.

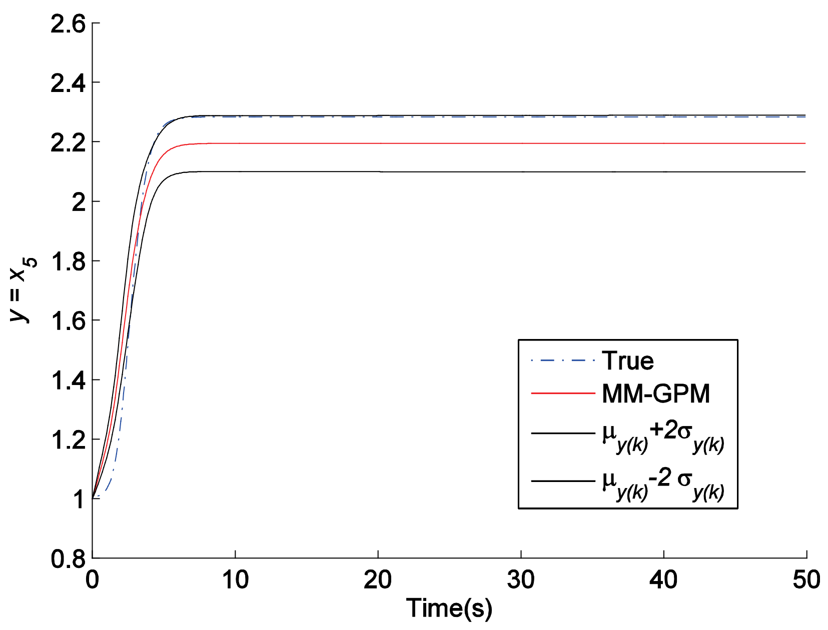


Figure 5-12: The RK4 solution of the 100th order 1-D Burgers model versus the 4th order meta-model solution with quantitative confidence intervals

Figure 5-12 shows a bad agreement ($\%fitness = 65\%$) between the 4th order meta-model solution and the 100th order 1-D Burgers equation solution. The reduced-order meta-model managed to capture the shape of the dynamics but failed to produce the correct magnitudes, this triggered the following diagnostic procedure to help understand what went wrong with this model:

- A full-order exact meta-model was solved using exact values of LLMs parameters obtained from original model Jacobians (appendix A.5), this produced almost matching response (over 99%) to the original NDS therefore proving the collected full-order LLMs data are correct and that the meta-model solver in full-order has no problem reproducing the correct system dynamics.

- A GP blended full-order meta-model was constructed using the same set of training data (only in full order) and the it did manage to produce almost matching response (over 99%) to the original model, this eliminated the possibility of a problem with the parameters GPM.

- Having done the above steps, there was clearly a problem with reduced order model itself, the results of the reduced-order meta-model training mentioned earlier leaves little doubt that the GPM failed to capture the underlying parameter function, so the only logical explanation is the parameters of the reduced-order meta-model were changing very fast for the VBL-LPV system to reproduce them over a single time step, the VBL-LPV system assumes that the time-varying parameter are constant over the time step during the solution (section 3.4) because of the implemented zero-order hold, this condition is met when using a full order meta-model of this particular system but became a problem when dealing with a reduced order version of the same system, decreasing the reduced-order meta-model solver time step has no effect on the quality of the meta-model predictions therefore affirms the above reasoning.

To demonstrate the fast changing dynamics of the reduced-order meta-model parameters, one of the time-varying parameters a_{21} was examined and plotted its value during the training and the solution of the meta-model (Figure 5-13), most of the reduced-order meta-model time varying parameters exhibits the same fast changing dynamics.

Figure 5-13 shows that at solver time zero, the meta-model produced the correct estimate for the parameter using the reduced-order canonical initial state of the system; this is true for the rest of the time-varying parameters of the meta-model. The problem occurred when the meta-model attempted a failed estimate for the next state at time

$0.1 second$ due to meta-model solver inability to track the fast change of model dynamics.

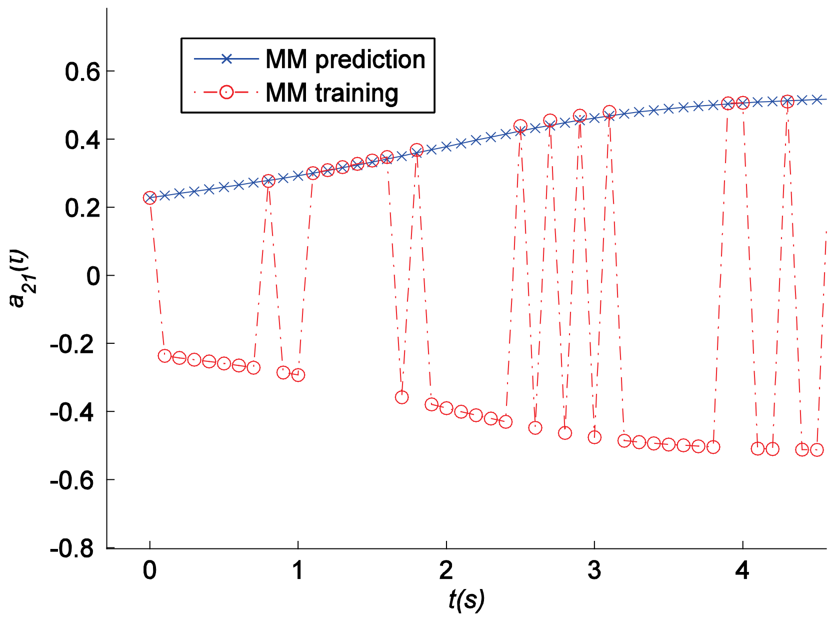


Figure 5-13: 100th order 1D-Burgers equation NDS time-varying parameter $a_{21}(t)$ part of the reduced-order canonical state matrix $\boldsymbol{A}_{rc}$, $a_{21}(t)$ was plotted for the first few seconds during the training and the solution of the meta-model

1D-Burgers equation NDS meta-modelling work by (Rewienski, 2003) using TPWL-MOR frame work (earlier described in section 4.3) managed to produce the correct dynamics of this system though their meta-models suffered in high orders of the system ($n = 1000$), others used POD-MOR method successfully (again low orders) to reproduced the correct dynamics with most recent example by (Jarvis, 2012). 1D-Burgers equation NDS is notorious when it comes to meta-modelling, the shock-wave dynamics are sensitive to initial perturbation of the system and might have been a tall order to meta-model this system using VBL-LPV system approach which if it had succeeded would have been a transparent reduced-order meta-model of this system.

The plot in Figure 5-12 also shows the meta-model quantitative confidence regions at two standard deviations around the mean of the model predictions. Most of the uncertainty in the meta-model output was contributed by the uncertainty of the solution vector $\boldsymbol{c}_{rc}$ with less contribution from the uncertainty in the meta-model velocity and state solutions due to the heavy scale down of the exact uncertainty by solver time step. There is no value in the acquired confidence regions in this case because the source of the meta-model uncertainty is characterized by model miss-specification.

5.5 Conclusions

Conclusions for the uncertainty propagation of the proposed GP blended VBL-LPV meta-model are summarized by the following:

1. Uncertainty propagation depends on the choice of the meta-model solver; for the fixed time step Forward-Euler solver, the solver time step has profound effect on the magnitudes of the uncertainty in the meta-model velocity and state solutions.

2. The meta-model variance can only detect regions where the GPMs of the system parameters fails to make a prediction. The rate of the expansion of the meta-model solution variance gives an indication to the correctness/incorrectness of the meta-model predicted solution.

3. Due to the sequential nature of any ODE solver, an error in a prediction at certain time step can affect the rest of the solution trajectory, and while the Gaussian Processes model faithfully provide accurate predictions at input spaces covered by its training points, it will provide the same faithful predictions for

the incorrect solution trajectory if it happens to pass on the same trained input spaces.

4. The information provided by the uncertainty propagation is a valuable tool when dealing with reduced order meta-model of complex NDS which generally have tens or even hundreds of GPMs for their VBL-LPV parameters. It conveys a lot about the meta-model solution and is a starting point to improve meta-model solution accuracy. This was demonstrated in the meta-modelling of the CSTR NDS.

5. Uncertainty propagation can be computed for the reduced-order meta-modelling of higher order NDS, this was demonstrated using the 100th order 1D-Burgers equation NDS. However, the proposed reduced-order meta-model did not manage to reproduce correct system dynamics due to the nature of the ZOH employed in the meta-model fixed-time step solver (not an issue for the full order meta-model).

Chapter 6 - Meta-Model Computational Complexity Analysis

6.1 Introduction

This chapter will analyse the computational time complexity analysis of the GP blended Full (or Reduced) Order VBL-LPV meta-model. Computational time complexity will be examined during the training phase of the meta-model, and later for the meta-model solver.

Big Oh asymptotic notation $\mathcal{O}(.)$ can be used to place an upper bound on how numerical algorithms scale with the increased size of input data and, it can be used to indicate asymptotic performance of the algorithm in terms of computational speed though this will mainly depend on the practical implementation of that algorithm. For real world performance of algorithms, benchmarking can be used to compare the run speed of two numerical algorithms, by taking a number of runtimes for each algorithm and averaging to get a comparative runtime between the two algorithms, however, this will be influenced by many factors such as code optimization and the available computational resources.

ODE solvers computational time complexity can be measured by either the number of function evaluations or the number of floating-point operations per second (flops) (Ilie et al., 2008), it varies greatly depending on the type of the solver (fixed-time step versus variable-time step) and the required accuracy (resolution in case of fixed-time step solver or error tolerance in case of variable-time step solver (Ilie et al., 2008)), the computational time complexity of a single function (the CI-NDS itself) evaluation will be part of the ODE solver computational time complexity.

The main source of computational time saving in the proposed meta-modelling approach is the dimensionality reduction of the problem and to a lesser extent by computational time savings obtained in evaluating linear (in the case of VBL-LPV system) set of equations against nonlinear ones. Regardless of the implemented ODE solver, the computational time complexity of a single evaluation of the CI-NDS set of ODEs is the baseline at which the computational time complexity of a single iteration of the reduced-order meta-model solver can be compared, both in terms of the problem order.

The computational time complexity of a single solver function evaluation for n order CI-NDS can be defined as a polynomial time complexity of $O(n^{\alpha})$ where α is a positive scalar. In the case of stiff CI-NDS, the variable-time step solver attempt advancing at smaller time scales in problematic regions of the solution with multiple evaluations of the NDS to establish the required solution accuracy, this greatly increases the computational time cost of the entire solution.

In the GP blended VBL-LPV meta-model solver, the number of flops in a single iteration of the meta-model solver can be computed and the order of computational time complexity can be evaluated, all meta-model calculations are vector-matrix arithmetic and their dimensionality can be expressed by a set of positive integers.

In the following sections, the computational time cost of the meta-model training phase will be discussed (section 6.2), followed by a discussion on the computational time cost for a single iteration of the meta-model solver (section 6.3), and finally a real world meta-modelling example of CI-NDS given by the NTL system of 5000 order will be examined along with benchmarking results of that system against different off shelf nonlinear ODE solvers.

6.2 The Meta-Model Training Cost

The meta-model training phase (section 3.3.4) can be divided into the following:

- Collection of Training data.
- Computing the LLMs of the associated training data.
- Training and validation of GPMs of the LLMs non-constant entries.

Each of the above sub-phases will be discussed in the following sections.

6.2.1 Collection of Training Data

As demonstrated over the course of the previous chapters, training data for the meta-model plays a significant role in the final success of making informed predictions.

In order to provide reliable predictions, training data (set of inputs, states and corresponding outputs) must be computed along the required CI-NDS trajectory. Training data should include steady state and transient behaviour of the complex system, both require evaluating the CI-NDS itself and one of the main questions that arise is how many training data are enough to do the job? There is no definitive answer to this, but one can hope to choose the minimum number of data points with the maximum variance to cover important parts of the solution trajectory.

The computational time complexity when collecting the meta-model training points will depend on the CI-NDS computational time complexity $\mathcal{O}(n^{\alpha})$, the type of NDS solver, the resolution/accuracy and length of the modelled trajectory which will determine the number of available meta-model training data N_t. In the case of the fixed-time step classical RK4 solver employed throughout the thesis, and for a single iteration of the solver, the CI-NDS is evaluated four times, for the entire length of the solution trajectory, the CI-NDS is evaluated $4N_t$ times.

6.2.2 Computing the LLMs

LLMs are constructed from first-principle or identified using a suitable system identification method in the case of equilibrium data. Unfortunately there is no valid system identification method that can construct LLMs for transient regions. Regardless of the type of operation (construction, linear MOR, and canonical transformation) being conducted on LLMs, it must be repeated N_t times each to cover the size of the meta-model training data set.

Constructing full-order LLMs for the CI-NDS from first-principle generally requires small computational times and they are in general sparse, therefore requiring small computer memory.

On the other hand, applying a linear MOR approach such as TBR or Krylov subspace methods will result in reduced-order LLMs (of order q) that are fully parameterized and may require larger memory to store them than the full-order Jacobians of the CI-NDS.

For the reduced-order LLMs using TBR-MOR, their full-order balanced realization has to be computed before applying state truncation (section 4.2.3); the balanced full-order Jacobians are dense and fully parameterized therefore requiring large amounts of computer memory. The balancing procedure for a single full-order LLM will have a computational cost on the order of $O(n^3)$ (Badía et al., 2006), making it impractical for large model order over few thousands.

Krylov subspace MOR method such as the one described by Algorithm 4-1 will require the inverse of the full-order LLM system matrix, at worst case, when dealing with dense representation of the full-order system matrix, the inverse will have computational time cost of $O(n^3)$, this however is very rare and the general case will usually have a sparse structured system matrix making the computational time complexity of the system matrix inverse with a suitable algorithm much lower than that, therefore Krylov subspace MOR algorithms is the preferred choice when dealing with model orders above few thousands.

The way to deal with very high order systems is to reduce them first to intermediate size problem with order $q_i \sim few\ hundereds$ using Krylov Subspace MOR and then applying the TBR MOR approach to obtain the final q order model of the LLMs to insure the stability of the final reduced-order LLMs.

The cost of applying canonical transformation to a reduced order LLMs is insignificant relative to the cost of computing full-order LLM MOR because it will only have to deal with LLM of reduced-order $q \ll n$.

6.2.3 GP Model Training

The collected set of meta-model training data and their corresponding VBL-LPV system time-varying components of size N_t is partitioned into two disjoint sets one for GPM hyper-parameters training of size N_{tt} and the other for GPM validation of size N_{tv}.

A GPM for each VBL-LPV system time-varying component must be trained as follows:

- A suitable covariance function must be chosen, this involve the examination of the collected training data and choosing the best covariance function to reflect our prior understanding of underlying meta-model parameter function.

- Training the GPM covariance function hyper-parameters, this step involves choosing initial guesses of the hyper- parameters and the optimization of those parameters. This is followed by the GPM validation cycle where the separated validation data set is used to make predictions using the newly trained GPM, the combined GPM training and validation cycles has a computational time complexity that depends directly on the number of the GPM training data N_{tt} (because of the inverse of the covariance matrix during the validation cycle) with cost of $\mathcal{O}({N_{tt}}^3)$ (Snelson, 2007). It should be noted that the GPM training-validation cycles are often repeated multiple times with different hyper-parameters initial values especially the additive noise hyper-parameter to achieve the highest possible confidence in the predictions.

