MULTI-RESOLUTION METHODS FOR HIGH FIDELITY MODELING AND CONTROL ALLOCATION IN LARGE-SCALE DYNAMICAL SYSTEMS

A Dissertation

by

PUNEET SINGLA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2006

Major Subject: Aerospace Engineering

MULTI-RESOLUTION METHODS FOR HIGH FIDELITY MODELING AND CONTROL ALLOCATION IN LARGE-SCALE DYNAMICAL SYSTEMS

A Dissertation

by

PUNEET SINGLA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	John L. Junkins
Committee Members,	Srinivas R. Vadali
	John E. Hurtado
	Goong Chen
	Kamesh Subbarao
Head of Department,	Helen Reed

May 2006

Major Subject: Aerospace Engineering

ABSTRACT

Multi-Resolution Methods for High Fidelity Modeling and Control Allocation in Large-Scale Dynamical Systems. (May 2006)

Puneet Singla, B.Tech, Indian Institute of Technology, Kanpur, India;

M.S., Texas A&M University

Chair of Advisory Committee: Dr. John L. Junkins

This dissertation introduces novel methods for solving highly challenging modeling and control problems, motivated by advanced aerospace systems. Adaptable, robust and computationally efficient, multi-resolution approximation algorithms based on Radial Basis Function Network and Global-Local Orthogonal Mapping approaches are developed to address various problems associated with the design of large scale dynamical systems. The main feature of the Radial Basis Function Network approach is the unique direction dependent scaling and rotation of the radial basis function via a novel Directed Connectivity Graph approach. The learning of shaping and rotation parameters for the Radial Basis Functions led to a broadly useful approximation approach that leads to global approximations capable of good local approximation for many moderate dimensioned applications. However, even with these refinements, many applications with many high frequency local input/output variations and a high dimensional input space remain a challenge and motivate us to investigate an entirely new approach. The Global-Local Orthogonal Mapping method is based upon a novel averaging process that allows construction of a piecewise continuous global family of local least-squares approximations, while retaining the freedom to vary in a general way the resolution (e.g., degrees of freedom) of the local approximations. These approximation methodologies are compatible with a wide variety of disciplines such as continuous function approximation, dynamic system modeling, nonlinear sig-

nal processing and time series prediction. Further, related methods are developed for the modeling of dynamical systems nominally described by nonlinear differential equations and to solve for static and dynamic response of Distributed Parameter Systems in an efficient manner. Finally, a hierarchical control allocation algorithm is presented to solve the control allocation problem for highly over-actuated systems that might arise with the development of embedded systems. The control allocation algorithm makes use of the concept of distribution functions to keep in check the "curse of dimensionality". The studies in the dissertation focus on demonstrating, through analysis, simulation, and design, the applicability and feasibility of these approximation algorithms to a variety of examples. The results from these studies are of direct utility in addressing the "curse of dimensionality" and frequent redundancy of neural network approximation.

To my parents for their love, encouragement and belief in me

ACKNOWLEDGMENTS

First of all, I would like to express my heartfelt thanks to my advisor, Dr. John L. Junkins for introducing me to the diverse field of modeling and control of dynamical systems and for giving me complete freedom in selecting my research topic. His invaluable suggestions and guidance have helped me greatly in achieving my goals at Texas A&M and will continue to do so throughout my career. His commitment and dedication to work have often been my source of inspiration. Working with him has not only been a great learning experience but also an exhilarating one due to the open and cordial atmosphere he provided in all our interactions and discussions.

I also thank my committee members, mentors and collaborators Dr. John Hurtado, Dr. Kamesh Subbarao, Dr. Srinivas R. Vadali, Dr. Goong Chen, Dr. Daniele Mortari and Dr. John Valasek for many constructive conversations on various topics directly/indirectly related to my dissertation work. Special thanks are due to Dr. Subbarao for not only providing the constructive feedback that significantly increased the quality and robustness of the results presented in this dissertation but also for helping me out as a friend during my stay at Texas A&M. The opportunity to work with all of you has been a very pleasant and inspiring experience. I am also grateful to Lisa Willingham for her cooperation in conducting my final examination and helping me with many administrative hurdles during my stay at Texas A&M. I am indebted to Dr. Junkins for giving me this opportunity to work with a most inspiring, ambitious and helpful group of people.

It is my pleasure to acknowledge the support of my friends, Praveen Bhojwani, Sarbjyot Singh Dali, Hement Mahawar, Sreekanth Reddy, Gaurav Singhal, Rohit Singhal, Arun Surendaran, Sesha Sai Vaddi, Veera and several others, who have made my stay at College Station a memorable event in my life. I also remember the

great company of my deceased friends Kishore Naik and his wife Garishma during my initial stay at TAMU. They lived together a happy and compassionate life. Christian Brucccoleri, Anup Katake, Waqar Malik, and Prasenjit Sengupta all had the misfortune of getting involved in many ill-conditioned technical discussions with me over a cup of coffee. My CHAAPUCTOP friends from IITK have been a good source of support and encouragement throughout my graduate and under-graduate studies. Their close friendship and belief in my abilities provided me strength whenever I found my confidence drooping. Thanks are also due to Troy Henderson and Mrinal Kumar for helping me out in generating some nice figures for the dissertation.

I also offer my gratitude to my parents for providing me the best of opportunities in school and college education. I have great admiration for my parents who have sacrificed a lot to help me in achieving my goals. Coming from a rural area which had no good schools, I needed to stay away from home at an early age. This dissertation would have not been possible without their support, love and encouragement. I am grateful to my elder sister Preeti, brother-in-law Anoop Goyal and my cousin brothers Rajeev Singla and Sanjeev Singla for their support and encouragement and letting me concentrate on my studies only, while I was away from my home for my education.

Finally, I acknowledge the support of Texas Institute for Intelligent Bio-Nano Materials and Structures (TIIMS) for Aerospace Vehicles - a NASA University Research, Engineering and Technology Institute (URETI) for funding the main part of this research work. The technical liaison of Dr. Tom Gates is warmly acknowledged. This research in part was also supported by Lockheed Martin and Harris under the Defense Advanced Projects Agencys Innovative Space-Based Radar Antenna Technology (ISAT) program and by NASA Langley under the EO-3 Geostationary Imaging Fourier Transform Spectrometer (GIFTS) mission contract. The collaboration of Mr. Thomas Depkovich of Lockheed Martin for the ISAT program and Mr. Michael Jacox

for the GIFTS program is highly appreciated.

Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the authors and do not necessarily reflect the views of the funding agencies.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

In all branches of engineering, various system processes are generally characterized by mathematical models. Controller design, optimization, fault detection, and many other advanced engineering techniques are based upon mathematical models of various system processes. The accuracy of the mathematical models directly effect the accuracy of the system design and/or control. As a consequence, there is a great demand for the development of advanced modeling algorithms that can adequately represent the system behavior. However, different system processes have their own unique characteristics which they do not share with other structurally different systems. Obviously the mathematical structure of engineering models are very diverse, they can be simple algebraic models, may involve differential, integral or difference equations, and it may be a hybrid of these. Further, many different factors, like intended use of the model, problem dimensionality, quality of the measurement data, offline or online learning etc., can result in ad-hoc decisions leading to an unappropriate model architecture. All these issues make system modeling a challenging task and motivates us to seek more general and universal modeling methods that can be applied to a wide class of structurally different systems.

Over the past few decades, Artificial Neural Networks (ANNs) have emerged as a powerful set of tools in pattern classification, time series analysis, signal processing, dynamical system modeling and control. The popularity of the ANN can be attributed to the fact that these network models are frequently able to learn behavior when traditional modeling is very difficult to generalize. Typically, a neural network consists

---

This dissertation follows the style of *IEEE Transactions on Automatic Control*.

of several computational nodes called perceptrons arranged in layers. The number of hidden nodes essentially determines the degrees of freedom of the non-parametric model. A small number of hidden units may not be enough to capture a given system's complex input-output mapping and alternately a large number of hidden units may overfit the data and may not generalize the behavior. Beside this, it is also natural to ask *"How many hidden layers are required to model the input-output mapping?"*. The answer to this question in a general sense is provided by Kolmogorov's theorem [1] (later modified by other researchers [2]); according to which any continuous function from an input subspace to an appropriate output subspace can be approximated by a two layer neural network. However, the optimal number of hidden units depends upon many factors, like the ability of the chosen basis functions to approximate the given systems behavior, the number of data points, the signal to noise ratio, and the complexity of the learning algorithms. While ANNs are frequently described using network architecture terminology and diagrams, the reality is that the ANNs results in a set of parametric interpolation functions representing the input-output behavior. Different learning algorithms are proposed in the literature [3–7] that utilize two-layered neural networks with sigmoid functions as activation functions for system modeling purposes. However, the traditional ANN learning algorithms have serious short-comings, including:

1. *Abstraction*: the estimated weights do not have physical significance.

2. *Interpolation versus Extrapolation*: How do we know when a given estimated model is sufficiently well-supported such that the network has converged (locally or globally), and has utilized sufficiently dense and accurate measurements neighboring the desired evaluation point?

3. *Issues Affecting Practical Convergence*: A priori learning versus on-line adapta-

tion? Actually, when the ANN architecture is fixed a priori, then the family of solvable problems is implicitly constrained, that means the *architecture of the network should be learned*, not merely weights adjusted, to ensure efficient and accurate modeling of the particular system behavior.

4. *Uncertainty in Prediction:* There is no exiting methodology that satisfactory captures the uncertainty in the prediction of the system behavior.

In short, the learning methods described in the literature for neural networks seek to minimize the error between network output and observations globally based upon the assumption that all the parameters of the network can be optimized simultaneously. However, the global nature of the distortions can lead to globally optimal network parameters which may minimize the approximation error on the training set but might not be robust when tested on some new data points, or more importantly, used for prediction. The variability of a particular nonlinear system may be particularly non-uniform in space and time; some regions may be highly irregular and others may be smooth and linear. Furthermore, for the case that the neural network is itself nonlinear, the issue of sub-optimal convergence to local minima must also be considered. As a consequence, it is improbable that a globally nonlinear input-output mapping parameterization can be guessed a priori that represents such phenomena accurately and efficiently.

An alternative to global learning is local learning [8, 9] based upon a *divide and conquer* strategy. The local learning algorithms involve estimation of network parameters using the observations in the local neighborhood of the operating point. Local learning lead naturally to localized adaptation of the approximation degree of freedom to represent the variations actually present. A potential downside of employing local approximation methods is that they may be computationally expensive as we

need to solve the optimization problem in each local neighborhood. Obviously, we also have to face the possible discrepancies between adjacent and overlapping local approximations.

If one considers the problem of approximating surfaces in a general n-dimensional space then thousands of evaluation points are required and this can be the one of the main reason for the relatively small emphasis by mathematicians and engineers to pursue local approximation methods for discrete data approximation. In last two decades, the method of Moving Least Squares (MLS) [9–11] has emerged as a powerful local learning algorithm. The main drawback of the moving least squares approximation is that it is valid only in some neighborhood of one evaluation point and may introduce systematic error due to neglected interaction between different local models. Further, in MLS approximation basis functions used to obtain different local approximation can not be independent from each other without introducing discontinuity across the boundary of different local regions. Basically, a main challenge is the lack of rigorous methods to merge different independent local approximations to obtain a desired order globally continuous approximation.

Another major challenge in nonlinear approximation theory and its applications is *high dimensionality*. The performance of various traditionally used modeling algorithms decreases drastically as the dimension of the system increases. Also, the ease with which a mathematical model can be used in various post-processing computations such as controller design may determine the suitability of an approximation method for a particular problem at hand. For example, the use of a traditional ANN for dynamic system identification typically leads to a non-affine control problem due to their inherent nonlinear and complex structure, which is not desirable for controller design purposes. In summary, factors like approximation accuracy, total number of parameters required to express the mathematical model, the computation

time to compute various parameters of the model, complexity of the mathematical model and efficiency of the learning algorithm play crucial roles in determining the performance of a particular approximation method. Finally, while the successes have been many with existing modeling techniques, the drawbacks of various fixed architecture implementations, essentially, have created the demand for improved, adaptive modeling techniques that base adaptation on monitoring the "health" of the overall behavior input-output models and learning algorithms.

Our aim in this dissertation is to come up with novel network model structures which minimize the approximation error in a robust manner, while considering the above mentioned points. This dissertation is being written with following six main objectives:

1. The first and the most important objective is to present an adaptable, robust and computationally efficient, multi-resolution based approximation algorithm which takes care of local and as well as global complexity of the problem.

2. The second objective is to develop a novel approach to merge different local and global approximations that guarantees a prescribed degree of piecewise continuity, while keeping the "curse of dimensionality" in check.

3. The third objective is to present a new adaptive learning algorithm to adjust in real time the various parameters of the unknown mathematical model while keeping the number of unknowns to be minimum.

4. The fourth objective is to set down a theoretical approximation framework including all assumptions to help understand the advantages, the drawbacks and the areas of applications of the new algorithms.

5. The fifth objective is to compare new approximation algorithms with some ex-

isting approximation algorithms while considering various benchmark problems in open literature.

6. The last but not the least objective is to assess the reliability and limitations of the newly established approximation methods by considering various academic and engineering problems where traditional methods either fail or perform very poorly.

To achieve the above objectives and document the results, the dissertation is structured in six chapters.