The time spent in the GPMs training-validation cycles of all the time-varying components of the VBL-LPV system will have to factor in the total number of those components which depends on the final order and the structure of the LLMs. To discuss this further, consider a hypothetical CI-NDS of order $n = 100$ with no forcing input and 100 output components for each state, upon gathering full-order LLMs at the model training points, the following linear system structure is revealed:

- $\boldsymbol{A}$ matrix is 100×100 sparse diagonal matrix, and the number of time-varying components in this matrix is 100.
- $\boldsymbol{B}$ matrix is empty because there is no forcing input.
- $\boldsymbol{C}$ matrix is 100×100 sparse diagonal matrix, and the number of time-varying components in this matrix is 100.

If the meta-model was to be trained at full-order, it will require 200 GPMs to describe all the time-varying components of the VBL-LPV system with computational time cost of $\mathcal{O}\left({N_{tt}}^{3}\right)$ each.

Now assume TBR-MOR was applied to the collected full-order LLMs, and it was found that a reduced-order LLM was sufficient to describe the dynamics say at order $q = 10$. The reduced order LLMs structure is as follows:

- $\boldsymbol{A}_r$ matrix is 10×10 dense matrix, and the number of time-varying components in this matrix is 100.
- $\boldsymbol{B}_r$ matrix is empty because there is no forcing input.
- $\boldsymbol{C}_r$ matrix is 100×10 dense matrix, and the number of time-varying components in this matrix is 1000.

In the case of the above reduced-order system, training the meta-model will require the training of 1100 GPMs to describe all the time-varying components of the VBL-LPV system with computational time cost of $\mathcal{O}\left({N_{tt}}^{3}\right)$ each, this is significant increase in computational time cost relative to the full order meta-model of the same system. It can be seen that the source of increase in computational time cost is the number of CI-NDS outputs. Assume modal canonical transformation was applied to the reduced-order LLMs and the following structure is obtained:

- $\boldsymbol{A}_{rc}$ matrix is 10×10 sparse diagonal matrix, and the number of time-varying components in this matrix is 10.
- $\boldsymbol{B}_{rc}$ matrix is empty because there is no forcing input.

- $\boldsymbol{C}_{rc}$ matrix is 100×10 dense matrix, and the number of time-varying components in this matrix is 1000.

Applying the canonical transformation works to reduce the number of time-varying components in LLM system matrix $\boldsymbol{A}_r$ (from 100 to 10), but it will not change the number of time-varying components of the LLM output matrix $\boldsymbol{C}_r$ and this will continue to be a problem in training of the meta-model.

Indeed most of the meta-modelling examples presented earlier used models with limited number of outputs (one or two at most) to justify MOR of full-order LLMs therefore reducing the overall computational time needed to train the meta-model time-varying parameters.

6.3 The Meta-Model Solver

The computational time complexity analysis is achieved by examining the steps the meta-model solver takes during a single iteration and computing the total number of flops per step. The worst case scenario is that no MOR has been applied to the meta-model and no canonical form was selected either.

In order to compute the number of flops involved in the meta-model solver calculations we have identified the number of flops in the following basic matrix-vector arithmetic as follows:

- Addition or subtraction of two n sized vectors requires n flops.

- Inner product of two n sized vectors requires n mutiplications and $n-1$ additions therefore require $2n-1$ flops.

- Multiplication of $m \times n$ sized matrix with n sized vector, each element in the solution requires inner product of n sized vector therefore the total operations is $m(2n-1)$ flops.

To simplify the computational time cost analysis for the meta-model solver, the following assumptions were made:

- A constant size of training data set N_{tt} has been employed for all the meta-model time-varying components.

- The covariance function is the same for all the meta-model time-varying components.

The GPM covariance function evaluation cost depends on the structure of the covariance function and is a function of the number of regression inputs (the number of states of the LLM so either n states for full-order LLM or q states for reduced-order LLM), this cost is negligible compared to the cost of computing the GPM prediction mean and variance.

Each GPM prediction involves computing the prediction mean m_p and its variance σ_p^2 (equation (3.15)) which is included again here

$$\begin{aligned} m_p &= \boldsymbol{k}_{tp}^T \boldsymbol{K}_{tt}^{-1} \boldsymbol{z} \\ \sigma_p^2 &= k_{pp} - \boldsymbol{k}_{tp}^T \boldsymbol{K}_{tt}^{-1} \boldsymbol{k}_{tp} + \sigma_n^2 \end{aligned} \tag{6.1}$$

$\boldsymbol{z}$ is the noisy training data vector of size N_{tt}, $\boldsymbol{k}_{tp}$ is the cross-covariance vector of size N_{tt} between the noisy training data and prediction target, and k_{pp} is the auto-covariance of the prediction target (a scalar).

$\boldsymbol{K}_{tt}$ is the noisy training data covariance matrix of size $N_{tt} \times N_{tt}$, the inverse of this covariance matrix has a computational time cost of $\mathcal{O}\left({N_{tt}}^3\right)$, therefore computing equation (6.1) will cost $\mathcal{O}\left({N_{tt}}^3\right)$ which is impractical. A more suited approach (Rasmussen and Williams, 2006) is to compute Cholesky decomposition of the covariance matrix and use this to compute the N_{tt} sized vector $\boldsymbol{alpha} = \boldsymbol{K}_{tt}^{-1}\boldsymbol{z}$ during the training phase of the GPM. This will reduce the GPM prediction to $m_p = \boldsymbol{k}_{tp}^T\boldsymbol{alpha}$ which is the inner product of two N_{tt} sized vectors therefore uses $(2N_{tt} - 1)$ flops and have a computional time cost of $\mathcal{O}(N_{tt})$. Using the pre-stored Cholesky decomposition of the covariance matrix to compute the GPM prediction variance σ_p^2, will have a computational time cost of $\mathcal{O}\left({N_{tt}}^2\right)$ (Snelson, 2007). Therefore, the computational time cost of one GPM prediction is $\mathcal{O}\left({N_{tt}}^2\right)$.

The analysis of the meta-model solver algorithm in single iteration is broken down to four distinctive operations:

1. Compute the GPM predictions for all time-varying components of the VBL-LPV meta-model, as it have been established by the above discussion, the computational time cost of one GPM prediction is $\mathcal{O}\left({N_{tt}}^2\right)$, the total number of meta-model time varying component depends on the LLMs structure. This will add a fixed computational time cost bias to the total cost of single iteration of the meta-model solver.

2. Compute the meta-model velocity vector $\boldsymbol{w}(k+1)$, when using Forward-Euler method (equation (3.27)), the number of flops required is given in Table 6-1.

Table 6-1 Total number of flops spent in the meta-model velocity vector $\boldsymbol{w}(k+1)$ computation using Forward-Euler method

$\boldsymbol{w}(k+1) =$	Number of flops
$\boldsymbol{w}(k) +$	n
$h \times ($	n
$\boldsymbol{A}(k)\boldsymbol{w}(k)$	$n(2n-1)$
$+$	n
$\boldsymbol{B}(k)\dot{\boldsymbol{u}}(k)$	$n(2p-1)$
$)$	Total flops $= 2n^2 + 2np + n$

Examining the total number of flops in Table 6-1 indicates that the computational time cost of the meta-model velocity vector depends mainly on the meta-model order (usually the number of model inputs $p \ll n$) therefore will have a computational time cost of $\mathcal{O}(n^2)$. For other types of meta-model velocity solvers described by section 3.4, the total number of flops required will be different but the computational cost will also be $\mathcal{O}(n^2)$.

3. Compute the meta-model state vector $\boldsymbol{x}(k+1)$ given by equation (3.30), the number of flops required is given in Table 6-2.

Table 6-2 Total number of flops spent in the meta-model state vector $\boldsymbol{x}(k+1)$ computation

$\boldsymbol{x}(k+1) =$	Number of flops
$\boldsymbol{x}(k) +$	n
$h \times \boldsymbol{w}(k+1)$	n
	Total flops $= 2n$

Examining the total number of flops in Table 6-2 indicates that the computational time cost of the meta-model state vector only depends on the meta-model order and therefore will have a computational time cost of $\mathcal{O}(n)$.

4. Finally, Compute the meta-model output vector $\boldsymbol{y}(k)$ given as part of equation (3.20), the number of flops required is given by Table 6-3.

Table 6-3 Total number of flops spent in the meta-model output vector $\boldsymbol{y}(k)$ computation

$\boldsymbol{y}(k) =$	Number of flops
$\boldsymbol{C}(k)\boldsymbol{x}(k)$	$m(2n-1)$
$+$	m
$\boldsymbol{D}(k)\boldsymbol{u}(k)$	$m(2p-1)$
	Total flops $= 2nm + 2pm + m$

Examining the total number of flops in Table 6-3 indicates that the computational time complexity of the evaluating the meta-model output vector depends mainly on the order n and the number of outputs m of the meta-model, as the number of outputs approaches the order of the meta-model (such as in the case of distributed parameters CI-NDS), the computational time cost will be $\mathcal{O}(n^2)$.

In conclusion, for a single iteration of the full-order meta-model solver, the total computational time cost will be a combination of a fixed computational time cost bias of GPM predictions given by $\mathcal{O}\left({N_{tt}}^2\right)$ plus a computational time cost of $\mathcal{O}(n^2)$. Since the meta-model computational time complexity depends partly on the LLMs order, reducing the order of the meta-model will provide computational time saving. Applying canonical transformation to the LLMs can also reduce the computational time cost to $\mathcal{O}(n)$ if it resulted in sparse state matrix. GPM predictions computational time cost overhead can be reduced from $\mathcal{O}\left({N_{tt}}^2\right)$ to $\mathcal{O}(N_{tt})$ if only the prediction mean is computed which is sufficient to obtain the meta-model solution. This can be implemented after verifying the uncertainty of the meta-model solution meets the experiment criteria.

The derived computational time cost of the proposed GP blended reduced order VBL-LPV meta-model has been anticipated due to the nature of the non-parametric meta-model blending by GPM regression specifically chosen earlier because it can deal with high dimensional CI-NDS whose training data are generally sparse therefore; the size of the meta-model training points is generally small.

Now the asymptotic computational time cost of the GP-blended VBL-LPV meta-model have been established, a couple of remarks concerning its practical applicability:

- LLMs structure plays an important role in the performance of the meta-model solver, reducing the LLM model order and applying MOR and canonical forms helps to reduce the number the meta-model time-varying parameters therefore sees the computational time cost reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$ because of LLMs sparsity. However, there is no practical way of knowing beforehand if MOR of the full-order LLM will lead to computational savings because this is model specific. Other parts of the LLMs structure such as the number of meta-model outputs could lead to increased computational costs defeating any gains made by MOR as demonstrated in section 6.2.3 and this section.

- The number of GPM training data contributes a computational overhead of $\mathcal{O}\left({N_{tt}}^2\right)$ to the meta-model solver cost, careful selection of a smallest-sized training data set greatly reduces this overhead, and optioning to predict the GPM mean during the solution will have a great positive effect on the computational speed of the meta-model solver (only $\mathcal{O}(N_{tt})$). There is a trade-off between the GP parameter model accuracy and the number of training data, more prior evidence (i.e more training data) lead to improved predictions, GPM does not suffer from over fitting like in parametric regression methods. All things being equal, reducing the prior evidence will reduce the GPM accuracy, however it is difficult to anticipate the effects of reducing GPM predictions accuracy on the meta-model solution because the GP static models provide predictions to a dynamic VBL-LPV meta-model which in many cases could alter the properties of the time-varying LLMs such as stability. Therefore, the general rule of thumb is to optimize the size of training data set to provide the highest GPM predication accuracy.

6.4 Meta-Modelling of a CI-NDS

The section is concerned with meta-modelling of a CI-NDS theoretical system given by the NTL model previously described in section 4.6, with a couple of notable differences:

- The model order is set at 5000 to demonstrate the computational complexity possible advantage of using the proposed GP-blended VBL-LPV meta-model.
- The NTL model being a distributed parameters model, it has potential outputs across its entire range of states, for the sake of simplicity the output was selected to be at node 1 $\left(y(t) = x_1(t)\right)$ to provide a large MOR span compared to full-order model, therefore reducing the potential number of meta-model time-varying parameters.

This section is further divided into three subsections, the first section will describe the meta-model training data and the application of MOR, the second section will describe the training of the meta-model time-varying parameters and the final section will present a number of test inputs to rate the meta-model computational time cost against those of some standard nonlinear ODE solvers.

6.4.1 Meta-Model Training Points

The meta-model training points have been collected using the same procedure previously described in section 4.6 for the 10th order NTL model, it used the same range of inputs for the collection of equilibrium data points and the same randomly generated pulse sequence (Figure 4-2) for the collection of the off-equilibrium data points. This resulted in 230 meta-model training points (input-state pairs) at the full-order LLMs were computed from the first-principle Jacobians of the NTL model (appendix A.4).

The computed full-order LLMs have the following structure:

- $\boldsymbol{A}$ state matrix is sparse tri-diagonal matrix of order $n = 5000$ with $(3n - 2)$ or 14998 time-varying elements.
- $\boldsymbol{b}$ input vector is a constant sparse 5000-long vector with only one element located at $\boldsymbol{b}(1)$.
- $\boldsymbol{c}$ output vector is a constant sparse 5000-long vector with only one element at located $\boldsymbol{c}(1)$.
- There is no feed-through input-output element in the system.

In order to produce a meta-model that is faster than the original CI-NDS model, linear MOR techniques must be applied to the collected full-order LLMs at the meta-model training data as explained previously in section 4.4.

Attempting to apply TBR-MOR directly on the full-order LLMs will have a computational cost of $O(n^3)$, impractical for this LLM order not to mention that the full-order balanced matrix is dense requiring tremendous amounts of storage prior to state truncation.

Krylov subspace MOR methods such as the one-sided method with Arnoldi iteration Algorithm 4-1 will require the inverse of the full-order state matrix that is also $O(n^3)$ computational complexity in the case of a dense state matrix.

However, the computed full-order state matrix is sparse therefore, a creating sparse matrix object and solving the linear system of equations in MATLAB will have only $O(n)$ when implementing the Arnoldi One-Sided Krylov Subspace MOR Algorithm which is a significant computational gain over the TBR-MOR approach. The problems with Krylov subspace MOR approach (as demonstrated before in section 4.2.2) are the lack of provable error bounds and the chance it may not preserve the stability of the full order LLMs. To overcome this, the full-order LLMs have been reduced to an intermediate model order $q_i = 150$ using Algorithm 4-1 and the results from this intermediate MOR step was reduced further using TBR-MOR to ensure stability of the reduced-order LLMs. Normalized local errors between outputs of the full-order and the reduced-order LLMs at the collected linearization points are computed using equation (4.16) and shown by Figure 6-1.

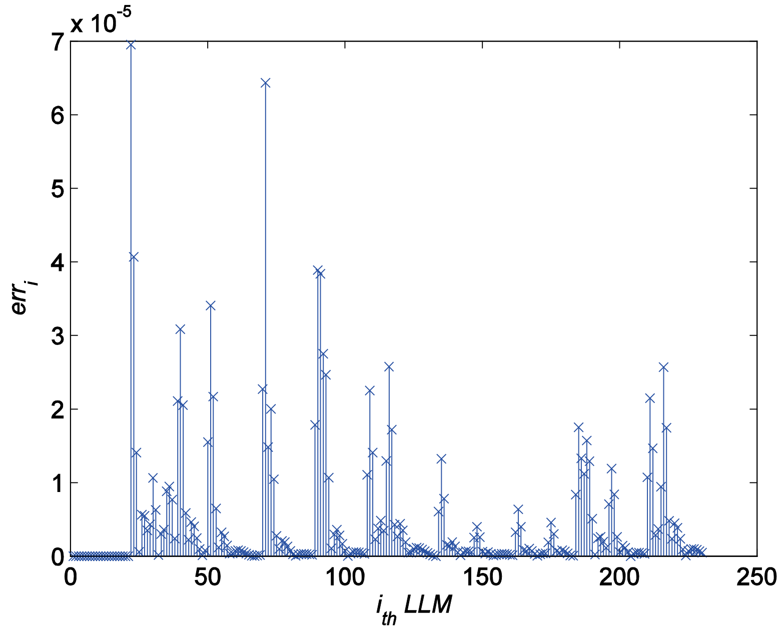


Figure 6-1 Normalized local output errors between the full-order LLMs and the Krylov-reduced LLMs at the collected training data of the NTL CI-NDS

Figure 6-1 Shows that the normalized output error err_i between the full-order LLMs output and those of the reduced-order LLMs, it indicates a very good agreement between the reduced-order LLMs and the full-order ones. The local error err_i of the LLMs outputs appears to approach zero in magnitude as it moves through the range of the collected equilibrium training points data set (the first 20 indices on the x-axis of Figure 6-1), since the Krylov MOR of Algorithm 4-1 is based on the Taylor's series expansion of the full-order LLM transfer function around $s = 0$ frequency; it will generally provide an excellent fit for the steady state dynamics of the system and hence the decrease of the computed output errors in the steady state regions of model.

At the end of the Krylov intermediate MOR step, the LLMs have the following structure:

- $\boldsymbol{A}_{kr}$ state matrix is a dense matrix of order $n = 150$ with 22500 time-varying elements.
- $\boldsymbol{b}_{kr}$ input vector is a dense with 150 time-varying elements.
- $\boldsymbol{c}_{kr}$ output vector is a with 150 time-varying elments.

Although the LLM order have been reduced from 5000 to 150, the meta-model time-varying components have risen from 14998 to 22800, making it even more difficult to train the meta-model in its present form. The computed reduced-order LLMs must be compressed further using TBR-MOR to reduce the number of meta-model time-varying components. This is accomplished by computing the balanced realization of the reduced-order LLMs, then computing the hsv values of state contributions to the reduced-order model outputs (section 4.2.3) to establish the level of state truncation.

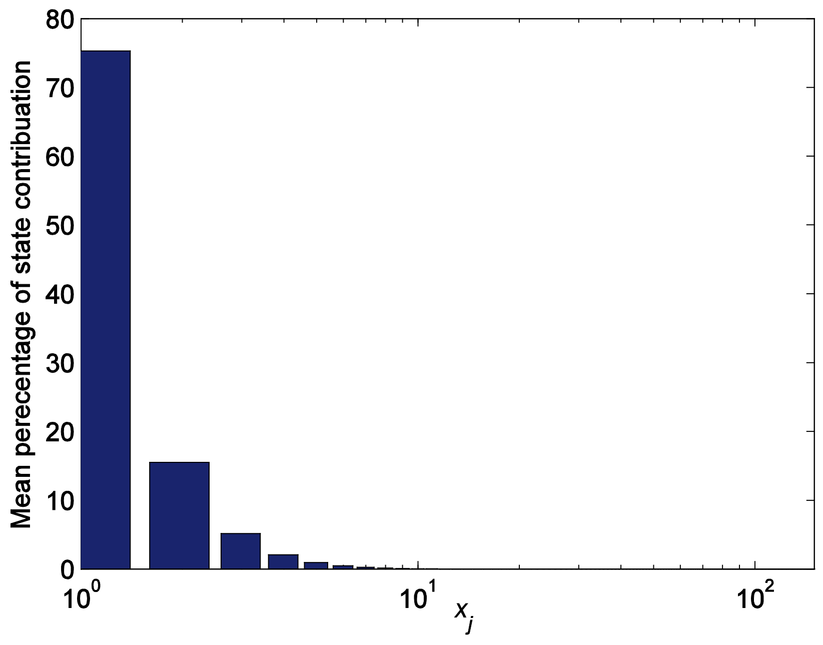


Figure 6-2 Mean percentage of state contributions of the balanced Krylov-reduced LLMs for the NTL meta-model collected training data

Figure 6-2 shows mean percentage of each state contribution in the balanced realization of the 150-order Krylov MOR of the full-order LLMs computed using equation (4.17). Figure 6-2 shows that the majority of state contribution is contained within the first four states (approximately 95.195%), selecting a TBR-MOR of order four will result in fast-changing meta-model time-varying parameters that are difficult to train using the GPM. A tenth order reduction will only discard 0.125% of model output energy and will provide easier to train meta-model parameters.

It has been decided to train two meta-model scenarios based on two different final reduced model orders of four and ten to showcase the trade off in predictions accuracy and meta-model computational time performance.

The 4th order LLMs have the following structure:

- $\boldsymbol{A}_r$ state matrix is a dense matrix of order $q = 4$ with 16 time-varying elements.

- $\boldsymbol{b}_r$ input vector is a dense with 4 time-varying elements.

- $\boldsymbol{c}_r$ output vector is a with 4 time-varying elments.

The number of LLMs time-varying parameters has seen a massive reduction from 22800 to 24, this number can be reduced more through the application of a suitable canonical form. The controllability staircase canonical form was applied to the fourth order LLMs computed by the TBR-MOR step resulting the following LLMs structure:

- $\boldsymbol{A}_{rc}$ state matrix is a sparse tri-diagonal matrix of order $q = 4$ with 10 time-varying elements.

- $\boldsymbol{b}_{rc}$ input vector is sparse 4 elements vector with only one time-varying component located at $\boldsymbol{b}_{rc}(4)$.

- $\boldsymbol{c}_{rc}$ output vector is sparse 4 elements vector with only one time-varying component located at $\boldsymbol{C}_{rc}(4)$.

The number of time-varying components in the reduced-order LLMs with canonical form has been reduced from 24 to 12. A close examination of the final meta-model time-varying parameters revealed that some of them is only changing on a small scale ($< 1 \times 10^{-4}$ variance) over the collected envelope of the training data therefore they were averaged and assumed to be constant, this further reduced the total number of meta-model parameter from 12 to 10 all contained in the canonical reduced-order state matrix $\boldsymbol{A}_{rc}$.

The 10th order LLMs have the following structure:

- $\boldsymbol{A}_r$ state matrix is a dense matrix of order $q = 10$ with 100 time-varying elements.

- $\boldsymbol{b}_r$ input vector is a dense with 10 time-varying elements.

- $\boldsymbol{c}_r$ output vector is a with 10 time-varying elments.

The number of reduced-order LLMs time-varying parameters has seen a massive reduction from 22800 to 120, this number can be reduced more through the application of a suitable canonical form. The controllability staircase canonical form was applied to the tenth order LLMs computed by the TBR-MOR step resulting the following LLMs structure:

- $\boldsymbol{A}_{rc}$ state matrix is a sparse tri-diagonal matrix of order $q = 10$ with 28 time-varying elements.

- $\boldsymbol{b}_{rc}$ input vector is sparse 10 elements vector with only one time-varying component located at $\boldsymbol{b}_{rc}(10)$.

- $\boldsymbol{c}_{rc}$ output vector is sparse 10 elements vector with only one time-varying component located at $\boldsymbol{C}_{rc}(10)$.

The number of time-varying components in the reduced-order LLMs with canonical form has been reduced from 120 to 30. A close examination of the final meta-model time-varying parameters revealed that some of them are only changing on a small scale ($< 1 \times 10^{-4}$ variance) over the collected envelope of the training data therefore they were averaged and assumed to be constant, this further reduced the total number of meta-model parameter from 30 to only 19 all contained in the canonical reduced-order state matrix $\boldsymbol{A}_{rc}$.

It can be seen that the tenth order meta-model have more time-varying parameters than the fourth order one (19 versus 10), this will produce a faster meta-model if the size of the meta-model training data set is equal between the two meta-models. However, this computational speed advantage will come at the cost of meta-model predictions accuracy and a harder to train meta-model parameters.

6.4.2 Meta-Model Parameters Training and Validation

The meta-model training points collected on the trajectory of the NTL model was split using into two disjoint randomly selected sets, one with 46 points for training of the GPMs and the other with 184 points for the validation of the trained models as part of the cross-validation process. Additive white noise of zero mean and very small variance (1×10^{-7}) was added to all training points' targets.

To level the computational overhead attributed by the GPMs, both meta-model scenarios have the same covariance function for all the time-varying parameter models (squared-exponential with ARD), they also share the same number of training and validation data. The GPM covariance function hyper-parameters are trained by minimizing the cost function $\boldsymbol{J}(\boldsymbol{\theta})$ in equation (3.10). The covariance function noise parameter was set to 1×10^{-7} after adding the same amount of zero-mean Gaussian noise to the collected training data targets.

The GPMs optimization results for the 4th order meta-model are given in Table 6-4. It is evident that the GPM struggled to model some of the time-varying parameters because of their fast changing dynamics. This will definitely reduce the quality of meta-model predictions.

Table 6-4: 4th order meta-model training results for the CI-NDS NTL model

$GPM(\boldsymbol{A}_{rc})$	$SMSE$	$\%fitness$
a_{11}	2.13×10^{-3}	95.37
$a_{12} = a_{21}$	6.17×10^{-2}	75.09
a_{22}	8.13×10^{-4}	97.14
$a_{23} = a_{32}$	1.25×10^{-4}	98.88
a_{33}	5.51×10^{-3}	92.56
$a_{34} = a_{43}$	1.05×10^{-2}	89.72
a_{44}	5.73×10^{-4}	97.6

The GPMs optimization results for the 10th order meta-model are given in Table 6-5.

Table 6-5: 10th order meta-model training results for the CI-NDS NTL model

$GPM(\boldsymbol{A}_{rc})$	$SMSE$	$\%fitness$
a_{44}	2.13×10^{-5}	99.54
$a_{45} = a_{54}$	3×10^{-5}	99.45
a_{55}	5.41×10^{-6}	99.77
$a_{56} = a_{65}$	4.13×10^{-6}	99.8
a_{66}	1.92×10^{-6}	99.56
$a_{67} = a_{76}$	6.28×10^{-6}	99.75
a_{77}	1.17×10^{-5}	99.66
$a_{78} = a_{87}$	3.62×10^{-6}	99.81
a_{88}	1.66×10^{-4}	98.71
$a_{89} = a_{98}$	2.64×10^{-4}	98.34
a_{99}	1.2×10^{-5}	99.56
$a_{9,10} = a_{10,9}$	7.55×10^{-6}	99.72
$a_{10,10}$	5.3×10^{-7}	99.93

The meta-model parameters training results in Table 6-5 proved to be successful, very small $SMSE$ values (small variances) coupled with very high $\%fitness$ indicate a very good agreement of the meta-model validation data set and the trained parameter model. In contrast to the 4th order meta-model training results, the 10th order meta-model parameters are smooth and easier to train.

6.4.3 Meta-Model Response to Test Inputs

This section will examine the response of the trained meta-models to several test inputs, the performance of the meta-model will be benchmarked against fixed and variable-time step nonlinear ODE solvers.

It has been established in section 6.3 that the meta-model computational time complexity is a function of its order and the size of GPM training data set. It is harder to train a meta-model below a certain reduced model order (evident by the 4th order meta-model training results in Table 6-4), because this will force the meta-model time-varying parameters to change rapidly making it difficult to train without using some 'exotic' GP covariance function. Therefore, it may limit the range of reduced-order model scenarios to benchmark the meta-model performance. For a certain reduced order meta-model, the other part of computational time complexity is controlled by the size of the meta-model training data set, however it is always important to get the trained meta-model time-varying parameters to faithfully reproduce their true function, because any lower quality predications of the time-varying parameters will induce unrecoverable errors in the meta-model sequential solution. Therefore it is not possible to rely on computational savings from reductions in the meta-model training data set.

The default solver time step for the NTL meta-model has been established at $h = 1 \times 10^{-4} s$ to provide an accurate solution.

All the meta-model test inputs have a time frame of one second to provide a comparison across the meta-model computational speeds. The meta-model solver used Exact-Discretization method (section 3.4.1) to solve the velocity equation, parameter similarity detection has been employed in the meta-model solver to detect similar meta-model time-varying parameters and reduce the meta-model solver processing time during the solution.

Classical RK4 fixed-time step solver has been used to obtain the CI-NDS response to different test inputs to produce exact CI-NDS response plot against that the two meta-models responses. All step-like discontinues-time test inputs were approximated using sigmoid function (appendix B).

Finally, the meta-model exact uncertainty propagation plots have been omitted because of the extremely small value of the meta-model solver time step (as explained by chapter 5).

The first test input is a Heaviside unit step function $H(.)$ given by $u(t) = H(t - 0.05)$, the results of the meta-models simulation with this test input are given by Figure 6-3. Figure 6-3 shows an excellent match between the 10th order meta-model output and the true CI-NDS model output achieving $99.76\ \%fitness$, with less accuracy for the 4th order meta-model achieving $96.23\ \%fitness$.

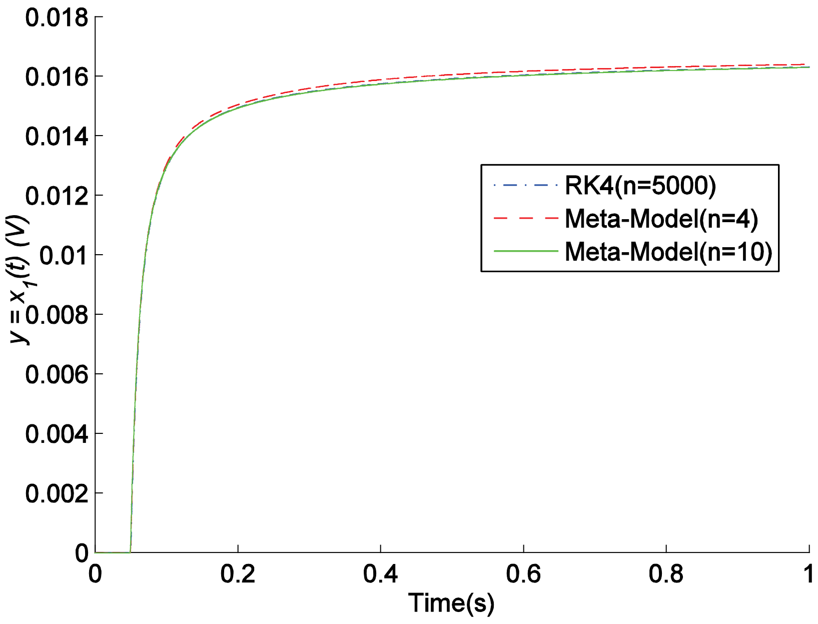


Figure 6-3 NTL meta-models response to test input $(t) = H(t - 0.05)$ versus the RK4 response of the CI-NDS

The second test input is a multi-step function given by Figure 6-4.

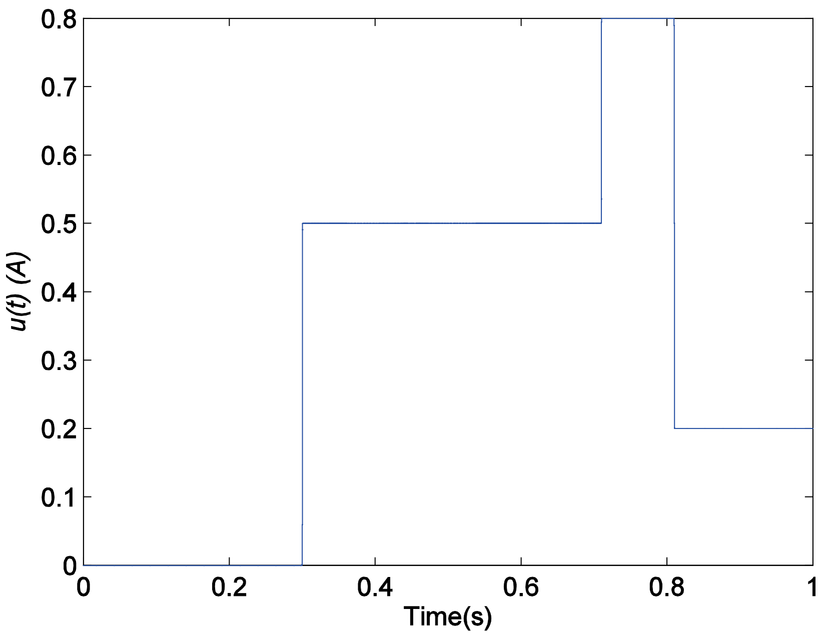


Figure 6-4 NTL meta-model multi-step test input

The meta-model response to the above test input is given by Figure 6-5.