Chapter II introduces a novel approach to adaptive approximation using radial basis functions. The approach introduces direction dependent scaling, shaping and rotation of the most generic Gaussian radial basis function for maximal trend sensing with minimal parameter representations for input output approximation. It is shown that shaping and rotation of each of the radial basis functions helps in reducing the total number of function units required to approximate any given input-output data, while greatly improving accuracy. While this novel radial basis function approach is a global method, the local optimization of the basis function leads to enhanced local convergence.

In Chapter III, a Global-Local Orthogonal MAPping (GLO-MAP) algorithm is introduced which is based upon a novel averaging process to determine a piecewise continuous global family of local least squares approximations while retaining the freedom to vary in a general way the resolution (e.g., degrees of freedom) of the local approximations. Also, the issues of model complexity, ill-conditioning of local least square approximations and curse of dimensionality are discussed in detail. The several ideas, discussed in this chapter, lay the foundation for rest of the dissertation, however; the numerical studies in this chapter just serve the purpose of demonstrating

the approximation capabilities of the GLO-MAP algorithm. Due to this reason, the reader can skip "Illustrative Engineering Applications" section in this chapter without any loss of continuity.

In Chapter IV, we make a transition from discussing numerical results of Chapter III to numerical analysis. In earlier chapters simulation results are used to validate the GLO-MAP algorithm, however, in this chapter, we discuss multi-resolution approximation capability and various other properties of the GLO-MAP algorithm from an analytical perspective.

Chapter V deals with the modeling of dynamical systems nominally described by nonlinear differential equations. A robust system identification algorithm is presented which makes combined use of existing linear system identification algorithms, such as the Eigensystem Realization Algorithm (ERA) or the Observer/Kalman IDentification (OKID), and a GLO-MAP based Artificial Neural Network (ANN). Being a combination of the ERA and the GLO-MAP algorithms, the resulting algorithm not only has the nonlinear approximation capability of ANN but also has model reduction capability of algorithms like Proper Orthogonal Decomposition (POD), in a setting where the system may be highly nonlinear.

In chapter VI, attention is focused on the use of the Galerkin discretization process and the GLO-MAP algorithm to solve for static and dynamic response of Distributed Parameter Systems (DPS) in an efficient manner. Two new meshless methods have been proposed based upon the GLO-MAP averaging process to solve for dynamics of DPS in an efficient way.

Finally, in Chapter VII, we consider the control allocation problem for a highly over-actuated systems which can arise with the development of embedded systems. Such an envisioned system can have quite a large number of actuators ($\sim 10^6$) which collectively produce the required control effort. While these systems are at present

futuristic, the advent of nano technology leads to a class of envisioned systems with multi-functional sensing and actuation engineered into materials at the micro and smaller length scales. The high dimensionality of the control distribution problem poses some challenges that are outside the reach of modern and classical control formulations. A recursive control distribution approach is discussed which makes use of adaptive distribution functions that can distribute control commands while the entire formulation remains compatible with real time computing.

CHAPTER II

DIRECTION DEPENDENT LEARNING APPROACH FOR RADIAL BASIS FUNCTION NETWORKS

A. Introduction

Radial Basis Function Networks (RBFN) are two-layer neural networks that approximate an unknown nonlinear function underlying given input-output data, as the weighted sum of a set of radial basis functions:

$$f(\mathbf{x}) = \sum_{i=1}^{h} w_i \phi_i(\|\mathbf{x} - \boldsymbol{\mu}_i\|) = \mathbf{w}^T \boldsymbol{\Phi}(\mathbf{x}, \boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_h) \tag{2.1}$$

where, $\mathbf{x} \in \mathcal{R}^n$ is an input vector, Φ is a vector of h radial basis functions with $\boldsymbol{\mu}_i \in \mathcal{R}^n$ as the center of i^{th} radial basis function and $\mathbf{w}$ is a vector of h linear weights or amplitudes. The two layers in an RBFN perform different tasks. The hidden layer with the radial basis function performs a non-linear transformation of the input space into a high dimensional hidden space whereas the outer layer of weights performs the linear regression of the function parameterized by this hidden space to achieve the desired approximation. The linear transformation, of Eq. (2.1) of a set of nonlinear basis functions is summarized in Cover's theorem [3] as follows,

Cover's Theorem. *A complex pattern classification problem or input/output problem cast in a high-dimensional space is more likely to be approximately linearly separable than in a low-dimensional space.*

Cover's theorem provides a theoretical motivation for using linear combination of a large number of nonlinear functions to approximate irregular phenomena. According to Cover and Kolmogorov's theorems [2, 3], Multilayered Neural Networks (MLNN) and RBFN can serve as "Universal Approximators" but in actuality, they offer no

guarantee on "accuracy in practice" for a reasonable dimensionality. While MLNN performs a global and distributed approximation at the expense of high parametric dimensionality, RBFN gives a global approximation but with locally dominant basis functions.

In recent literature [4, 12–14], various choices for radial basis functions are discussed. The Gaussian function is most widely used because, among other reasons, the arguments are the space of inputs and the associated parameters correlate to the local features and therefore all of the network parameters have physical and heuristic interpretations. These heuristic local interpretations lead directly to approximations that generate good starting estimates from local measurement data. The use of Gaussian functions to approximate given input-output data can be theoretically supported using the following nice characteristic of the Dirac-Delta function:

$$\delta(f) = \int_{-\infty}^{\infty} \delta_0(x) f(x) dx = f(0) \tag{2.2}$$

In other words, we can think of the above as "$f * \delta \rightarrow f$", where "$*$" denotes the convolution operator. Strictly speaking $\delta(x)$ is not a function but is a distribution [15]. Further, according to the following Lemma, such "localized bumps" can be well-approximated by Gaussian functions (illustrated in Fig. 1.):

Lemma 1. *Let* $\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ *and* $\phi_{(\sigma)}(x) = \frac{1}{\sigma}\phi(\frac{x}{\sigma})$. *If* $C_b(\mathcal{R})$ *denotes the set of continuous, bounded functions over* $\mathcal{R}$*), then*

$$\forall f \in C_b(\mathcal{R}), \quad \lim_{\sigma \rightarrow 0} \phi_{(\sigma)} * f(x) = \delta(f) \tag{2.3}$$

Proof. Let us consider, $|f(x) - \phi_{(\sigma)} * f(x)|$. Now using the fact that $\int \phi_{(\sigma)}(x)dx = 1$

and the definition of convolution, we have:

$$\begin{aligned} |f(x) - \phi_{(\sigma)} * f(x)| &= |\int \phi_{(\sigma)}(y) f(x) dy - \int \phi_{(\sigma)}(y) * f(x-y) dy| \\ &\leq \int |\phi_{(\sigma)}(y)||f(x) - f(x-y)| dy \end{aligned}$$

Since f is a continuous function, for any given $\epsilon > 0$, there is an $\eta > 0$ such that if $|y| < \eta$ then $|f(x) - f(x-y)| < \epsilon$. This yields the estimate

$$|f(x) - \phi_{(\sigma)} * f(x)| \leq \epsilon \int_{|y|<\eta} |\phi_{(\sigma)}(y)| dy + 2 f_{max} \int_{|y|\geq\eta} |\phi_{(\sigma)}(y)| dy$$

Further, let us compute

$$\int_{|y|\geq\eta} |\phi_{(\sigma)}(y)| dy = \frac{1}{\sigma} \int_{|y|\geq\eta} |\phi(\frac{y}{\sigma})| dy = \int_{|u|\geq\frac{\eta}{\sigma}} |\phi(u)| du$$

Now, this last term tends to 0 as σ tends to 0 since $\eta > 0$. Further, as f is a bounded continuous function, $|f(x) - \phi_{(\sigma)} * f(x)| < \epsilon$ as $\sigma \to 0$ and thus we obtain our desired result as ϵ can be chosen as small as we wish. □

So, theoretically, we can approximate any bounded continuous function with an infinite sum of Gaussian functions but practically, this may lead to very high dimensioned estimation problem. That said, one can always truncate this infinite sum to some finite number and learn the number of terms required along with other parameters of the Gaussian functions to minimize an appropriate approximation error norm i.e.

$$\inf_{\mathbf{p}} \left\{ \|f - \sum_{i=1}^{h} w_i \phi_i(\mathbf{x}, \mathbf{p})\| \right\} \tag{2.4}$$

where, $\mathbf{p}$ is a vector of following free network parameters needed to construct an RBFN:

1. Number of RBFs, h

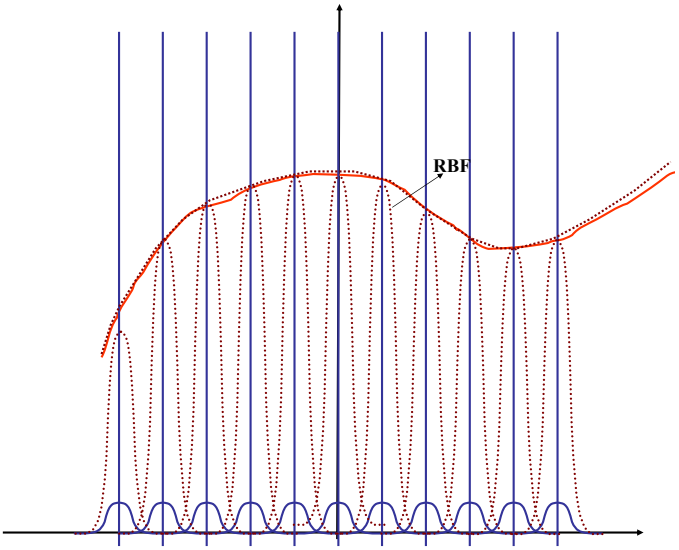


Fig. 1. Illustration of function approximation by localized bumps and RBF.

2. The centers of RBFs, μ_i

3. The spread of RBFs (σ_i in case of Gaussian function)

4. The linear weights between hidden layer and the output layer, w_i

Recently, Narcowich et al. [16] have found sobolev bounds on approximation error using RBF's as interpolates. More discussion on the approximation characteristics of RBF networks can be found in Refs [2, 3, 15, 17–19].

In this chapter, we seek to construct an adaptable, intelligent network that is designed such that it seeks to update/learn some or all of the above mentioned parameters. To learn various network parameters, different learning algorithms have been suggested in the literature [4–7, 12, 13, 20–22]. Further, adaptation of the architecture of an RBF network, as suggested in Refs. [5, 7, 12, 13, 21, 23], has lead to a new class of approximators suitable for multi-resolution modeling applications. While the adaptive nature of these algorithms aids in improving the resolution, it does not

necessarily help in the reduction of the number of basis functions required. For all available adaptive RBF networks, the network size can grow indefinitely if high accuracy is sought, due to the fact that the choice of the (fixed) basis function's shape and initial distribution over the input space may bear no correlation to the function to be represented. One important root difficulty for most of the methods in the existing literature lies in the fact that the basis functions are chosen to be circular (i.e. width of basis function is assumed to be same along each direction) and thus many neighboring circular functions of various sizes must ultimately add and subtract to approximate accurately even moderately non-circular features. In other words, the existing literature provides no consistent means for adaptive reshaping, scaling and rotation of non-circular basis functions to learn from current and past data points. The high degree of redundancy and lack of adaptive reshaping and scaling of RBF's are felt to be serious disadvantages of existing algorithms and provides the motivation for this chapter.

The objectives of this chapter are threefold. First, means for reshaping and rotation of Gaussian function are introduced to learn the local shape and orientation of the function measured by a given data set. The orientation of each radial basis function is parameterized through a rotation parameter vector, the magnitude of which for the two and three dimensional cases can be shown to be equal to the tangent of the half angle of the principal rotation angle [24]. The principal rotation vector defines the orientation of the principal axes of the quadratic coefficient function of the Gaussian RBF through parameterization of the direction cosine matrix. The shape is captured by solving independently for the principal axis scale factors. We mention that qualitatively, considering a sharp ridge or canyon feature in an input-output map, we can expect the principal axes of the local basis functions to approximately align along and perpendicular to the ridge. Secondly, an "intelligent" adaptation scheme is

proposed that learns the optimal shape and orientation of the basis functions, along with tuning of the centers and widths to enlarge the size of a single basis function as appropriate to approximate as much of the data possible. Thirdly, we modify existing learning algorithms to incorporate the concept of rotation and re-shaping of the basis functions to enhance their performance. This objective is achieved by modifying a conventional Modified Resource Allocating Network (MRAN) [12] learning algorithm.

The rest of the chapter is structured as follows: In the next section, the notion of rotation and shape optimization of a Gaussian function in the general case is introduced. Next, a novel learning algorithm is presented to learn the rotation parameters along with the parameters that characterize a regular RBFN. A modification to the MRAN algorithm is introduced to incorporate rotation of the Gaussian basis functions and finally, the results from various numerical studies are presented to illustrate the efficacy of the algorithm presented in this chapter.

B. Direction Dependent Approach

In Ref. [25], we introduce the concept of rotation of generally non-circular radial basis functions. Our approach of representing the rotation is motivated through developments in rigid body rotational kinematics [24]. The development is novel because we believe this represents the first application of the rotation ideas to the function approximation problem. We seek the optimal center location as well as rotation and shape for the Gaussian basis functions to expand coverage and approximately capture non-circular local behavior, thereby reducing the total number of basis functions required for learning. We mention that this approach can lead to most dramatic improvements when sharp "ridges" or "canyons" exist in the input-output map.