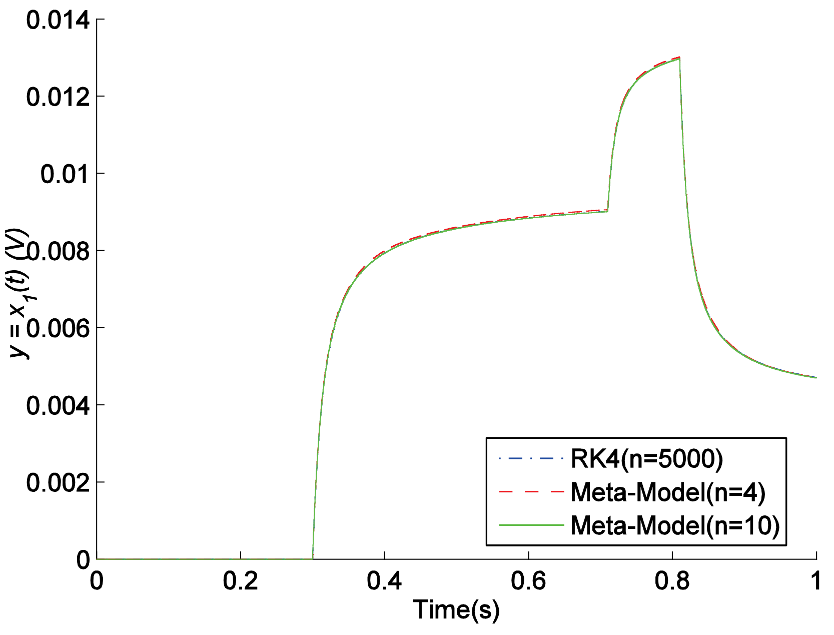


Figure 6-5 NTL meta-models response to the multi-step test input versus the RK4 response of the CI-NDS

Figure 6-5 shows an excellent match between the 10th order meta-model output and the true CI-NDS model output achieving 99.83 $\%fitness$, with less accuracy for the 4th order meta-model achieving 96.92 $\%fitness$.

The third meta-model test input is a cosine function given by $u(t) = \frac{(\cos(2\pi t)+1)}{2}$., the results of the meta-model simulation are shown in Figure 6-6.

Figure 6-6 shows an excellent match between the 10th order meta-model output and the true CI-NDS model output achieving 98.82 $\%fitness$, with a slightly higher accuracy for the 4th order meta-model achieving 98.83 $\%fitness$. It is plausible in the case of the 4th order meta-model that the trajectory of a test input might have passed through part of well-trained meta-model parameters operating space and produce good model accuracy.

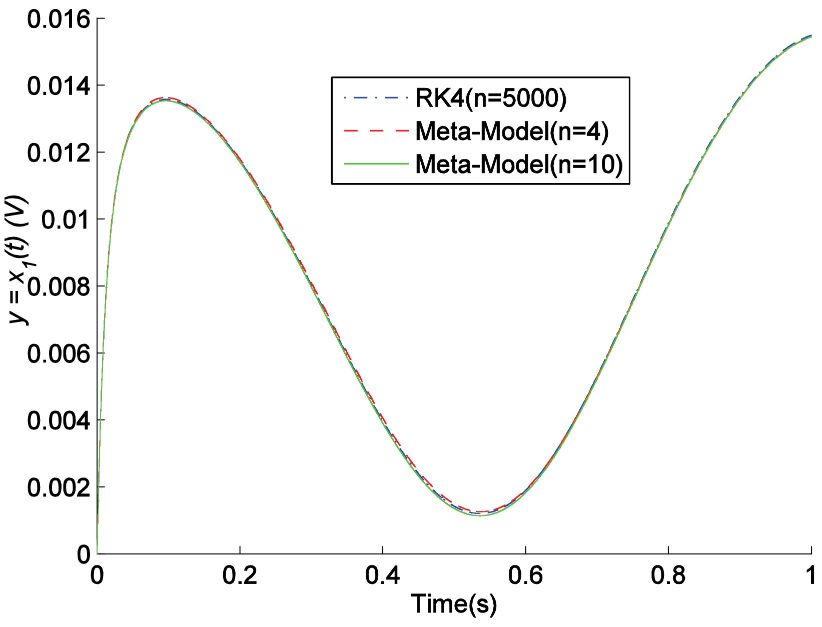


Figure 6-6 NTL meta-models response to cosine test input versus the RK4 response of the CI-NDS

The fourth meta-model test input is an exponential function given by $u(t) = e^{-10t}$. The results of the meta-model simulation with this test input are given by Figure 6-7.

Figure 6-7 shows an excellent match between the 10th order meta-model output and the true CI-NDS model output achieving $98.1\ \%fitness$, with less accuracy for the 4th order meta-model achieving $95.94\ \%fitness$.

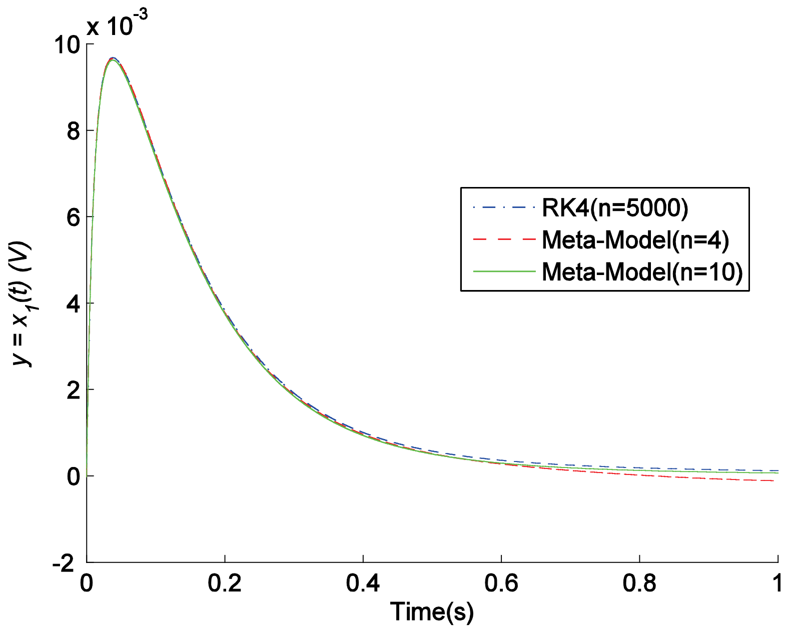


Figure 6-7 NTL meta-model response to exponential test input versus the RK4 response of the CI-NDS

The fifth meta-model test input is a 1A$_{p\text{-}p}$, $5Hz$ sinusoidal function, the results of the meta-model simulation with this test input are given by Figure 6-8. This test input is interesting because it spans meta-model input range $(-1, 1)$ beyond that used in the training phase $(0, 1)$, the both meta-models saw degraded prediction accuracy, the 10th order meta-model achieved $88.01\ \%fitness$, while the 4th order meta-model only achieved $41.44\ \%fitness$ While the meta-model is not generally expected to deal with inputs outside its training data range, this massive reduction in the 4th order meta-model accuracy is mostly attributed to the inadequate training results rather the actual model order.

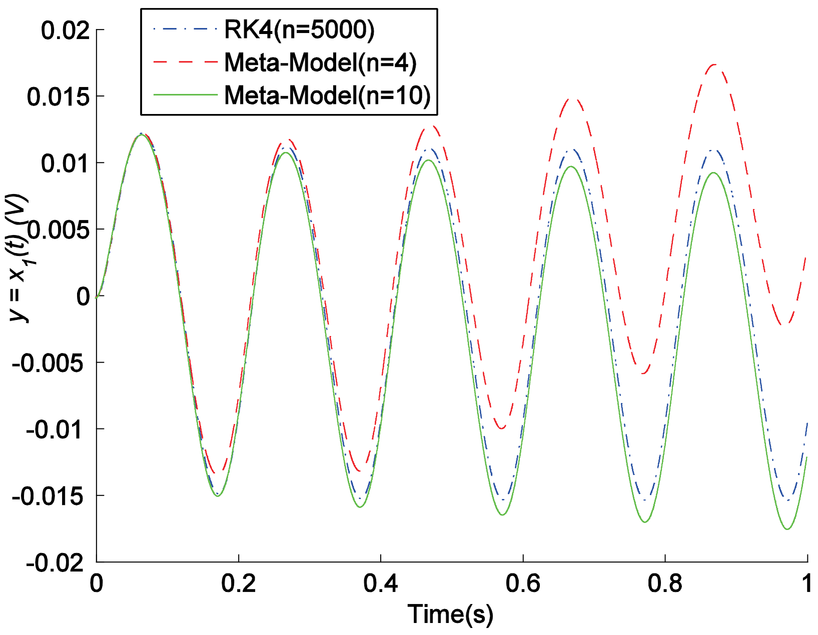


Figure 6-8 NTL meta-models response to sinusoidal test input versus the RK4 response of the CI-NDS

The final test input the linear frequency sweep previously described in section 4.6 (Figure 4-5) used in testing the 10th order NTL meta-model. This test input also spans meta-model input range $(-1, 1)$ beyond that used in the training phase $(0, 1)$.

The results of the meta-model simulation with this test input are given by Figure 6-9. Again both meta-models saw degraded solution accuracy. The 10th order meta-model achieved 93.83 $\%fitness$ and the 4th order meta-model only achieved 62.6 $\%fitness$.

In terms of prediction accuracy, the 10th order meta-model has an apparent advantage on the 4th order meta-model in almost all the six test cases. The 4th order meta-model could have led to high prediction accuracy if was not for it's difficult to train rapidly-changing time-varying parameters. Testing the meta-models with inputs outside its range of training inputs reveals how much they have uncovered of the underlying time-varying parameter functions during the training phase.

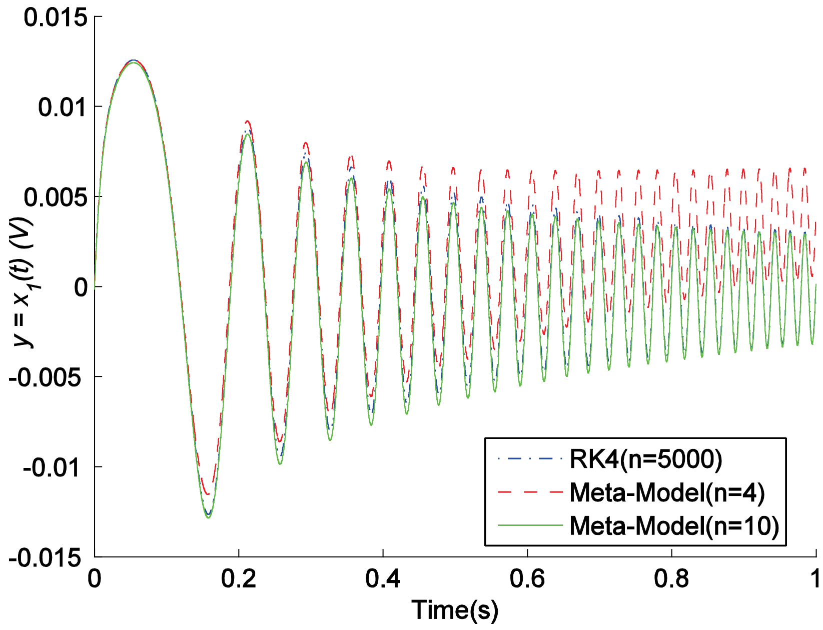


Figure 6-9 NTL meta-models response to sinusoidal linear frequency sweep test input versus the RK4 response of the CI-NDS

The computational time performance of the two meta-modelling scenarios versus the fixed-time step RK4 method has been computed by measuring the time to obtain the solution in all test input cases. This has been done twice for each meta-model case; one without uncertainty propagation and the other with it to showcase the difference in speed. Since the fixed-time step solver total time to obtain a solution only depends on the solver time step and the length of the modelled trajectory, and since of all of the previous test inputs share the same trajectory length and solver time step, the final total time to obtain the solution was averaged and the time for each solution point can be computed by simple division on the number of solution points (in this case is 9999 based on a trajectory length of $1s$ and a solver time step of $1 \times 10^{-4}s$). Computational time performances of the two meta-model scenarios versus the RK4 method for the NTL model of order 5000 are given in Table 6-6.

Table 6-6: Computational time performance of the of the two meta-model scenarios versus the RK4 method for the NTL model of order 5000 (all measurements are in seconds)

	RK4 (order 5000)	10th order meta-model		4th order meta-model	
		Mean only	Mean and variance	Mean only	Mean and variance
Heaviside unit step	158.85	141.98	268.49	82.57	145.03
Multi-step	156.22	140.69	271.51	81.12	148.54
Cosine function	149.55	143.03	276.28	81.65	149.09
Exponential function	152.40	143.00	278.49	82.36	151.19
Sinusoidal	161.76	147.12	286.44	82.57	151.75
Sinusoidal linear frequency sweep	176.37	158.19	303.63	81.94	149.77
Average solver time					
	159.19	145.67	280.81	82.04	149.23
Average solver time for single solution point (based on 9999 points)					
	1.59×10^{-02}	1.46×10^{-02}	2.81×10^{-02}	8.20×10^{-03}	1.49×10^{-02}

Table 6-6 shows the computational speed advantage of the 4th order meta-model solver (without uncertainty propagation) over the RK4 method (almost twice as fast) while the 10th order meta-model solver only achieved a marginally better performance to the RK4 solver. Table 6-6 also shows that obtaining the meta-model solution without uncertainty propagation (i.e. without the GPM predictions variance computation) is almost twice as fast as when it computed with uncertainty propagation.

The 5000 order NTL model was solved using a set of MATLB variable-time step solvers. The MATLAB ODE solvers suit (Shampine and Reichelt, 1997) contains a number of explicit and implicit ODE solvers. Explicit solvers as such ODE23, ODE45 and ODE113 are used to solve non-stiff NDS. Implicit solvers such ODE23s and ODE15s are used to solve both stiff and non-stiff NDS. ODE113 and ODE15s are variable-order solvers in addition of being variable-time step solvers.

The solution error (which reflects on the accuracy of the solver solution) at each step taken by the variable-time step solver is controlled by two quantities:

- Relative error tolerance ($RelTol$), controls the number of correct digits in all solution components bellow the Absolute error tolerance ($AbsTol$). The default value of $RelTol$ is 1×10^{-3}.

- Absolute error tolerance ($AbsTol$) is a threshold below which the value of a solution component is unimportant. $AbsTol$ determine the accuracy when the solution approaches zero. The default value of $AbsTol$ is 1×10^{-6}.

Default relative and absolute error tolerances were used for all the above ODE solvers. The Jacobian matrix pattern (a sparse tri-diagonal matrix of order 5000) of the full-order NTL model was supplied to the stiff solvers (ODE23s and ODE15s) to speed up the computations. The computational time performance of the MATLAB ODE suit of solvers for the NTL model of order 5000 is given in Table 6-7. The tested ODE solvers solution (if successful) achieved more than $99\% fitness$ referenced to the solution of the RK4 solver.