We propose adoption of the following most general n-dimensional Gaussian func-

tion:

$$\Phi_i(\mathbf{x}, \mu_i, \sigma_i, q_i) = \exp\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{R}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\} \tag{2.5}$$

Where, $\mathbf{R} \in \mathcal{R}^{n \times n}$ is a *fully populated* symmetric positive definite matrix instead of a diagonal one as in the case of the conventional Gaussian function representation used in various existing learning algorithms. The assumption of a diagonal $\mathbf{R}$ matrix is valid if the variation of output with x_j is uncoupled to x_k i.e. if different components of input vector are independent. In this case, the generalized Gaussian function reduces to the product of n independent Gaussian functions. However, if the parameters are not independent, there are terms in the resulting output that depend on off-diagonal terms of the matrix, $\mathbf{R}$. So it becomes useful to learn the off-diagonal terms of the matrix $\mathbf{R}$ for more accurate results (the local basis functions size, shape and orientation can be tailored adaptively to approximate the local behavior).

Now, using spectral decomposition the matrix $\mathbf{R}^{-1}$ can be written as a product of orthogonal matrices and a diagonal matrix:

$$\mathbf{R}_i^{-1} = \mathbf{C}^T(\boldsymbol{q}_i)\mathbf{S}(\boldsymbol{\sigma}_i)\mathbf{C}(\boldsymbol{q}_i) \tag{2.6}$$

Where $\mathbf{S}$ is a diagonal matrix containing the eigenvalues, σ_{i_k} of the matrix $\mathbf{R}_i$ which dictates the spread of the Gaussian function Φ_i and $\mathbf{C}(\boldsymbol{q}_i)$ is an $n \times n$ orthogonal rotation matrix consisting of eigenvectors of $\mathbf{R}^{-1}$. Now, it is easy to see that contour plots corresponding to a constant value of a generalized Gaussian function, Φ_i are hyperellipsoids in $\mathbf{x}$-*space*, given by following equation:

$$(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{R}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) = c^2 \text{ (a constant)} \tag{2.7}$$

Further, substituting for Eq. (2.6) in Eq. (2.7), we get an equation for an another

hyperellipsoid in a rotated coordinate system, $\mathbf{y}_i = \mathbf{C}(\mathbf{x} - \boldsymbol{\mu})$.

$$[\mathbf{C}(\boldsymbol{q}_i)(\mathbf{x} - \boldsymbol{\mu}_i)]^T \mathbf{S}(\boldsymbol{\sigma}_i) [\mathbf{C}(\boldsymbol{q}_i)(\mathbf{x} - \boldsymbol{\mu}_i)] = \mathbf{y}_i^T \mathbf{S}(\boldsymbol{\sigma}_i)\mathbf{y}_i = c^2 \text{ (a constant)} \tag{2.8}$$

From Eq. (2.8), we conclude that the orthogonal matrix, $\mathbf{C}$ represents the rotation of the basis function, Φ_i. Since the eigenvectors of the matrix $\mathbf{R}$ point in the direction of extreme principal axes of the data set, it naturally follows that learning the optimum rotation matrix, $\mathbf{C}$ (whose columns are the eigenvectors of $\mathbf{R}$) is most helpful in maximal local trend sensing. Though $\mathbf{C}(\boldsymbol{q}_i)$ is an $n \times n$ square matrix, we require only $n(n-1)/2$ parameters to describe it's most general variation due to the orthogonality constraint ($\mathbf{C}^T\mathbf{C} = \mathbf{I}$). So, in addition to the parameters that characterize a regular RBFN, we now have to account for the additional parameters characterizing the orthogonal rotation matrix making a total of $(n+2)(n+1)/2$ parameters for a minimal parameter description of the most general Gaussian function for an n input single output system. We will find that the apparent increase in the number of parameters is not usually a cause for concern because the total number of generalized Gaussian functions required for the representation typically reduces greatly, thereby bringing down the total number of parameters along with them. Also, we will see that the increased accuracy with a reduced number of RBFs provides a powerful heuristic argument for this approach. For each RBFN, we require the following parameters:

1. n parameters for the centers of the Gaussian functions i.e. $\boldsymbol{\mu}$.

2. n parameters for the spread (shape) of the Gaussian functions i.e. $\boldsymbol{\sigma}$.

3. $n(n-1)/2$ parameters for rotation of the principal axis of the Gaussian functions.

4. Weight w_i scaling $\phi_i(.)$'s contribution to the output.

To enforce the positive definiteness and symmetry constraint of matrix $\mathbf{R}$, we propose following three different parameterizations for the covariance matrix, $\mathbf{R}$.

1. To enforce the orthogonality constraint of the rotation matrix, $\mathbf{C}$, the following result in matrix theory that is widely used in rotational kinematics namely, the Cayley Transformation [24] is proposed:

   Cayley Transformation. *If* $\mathbf{C} \in \mathbb{R}^{n\times n}$ *is any proper orthogonal matrix and* $\mathbf{Q} \in \mathbb{R}^{n\times n}$ *is a skew-symmetric matrix then the following transformations hold:*

   (a) Forward Transformations

   i. $\mathbf{C} = (\mathbf{I} - \mathbf{Q})(\mathbf{I} + \mathbf{Q})^{-1}$

   ii. $\mathbf{C} = (\mathbf{I} + \mathbf{Q})^{-1}(\mathbf{I} - \mathbf{Q})$

   (b) Inverse Transformations

   i. $\mathbf{Q} = (\mathbf{I} - \mathbf{C})(\mathbf{I} + \mathbf{C})^{-1}$

   ii. $\mathbf{Q} = (\mathbf{I} + \mathbf{C})^{-1}(\mathbf{I} - \mathbf{C})$

   Remarkably, the forward and inverse transformations are identical. Since any arbitrary proper orthogonal matrix $\mathbf{C}$ (or skew-symmetric matrix $\mathbf{Q}$) can be substituted into the above written transformations, the Cayley Transformations can be used to parameterize the entire $\mathbf{O}(n)$ rotational group by skew symmetric matrices. The number of distinct elements in $\mathbf{Q}$ is precisely $n(n-1)/2$, so this is a minimal parameter representation. The forward transformation is always well behaved, however the inverse transformation encounters a difficulty only near the 180° rotation where $det\,(\mathbf{I} + \mathbf{C}) \to 0$. Thus $\mathbf{Q}$ is a unique function of $\mathbf{C}$ except at 180° rotation and $\mathbf{C}$ is always a unique function of $\mathbf{Q}$. Thus as per the Cayley transformation, we can parameterize the orthogonal matrix $\mathbf{C}(\mathbf{q}_i)$

in Eq. (2.6) as:

$$\mathbf{C}(\mathbf{q}_i) = (\mathbf{I} + \mathbf{Q}_i)^{-1}(\mathbf{I} - \mathbf{Q}_i) \tag{2.9}$$

where, $\mathbf{q}_i$ is a vector of $n(n-1)/2$ distinct elements of a skew symmetric matrix $\mathbf{Q}_i$ i.e. $\mathbf{Q}_i = -\mathbf{Q}_i^T$. Note $\mathbf{q}_i \to 0$ for $\mathbf{C} = \mathbf{I}$ and $-\infty \leq \mathbf{q}_i \leq \infty$ where $\mathbf{q}_i \to \pm\infty$ corresponds to a 180° rotation about any axis. Although using the Cayley transformation, the orthogonality constraint on the matrix $\mathbf{C}$ can be implicitly guaranteed, one still needs to check for the positive definiteness of $\mathbf{R}$ by requiring $\sigma_i > 0$.

2. We also introduce the following alternate minimal parameter representation of positive definite matrices that is motivated by the definition of a correlation matrix normally encountered in the theory of statistics.

Additive Decomposition. *Let* $\mathbf{R} \in \mathcal{R}^{n\times n}$ *be a symmetric positive definite matrix then* $\mathbf{R}^{-1}$ *is also symmetric and positive definite and can be written as a sum of a diagonal matrix and a symmetric matrix:*

$$\mathbf{R}_k^{-1} = \mathbf{\Gamma}_k + \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{e}_i\mathbf{e}_j^T q_{k_{ij}} \tag{2.10}$$

where $\boldsymbol{e}_i$ *is an* $n \times 1$ *vector with only the* i^{th} *element equal to one and rest of them zeros and* $\mathbf{\Gamma}_k$ *is a diagonal matrix given by:*

$$\mathbf{\Gamma}_k = \frac{1}{\sigma_k^2}\mathbf{I} \tag{2.11}$$

subject to following constraints:

$$q_{k_{ij}} = q_{k_{ji}} \tag{2.12}$$

$$\sigma_k > 0 \tag{2.13}$$

$$q_{k_{ii}} > 0 \tag{2.14}$$

$$-1 < \frac{q_{k_{ij}}}{(\sigma_k + q_{k_{ii}})(\sigma_k + q_{k_{jj}})} < 1 \tag{2.15}$$

It is worthwhile to mention that $q_{k_{ij}} \neq 0$ generates the stretching and rotation of the Gaussian function. If $q_{k_{ij}} = 0$ then we obviously obtain the circular Gaussian function. It is mentioned that even though the learning of the matrix, $\mathbf{R}$ is greatly simplified by this parameterization, one needs to impose the constraints defined in Eqs. (2.12)-(2.15) during the parameter learning process.

3. To explicitly enforce the positive definiteness and symmetry of the covariance matrix, $\mathbf{R}$ one could alternatively use the Cholesky decomposition [24]

   Cholesky Decomposition. *Let* $\mathbf{R} \in \mathcal{R}^{n \times n}$ *be a symmetric positive definite matrix then* $\mathbf{R}^{-1}$ *is also symmetric and positive definite and can be factored into a lower triangular matrix times its transpose such that:*

$$\mathbf{R}^{-1} = \mathbf{L}\mathbf{L}^T \tag{2.16}$$

where, $\mathbf{L}$ *is an lower triangular matrix given by following expression:*

$$\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix}$$

Notes: The Cholesky upper triangular matrix, $\mathbf{L}^T$, is also known as the matrix

square root of positive definite matrix, $\mathbf{R}^{-1}$. There are $n + \frac{n(n-1)}{2}$ distinct elements in $\mathbf{L}$, so Eq. (2.16) is a minimal parameter representation of $\mathbf{R}$.

The Cholesky decomposition-based parameterization of the matrix $\mathbf{R}$ is computationally more attractive than the other two parameterizations because the symmetry and positive definiteness properties of $\mathbf{R}^{-1}$ are explicitly enforced in this case to get rid of any kind of constraints. However, to our knowledge, the use of any of the three above parameterizations for aiding parameter updating in radial basis function network approximation applications is an innovation introduced in this dissertation. We have experimented with all three approaches and studies to date favor the Cholesky decomposition mainly because of programming convenience. Preliminary studies indicate a significant reduction in the number of basis functions required to accurately model unknown functional behavior of the actual input output data. In the subsequent sections, we report a novel learning algorithm and a modified version of the MRAN algorithm to learn this extended set of parameters, we also report the results of applications to five benchmark problems and comparison with existing algorithms.

C. Directed Connectivity Graph

A common main feature of the proposed learning algorithms is a judicious starting choice for the location of the RBFs via a Directed Connectivity Graph (DCG) approach which allows a priori adaptive sizing of the network for off-line learning and zeroth order network pruning. Because the Gaussian RBFN is a nonlnear representation, we know that finding the global minimum of approximation error is challenging; for this reason, means to initiate learning with a good approximation is very important. Direction dependent scaling and rotation of basis functions are initialized for maximal local trend sensing with minimal parameter representations and adaptation

of the network parameters is implemented to account for on-line tuning.

The first step towards obtaining a zeroth order off-line model is the judicious selection of a set of basis functions and their center locations, followed by proper initialization of the shape and orientation parameters. This exercise is the focus of this section.

To choose the locations for the RBF centers, we make use of following Lemma that essentially states that "*the center of a Gaussian function is an extremum point*".

Lemma 2. *Let* $\Phi(\mathbf{x}) : \mathcal{R}^n \rightarrow \mathcal{R}$ *represents a Gaussian function i.e.* $\Phi(\mathbf{x}) = \exp\Big(-(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{R}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big)$ *then* $\mathbf{x} = \boldsymbol{\mu}$ *is the only extremum point of* $\Phi(\mathbf{x})$ *i.e.* $\frac{d\Phi}{d\mathbf{x}}|_{\mathbf{x}=\boldsymbol{\mu}} = 0$. *Further,* $\mathbf{x} = \boldsymbol{\mu}$ *is the global maximum of* Φ

Proof. This Lemma is pretty obvious, but formally we see the gradient of Φ is

$$\frac{d\Phi}{d\mathbf{x}} = \exp\Big(-(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{R}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big)\mathbf{R}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{2.17}$$

Now, since $\mathbf{R}^{-1}$ is a positive definite symmetric covariance matrix, from equation (2.17), it is clear that $\frac{d\Phi}{d\mathbf{x}} = 0$ iff $\mathbf{x} = \boldsymbol{\mu}$. Further, it is easy to check that

$$\frac{d\log\Phi}{d\mathbf{x}} = -\mathbf{R}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \tag{2.18}$$

$$\nabla^2 \log\Phi(\mathbf{x}) = -\mathbf{R}^{-1} \tag{2.19}$$

Since $\frac{d\log\Phi}{d\mathbf{x}}|_{\mathbf{x}=\boldsymbol{\mu}} = 0$ and $\nabla^2\log\Phi(\mathbf{x}) < 0$ we conclude that $\mathbf{x} = \boldsymbol{\mu}$ is the only maximum point of $\log\Phi$. Since log is a monotonically increasing function of $(\mathbf{x}-\boldsymbol{\mu})$, so the center of the Gaussian function, $\boldsymbol{\mu}$ is also a global maximum point of the Gaussian function. □

Thus, from the above-mentioned Lemma, all the interior extremum points of the given surface data should naturally be the first choice for location of Gaussian functions with the $\mathbf{R}$ matrix determined to first order by the covariance of the data

confined in a judicious local mask around a particular extremum point. Therefore, the first step of the learning algorithm for an RBFN should be *to find the extremum points of a given input-output map.* It should be noticed that as the functional expression for the input-output map is unknown, to find the extremum points from discrete surface data, we need to check the necessary condition that first derivative of the unknown input-output map should be zero at each and every data point. We mention that the process of checking this condition at every data point is very tedious and computationally expensive. Note it is nt difficult to test for relative extrema of adjacent function values by direct comparison. Hence, we list the following Lemma 3 that provides an efficient way to find the extremum points of the given input-output map.