Table 6-7: Computational time performance of the MATLAB ODE suit of solvers for the NTL model of order 5000 (all measurements are in seconds)

	ODE23	ODE45	ODE113	ODE23s	ODE15s
Heaviside unit step	14.04	Fail	18.0659	18.38	16.46
Multi-step	13.99	Fail	17.8509	19.56	17.35
Cosine function	13.20	14.26	16.31	20.39	15.82
Exponential function	13.19	13.38	17.59	18.87	15.67
Sinusoidal	25.30	17.66	49.73	37.94	29.41
Sinusoidal linear frequency sweep	148.63	35.52	1288.90	125.92	668.29

Table 6-7 shows a good overall performance of the variable-time step solvers, the following highlights the performance:

- The ODE23 solver achieved the best performance across the first four test inputs.
- The ODE45 solver failed to make a prediction for the first two step-like test inputs, the rest of the solvers had a comparable performance for those test inputs.
- The fixed and variable frequency sinusoidal test inputs (five and six) had proved to be more problematic for all the tested solvers except the ODE45 which achieved the best performance in these inputs.

While it is unfair to compare the computational time performance of variable-time step solvers against fixed-time step solvers (because they will traverse the length of the solution trajectory much faster due to the variable-time steps), comparing the average total solver time of the meta-model in Table 6-6 to the variable-time step solvers performance in Table 6-7 highlights the following points:

- For all the tested inputs, there exists at least one variable-time step solver that has a better performance than the two meta-model scenarios (orders 10 and 4).
- The 4th order meta-model computational time performance for the sinusoidal linear frequency sweep input was better than four of the five variable-time step solvers except for ODE45. However, the 4th order meta-model achieved only $62.6\ \%fitness$ of the true model.
- The 10th order meta-model computational time performance for the sinusoidal linear frequency sweep input was comparable to the ODE23 and ODE23s solvers, and was better than ODE113 and ODE15s solvers. The 10th order meta-model achieved $93.83\ \%fitness$ for this test input.

6.5 Conclusions

The computational time cost of the meta-model during the collection of training data, training the meta-model and making predictions was analysed and the following points have been observed:

- When collecting meta-model training data, the CI-NDS must be solved, and the cost of this operation will follow the cost of the employed ODE solver for this system.
- The computational time cost of meta-model training is the cost of training a GPM which depends on the size of the training data set (N_{tt}), the process involves the inversion of the covariance matrix with cost of $O\left({N_{tt}}^3\right)$ but with

the application of Cholesky decomposition on the covariance matrix, this cost can be reduced to $\mathcal{O}\left(\frac{N_{tt}^3}{6}\right)$ (Rasmussen and Williams, 2006). The overall time to train the meta-model depends on the structure LLMs and the total number of meta-model time-varying parameters both specific to the Jacobians of the CI-NDS.

- The computational time cost of the meta-model fixed-time step solver is a combination of the meta-model order (with cost of $\mathcal{O}(n^2)$) and a computation time cost ($\mathcal{O}\left(N_{tt}^2\right)$) overhead attributed by the GPM predictions (this depends on the meta-model training data set). The meta-model computational time cost can be reduced if the order of the meta-model is reduced, or if the size of the training data set is reduced. Use of sparse-structured LLMs (through the application of a suitable canonical form) and omitting the uncertainty predictions in the GPMs can reduce both costs to $\mathcal{O}(n)$ and $\mathcal{O}(N_{tt})$ respectively.
- The final number of meta-model time-varying parameters affects the speed of the meta-model solver. Care must be taken when dealing with CI-NDS of many outputs as this can significantly decrease the meta-model computational time performance.

A case study for the practical computational time performance of the meta-model was conducted for the NTL model of order 5000. Two meta-model scenarios were trained of orders four and ten. The 10th order meta-model was easier to train than the 4th order meta-model because the meta-model time-varying parameters were smooth. The meta-model solver being a fixed-time step solver was tested against the RK4 fixed-time step solver. The 10th order meta-model have showed a great modelling accuracy and a comparable time to the full-order RK4 solver. The 4th order meta-model achieved had less accuracy than the 10th order meta-model for test inputs within the range of its training, but achieved considerable (~50%) computational time saving against the full-order RK4 solver.

The performance of the meta-model was rated against a set of MATLAB ODE variable-time step solvers, overall, the variable-time steps had a better performance solving the CI-NDS except for a couple of test inputs (sinusoidal inputs). However this performance comparison is not fair due to the different nature of fixed and variable-time step solvers.

Chapter 7 - Conclusions and Recommendations

7.1 Conclusions

Within the literature of meta-modelling approaches, an area of concern was the transparency of the meta-model structure, and the failure to utilize the transparent structure of analytical CI-NDS. Therefore, this thesis sat out to develop a meta-modelling framework tailored for analytical CI-NDS.

After consulting the literature on NDS mathematical modelling approaches, it was found that the VBL-LPV system was the only structure that fully-preserved the transparency of the analytical NDS. The VBL-LPV system utilized the underlying structure of the analytical NDS to construct VBL-LLMs. The difficulty with VBL-LPV system is it requires the numerical derivative of the forcing input of the NDS, which may pose a problem if the NDS inputs were corrupted by noise. The derivative of discontinues noise-free piecewise inputs can be computed after input approximation by a sigmoid like function.

GP models have been used to blend the parameters of the VBL-LPV system because they do not suffer from over-fitting problems with limited training data and they provide uncertainty information about predictions.

The GP blended VBL-LPV system was never used to meta-model any type of CI models, because the order of the GP blended VBL-LPV system matches that of the CI-NDS, therefore it was not going to offer any computational saving over the CI-NDS.

The *first contribution* of the thesis was to apply projection-based linear MOR to the LLMs of the GP blended VBL-LPV system to reduce their order therefore permitting a possible computational speed advantage over the CI-NDS. The GP blended reduced-order VBL-LPV system has been applied to meta-model a 100th order 1D-Burgers equations and two scenarios of the NTL model (10th order and 5000th order). The following was observed:

i. MOR of the meta-model LLMs below a certain (model specific) reduced model order can cause the meta-model time-varying parameters to rapidly change with time, this has two consequences:

- The GP model may struggle to model the underlying meta-model time-varying parameters due to the lost smoothness of the underlying function, a possible solution is to invest in a more complicated GP model covariance function, but this takes more design time and may not necessarily give adequate results. The GP model predictions inaccuracy will result in the meta-model predictions being inaccurate.

- Due to the above, the dynamic capability of the meta-model to produce different accuracy grades for different reduced-order LLMs is limited, this in turn will restrict the range of computational time savings to just one corresponding to the one reduced-order meta-model with smooth time-varying parameters.

ii. The reduced-order meta-model of the 100th order 1D-Burgers equations model did not perform well despite the excellent training results of the reduced-order meta-model parameters. The reason for this is the very fast dynamics contained in the reduced order VBL-LPV model of the 100th order 1D-Burgers equations did not comply with the meta-model solver rule of constant parameters within one period of the fixed-time step (due to solver's ZOH). Therefore, the thesis concluded that this meta-modelling approach may not be suitable for analytical CI-NDS with rapidly changing dynamics or at least for this particular system.

iii. The computational time complexity of the meta-model was shown to be a combination of a computational time bias introduced by the GP model of the meta-model parameters (function of the size of the training data set) and the order of the meta-model. The 5000 order NTL was used to test the meta-model computational time saving, the following was observed:

- The number of meta-model time-varying parameters depends on its order, number of inputs and outputs, while MOR and canonical transformation deals with reducing the number of parameters in the LLM state matrix only, the output matrix will be fully-parameterised. The additional parameters in the output matrix will considerably increase the meta-model training time and will impact negatively on the meta-model solver computational time. This will limit its usefulness for CI-NDSs that arise from CFD, FEA and practically any CI-NDS with large number of outputs. Therefore this meta-modelling structure is not recommended for modelling a CI-NDS with more than one or two outputs at a time.

- The meta-model solver implementation around a fixed-time step solver is inadequate to compete with the standard NDS variable-time step solvers, however the positive results of the 4th order NTL meta-model computational time in comparison with the RK4 fixed-time step solver indicates that there is a potential for computational time saving if the meta-model solver was implemented around a variable-time step topology.

The *second contribution* of the thesis is the propagation of uncertainty from the meta-model time-varying parameters to the meta-model predictions. The following was observed for the fixed-time step meta-model solver:

i. Quantitative uncertainty in the meta-model velocity and state solutions is heavily scaled down by the meta-model solver's fixed-time step. A logical result for this kind of solver that the uncertainty diminishes as the time step get smaller however, this will obscure the underlying uncertainty of the meta-model time-varying parameters.

ii. A qualitative measure of uncertainty can be computed instead the quantitative one to visualize the uncertainty in the meta-model velocity and state solutions. It can also be used to improve the accuracy of the meta-model predictions evident by the CSTR meta-model example.

iii. Meta-modelling errors due to the selection of the wrong order for the reduction of the CI-NDS LLMs will not show in the computed meta-model uncertainty of the solution, because the source of uncertainty is the blending of the GP regression models which are independent of the suitability of a certain reduced order LLM choice in describing the local dynamics of the full-order one.

Despite the above limitations, the GP blended VBL-LPV system, it was a transplant model and did utilize the underlying structure of the analytical NDS, it also produced accurate predictions for most of the test problems with the exception of the 1D-Burgers equations.

7.2 Recommendations

The thesis recommends the following to address theoretical and practical shortcomings in the proposed meta-modelling framework:

- The VBL-LPV system can only be applied to analytical CI-NDS because of the requirement to find a valid LLM at operating points of interest. This places a restriction on the type of CI-NDS that can benefit from the proposed meta-modelling approach. So the idea should be extended to cover black-box CI-NDS. LLM Jacobians described by a pre-specified structure can be found empirically at equilibrium points of the trajectory with the aid of many linear system identification methods. The goal is to extend this structure to identify state-space LLMs at off-equilibrium points as well.

- In depth analysis of the deformation (in some) of the reduced-order time-varying parameter space and the proposal of robust covariance functions to deal with this. This is of particular importance since the meta-model accuracy depends on the correct blending of the time-varying parameters predicted by the GP regression models.

- Implementation of variable-time step numerical solvers for the proposed meta-modelling structure, will improve the overall computational speed of the solver. Variable-time step solver can be applied to the meta-model velocity equation and it can consist from low order solver such as Forward-Euler method implemented against Exact-Discretization method or RK4 method to produce local truncation error measure to estimate the local step size.

- Redesign the meta-model solver algorithm to utilize parallel-processing methods to reduce the computational bias introduced by the size of the meta-model training data sets and improve the computational speed of the meta-model and provide simultaneous meta-model time-varying parameters predictions. The meta-model time-varying parameters have to be predicted at each time step taken by the solver and their estimation is independent from each other so it is possible to apply parallel-processing methods such as matrix-vector multiplications through GPU cores or multiple CPU cores will reduce the overall-computational speed of the meta-model.

- Implementation of a tool-box for the proposed meta-modelling approach with suitable programming languages.

List of References

AGUIRRE, L. A. & LETELLIER, C. 2009. Modeling Nonlinear Dynamics and Chaos: A Review. *Mathematical Problems in Engineering,* 2009.

ANTOULAS, A. C. 2005. *Approximation of large-scale dynamical systems*, Siam.

ASCHER, U. M. & PETZOLD, L. R. 1998. *Computer methods for ordinary differential equations and differential-algebraic equations*, Siam.

AUBRY, N. 1991. On the hidden beauty of the proper orthogonal decomposition. *Theoretical and Computational Fluid Dynamics,* 2**,** 339-352.

AZMAN, K. & KOCIJAN, J. 2006. An application of Gaussian process models for control design. *International Control Conference (ICC2006).*

AŽMAN, K. & KOCIJAN, J. 2009. Fixed-structure Gaussian process model. *International Journal of Systems Science,* 40**,** 1253-1262.

BADÍA, J. M., BENNER, P., MAYO, R. & QUINTANA-ORTÍ, E. S. 2006. Parallel algorithms for balanced truncation model reduction of sparse systems. *Applied Parallel Computing. State of the Art in Scientific Computing.* Springer.

BAI, Z. 2002. Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems. *Applied Numerical Mathematics,* 43**,** 9-44.

BALDACCHINO, T., ANDERSON, S. R. & KADIRKAMANATHAN, V. 2012. Structure detection and parameter estimation for NARX models in a unified EM framework. *Automatica,* 48**,** 857-865.

BALDI ANTOGNINI, A. & ZAGORAIOU, M. 2010. Exact optimal designs for computer experiments via Kriging metamodelling. *Journal of Statistical Planning and Inference,* 140**,** 2607-2617.

BARTON, R. R. Simulation metamodels. Proceedings of the 30th conference on Winter simulation, 1998. IEEE Computer Society Press, 167-176.

BECHTOLD, T., STRIEBEL, M., MOHAGHEGH, K. & TER MATEN, E. 2008. Nonlinear model order reduction in nanoelectronics: combination of POD and TPWL. *PAMM,* 8**,** 10057-10060.

BEQUETTE, B. W. & BEQUETTE, W. B. 1998. *Process dynamics: modeling, analysis, and simulation*, Prentice Hall PTR Upper Saddle River, NJ.

BERKOOZ, G., HOLMES, P. & LUMLEY, J. L. 1993. The proper orthogonal decomposition in the analysis of turbulent flows. *Annual review of fluid mechanics,* 25**,** 539-575.

BILLINGS, S. A. & ZHU, Q. M. 1991. Rational model identification using an extended least-squares algorithm. *International Journal of Control,* 54**,** 529-546.

BLANNING, R. W. 1974. The sources and uses of sensitivity information. *Interfaces,* 4**,** 32-38.

BOLEY, D. L. 1994. Krylov space methods on state-space control models. *Circuits, Systems and Signal Processing,* 13**,** 733-758.

BOND, B. & DANIEL, L. Parameterized model order reduction of nonlinear dynamical systems. Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design, 2005. IEEE Computer Society, 487-494.

BROMILEY, P. 2013. Products and Convolutions of Gaussian Probability Density Functions. Tina-Vision Memo.

CAI, G., DUAN, G. & HU, C. 2011. A velocity-based LPV modeling and control framework for an air-breathing hypersonic vehicle. *International Journal of Innovative Computing, Information and Control,* 7**,** 2269-2281.

CAO, D., LIU, D. & WANG, C. H. 2005. Nonlinear dynamic modelling for MEMS components via the Cosserat rod element approach. *Journal of Micromechanics and Microengineering,* 15**,** 1334.

CARDOSO, M. & DURLOFSKY, L. J. 2010. Linearized reduced-order models for subsurface flow simulation. *Journal of Computational Physics,* 229**,** 681-700.

CASTELLETTI, A., GALELLI, S., RATTO, M., SONCINI-SESSA, R. & YOUNG, P. 2012. A general framework for dynamic emulation modelling in environmental problems. *Environmental Modelling & Software,* 34**,** 5-18.

CHEN, S. & BILLINGS, S. A. 1989. Representations of non-linear systems: the NARMAX model. *International Journal of Control,* 49**,** 1013-1032.

CHEN, S., BILLINGS, S. A., COWAN, C. F. N. & GRANT, P. M. 1990a. Non-linear systems identification using radial basis functions. *International Journal of Systems Science,* 21**,** 2513-2539.

CHEN, S., BILLINGS, S. A., COWAN, C. F. N. & GRANT, P. M. 1990b. Practical identification of NARMAX models using radial basis functions. *International Journal of Control,* 52**,** 1327-1350.

CHEN, Y. & WHITE, J. 2000. A quadratic method for nonlinear model order reduction. *Proceedings of the International Conference on Modeling and Simulation of Microsystems***,** pp. 477-80.

CONTI, S., ANDERSON, C. W., KENNEDY, M. C. & O'HAGAN, A. A Bayesian analysis of complex dynamic computer models. 2004.