Lemma 3. *Let $f : \mathcal{X} \rightarrow \mathcal{R}$ be a continuous function, where $\mathcal{X}$ is a paracompact space with $\mathcal{U} = \{U_\alpha\}_{\alpha \in \mathcal{A}}$ as an open covering i.e. $\mathcal{X} \subset \cup_{\alpha \in \mathcal{A}} U_\alpha$. If $\mathcal{S}$ denotes the set of all extremum points of f then there exists a refinement, $\mathcal{V} = \{V_\beta\}_{\beta \in \mathcal{B}}$, of the open covering $\mathcal{U}$, such that $\mathcal{S} \subseteq \mathcal{W}$, where $\mathcal{W}$ is the set of the relative maxima and minima of f in open sets V_α.*

Proof. The proof of this lemma follows from the fact that the input space $\mathcal{X}$ is a paracompact space because it allows us to refine any open cover $\mathcal{U} = \{U_\alpha\}_{\alpha \in \mathcal{A}}$ of $\mathcal{X}$. Let $\mathcal{U} = \{U_\alpha\}_{\alpha \in \mathcal{A}}$ be the open cover of the input space $\mathcal{X}$. Further, assume that x_{max_α} and x_{min_α} define the maximum and minimum values of f in each open set $\mathcal{U}_\alpha$ respectively and $\mathcal{W}$ is the set of all such points i.e. $card(\mathcal{W}) = 2card(\mathcal{A})$. Now, we know that the set of all local maximum and minimum points of any function is the same as the set $\mathcal{S}$, of extremum points of that function. Further, without loss of generality we can assume that the set, $\mathcal{W}$, of all local maxima and minima of the function in each open set, $\mathcal{U}_\alpha$, is a subset of $\mathcal{S}$ because if it is not, then we can refine

the open cover $\mathcal{U}$ further until this is true. □

According to the above Lemma 3, for mesh sizes less than a particular value, the set $\mathcal{S}$, of the extremum points of the unknown input-output map f, should be subset of the set $\mathcal{W}$, consisting of the relative maxima and minima of the data points in each grid element. Now, the set $\mathcal{S}$, can be extracted from set $\mathcal{W}$ by checking the necessary condition that first derivative of f should be zero at extremum points. This way one need only approximate the first derivative of the unknown map at $2M$ points, where M, is the total number of elements in which data has been divided. It should be noticed that M is generally much smaller than the total number of data points available to approximate the unknown input-output map.

Further, to choose the centers from the set $\mathcal{S}$, we construct directed graphs $\mathcal{M}$ and $\mathcal{N}$ of all the relative maxima sorted in descending order and all the relative minima sorted in ascending order respectively. We then choose the points in $\mathcal{M}$ and $\mathcal{N}$ as candidates for Gaussian function centers with the extreme function value as the corresponding starting weight of the Gaussian functions. The centers at the points in $\mathcal{M}$ and $\mathcal{N}$ are introduced recursively until some convergence criteria is satisfied. The initial value of each local covariance matrix $\mathbf{R}$ is computed from statistical covariance of the data in a local mask around the chosen center. Now, using all the input data, we adapt the parameters of the chosen Gaussian functions and check the error residuals for the estimation error. If the error residuals do not satisfy a predefined bound, we choose the next set of points in the directed graphs $\mathcal{M}$ and $\mathcal{N}$ as center locations for additional Gaussian RBFs and repeat the whole process. The network only grows in dimensionality when error residuals can not be made sufficiently small, and thus the increased dimensionality grows only incrementally with the introduction of a judiciously shaped and located basis function. The initial location parameters

are simply the starting estimates for the learning algorithm; we show below that the combination of introducing basis functions sequentially and estimating their shape and location from local data to be highly effective.

1. Estimation Algorithm

The heart of any learning algorithm for RBFN is an estimation algorithm to adapt initially defined network parameters so that approximation errors are reduced to smaller than some specified tolerance. Broadly speaking, none of the nonlinear optimization algorithms available guarantee the global optimum will be achieved. Estimation algorithms based on the least squares criteria are the most widely used methods for estimation of the constant parameter vector from a set of redundant observations. According to the least square criteria, the optimum parameter value is obtained by minimizing the sum of squares of the vertical offsets ("Residuals") between the observed and computed approximations. In general, for nonlinear problems, successive corrections are made based upon local Taylor series approximations. Further, any estimation algorithm generally falls into the category of a Batch Estimator or a Sequential Estimator, depending upon the way in which observation data is processed. A batch estimator processes a usually large "batch" of data taken from a fixed span of the independent variable (usually time) to estimate the optimum parameter vector while a sequential estimator is based upon a recursive algorithm, which updates the parameter vector in a recursive manner after receipt of each observation. Due to their recursive nature, sequential estimators are preferred for real time estimation problems, however, batch estimators are usually preferable for offline learning.

To adapt the various parameters of the RBFN as defined in the previous section, we use an extended Kalman filter [26] for on-line learning while the Levenberg-Marquardt [27,28] batch least squares algorithm is used for off-line learning. Kalman

filtering is a modern (1960) development in the field of estimation [29, 30] though it has its roots as far back as in Gauss' work in the 1800's. In the present study, the algebraic version of the Kalman filter is used, since our model does not involve differential equations. On other hand, the Levenberg-Marquardt estimator, being the combination of *method of steepest descent* and *method of differential correction*, is a powerful batch estimator tool in the field of nonlinear least squares [29]. We mention that both the algorithms are very attractive for the problem at hand and details of both the algorithms can be found in Ref. [29]. Further, for some problems, the Kalman filter is attractive as a means to update the off-line a priori learned network parameters in real time whenever new measurements are available. The implementation equations for the extended Kalman filter or "*Kalman-Schmidt filter*" are given in Table I. To learn the different parameters of the RBFN using any estimation algorithm, the sensitivity (Jacobian) matrix $\mathbf{H}$ needs be computed. Since the covariance update is based upon an assumption of linearity, it is typically useful to impose a lower bound on the eigenvalues of $\mathbf{P}_k^+$ to keep the Kalman filter from becoming "too optimistic" and rejecting new measurements. The various partial derivatives required to synthesize the sensitivity matrix are outlined in subsequent subsections for all three parameterizations described in section B:

2. Cayley Transformation

In this case, the sensitivity matrix, $\mathbf{H}$, can be defined as follows:

$$\mathbf{H} = \frac{\partial f(\mathbf{x}, \mu, \sigma, \mathbf{q})}{\partial \mathbf{\Theta}} \tag{2.20}$$

Table I. Kalman-Schmidt filter.

Measurement Model	$$\tilde{\boldsymbol{y}} = \mathbf{h}(\mathbf{x}_k) + \boldsymbol{\nu}_k$$ with $$\begin{aligned} \mathbf{E}(\boldsymbol{\nu}_k) &= 0 \\ \mathbf{E}(\boldsymbol{\nu}_l \boldsymbol{\nu}_k^T) &= \mathbf{R}_k \delta(l-k) \end{aligned}$$
Update	$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \hat{\boldsymbol{x}}_k^+ &= \hat{\boldsymbol{x}}_k^- + \mathbf{K}_k (\tilde{\boldsymbol{y}} - \mathbf{H}_k \hat{\boldsymbol{x}}_k^-) \\ \mathbf{P}_k^+ &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^- \end{aligned}$$ where $$\mathbf{H}_k = \frac{\partial \mathbf{h}(\mathbf{x}_k)}{\partial \mathbf{x}}\Big|_{\mathbf{x}=\hat{\boldsymbol{x}}_k^-}$$

where, $f(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{q}) = \sum_{i=1}^{N} w_i \Phi_i(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i, \boldsymbol{q}_i)$ and $\boldsymbol{\Theta}$ is a $N \times \frac{(n+1)(n+2)}{2}$ vector given by:

$$\boldsymbol{\Theta} = \left\{ \begin{array}{ccccccccc} w_1 & \boldsymbol{\mu}_1 & \boldsymbol{\sigma}_1 & \boldsymbol{q}_1 & \cdots & w_N & \boldsymbol{\mu}_N & \boldsymbol{\sigma}_N & \boldsymbol{q}_N \end{array} \right\} \tag{2.21}$$

Here, $\mathbf{q}$ is a $n(n-1)/2$ vector used to parameterize the rank deficient skew-symmetric matrix $\mathbf{Q}$ in Eq. (2.9).

$$\begin{aligned} Q_{ij} &= 0,\ i = j \\ &= q_k\ i < j \end{aligned} \tag{2.22}$$

where, $k = \|i - j\|$ if $i = 1$ and $k = \|i - j\| + \|i - 1 - n\|$ for $i > 1$. Notice that the lower triangular part of $\mathbf{Q}$ can be formed using the skew-symmetry property of $\mathbf{Q}$. The partial derivatives required for the computation of the sensitivity matrix, $\mathbf{H}$ are obtained using Eqs. (2.5), (2.6) and (2.9), as follows:

$$\frac{\partial f}{\partial w_k} = \phi_k \tag{2.23}$$

$$\frac{\partial f}{\partial \boldsymbol{\mu}_k} = \left[w_k \phi_k \mathbf{R}_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right]^T \tag{2.24}$$

$$\frac{\partial f}{\partial \boldsymbol{\sigma}_{k_i}} = w_k \phi_k \frac{\mathbf{y}_i^2}{\sigma_{k_i}^3}, \mathbf{y}_i = \mathbf{C}_k (\mathbf{x} - \boldsymbol{\mu}_k),\ i = 1 \ldots n \tag{2.25}$$

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{q}_{k_l}} &= -\frac{w_k}{2} \phi_k \left[(\mathbf{x} - \boldsymbol{\mu}_k)^T \frac{\partial \mathbf{C}_k^T}{\partial \mathbf{q}_{k_l}} \mathbf{S}_k \mathbf{C}_k (\mathbf{x} - \boldsymbol{\mu}_k) + (\mathbf{x} - \boldsymbol{\mu}_k)^T \mathbf{C}_k^T \mathbf{S}_k \frac{\partial \mathbf{C}_k}{\partial \mathbf{q}_{k_l}} (\mathbf{x} - \boldsymbol{\mu}_k) \right], \\ & l = 1 \ldots n(n-1)/2 \end{aligned} \tag{2.26}$$

Further, the partial $\frac{\partial \mathbf{C}_k^T}{\partial \mathbf{q}_{k_l}}$ in Eq. (2.26) can be computed by substituting for $\mathbf{C}$ from Eq. (2.9):

$$\frac{\partial \mathbf{C}_k}{\partial \mathbf{q}_{k_l}} = \frac{\partial}{\partial \mathbf{q}_{k_l}} \left(\mathbf{I} + \mathbf{Q}_k \right)^{-1} \left(\mathbf{I} - \mathbf{Q}_k \right) + \left(\mathbf{I} + \mathbf{Q}_k \right)^{-1} \frac{\partial}{\partial \mathbf{q}_{k_l}} \left(\mathbf{I} - \mathbf{Q}_k \right) \tag{2.27}$$

Making use of the fact that $(\mathbf{I}+\mathbf{Q})^{-1}(\mathbf{I}+\mathbf{Q})=\mathbf{I}$, we get:

$$\frac{\partial}{\partial \mathbf{q}_{k_l}}(\mathbf{I}+\mathbf{Q}_k)^{-1} = -(\mathbf{I}+\mathbf{Q}_k)^{-1}\frac{\partial \mathbf{Q}_k}{\partial \mathbf{q}_{k_l}}(\mathbf{I}+\mathbf{Q}_k)^{-1} \tag{2.28}$$

substitution of Eq. (2.28) in Eq. (2.27) gives:

$$\frac{\partial \mathbf{C}_k}{\partial \mathbf{q}_{k_l}} = -(\mathbf{I}+\mathbf{Q}_k)^{-1}\frac{\partial \mathbf{Q}_k}{\partial \mathbf{q}_{k_l}}(\mathbf{I}+\mathbf{Q}_k)^{-1}(\mathbf{I}-\mathbf{Q}_k) - (\mathbf{I}+\mathbf{Q}_k)^{-1}\frac{\partial \mathbf{Q}_k}{\partial \mathbf{q}_{k_l}} \tag{2.29}$$

Now, Eqs. (2.23)-(2.26) constitute the sensitivity matrix $\mathbf{H}$ for the Extended Kalman Filter. We mention that although Eq. (2.6) provides a minimal parameterization of the matrix $\mathbf{R}$, we need to make sure that the scaling parameters denoted by σ_i are always greater than zero. So in case of any violation of this constraint, we need to invoke the parameter projection method to project inadmissible parameters onto the boundary of the set they belong to, thereby ensuring that the matrix $\mathbf{R}$ remains symmetric and positive definite at all times. Further, based on our experience with this parameterization, it is highly nonlinear in nature and sometimes causes unreliable convergence of the estimation algorithm. We found that this difficulty is alleviated by considering the two alternate representations, discussed earlier. We summarize the sensitivity matrices for these alternate parameterizations in the next subsections.