CONTI, S. & O'HAGAN, A. 2010. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference,* 140**,** 640-651.

DE SILVA, C. W. 2009. *Modeling and control of engineering systems,* Boca Raton, CRC Press.

DENG, H., SHAO, W., MA, Y. & WEI, Z. 2012. Bayesian Metamodeling for Computer Experiments Using the Gaussian Kriging Models. *Quality and Reliability Engineering International,* 28**,** 455-466.

DONG, N. & ROYCHOWDHURY, J. Piecewise polynomial nonlinear model reduction. Design Automation Conference, 2003. Proceedings, 2003. IEEE, 484-489.

DONG, N. & ROYCHOWDHURY, J. 2008. General-purpose nonlinear model-order reduction using piecewise-polynomial representations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* 27**,** 249-264.

DRUSKIN, V. & SIMONCINI, V. 2011. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems & Control Letters,* 60**,** 546-560.

EFE, M. O. & OZBAY, H. Proper orthogonal decomposition for reduced order modeling: 2D heat flow. Control Applications, 2003. CCA 2003. Proceedings of 2003 IEEE Conference on, 2003. IEEE, 1273-1277.

EGEA, J. A., VRIES, D., ALONSO, A. A. & BANGA, J. R. 2007. Global optimization for integrated design and control of computationally expensive process models. *Industrial & Engineering Chemistry Research,* 46**,** 9148-9157.

EL-BELTAGY, M. A. & KEANE, A. 1999. Metalmodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations.

EL-BELTAGY, M. A. & KEANE, A. J. Evolutionary optimization for computationally expensive problems using gaussian processes. Proc. Int. Conf. on Artificial Intelligence, 2001. Citeseer, 708-714.

FARHANG-MEHR, A. & AZARM, S. 2005. Bayesian meta-modelling of engineering design simulations: a sequential approach with adaptation to irregularities in the response behaviour. *International Journal for Numerical Methods in Engineering,* 62**,** 2104-2126.

FARINA, M. & PIRODDI, L. 2011. Identification of polynomial input/output recursive models with simulation error minimisation methods. *International Journal of Systems Science,* 43**,** 319-333.

FISCHER, M. & EBERHARD, P. 2014. Linear model reduction of large scale industrial models in elastic multibody dynamics. *Multibody System Dynamics,* 31**,** 27-46.

FREUND, R. W. 2000. Krylov-subspace methods for reduced-order modeling in circuit simulation. *Journal of Computational and Applied Mathematics,* 123**,** 395-421.

FREUND, R. W. 2003. Model reduction methods based on Krylov subspaces. *Acta Numerica,* 12**,** 267-319.

FREUND, R. W. 2008. Structure-preserving model order reduction of RCL circuit equations. *Model Order Reduction: Theory, Research Aspects and Applications.* Springer.

GALLIVAN, K., GRIMME, E. & VAN DOOREN, P. 1994. Padé approximation of large-scale dynamic systems with Lanczos methods. *Urbana,* 51**,** 61801.

GALLIVAN, K., GRIMME, G. & VAN DOOREN, P. 1996. A rational Lanczos algorithm for model reduction. *Numerical Algorithms,* 12**,** 33-63.

GIUNTA, A. A., WATSON, L. T. & KOEHLER, J. 1998. A comparison of approximation modeling techniques: polynomial versus interpolating models. *AIAA paper.*

GOYAL, S. & BAROOAH, P. 2012. A method for model-reduction of non-linear thermal dynamics of multi-zone buildings. *Energy and Buildings,* 47**,** 332-340.

GREGORČIČ, G. & LIGHTBODY, G. 2007. Local model network identification with Gaussian processes. *IEEE Transactions on Neural Networks,* 18**,** 1404-1423.

GRIMME, E. J., SORENSEN, D. C. & VAN DOOREN, P. 1996. Model reduction of state space systems via an implicitly restarted Lanczos method. *Numerical Algorithms,* 12**,** 1-31.

GUANG-BIN, C., GUANG-REN, D. & CHANG-HUA, H. A velocity-based LPV modeling and control framework for nonlinear tracking. Control Conference (CCC), 2010 29th Chinese, 2010. IEEE, 286-291.

GUGERCIN, S. & ANTOULAS, A. C. 2004. A survey of model reduction by balanced truncation and some new results. *International Journal of Control,* 77**,** 748-766.

GUGERCIN, S., ANTOULAS, A. C. & BEATTIE, C. 2008. H_2 model reduction for large-scale linear dynamical systems. *SIAM journal on matrix analysis and applications,* 30**,** 609-638.

HAHN, J. & EDGAR, T. F. 2002. Balancing approach to minimal realization and model reduction of stable nonlinear systems. *Industrial & engineering chemistry research,* 41**,** 2204-2212.

HAHN, J., EDGAR, T. F. & MARQUARDT, W. 2003. Controllability and observability covariance matrices for the analysis and order reduction of stable nonlinear systems. *Journal of Process Control,* 13**,** 115-127.

HINZE, M. & VOLKWEIN, S. 2005. Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: Error estimates and suboptimal control. *Dimension Reduction of Large-Scale Systems.* Springer.

HONG, X., MITCHELL, R. J., CHEN, S., HARRIS, C. J., LI, K. & IRWIN, G. W. 2008. Model selection approaches for non-linear system identification: a review. *International Journal of Systems Science,* 39**,** 925-946.

ILIE, S., SÖDERLIND, G. & CORLESS, R. M. 2008. Adaptivity and computational complexity in the numerical solution of ODEs. *Journal of Complexity,* 24**,** 341-361.

JARVIS, C. 2012. *Reduced order model study of Burgers' equation using proper orthogonal decomposition.* Virginia Polytechnic Institute and State University.

JIN, R., CHEN, W. & SIMPSON, T. W. 2001. Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization,* 23**,** 1-13.

JIN, Y. 2005. A comprehensive survey of fitness approximation in evolutionary computation. *Soft computing,* 9**,** 3-12.

JOHANSEN, T. A. & FOSS, B. A. 1992. A Narmax Model Representation for Adaptive-Control Based on Local Models. *Modeling Identification and Control,* 13**,** 25-39.

JOHANSEN, T. A., HUNT, K. J., GAWTHROP, P. J. & FRITZ, H. 1998. Off-equilibrium linearisation and design of gain-scheduled control with application to vehicle speed control. *Control Engineering Practice,* 6**,** 167-180.

KALMAN, R. E., FALB, P. L. & ARBIB, M. A. 1969. *Topics in mathematical system theory,* New York,, McGraw-Hill.

KARIMI, K. J., BOOKER, A. & MONG, A. Modeling, simulation, and verification of large DC power electronics systems. Power Electronics Specialists Conference, 1996. PESC'96 Record., 27th Annual IEEE, 1996. IEEE, 1731-1737.

KENNEDY, M. C. & O'HAGAN, A. 2001. Bayesian Calibration of Computer Models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology),* 63**,** 425-464.

KERSCHEN, G., GOLINVAL, J.-C., VAKAKIS, A. F. & BERGMAN, L. A. 2005. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear dynamics,* 41**,** 147-169.

KLEIJNEN, J. P. 1975. A comment on Blanning's "Metamodel for sensitivity analysis: the regression metamodel in simulation". *Interfaces,* 5**,** 21-23.

KLEIJNEN, J. P. C. 2009. Kriging metamodeling in simulation: A review. *European Journal of Operational Research,* 192**,** 707-716.

LALL, S., MARSDEN, J. E. & GLAVAŠKI, S. Empirical model reduction of controlled nonlinear systems. 1999. International Federation of Automatic Control.

LAUB, A. J., HEATH, M. T., PAIGE, C. & WARD, R. 1987. Computation of system balancing transformations and other applications of simultaneous diagonalization algorithms. *Automatic Control, IEEE Transactions on,* 32**,** 115-122.

LEITH, D. 1999. Input-output linearization by velocity-based gain-scheduling. *International Journal of Control,* 72**,** 229-246.

LEITH, D. & LEITHEAD, W. 1998a. Appropriate realisation of MIMO gain-scheduled controllers. *International Journal of Control,* 70**,** 13-50.

LEITH, D. & LEITHEAD, W. 1998b. Gain-scheduled controller design: an analytic framework directly incorporating non-equilibrium plant dynamics. *International Journal of Control,* 70**,** 249-269.

LEITH, D., TSOURDOS, A., WHITE, B. & LEITHEAD, W. 2001. Application of velocity-based gain-scheduling to lateral auto-pilot design for an agile missile. *Control Engineering Practice,* 9**,** 1079-1093.

LEITH, D. J. & LEITHEAD, W. E. 1996. Appropriate realization of gain-scheduled controllers with application to wind turbine regulation. *International Journal of Control,* 65**,** 223-248.

LEITH, D. J. & LEITHEAD, W. E. 1998c. Gain-scheduled and nonlinear systems: Dynamic analysis by velocity-based linearization families. *International Journal of Control,* 70**,** 289-317.

LEITH, D. J. & LEITHEAD, W. E. 1999. Analytic framework for blended multiple model systems using linear local models. *International Journal of Control,* 72**,** 605-619.

LEITH, D. J. & LEITHEAD, W. E. 2000. Survey of gain-scheduling analysis and design. *International Journal of Control,* 73**,** 1001-1025.

LEITH, D. J. & LEITHEAD, W. E. Global reconstruction of nonlinear systems from families of linear systems. *In:* BASAÑEZ, L. & PUENTE, J. A. D. L., eds. Proceedings of the 15th IFAC World Congress 2002 Barcelona, Spain. World Congress.

LEITH, D. J. & LEITHEAD, W. E. 2003. Necessary & Sufficient Conditions for the Minimal State-Space Realisation of Nonlinear Systems from Input-Output Information. 2003. 2937-2942.

LEITH, D. J., LEITHEAD, W. E., SOLAK, E. & MURRAY-SMITH, R. 2002. Divide & conquer identification using Gaussian process priors. 2002 2002. 624-629.

LENAERTS, V., KERSCHEN, G. & GOLINVAL, J.-C. 2001. Proper orthogonal decomposition for model updating of non-linear mechanical systems. *Mechanical Systems and Signal Processing,* 15**,** 31-43.

LEONTARITIS, I. J. & BILLINGS, S. A. 1985a. Input-output parametric models for non-linear systems Part I: deterministic non-linear systems. *International Journal of Control,* 41**,** 303-328.

LEONTARITIS, I. J. & BILLINGS, S. A. 1985b. Input-output parametric models for non-linear systems Part II: stochastic non-linear systems. *International Journal of Control,* 41**,** 329-344.

LEVY, S. & STEINBERG, D. 2010. Computer experiments: a review. *AStA Advances in Statistical Analysis,* 94**,** 311-324.

LI, J.-R. 2000. *Model reduction of large linear systems via low rank system gramians.* Massachusetts Institute of Technology.

LIANG, Y., LEE, H., LIM, S., LIN, W., LEE, K. & WU, C. 2002a. Proper orthogonal decomposition and its applications—Part I: Theory. *Journal of Sound and Vibration,* 252**,** 527-544.

LIANG, Y., LIN, W., LEE, H., LIM, S., LEE, K. & SUN, H. 2002b. Proper orthogonal decomposition and its applications–part II: Model reduction for MEMS dynamical analysis. *Journal of Sound and Vibration,* 256**,** 515-532.

LIANGYUE, C., YIGUANG, H., HAIPING, F. & GUOWEI, H. 1995. Predicting chaotic time series with wavelet networks. *Physica D: Nonlinear Phenomena,* 85**,** 225-238.

LJUNG, L. 1998. *System identification*, Springer.

LOHMANN, B. & SALIMBAHRAMI, B. 2000. Introduction to Krylov subspace methods in model order reduction. *Methods and Applications in Automation***,** 1-13.

MARTIN, J. D. & SIMPSON, T. W. 2005. Use of kriging models to approximate deterministic computer models. *AIAA journal,* 43**,** 853-863.

MARTONE, R., FORMISANO, A., HAWE, G. & SYKULSKI, J. 2007. A hybrid one-then-two stage algorithm for computationally expensive electromagnetic design optimization. *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering,* 26**,** 236-246.

MCLOONE, S. C., IRWIN, G. W. & MCLOONE, S. F. 2001. Constructing networks of continuous-time velocity-based models. *IEE Proceedings: Control Theory and Applications,* 148**,** 397-405.

MECKESHEIMER, M., BOOKER, A. J., BARTON, R. R. & SIMPSON, T. W. 2002. Computationally inexpensive metamodel assessment strategies. *AIAA journal,* 40**,** 2053-2060.

MENDEZ, E. M. A. M. & BILLINGS, S. A. 2001. An alternative solution to the model structure selection problem. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* 31**,** 597-608.

MOORE, B. 1981. Principal component analysis in linear systems: Controllability, observability, and model reduction. *Automatic Control, IEEE Transactions on,* 26**,** 17-32.

MURRAY-SMITH, R., JOHANSEN, T. A. & SHORTEN, R. 1999. On transient dynamics, off-equilibrium behaviour and identification in blended multiple model structures. *In:* FRANK, P. M. (ed.) *5th European Control Conference.* Karlsruhe, Germany: Springer.

NAHVI, S., NABI, M. & JANARDHANAN, S. 2013. Nonlinearity-aware sub-model combination in trajectory based methods for nonlinear Mor. *Mathematics and Computers in Simulation,* 94**,** 127-144.

NARENDRA, K. S. & PARTHASARATHY, K. 1990. Identification and control of dynamical systems using neural networks. *IEEE Trans Neural Netw,* 1**,** 4-27.

NEAL, R. M. 1996. *Bayesian learning for neural networks,* New York, Springer.

O'HAGAN, A. & KINGMAN, J. F. C. 1978. Curve Fitting and Optimal Design for Prediction. *Journal of the Royal Statistical Society. Series B (Methodological),* 40**,** 1-42.

PERNEBO, L. & SILVERMAN, L. 1982. Model reduction via balanced state space representations. *Automatic Control, IEEE Transactions on,* 27**,** 382-387.

PHILLIPS, J. R. & SILVEIRA, L. M. 2005. Poor man's TBR: a simple model reduction scheme. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* 24**,** 43-55.

RASMUSSEN, C. E. & NICKISCH, H. 2010. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research,* 11**,** 3011-3015.

RASMUSSEN, C. E. & WILLIAMS, C. K. I. 2006. *Gaussian processes for machine learning,* Cambridge, Mass., MIT Press.

REBERGA, L., HENRION, D., BERNUSSOU, J. & VARY, F. LPV modeling of a turbofan engine. Proc. 16th IFAC World Congress, Prague, Czech Republic,(CD-ROOM), 2005.

REED, J. A. & AFJEH, A. A. 2000. Computational simulation of gas turbines: part 1—Foundations of component-based models. *Journal of engineering for gas turbines and power,* 122**,** 366-376.

REWIEŃSKI, M. & WHITE, J. 2006. Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations. *Linear algebra and its applications,* 415**,** 426-454.

REWIENSKI, M. J. 2003. *A Trajectory Piecewise-Linear Approach to Model Order Reduction of Nonlinear Dynamical Systems.* Doctor of Philosophy, MASSACHUSETTS INSTITUTE OF TECHNOLOGY.

ROWLEY, C. 2005. Model reduction for fluids, using balanced proper orthogonal decomposition. *International Journal of Bifurcation and Chaos,* 15**,** 997-1013.

SACKS, J., WELCH, W. J., TOBY, J. M. & WYNN, H. P. 1989. Design and Analysis of Computer Experiments. *Statistical Science,* 4**,** 409-423.

SANDBERG, H. Model reduction of linear systems using extended balanced truncation. American Control Conference, 2008, 2008. IEEE, 4654-4659.