3. Additive Decomposition of the "Covariance" Matrix, $\mathbf{R}$

Using the additive decomposition for the $\mathbf{R}_i$ matrix in Eq. (2.5) the different partial derivatives required for synthesizing the sensitivity matrix $\mathbf{H}$ can be computed. We define following parameter vector $\mathbf{\Theta}$

$$\mathbf{\Theta} = \left\{ \begin{array}{cccccccc} w_1 & \boldsymbol{\mu}_1 & \sigma_1 & \mathbf{q}_1 & \cdots & w_N & \boldsymbol{\mu}_N & \sigma_N & \mathbf{q}_N \end{array} \right\} \tag{2.30}$$

The required partials with respect tot he elements of $\mathbf{\Theta}$ are then given as follows:

$$\frac{\partial f}{\partial w_k} = \phi_k \tag{2.31}$$

$$\frac{\partial f}{\partial \boldsymbol{\mu}_k} = \left[w_k \phi_k \mathbf{P}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]^T \tag{2.32}$$

$$\frac{\partial f}{\partial \sigma_{k_i}} = w_k \phi_k \frac{(\mathbf{x}_i - \boldsymbol{\mu}_{k_i})^2}{\sigma_{k_i}^3}, i = 1 \ldots n \tag{2.33}$$

$$\frac{\partial f}{\partial \mathbf{q}_{k_l}} = -w_k \phi_k (\mathbf{x}_i - \boldsymbol{\mu}_{k_i})^T (\mathbf{x}_j - \boldsymbol{\mu}_{k_j}),\ l = 1 \ldots n(n+1)\backslash 2, i, j = 1 \ldots n. \tag{2.34}$$

Thus, Eqs. (2.31)-(2.34) constitute the sensitivity matrix $\mathbf{H}$. It is to be mentioned that even though the synthesis of the sensitivity matrix is greatly simplified, one needs to check the constraint satisfaction defined in Eqs. (2.12)-(2.15) at every update. In case these constraints are violated, we once again invoke the parameter projection method to project the parameters normal to the constraint surface to nearest point on the set they belong to, thereby ensuring that the covariance matrix remains symmetric and positive definite at all times.

4. Cholesky Decomposition of "Covariance" Matrix, $\mathbf{R}$

Like in previous two cases, once again the sensitivity matrix, $\mathbf{H}$, can be computed by defining the parameter vector, $\mathbf{\Theta}$, as:

$$\mathbf{\Theta} = \left\{ \begin{array}{ccccccc} w_1 & \boldsymbol{\mu}_1 & \mathbf{l}_1 & \cdots & w_n & \boldsymbol{\mu}_n & \mathbf{l}_n \end{array} \right\} \tag{2.35}$$

where, $\mathbf{l}_i$ is the vector of elements parameterizing the lower triangular matrix, $\mathbf{L}$.

Carrying out the algebra the required partials can be computed as:

$$\frac{\partial f}{\partial w_k} = \phi_k \tag{2.36}$$

$$\frac{\partial f}{\partial \boldsymbol{\mu}_k} = \left[w_k \phi_k \mathbf{R}_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)\right]^T \tag{2.37}$$

$$\frac{\partial f}{\partial \mathbf{l}_{k_l}} = -\frac{w_k}{2}\phi_k \left[(\mathbf{x} - \boldsymbol{\mu}_k)^T \left(\frac{\partial \mathbf{L}_k}{\partial \mathbf{l}_{k_l}}\mathbf{L}_k^T + \mathbf{L}_k \frac{\partial \mathbf{L}_k^T}{\partial \mathbf{l}_{k_l}}\right)(\mathbf{x} - \boldsymbol{\mu}_k)\right], \quad l = 1 \ldots n(n-1)\backslash 2 \tag{2.38}$$

Further, $\mathbf{L}_k$ can be written as:

$$\mathbf{L}_k = \sum_{i=1}^{n}\sum_{j=i}^{n} e_i e_j L_{k_{ij}} \tag{2.39}$$

Therefore, $\frac{\partial \mathbf{L}_k}{\partial l_{k_l}}$ can be computed as:

$$\frac{\partial \mathbf{L}_k}{\partial l_{k_l}} = \sum_{i=1}^{n}\sum_{j=i}^{n} e_i e_j \tag{2.40}$$

Thus, Eqs. (2.36)-(2.38) constitute the sensitivity matrix $\mathbf{H}$. It is to be mentioned that unlike the Cayley transformation and the Additive decomposition, Cholesky decomposition guarantees the symmetry and positive definiteness of matrix, $\mathbf{R}$, without any additional constraints and so is more attractive for learning the matrix, $\mathbf{R}$.

It should be noted that although these partial derivatives are computed to synthesize the sensitivity matrix for the extended Kalman filter they are required in any case, even if a different parameter estimation algorithm is used (the computation of these sensitivity partials is inevitable).

Finally, the steps for implementing the Directed Connectivity Graph Learning Algorithm are summarized as follows:

Step 1 Find the interior extremum points i.e. global maximum and minimum of the given input-output data.

Step 2 Grid the given input space, $\mathcal{X} \in \mathcal{R}^n$ using hypercubes of length l.

Step 3 Find the relative maximum and minimum of given input-output data on the grid points in the region covered by each hypercube.

Step 4 Make a directed graph of all maximum and minimum points sorted in descending and ascending order respectively. Denote the directed graph of maximum points and minimum points by $\mathcal{M}$ and $\mathcal{N}$.

Step 5 Choose first point from graphs $\mathcal{M}$ and $\mathcal{N}$, denoted by $\mathbf{x}_M$ and $\mathbf{x}_N$ repectively, as candidates for Gaussian center and respective function values as the initial weight estimate of those Gaussian functions because at the center, the Gaussian function response is 1.

Step 6 Approximate the initial covariance matrix estimate, $\mathbf{R}$, directly from the statistical covariance matrix using the observations in a local mask around points $\mathbf{x}_M$ and $\mathbf{x}_N$.

Step 7 Parameterize the covariance matrix, $\mathbf{R}$, using one of the three parameterizations defined in section B.

Step 8 Use the Extended Kalman filter (Table I) or the Levenberg-Marquardt algorithm to refine the parameters of the network using the given input-output data.

Step 9 On each iteration, use parameter projection to enforce parametric constraints, if any, depending upon the covariance matrix decomposition.

Step 10 Check the estimation error residuals. If they do not satisfy the prescribed accuracy tolerance then choose the next point in the directed graphs $\mathcal{M}$ and $\mathcal{N}$ as the Gaussian center and restart at step 5.

The grid generation in step 2 is computationally costly, unless careful attention is paid to efficiency. To grid the input space $\mathcal{X} \in \mathcal{R}^n$, in a computationally efficient way, we designate a unique cell number to each input point in N^{th} decimal system, depending upon its coordinates in $\mathcal{R}^n$. Here, $N = \max\{N_1, N_2, \cdots, N_n\}$ and N_i denotes the number of cells required along i^{th} direction. The pseudo-code for the grid generation is given below:

Psuedo-code for grid generation

$for \quad ct = 1 : n$

$\quad xlower(ct) = min(inputdata(:, ct))$

$\quad xupper(ct) = max(inputdata(:, ct))$

end

deltax=(xupper-xlower)/N

$for \quad ct = 1 : Npoints$

$\quad cellnum(ct) = ceil((inputdata(ct, :) - xlower)./deltax)$

$\quad cellIndex(ct) = getindex(cellnum(ct))$

end

The relative maxima and minima in each cell are calculated by using all the data points with the same cell number. Though this process of finding the centers and evaluating the local covariance followed by the function evaluation with adaptation and learning seems computationally extensive, it helps in reducing the total number of Gaussian functions and therefore keeps the "*curse of dimensionality*" in check. Further, the rotation parameters and shape optimization of the Gaussian functions enables us to approximate the local function behavior with improved accuracy. Since we use the Kalman filter to refine the parameters of the RBF network, the selection of starting estimates for the centers can be made off-line with some training data and the

same algorithm can be invoked online as new measurements are processed to adapt the parameters from the off-line (a priori) network. Obviously, we can choose to constrain any subset of the network parameters, if necessary, to implicitly obtain a sub-optimal approximation but with reduce dimensionality. Any new Gaussian centers can be added to the existing network, these can be introduced based upon the statistical information of the approximation errors. Additional localization and reduction in the computational burden can be achieved by exploiting the local dominance near a given point by adjusting only a small subset of locally dominant RBFN parameters.

D. Modified Minimal Resource Allocating Algorithm (MMRAN)

In this section, we illustrate how the rotation parameters can be incorporated into existing RBF learning algorithms as well as the attractive consequences, by modifying the popular Minimal Resource Allocating Network (MRAN). To show the effectiveness of this modification, we include the rotation parameters also as adaptable parameters while keeping the same center selection and pruning strategy as in the conventional MRAN. For sake of completion, we give a brief introduction to MRAN and the reader should refer to Ref. [12] for more details (note that, MRAN is generally accepted as a significant improvement of the Resource Allocating Network (RAN) of Platt [5]). It adopts the basic idea of adaptively "growing" the number of radial basis functions where needed to null local errors, and also includes a "pruning strategy" to eliminate little-needed radial basis functions (those with weights smaller than some tolerance), with the overall goal of finding a minimal RBF network. RAN allocates new units as well as adjusts the network parameters to reflect the complexity of function being approximated. The problem of allocating RBF functions sequentially was stated as follows in Ref. [7]: *Given the prior approximation f^{n-1} and the new observation*

(x_n, y_n), how do we combine these two information sets to obtain the posterior approximation f^n? The optimal approximation for f^n is to add an impulse function at $\mathbf{x}_n$ to f^{n-1} which compensates for the difference in the estimated response and the actual response.

$$f^n(\mathbf{x}) = f^{n-1}(\mathbf{x}) + \delta_n(y_n - f^{n-1}(\mathbf{x}_n)) \tag{2.41}$$

This will ensure that the existing features of a prior network are maintained and error for the new added unit is zero. But such a solution lacks smoothness of the underlying function. We might anticipate that this approach is also prone to error when the new measurement contains measurement errors. Therefore, we use Gaussian functions centered at $\mathbf{x}_n$ instead of an impulse function to get a smooth approximation.

$$\phi_n(\mathbf{x}) = \exp(-\frac{1}{\sigma_n^2}\|\mathbf{x} - \mathbf{x}_n\|^2) \tag{2.42}$$

Let the number of hidden units required to approximate f^{n-1} be h then we can write:

$$\begin{aligned} f^n(\mathbf{x}) &= \sum_{i=1}^{h} w_i\phi_i(\mathbf{x}) + (y_n - f^{n-1}(\mathbf{x}_n))\phi_n(\mathbf{x}) \\ &= \sum_{i=1}^{h+1} w_i\phi_i(\mathbf{x}) \end{aligned} \tag{2.43}$$

Therefore the parameters associated with the new hidden unit are given as follows:

$$w_{h+1} = y_n - f^{n-1}(\mathbf{x}_n) \tag{2.44}$$

$$\boldsymbol{\mu}_{h+1} = \mathbf{x}_n \tag{2.45}$$

$$\sigma_{h+1} = \sigma_n \tag{2.46}$$

Heuristically, the estimated width of new Gaussian function, σ_n, is chosen in MRAN to be proportional to the shortest distance between $\mathbf{x}_n$ and the existing centers i.e.

$$\sigma_n = \kappa\|\mathbf{x}_n - \boldsymbol{\mu}_{nearest}\| \tag{2.47}$$

κ should be chosen judiciously to account for the amount of overlap between different Gaussian functions.

The main difficulty with this kind of approach is that we may go on adding new hidden units that contribute little to the final estimate. Therefore, a new hidden unit is actually added to existing network only if it satisfies following criteria [5]:

$$\|\mathbf{x}_i - \boldsymbol{\mu}_{nearest}\| > \epsilon \tag{2.48}$$

$$\|e_i\| = \|y_i - f(\mathbf{x}_i)\| > e_{min} \tag{2.49}$$

$$e_i^{rms} = \sqrt{\sum_{j=i-(N_w-1)}^{i} \frac{\|e_j\|^2}{N_w}} > e_{r_{min}} \tag{2.50}$$

Eq. (2.48) ensures that a new RBF node is added if it is sufficiently far from all the existing nodes. If the inequality of Eq. (2.49) is satisfied, then the approximation error using existing nodes meet the error specification and no new node is added. Eq. (2.50) takes care of noise in the observations by checking the sum squared error for past N_w observations. ϵ, e_{min} and $e_{r_{min}}$ are different thresholds which should be chosen appropriately to achieve desired accuracy.