SHAMPINE, L. F. & REICHELT, M. W. 1997. The matlab ode suite. *SIAM journal on scientific computing,* 18**,** 1-22.

SHAN, S. & WANG, G. G. 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization,* 41**,** 219-241.

SHORTEN, R., MURRAY-SMITH, R., BJØRGAN, R. & GOLLEE, H. 1999. On the interpretation of local models in blended multiple model structures. *International Journal of Control,* 72**,** 620-628.

SIFAKIS, E. & BARBIC, J. FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction. ACM SIGGRAPH 2012 Courses, 2012. ACM, 20.

SIMPSON, T. W., BOOKER, A. J., GHOSH, D., GIUNTA, A. A., KOCH, P. N. & YANG, R.-J. 2004. Approximation methods in multidisciplinary analysis and optimization: a panel discussion. *Structural and multidisciplinary optimization,* 27**,** 302-313.

SIMPSON, T. W., PEPLINSKI, J. D., KOCH, P. N. & ALLEN, J. K. 2001. Metamodels for computer-based engineering design: survey and recommendations. *Engineering with Computers,* 17**,** 129-150.

SJÖBERG, J., ZHANG, Q., LJUNG, L., BENVENISTE, A., DELYON, B., GLORENNEC, P.-Y., HJALMARSSON, H. & JUDITSKY, A. 1995. Nonlinear black-box modeling in system identification: a unified overview. *Automatica,* 31**,** 1691-1724.

SNELSON, E. L. 2007. *Flexible and efficient Gaussian process models for machine learning.* Citeseer.

STORLIE, C. B., SWILER, L. P., HELTON, J. C. & SALLABERRY, C. J. 2009. Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models. *Reliability Engineering & System Safety,* 94**,** 1735-1763.

SUDRET, B. 2012. Meta-models for structural reliability and uncertainty quantification. *arXiv preprint arXiv:1203.2062*.

TOIVONEN, H. T., SANDSTRÖM, K. V. & NYSTRÖM, R. H. 2003. Internal model control of nonlinear systems described by velocity-based linearizations. *Journal of Process Control,* 13**,** 215-224.

TROCH, I. & BREITENECKER, F. 2000. Modeling and Simulation of Dynamic Systems. *Control Systems, Robotics and Automation,* IV.

VASILYEV, D., REWIENSKI, M. & WHITE, J. A TBR-based trajectory piecewise-linear algorithm for generating accurate low-order models for nonlinear analog circuits and MEMS. Design Automation Conference, 2003. Proceedings, 2003. IEEE, 490-495.

VASILYEV, D. M. 2007. *Theoretical and practical aspects of linear and nonlinear model order reduction techniques.* Massachusetts Institute of Technology.

VILLA-VIALANEIX, N., FOLLADOR, M., RATTO, M. & LEIP, A. 2012. A comparison of eight metamodeling techniques for the simulation of N 2 O fluxes and N leaching from corn crops. *Environmental Modelling & Software,* 34, 51-66.

WILLCOX, K. & PERAIRE, J. 2002. Balanced model reduction via the proper orthogonal decomposition. *AIAA journal,* 40, 2323-2330.

WILLIAMS, C. K. I. & RASMUSSEN, C. E. 1996. Gaussian processes for regression. *Advances in Neural Information Processing Systems 8 proceedings of the 1995 conference.* Cambridge, Mass.; London: MIT Press.

YU, D., ZHAO, H., XU, Z., SUI, Y. & LIU, J. 2011. An approximate non-linear model for aeroengine control. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 0954410011400959.

YUE, Y. & MEERBERGEN, K. 2012. Using Krylov-Padé model order reduction for accelerating design optimization of structures and vibrations in the frequency domain. *International Journal for Numerical Methods in Engineering,* 90, 1207-1232.

ZHOU, H., CEN, K. & FAN, J. 2004. Modeling and optimization of the NOx emission characteristics of a tangentially fired boiler with artificial neural networks. *Energy,* 29, 167-183.

ZHOU, Z., ONG, Y. S., LIM, M. H. & LEE, B. S. 2007. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing,* 11, 957-971.

Appendix A

A.1 Introduction

This appendix contains supplemental information about the nonlinear dynamical systems used as examples in the preparation of this thesis. Each section contains the nonlinear model, its mathematical description from first-principle, the calculation of its equilibrium points and Jacobians of the system at any operating point.

A.2 Two-Tanks NDS

System Description

The system is given

$$\begin{aligned}
\dot{x_1}(t) &= \frac{1}{A_1}\left(ku(t) - a_1\sqrt{2gx_1(t)}\right) \\
\dot{x_2}(t) &= \frac{1}{A_2}\left(a_1\sqrt{2gx_1(t)} - a_2\sqrt{2gx_2(t)}\right) \\
\boldsymbol{x}(t) &= [x_1(t) \quad x_2(t)]^T, y(t) = x_2(t) \\
\boldsymbol{x}(0) &= [0.05 \quad .1]^T
\end{aligned} \tag{A. 1}$$

$$a_1 = 0.02, a_2 = 0.015, A_1 = 0.5, A_2 = .25, g = 9.81 \textit{ and}$$
$$k = 0.005$$

The NDS in equation (A. 1) is re-arranged to the form below

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(\boldsymbol{x}(t), u(t))$$

$$= \begin{bmatrix} \frac{-a_1\sqrt{2g}}{A_1} & 0 \\ \frac{a_1\sqrt{2g}}{A_2} & \frac{-a_2\sqrt{2g}}{A_2} \end{bmatrix} \begin{bmatrix} \sqrt{x_1(t)} \\ \sqrt{x_2(t)} \end{bmatrix} + \begin{bmatrix} \frac{k}{A_1} \\ 0 \end{bmatrix} u(t) \qquad \text{(A. 2)}$$

$$y(t) = g(\boldsymbol{x}(t)) = [0 \quad 1]\boldsymbol{x}(t)$$

$u(t)$ is the input to the system with range $\{1 \leq u(t) \leq 10\}$, the input is a step like function and is discontinuous, the calculation in Appendix B was applied to approximate the input using a sigmoid like function.

System Equilibrium points

Let the input to the system at equilibrium be $\boldsymbol{u}_e$ and the state corresponding to that input $\boldsymbol{x}_e$ then the system at equilibrium is given by

$$\dot{\boldsymbol{x}_e} = \boldsymbol{F}(\boldsymbol{x}_e, \boldsymbol{u}_e) = 0 \qquad \text{(A. 3)}$$

For the Two Tanks system in equation (A. 1), at equilibrium the states are given by

$$\dot{x}_{e1} = \frac{1}{A_1}\left(ku_e - a_1\sqrt{2gx_{e1}}\right) = 0$$

$$\dot{x}_{e2} = \frac{1}{A_2}\left(a_1\sqrt{2gx_{e1}} - a_2\sqrt{2gx_{e2}}\right) = 0 \qquad \text{(A. 4)}$$

$$\dot{\boldsymbol{x}}_{\boldsymbol{e}} = [\dot{x}_{e1} \quad \dot{x}_{e2}]^T$$

Equation (A. 4) is solved to get the analytical expression for the equilibrium points state given by equation (A. 5).

$$x_{e1} = \frac{1}{2g}\left(\frac{ku_e}{a_1}\right)^2, x_{e2} = \frac{1}{2g}\left(\frac{ku_e}{a_2}\right)^2$$

$$\boldsymbol{x}_e = [x_{e1} \quad x_{e2}]^T \qquad \text{(A. 5)}$$

The equilibrium states corresponding to a certain input can be calculated using equation (A. 5) above.

System Jacobian Matrices

The LPV system parameters are given by

$$\boldsymbol{A}(t_i) = \nabla_{\boldsymbol{x}}\boldsymbol{F}\big(\boldsymbol{x}(t_i), u(t_i)\big) = \frac{1}{2}\begin{bmatrix} \frac{-a_1\sqrt{2g}}{A_1\sqrt{x_1(t_i)}} & 0 \\ \frac{a_1\sqrt{2g}}{A_2\sqrt{x_1(t_i)}} & \frac{-a_2\sqrt{2g}}{A_2\sqrt{x_2(t_i)}} \end{bmatrix}$$

$$\boldsymbol{b}(t_i) = \nabla_u\boldsymbol{F}\big(\boldsymbol{x}(t_i), u(t_i)\big) = \begin{bmatrix} \frac{k}{A_1} & 0 \end{bmatrix}^T \qquad \text{(A. 6)}$$

$$\boldsymbol{c}(t_i) = \nabla_{\boldsymbol{x}}g\big(\boldsymbol{x}(t_i)\big) = [0 \quad 1]$$

A.3 Continuously-Stirred Tank Reactor (CSTR) NDS

System Description

The system is given by

$$\dot{x_1}(t) = \frac{F}{V}\big(u_1(t) - x_1(t)\big) - k_0 x_1(t) e^{-\frac{E}{Rx_2(t)}}$$

$$\dot{x_2}(t) = \frac{F}{V}\big(u_2(t) - x_2(t)\big) - \frac{Hk_0}{H_D} x_1(t) e^{-\frac{E}{Rx_2(t)}} - \frac{H_A}{H_D V}\big(x_2(t) - u_3(t)\big) \quad \text{(A. 7)}$$

$$y_1(t) = x_1(t), y_2(t) = x_2(t)$$

$$x_1(0) = 8.5695, x_2(0) = 311.267$$

$F = 1, V = 1, k_0 = 35 \times 10^6, E = 11850, R = 1.98589, H = -5960, H_A = 145$ and $H_D = 480$

Equation (A. 7) can be simplified using the following set of parameters

$$p_1 = \frac{F}{V}, p_2 = -k_0, p_3 = -\frac{E}{R}, p_4 = \frac{H}{H_D}, p_5 = -\frac{H_A}{H_D V} \quad \text{(A. 8)}$$

Re-writing equation (A. 8) using the introduced parameters as

$$\dot{x_1}(t) = p_1\big(u_1(t) - x_1(t)\big) + p_2 x_1(t) e^{\frac{p_3}{x_2(t)}}$$

$$\dot{x_2}(t) = p_1\big(u_2(t) - x_2(t)\big) + p_2 p_4 x_1(t) e^{\frac{p_3}{x_2(t)}} + p_5\big(x_2(t) - u_3(t)\big) \quad \text{(A. 9)}$$

Let,

$$\boldsymbol{u}(t) = [u_1(t) \quad u_2(t) \quad u_3(t)]^T$$

$$\boldsymbol{x}(t) = [x_1(t) \quad x_2(t)]^T$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \quad \text{(A. 10)}$$

$$\boldsymbol{y}(t) = \boldsymbol{G}(\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) = [y_1(t) \quad y_2(t)]^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{x}(t)$$

$\boldsymbol{u}(t)$ is the input to the system, the input is a step like function and is discontinuous, the calculation in Appendix B was used to approximate the input using a sigmoid like function. The input is a group of three control inputs. The range of the inputs is $\{u_1(t) = 10, u_2(t) = 298\,, 273 \leq u_3(t) \leq 350\}$.

System Equilibrium points

Let the input to the system at equilibrium be $\boldsymbol{u}_e$ and the state corresponding to that input $\boldsymbol{x_e}$ then CSTR system at equilibrium is given by

$$\dot{\boldsymbol{x_e}} = \boldsymbol{F}(\boldsymbol{x_e}, \boldsymbol{u}_e) = 0 \qquad \text{(A. 11)}$$

For the CSTR system described by equation (A. 9), the equilibrium manifold is given by

$$\begin{aligned} \dot{x_{e1}} &= p_1(u_{e1} - x_{e1}) + p_2 x_{e1} e^{\frac{p_3}{x_{e2}}} = 0 \\ \dot{x_{e2}} &= p_1(u_{e2} - x_{e2}) + p_2 p_4 x_{e1} e^{\frac{p_3}{x_{e2}}} + p_5(x_{e2} - u_{e3}) = 0 \\ \dot{\boldsymbol{x}}_{\boldsymbol{e}} &= [\dot{x}_{e1} \quad \dot{x}_{e2}\,]^T \end{aligned} \qquad \text{(A. 12)}$$

Solving for $\dot{x_{e1}}$ results in

$$\begin{aligned} \dot{x_{e1}} &= \left[p_1(u_{e1} - x_{e1}) + p_2 x_{e1} e^{\frac{p_3}{x_{e2}}} = 0\right]\left(\frac{1}{p_1}\right) \\ u_{e1} - x_{e1} &+ \frac{p_2}{p_1} x_{e1} e^{\frac{p_3}{x_{e2}}} = 0 \\ x_{e1} &= \frac{u_{e1}}{1 - z_1 e^{\frac{p_3}{x_{e2}}}} \\ z_1 &= \frac{p_2}{p_1} \end{aligned} \qquad \text{(A. 13)}$$

Solving for $\dot{x_{e2}}$ results in

$$\left[\dot{x_{e2}} = p_1(u_{e2} - x_{e2}) + p_2 p_4 x_{e1} e^{\frac{p_3}{x_{e2}}} + p_5(x_{e2} - u_{e3}) = 0\right]\left(\frac{1}{p_1}\right)$$

$$u_{e2} - x_{e2} + z_1 p_4 x_{e1} e^{\frac{p_3}{x_{e2}}} + z_2(x_{e2} - u_{e3}) = 0 \qquad \text{(A. 14)}$$

$$z_2 = \frac{p_5}{p_1}$$

Substituting equation (A. 14) in equation (A. 13) and re-arranging to get

$$x_{e2}(z_2 - 1) + \frac{z_1 p_4 u_{e1} e^{\frac{p_3}{x_{e2}}}}{1 - z_1 e^{\frac{p_3}{x_{e2}}}} + z_3 = 0 \qquad \text{(A. 15)}$$

$$z_3 = u_{e2} - z_2 u_{e3}$$

Further re-arrange equation (A. 15) to get

$$x_{e2}(z_2 - 1) - \frac{p_4 u_{e1}}{1 - \frac{1}{z_1 e^{\frac{p_3}{x_{e2}}}}} + z_3 = 0 \qquad \text{(A. 16)}$$

Equation (A. 16) can be solved numerically to find x_{e2} value, then it can be substituted in equation (A. 13) to get x_{e1}.