If the above-mentioned criteria are not met, then the following network parameters are updated using the gradient descent approach or extended Kalman filter as suggested by Sundararajan [12].

$$\boldsymbol{\Theta} = \left\{ \begin{array}{ccccccc} w_1 & \boldsymbol{\mu}_1^T & \sigma_1 & \cdots & w_h & \boldsymbol{\mu}_h^T & \sigma_h \end{array} \right\} \tag{2.51}$$

Note, the advantages of MRAN over other learning algorithms can be summarized as follows.

- It is inherently sequential in nature and therefore can be used recursively in real-time to update the estimated model

- The network architecture itself is adapted in contrast to adjusting weights in a fixed architecture network. The ability of the network to capture the input-output behavior typically improves as more measurements are available.

The adaptive architecture feature and the inherent recursive structure of the learning algorithm makes this approach ideal for multi-resolution modeling [7, 23, 31]. While the methodology is very effective in some cases, it still suffers from the drawback of potential explosion in the number of basis functions utilized to approximate the functional behavior. A primary reason, we believe, for this is because the basis functions are traditionally chosen to be circular, though in some cases, the widths of the basis functions are adapted. While varying the width (sharpness) of the RBFs aids in improving the resolution, it still may not sufficiently help in the reduction of the number of basis functions required because many circular shaped basis functions are required to approximate sharp non-circular features.

To generalize the adaptation in the present study, we augment the parameter vector with a rotation parameter vector, $\mathbf{q}$ and different spread parameters, σ_{i_k} as described in section B.

$$\mathbf{\Theta} = \left\{ \begin{array}{cccccccc} w_1 & \boldsymbol{\mu}_1^T & \boldsymbol{\sigma}_1 & \mathbf{q} & \cdots & w_h & \boldsymbol{\mu}_h^T & \boldsymbol{\sigma}_h \quad \mathbf{q} \end{array} \right\} \tag{2.52}$$

Whenever a new node or Gaussian function is added to the MMRAN network, the corresponding rotation parameters are first set to zero and the spread parameters along different directions are assumed to be equal i.e. initially, the Gaussian functions are assumed to be circular.

The last step of the MRAN algorithm is the pruning strategy as proposed in Ref. [12]. The basic idea of the pruning strategy is to prune those nodes that contribute less than a predetermined number, δ, for S_w consecutive observations. Finally,

the modified MRAN algorithm (MMRAN) can be summarized as follow:

Step 1 Compute the RBF network output using following equation:

$$\mathbf{y} = \sum_{i=1}^{h} w_i \Phi_i(\mathbf{x}, \mathbf{\Theta}) \tag{2.53}$$

$$\Phi_i(\mathbf{x}, \mathbf{\Theta}) = \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \mathbf{R}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)\right) \tag{2.54}$$

Step 2 Compute different error criteria as defined in Eqs. (2.48)-(2.49).

Step 3 If all the error criteria hold then create a new RBF center with different network parameters assigned according to the following:

$$\mathbf{w}_{h+1} = \mathbf{e}_i \tag{2.55}$$

$$\boldsymbol{\mu}_{h+1} = \mathbf{x}_i \tag{2.56}$$

$$\boldsymbol{\sigma}_{h+1_k} = \kappa \|\mathbf{x}_i - \boldsymbol{\mu}_{nearest}\|, \ \forall k = 1, 2, \cdots, n \tag{2.57}$$

$$\mathbf{q} = 0 \tag{2.58}$$

Step 4 If all criterion for adding a new node to the network are not met, then update different parameters of the network using an EKF, as described in section 1.

Step 5 Remove those nodes of the RBF network that contribute negligibly to the output of the network for a certain number of consecutive observations.

E. Numerical Simulations and Results

The advantages of rotation and re-shaping the Gaussian basis functions are evaluated by implementing the DCG and modified MRAN algorithm using a variety of test examples in the areas of function approximation, chaotic time series prediction and dynamical system identification problems. Most of the test case examples are either

taken from the open literature or from the recently set up data modeling benchmark group [32] by IEEE Neural Network Council. In this section, we provide a comprehensive comparison of DCG and modified MRAN algorithm with various other conventional learning algorithms. At same time, these results also, demonstrate that the inclusion of rotation and re-shaping parameters significantly enhances the performance of the MRAN algorithm, for all five test problems.

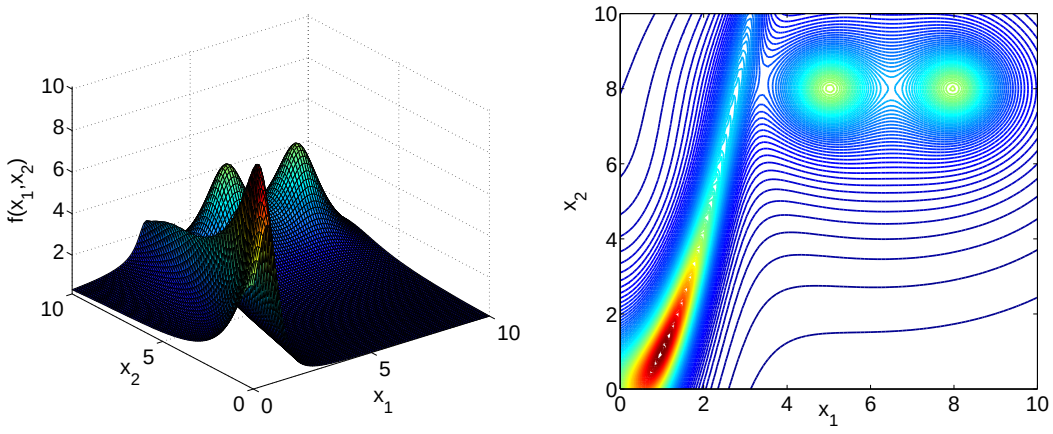
1. Test Example 1: Function Approximation

The first Test Example for the function approximation is constructed by using the following analytic surface function [33].

$$\begin{aligned} f(x_1, x_2) &= \frac{10}{(x_2 - x_1^2)^2 + (1 - x_1)^2 + 1} + \frac{5}{(x_2 - 8)^2 + (5 - x_1)^2 + 1} \\ &+ \frac{5}{(x_2 - 8)^2 + (8 - x_1)^2 + 1} \end{aligned} \tag{2.59}$$

Figs. 2(a) and 2(b) show the true surface and contour plots of the above functional expression respectively. According to our experience, this particular function has many important features including the sharp ridge that is very difficult to learn accurately with existing function approximation algorithms, with a reasonable number of nodes. To approximate the function given by Eq. (3.52), a training data set is generated by taking $10,000$ uniform random sampling in the interval [0-10] $\times$ [0-10] in the X_1-X_2 space while test data consists of $5,000$ other uniform samples of the interval [0-10] $\times$ [0-10].

To show the effectiveness of the rotation of Gaussian basis functions, we first use the standard MRAN algorithm without the rotation parameters, as discussed in Ref. [12]. Since the performance of MRAN algorithm depends upon the choice of various tuning parameters, several simulations were performed for various values of the tuning



(a) True surface plot (b) True contour plots

Fig. 2. True surface and contour plots for test example 1.

Table II. Various tuning parameters for MRAN and modified MRAN algorithms.

Algorithm	ϵ_{max}	ϵ_{min}	γ	e_{min}	$e_{r_{min}}$	κ	p_0	R	N_w	S_w	δ
Std. MRAN	3	1	0.66	0.002	0.0015	0.45	10^{-1}	10^{-5}	200	500	0.005
Mod.-MRAN	3	1.65	0.66	0.002	0.0015	0.45	10^{-1}	10^{-5}	200	500	0.005

Table III. Comparative results for test example 1.

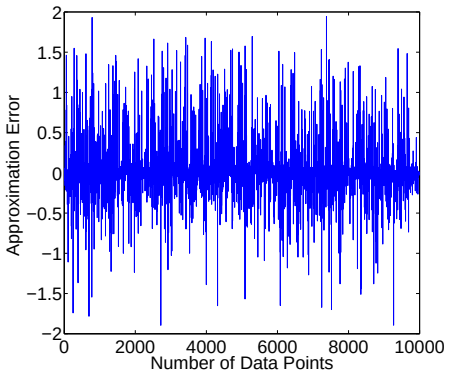
Algorithm	Mean Error	Std. Deviation (σ)	Max. Error	Number of Network Parameters
Std. MRAN	32×10^{-4}	0.1811	2.0542	280
Modified MRAN	6.02×10^{-4}	0.0603	0.7380	232
DCG	5.14×10^{-4}	0.0515	0.5475	144

parameters before selecting the tuning parameters (given in Table II) which gives us a suitably small approximation error. Figs. 3(a) and 3(b) show the approximation error for the training data set and the evolution of the number of centers with number of data points. From these figures, it is clear that approximation errors are quite high even for the training data set, even though the number of Gaussian functions settled down to 70 approximately after 3000 data points. Further, Figs. 3(c) and 3(d) show the approximated test surface and contours plots respectively, whereas Figs 3(e) and 3(f) show the percentage error surface and error contour plots corresponding to test data respectively. From these figures, it is apparent that approximation errors are pretty large ($\approx$ 15%) along the knife edge of the sharp ridge line while they are $<$ 1% in other regions. Actually, this is also the reason for the high value of the

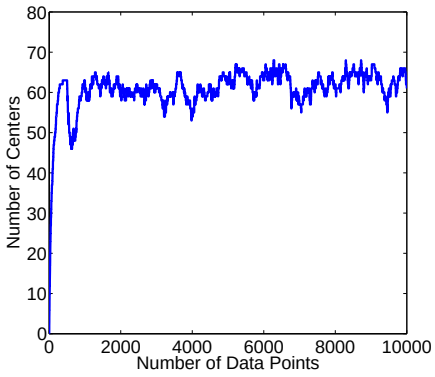
standard deviation of the approximation error for MRAN in Table III, the errors along the sharp ridge dominate the statistics. The failure of MRAN type learning algorithms in this case can be attributed directly to the *inability of the prescribed circular Gaussian basis function* to approximate the sharp ridge efficiently.

Further, to show the effectiveness of the shape and rotation parameters, we modify the MRAN algorithm, as discussed in section D, by simply including the shape and rotation parameters also as adaptable while keeping the same center selection and pruning strategy. The modified MRAN algorithm is trained and tested with the same training data sets that we used for the original algorithm. In this case too, a judicious selection of various tuning parameter is made by performing a few different preliminary simulations and selecting final tuning parameters (given in Table II) which give us a near-minimum approximation error. Figs. 4(a) and 4(b) show the approximation error for the training data set and the evolution of number of centers with the number of data points. *From these figures, it is clear that by learning the rotation parameters, the approximation errors for the training data set is reduced by an almost order of magnitude whereas the number of Gaussian functions is reduced by half.* It should be noted, however, that $\sim 50\%$ reduction in number of Gaussian functions corresponds to only a 17% reduction in the number of network parameters to be learned. Figs. 4(c) and 4(d) show the approximated surface and contours plots respectively whereas Figs 4(e) and 4(f) show the percentage error surface and error contour plots respectively. As suspected, the approximation errors are significantly reduced ($\approx 5\%$) along the knife edge of the sharp ridge line while they are still $< 1\%$ in other regions. From Table III, it is apparent that the mean and standard deviation of the approximation errors are also reduced very significantly.

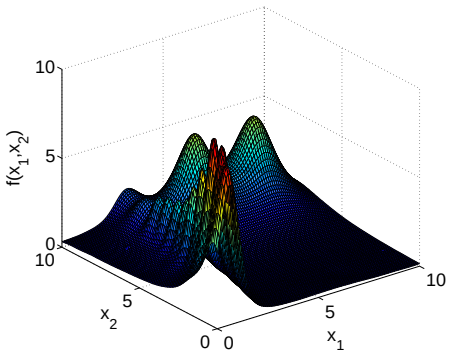
Finally, the DCG algorithm, proposed in section C, is used to approximate the analytical function given by Eq. (3.52). As mentioned in section C, we first divide



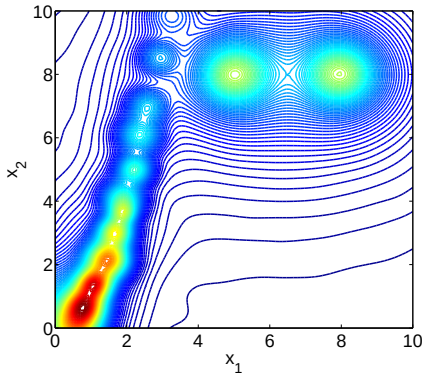
(a) Training data set error plot



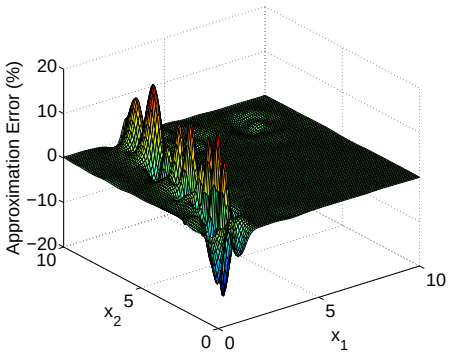
(b) Number of RBFs vs number of data points



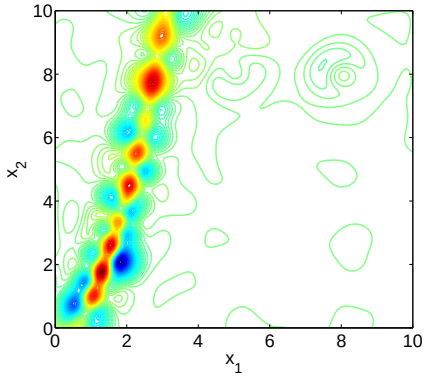
(c) Estimated surface plot



(d) Estimated contour plots

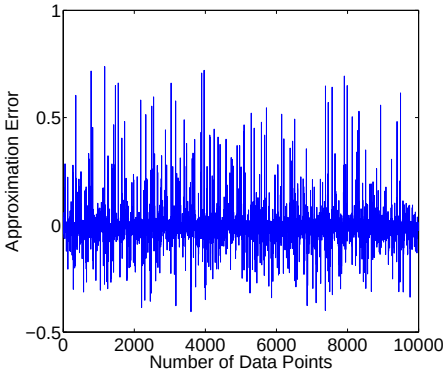


(e) Approximation error surface (%) plot

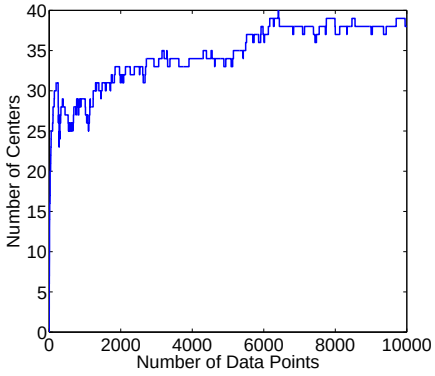


(f) Approximation error contour plots

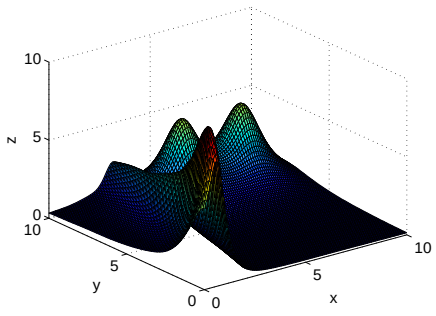
Fig. 3. MRAN approximation results for test example 1.



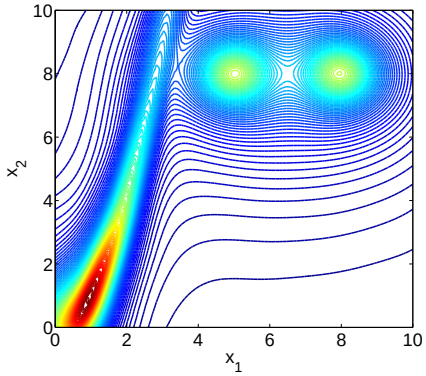
(a) Training data set error plot



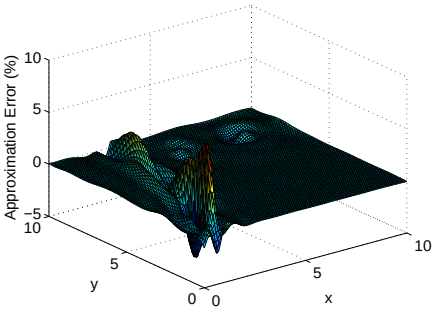
(b) Number of RBFs vs number of data points



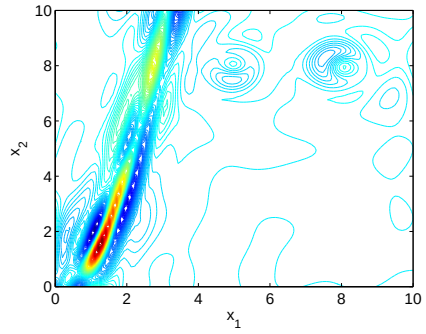
(c) Estimated surface plot



(d) Estimated contour plots



(e) Approximation error surface (%) plot

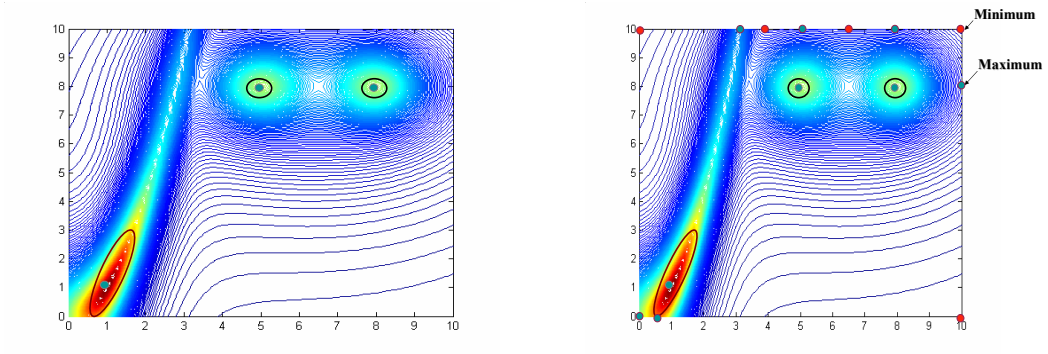


(f) Approximation error contour plots

Fig. 4. Modified MRAN approximation results for test example 1.

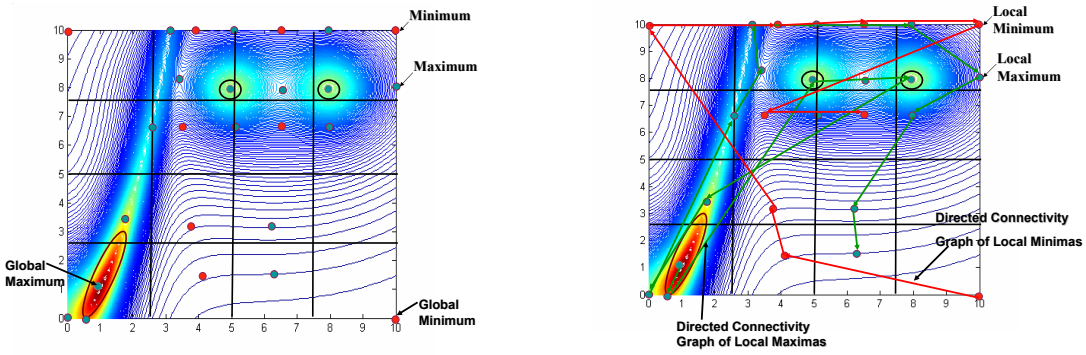
the whole input region into a total of 16 square regions (4×4 cells); this decision was our first trial, better results might be obtained by tuning. Then we generated a directed connectivity graph of the local maxima and minima in each sub-region that finally lead to locating and shaping the 24 radial basis functions that, after parameter optimization gave approximation errors less than 5%. This whole procedure is illustrated in Fig. 5. The DCG algorithm is also trained and tested with the same data sets that we use for the MRAN algorithm training and testing. Figs. 6(a) and 6(b) show the estimated surface and contour plots respectively for the test data. From these figures, it is clear that we are able to learn the analytical function given in Eq. (3.52) very well. In Fig. 6(b) the circular ($\circ$) and asterisk ($*$) marks denote the initial and final positions (after learning process is over) of the Gaussian centers. As expected, initially the center locations cover the global and local extremum points of the surface and finally some of those centers, shape and rotation parameters move a significantly. The optimum location, shape, and orientation of those functions along the sharp ridge are critical to learn the surface accurately with a small number of basis functions. Figs. 6(c) and 6(d) show the error surface and error contour plots for the DCG approximated function. From Fig. 6(c), it is clear that approximation errors are less than 5% whereas from Fig. 6(d) it is clear that even though we have approximated the sharp surface very well, the largest approximation errors are still confined to the vicinity of the ridge. Clearly, we can continue introducing local functions along the ridge until the residual errors are declared small enough. Already, however, advantages relative to competing methods are quite evident (the smallest approximation error and the fewest number of network parameters).

For comparison sake, the mean approximation error, standard deviation of approximation error and total number of network parameters learned are listed in Table III for MRAN (with and without rotation parameters) and DCG algorithms. From



(a) Step 1: Locate interior extremum points

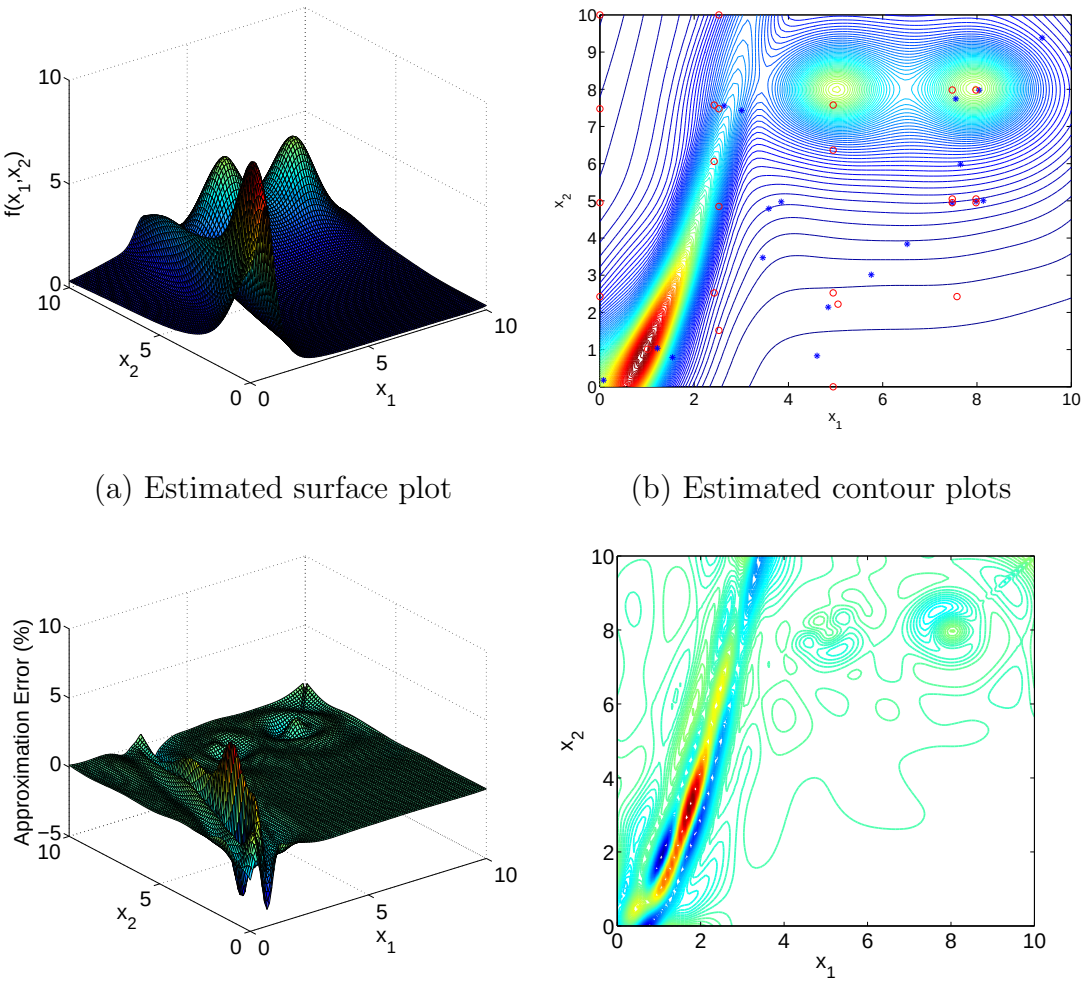
(b) Step 2: Locate extremum points along the boundary of input-space



(c) Step 3: Locate local extremum points

(d) Step 4: Make a DCG of local extremum points

Fig. 5. Illustration of center selection in the DCG network.



(a) Estimated surface plot

(b) Estimated contour plots

(c) Approximation error surface plot (%)

(d) Approximation error contour plots

Fig. 6. DCG simulation results for test example 1.

these numbers, it is very clear that the mean approximation error and standard deviation decreases by factors from three to five if we include the rotation and shape parameters. Further, this reduction is also accompanied by a considerable *decrease* in number of learned parameters required to define the RBFN network in each case. It is noted that this very substantial improvement in performance of the modified MRAN algorithm over the standard MRAN can be attributed directly to the *inclusion of shape and rotation parameters*, because the other parameter selections and learning criteria for the modified MRAN algorithm are held the same as for the original MRAN algorithm. Although, there is not much difference between the modified MRAN and DCG algorithm results, in terms of accuracy, in the case of the DCG algorithm, a total of only 144 network parameters are required to be learned as compared to 232 in case of the modified MRAN. This 33% decrease in number of network parameters to be learned in the case of the DCG can be attributed to the *judicious selection of centers*, using the graph of maxima and minima, and the avoidance of local convergence to sub-optimal values of the RBF parameters. It is anticipated that persistent optimization and pruning of the modified MRAN may lead to results comparable to the DCG results. In essence DCG provides more nearly the global optimal location, shape and orientation parameters for the Gaussian basis functions to start the modified MRAN algorithm.

2. Test Example 2: 3 Input- 1 Output Continuous Function Approximation

In this section, the effectiveness of the shape and rotation parameters is shown by comparing the modified MRAN and DCG algorithms with the *Dependence Identification* (DI) algorithm [34]. The DI algorithm bears resemblance to the boolean network construction algorithms and it transforms the network training problem into a set of quadratic optimization problems that are solved by a number of linear equations.