System Jacobian Matrices

The LPV system parameters are given by

$$\boldsymbol{A}(t_i) = \nabla_{\boldsymbol{x}} \boldsymbol{F}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) = \begin{bmatrix} \frac{\partial \dot{x_1}(t_i)}{\partial x_1(t_i)} & \frac{\partial \dot{x_1}(t_i)}{\partial x_2(t_i)} \\ \frac{\partial \dot{x_2}(t_i)}{\partial x_1(t_i)} & \frac{\partial \dot{x_2}(t_i)}{\partial x_2(t_i)} \end{bmatrix}$$

$$\frac{\partial \dot{x_1}(t_i)}{\partial x_1(t_i)} = -p_1 + p_2 e^{\frac{p_3}{x_2(t_i)}}$$

$$\frac{\partial \dot{x_1}(t_i)}{\partial x_2(t_i)} = -p_2 p_3 \frac{x_1(t_i)}{x_2(t_i)^2} e^{\frac{p_3}{x_2(t_i)}} \qquad \text{(A. 17)}$$

$$\frac{\partial \dot{x_2}(t_i)}{\partial x_1(t_i)} = p_2 p_4 e^{\frac{p_3}{x_2(t_i)}}$$

$$\frac{\partial \dot{x_2}(t_i)}{\partial x_2(t_i)} = -p_1 - p_2 p_3 p_4 \frac{x_1(t_i)}{x_2(t_i)^2} e^{\frac{p_3}{x_2(t_i)}} + p_5$$

And,

$$\boldsymbol{B}(t_i) = \nabla_{\boldsymbol{u}} \boldsymbol{F}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) = \begin{bmatrix} p_1 & 0 & 0 \\ 0 & p_1 & -p_5 \end{bmatrix}$$

$$\boldsymbol{C}(t_i) = \nabla_{\boldsymbol{x}} \boldsymbol{G}\big(\boldsymbol{x}(t_i), \boldsymbol{u}(t_i)\big) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \text{(A. 18)}$$

A.4 Nonlinear Transmission Line NDS

System Description

The considered nonlinear transmission line system model of order n is given by

$$
\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{F}\big(\boldsymbol{x}(t), u(t)\big) \\
&= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{N}\big(\boldsymbol{x}(t)\big) + \boldsymbol{b}u(t) \\
y(t) &= g\big(\boldsymbol{x}(t)\big) = \boldsymbol{c}\boldsymbol{x}(t) \\
\boldsymbol{A} &= \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -1 \end{bmatrix} \in \mathbb{R}^{n\times n} \\
\boldsymbol{N}\big(\boldsymbol{x}(t)\big) &= \begin{bmatrix} 2 - e^{40x_1} - e^{40(x_1-x_2)} \\ e^{40(x_1-x_2)} - e^{40(x_2-x_3)} \\ \vdots \\ e^{40(x_{n-2}-x_{n-1})} - e^{40(x_{n-1}-x_n)} \\ e^{40(x_{n-1}-x_n)} - 1 \end{bmatrix} \in \mathbb{R}^{n\times 1} \\
\boldsymbol{b} &= \boldsymbol{c}^T = [1 \quad 0 \quad \cdots \quad 0] \in \mathbb{R}^{n\times 1} \\
\boldsymbol{x}(0) &= \boldsymbol{x}_0 = [0 \quad \cdots \quad 0] \in \mathbb{R}^{n\times 1} \\
u(t) &\in \mathbb{R}
\end{aligned}
\tag{A. 19}
$$

System Equilibrium points

Let the input to the system at equilibrium be u_e and the state corresponding to that input $\boldsymbol{x_e}$ then NTL system at equilibrium is given by

$$\dot{\boldsymbol{x_e}} = \boldsymbol{F}(\boldsymbol{x_e}, u_e) = 0 \tag{A. 20}$$

Examining equation (A. 19); $\big(\boldsymbol{x(t)}, \boldsymbol{u(t)}\big) = 0 \ \forall x_1 = x_2 = \cdots = x_n = x_e$; therefore the equilibrium manifold can be computed numerically by solving

$$\dot{x_e} = -x_e + \mathrm{e}^{40x_e} + 1 + u_e = 0 \qquad \text{(A. 21)}$$

System Jacobian Matrices

The Jacobians of the NTL system parameters are given by

$$\begin{aligned} \boldsymbol{A}(t_i) &= \nabla_{\boldsymbol{x}} \boldsymbol{F}\big(\boldsymbol{x}(t_i), u(t_i)\big) \\ \boldsymbol{b}(t_i) &= \nabla_u \boldsymbol{F}\big(\boldsymbol{x}(t_i), u(t_i)\big) \\ \boldsymbol{c}(t_i) &= \nabla_{\boldsymbol{x}} g\big(\boldsymbol{x}(t_i)\big) \end{aligned} \qquad \text{(A. 22)}$$

A.5 1-D Burgers Equations

System Description

The 1-D Burgers equations are partial differential equations given by

$$\frac{\partial x(d,t)}{\partial t} + \frac{\partial f\big(x(d,t)\big)}{\partial d} = \varphi(d) \qquad \text{(A. 23)}$$

With $f(x) = 0.5x^2$ and $\varphi(d) = 0.02e^{0.02d}$

The initial and boundary conditions are given

$$\begin{aligned} x(d,0) &= 1 \\ x(0,t) &= u(t) \end{aligned} \qquad \text{(A. 24)}$$

For all $x \in (0, L)$ and $t > 0$, $u(t)$ is the incoming flow and L is the length of the modelled region.

Applying spatial discretization to the length of modelled region at n nodes with $\Delta d = \frac{L}{n}$ results in n set of nonlinear ordinary differential equations and the nth order CI-NDS is given by

$$\begin{aligned}
\dot{\boldsymbol{x}}(t) &= \boldsymbol{F}\big(\boldsymbol{x}(t), u(t)\big) \\
&= \boldsymbol{\varphi}\big(\boldsymbol{x}(t)\big) + \boldsymbol{N}\big(\boldsymbol{x}(t)\big) + \boldsymbol{b}u(t)^2 \\
y(t) &= g\big(\boldsymbol{x}(t)\big) = \boldsymbol{c}\boldsymbol{x}(t) \\
\boldsymbol{\varphi} &= 0.02[e^{0.02\Delta d}, \dots, e^{0.02\Delta d}]^T \in \mathbb{R}^{n\times 1} \\
\boldsymbol{N}\big(\boldsymbol{x}(t)\big) &= \frac{1}{\Delta d}\begin{bmatrix} -0.5x_1(t)^2 \\ 0.5(x_1(t)^2 - (x_2(t)^2)) \\ \vdots \\ 0.5(x_{n-1}(t)^2 - (x_n(t)^2)) \end{bmatrix} \\
\boldsymbol{b} &= [\frac{1}{\Delta d} \quad 0 \quad \cdots \quad 0]^T \in \mathbb{R}^{n\times 1} \\
\boldsymbol{x}(0) &= \boldsymbol{x}_0 = [1 \quad \cdots \quad 1] \in \mathbb{R}^{n\times 1} \\
u(t) &\in \mathbb{R}
\end{aligned} \tag{A. 25}$$

$\boldsymbol{c}$ is a vector containing the required location of the node output.

System Equilibrium points

Let the input to the system at equilibrium be u_e and the state corresponding to that input $\boldsymbol{x_e}$ then NTL system at equilibrium is given by

$$\dot{\boldsymbol{x}_e} = \boldsymbol{F}(\boldsymbol{x_e}, u_e) = 0 \tag{A. 26}$$

The equilibrium states are given by

$$x_{e1} = \sqrt{2\Delta d(\varphi_1 + b_1 {u_e}^2)}$$

$$x_{ei} = \sqrt{{x_{e(i-1)}}^2 + 2\Delta d\varphi_i}\ \forall 2 \leq i \leq n \tag{A. 27}$$

System Jacobian Matrices

The Jacobians of the NTL system parameters are given by

$$\begin{aligned} \boldsymbol{A}(t_i) &= \nabla_{\boldsymbol{x}} \boldsymbol{F}\big(\boldsymbol{x}(t_i), u(t_i)\big) \\ \boldsymbol{b}(t_i) &= \nabla_u \boldsymbol{F}\big(\boldsymbol{x}(t_i), u(t_i)\big) \\ \boldsymbol{c}(t_i) &= \nabla_{\boldsymbol{x}} g\big(\boldsymbol{x}(t_i)\big) \end{aligned} \tag{A. 28}$$

Appendix B

B.1 Approximation of the Input Derivative

The thesis pointed out that the input signal derivative need to be calculated in order to use in the GP blended VBL-LPV system. Sometimes; the CI-NDS forcing input might be discontinuous like the in the case of a Heaviside step function. A step function given by

$$u(t-t_0) = \begin{cases} c_2 & \forall t > t_0 \\ c_1 & \forall t < t_0 \end{cases} \tag{B. 1}$$

c_1 and c_2 are constants that define the lower and upper limits of the step function and t_0 is an arbitrary value of time at which the change in the step occurs. This function is shown in Figure B. 1.

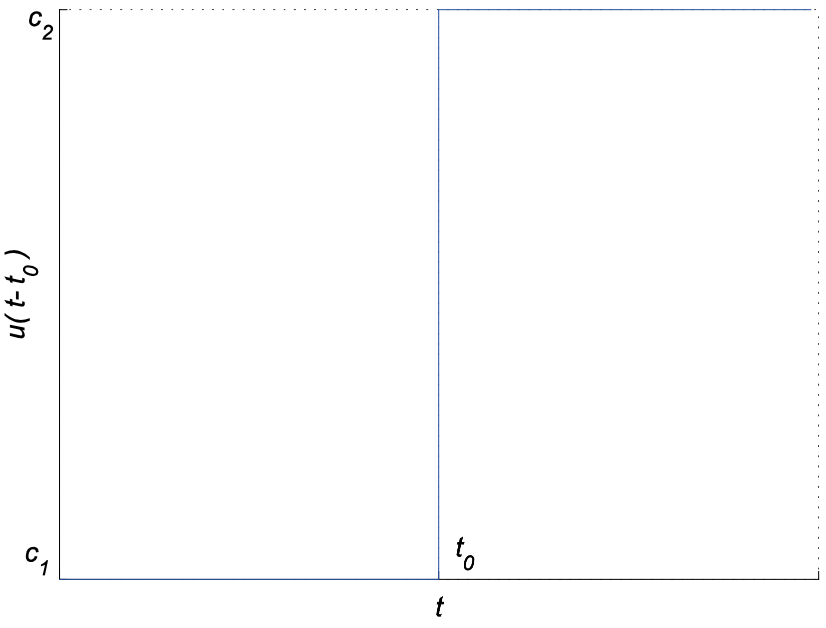


Figure B. 1 a Step function.

The derivative of this function w.r.t. time is given by

$$\frac{du(t-t_0)}{dt} = \begin{cases} 0 & \forall t > t_0 \\ \delta(0) = \infty & t = t_0 \\ 0 & \forall t < t_0 \end{cases} \qquad \text{(B. 2)}$$

$\delta(t)$ is the Dirac delta function.

Equation (B. 2) has a value of infinity at $t = t_0$. For applications where the derivative of a step function needs to be continuous over time, the study proposes to approximate the step function using a sigmoid function (Figure B. 2) given by

$$f(t-t_0) = c_1 + \frac{(c_2 - c_1)}{1 + e^{-2K(t-t_0)}} \qquad \text{(B. 3)}$$

K is a constant

Now examine equation (B. 3) to see how it behaves at different time values as in

$$\lim_{t\to-\infty} f(t-t_0) = \lim_{t\to-\infty} (c_1 + \frac{(c_2 - c_1)}{1 + e^{-2K(t-t_0)}}) = c_1$$

$$f(t-t_0)|_{t=t_0} = c_1 + \frac{(c_2 - c_1)}{1 + e^{-2K(0)}}) = \frac{c_1 + c_2}{2} \qquad \text{(B. 4)}$$

$$\lim_{t\to+\infty} u(t-t_0) = \lim_{t\to+\infty} (c_1 + \frac{(c_2 - c_1)}{1 + e^{-2K(t-t_0)}}) = c_2$$

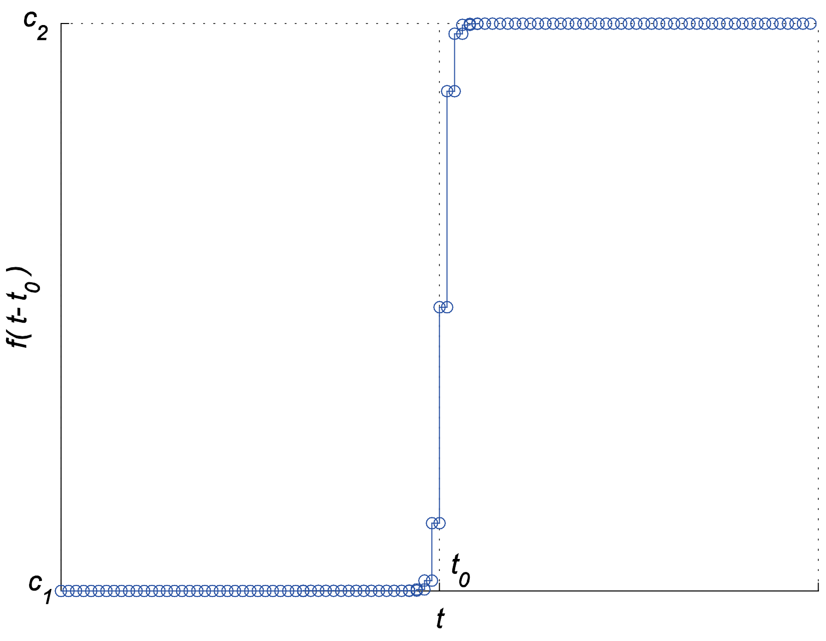


Figure B. 2 Step function approximation ($K = 1$).

From equation (B. 4) we can see how the constant K determine how fast f approach its lower and upper limits in both directions, the higher the value of K the faster the approach is. The derivative of equation (B. 3) is given by

$$\begin{aligned}\frac{df(t-t_0)}{dt} &= 0 + \frac{-(c_2-c_1)\frac{d}{dt}\left(1+e^{-2K(t-t_0)}\right)}{(1+e^{-2K(t-t_0)})^2} \\ &= \frac{2K(c_2-c_1)e^{-2K(t-t_0)}}{(1+e^{-2K(t-t_0)})^2}\end{aligned} \tag{B. 5}$$

Next examine equation (B. 5) behaviour with time in

$$\lim_{t\to-\infty}\frac{df(t-t_0)}{dt}=\lim_{t\to-\infty}\frac{2K(c_2-c_1)e^{-2K(t-t_0)}}{(1+e^{-2K(t-t_0)})^2}=0$$

$$\frac{df(t-t_0)}{dt}|_{t=t_0}=\frac{2K(c_2-c_1)e^{-2K(0)}}{(1+e^{-2K(0)})^2}=\frac{K(c_2-c_1)}{2} \qquad \text{(B. 6)}$$

$$\lim_{t\to+\infty}\frac{df(t-t_0)}{dt}=\lim_{t\to+\infty}\frac{2K(c_2-c_1)e^{-2K(t-t_0)}}{(1+e^{-2K(t-t_0)})^2}=0$$

It can be seen that the derivative of f w.r.t. time is continues and has a finite value defined at $t=t_0$.

B.2 Adapting the Approximation for Use with the Fixed Time Step Solver

This approximation was developed to be used with the proposed fixed-time step meta-model solver

Re-writing equation (B. 3) at fixed time instances ($t = kh, \{k = 0,1,2 \dots\}$) in

$$f\big((k-k_0)h\big)=c_1+\frac{(c_2-c_1)}{1+e^{-2K(k-k_0)h}} \qquad \text{(B. 7)}$$

k_0 is an arbitrary time sample

The derivative of (B. 7) w.r.t. discrete time was approximated and is given by

$$\frac{df((k-k_0)h)}{dkh} \approx \frac{f\big((k+1-k_0)h\big) - f\big((k-k_0)h\big)}{h}$$
$$= \frac{1}{h}\left(c_1 + \frac{(c_2-c_1)}{1+e^{-2K(k+1-k_0)h}} - c_1 - \frac{(c_2-c_1)}{1+e^{-2K(k-k_0)h}}\right)$$
$$= \frac{(c_2-c_1)}{h}\left(\frac{1}{1+e^{-2K(k+1-k_0)h}} - \frac{1}{1+e^{-2K(k-k_0)h}}\right) \tag{B. 8}$$

Equation (B. 8) performs similarly to equation (B. 5) when k approaches $(\pm)\infty$, and at time $k = k_0$ it has a value given by

$$\frac{df\big((k-k_0)h\big)}{dkh}\Big|_{k=k_0} \approx \frac{(c_2-c_1)}{h}\left(\frac{1}{1+e^{-2Kh}} - \frac{1}{1+e^{0}}\right)$$
$$= \frac{(c_2-c_1)}{2h}\left(\frac{1-e^{-2Kh}}{1+e^{-2Kh}}\right) \tag{B. 9}$$

The value of the constant K can be set to be $1/h$ to cancel the effect of the time step on the rising and settling time of the sigmoid function approximation.