The particular test example considered here is borrowed from Ref. [34] and involves the approximation of a highly nonlinear function given by following equation:

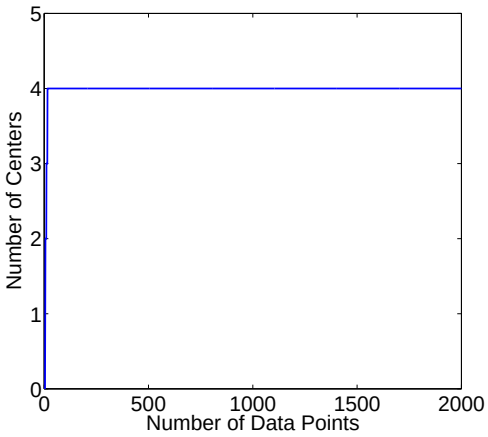
$$y = \frac{1}{10}\left(e^{x_1} + x_2 x_3 \cos(x_1 x_2) + x_1 x_3\right) \tag{2.60}$$

Here, $x_1 \in [0, 1]$ and $x_2, x_3 \in [-2, 2]$. We mention that in Ref. [12], Sundarajan et al. compared MRAN algorithm with DI algorithm. Like in Ref. [12, 34], the input vector for MMRAN and DCG is $\mathbf{x} = \left\{ \begin{array}{ccc} x_1 & x_2 & x_3 \end{array} \right\}^T$ and the training data set for network learning is generated by taking 2000 uniformly distributed random values of the input vector and calculating the associated value of y according to Eq. (2.60). The several tuning parameters for the MMRAN algorithm are given in Table IV.

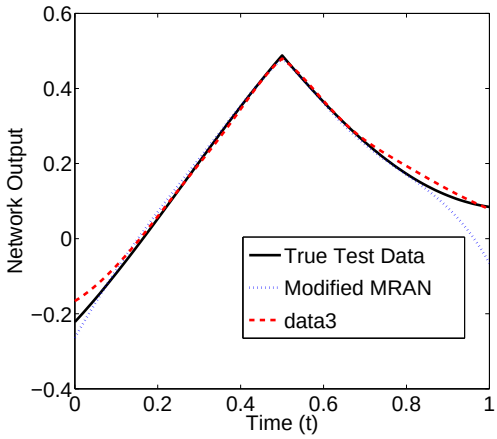
Table IV. Various tuning parameters for modified MRAN algorithm for test example 2.

Algorithm	ϵ_{max}	ϵ_{min}	γ	e_{min}	$e_{r_{min}}$	κ	p_0	q_0	N_w	S_w	δ
MMRAN	3	0.3	0.97	0.002	0.12	0.70	1	10^{-1}	10^2	2000	10^{-4}

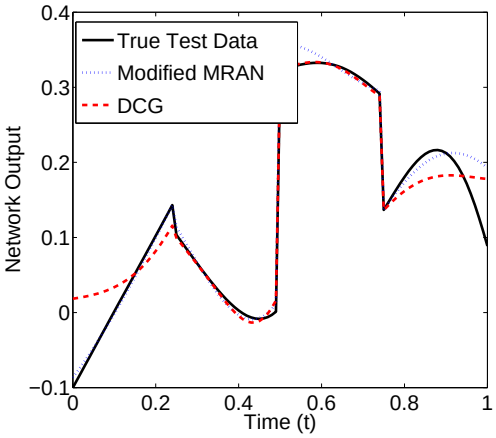
Fig. 7(a) shows the growth of the modified MRAN network. In case of the DCG network, the whole input space is divided into $2 \times 2 \times 2$ grid so giving us a freedom to choose the connectivity graph of 16 centers. However, finally we settled down to a total 4 basis functions to have mean training data set errors of the order of 10^{-3}. Further, Fig. 7 shows the result of testing the modified MRAN and DCG network with the input vector $\mathbf{x}$ set to following three parameterized functions of t as described



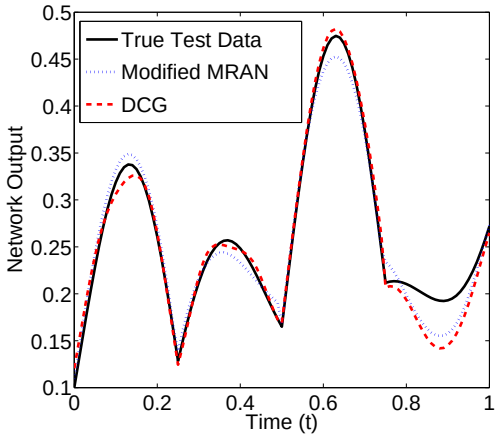
(a) Network growth for modified MRAN algorithm



(b) Test case 1



(c) Test case 2



(d) Test case 3

Fig. 7. Simulation results for test example 2.

in Ref. [12, 34].

Test Case 1

$$\begin{aligned} x_1(t) &= t \\ x_2(t) &= 1.61 \\ x_3(t) &= \begin{cases} 8t-2 & 0 \le t < \frac{1}{2} \\ -8t+6 & \frac{1}{2} \le t < 1 \end{cases} \end{aligned} \tag{2.61}$$

Test Case 2

$$\begin{aligned} x_1(t) &= t \\ x_2(t) &= \begin{cases} 8t-2 & 0 \le t < \frac{1}{2} \\ -8t+6 & \frac{1}{2} \le t < 1 \end{cases} \\ x_3(t) &= step(t) - 2step(t-0.25) + 2step(t-0.5) - \cdots \end{aligned} \tag{2.62}$$

Test Case 3

$$\begin{aligned} x_1(t) &= t \\ x_2(t) &= step(t) - 2step(t-0.25) + 2step(t-0.5) - \cdots \\ x_3(t) &= 2\sin(4\pi t) \end{aligned} \tag{2.63}$$

$$\tag{2.64}$$

As in Ref. [12, 34], in all 3 test cases t takes on 100 evenly spaced values in the $[0, 1]$ interval. In Table V, comparative results are shown in terms of percentage squared error for each test case and set of network parameters. The performance numbers for MRAN and DI algorithms are taken from Ref. [12]. From this Table and Fig. 7, it is clear that modified MRAN and DCG achieve smaller approximation error with a smaller number of network parameters. Once again, the effectiveness of the shape and rotation parameters is clear from the performance difference between the standard MRAN and the modified MRAN algorithms, although the advantage is not

as dramatic as in the first example.

Table V. Comparative results for 3-input, 1-output nonlinear function case.

Algorithm	Network Architecture	Squared Percentage Error for all Testing Sets	Number of Network Parameters
Modified MRAN	3-4-1	0.0265	40
DCG	3-4-1	0.0237	40
Std. MRAN	3-9-1	0.0274	45
DI	4-280-1	0.0295	1400

3. Test Example 3: Dynamical System Identification

In this section, a nonlinear system identification problem is considered to test the effectiveness of the shape and rotation parameters. The nonlinear dynamical system is described by the following equation and is borrowed from Refs. [12, 35]

$$y_{n+1} = \frac{1.5y_n}{1+y_n^2} + 0.3\cos y_n + 1.2u_n \tag{2.65}$$

The particular system considered here was originally proposed by Tan et al. in Ref. [35]. In Ref. [35], a recursive RBF structure (with fixed 42 neurons and one width value (0.6391)) is used to identify the discrete-time dynamical system given by Eq. (2.65). Further, in Ref. [12] the standard MRAN algorithm is employed to predict the value of $y(n+1)$ with 11 hidden units. It should be noticed that while the number of hidden units was reduced by a factor of three, the total number of parameters (44 in case of MRAN) to be learned was increased by 2 as compared to total number of parameters learned in Ref. [35].

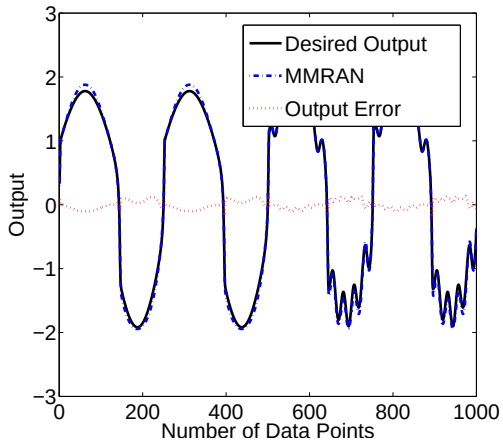
Like in the previous test examples, to show the effectiveness of the shape and rotation parameters, we first use the modified MRAN algorithm to identify the particular discrete-time system. Like in Refs. [12, 35], the RBF network is trained by taking 200 uniformly distributed random samples of input signals, u_n, between -2 and 2. The network input vector, $\mathbf{x}$, is assumed to consist of y_{n-1}, and u_n, i.e.

$$\mathbf{x} = \left\{ \begin{array}{cc} y_{n-1} & u_n \end{array} \right\} \tag{2.66}$$

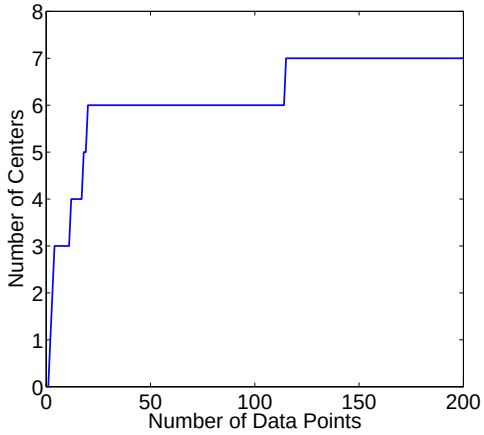
To test the learned RBF network, test data is generated by exciting the nonlinear system by a sequence of periodic inputs [12, 35]:

$$u(n) = \begin{cases} \sin(2\pi n/250) & 0 < n \leq 500 \\ 0.8\sin(2\pi n/250) + 0.2\sin(2\pi n/25) & n > 500 \end{cases} \tag{2.67}$$

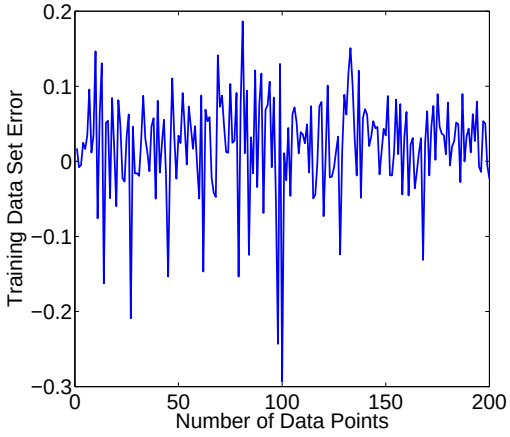
The different tuning parameters for the modified MRAN algorithms are given in Table VI. Fig. 8(a) shows the actual system excitation, the RBF network output learned by modified MRAN algorithm with shape and rotation parameters and the approximation error. Fig. 8(b) shows the plot of the evolution of RBF network with number of data points. From, these plots, we can conclude that number of hidden units required to identify the discrete-time system accurately reduces to 7



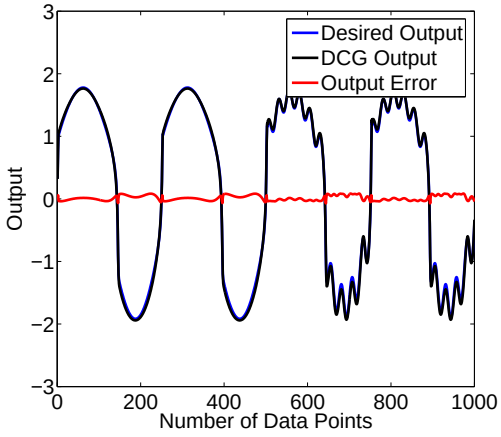
(a) Test data approximation result for modified MRAN algorithm



(b) Number of centers vs data points



(c) Training set approximation error for DCG algorithm



(d) Test data approximation result for DCG algorithm

Fig. 8. Simulation results for test example 3.

Table VI. Various tuning parameters for modified MRAN algorithm for test example 3.

Algorithm	ϵ_{max}	ϵ_{min}	γ	e_{min}	$e_{r_{min}}$	κ	p_0	R	N_w	S_w	δ
MMRAN	3	1	0.6	0.04	0.4	0.50	1	10^{-2}	25	200	10^{-4}

from 11 if we introduce shape and rotation optimization of the Gaussian functions in the standard MRAN algorithm. However, in terms of the total number of learning parameters there is a reduction of only 2 parameters when we include the shape and rotation parameters in the MRAN algorithm.

Finally, the Directed Connectivity Graph Learning Algorithm is used to learn the unknown nonlinear behavior of the system described by Eq. (2.65). For approximation purposes, the input space is divided into 2×2 grid giving us a freedom to choose a maximum 8 radial basis functions. However, the final network structure requires only 6 neurons to have approximation errors less than 5%. Fig. 8(c) shows the plot of training data set approximation error with 6 basis functions while Fig. 8(d) shows the actual system excitation for test data, the RBF network output learned by the DCG algorithm and the approximation error. From these plots, we conclude that the DCG algorithm is by far the most advantageous since it requires only 6 Gaussian centers to learn the behavior of the system accurately as compared to 42 and 11 Gaussian centers used in Refs. [35] and [12] respectively. In terms of the total number of learning parameters, the DCG algorithm is also preferable. For DCG, we need to learn only $6 \times 6 = 36$ parameters as compared to 42 and 44 parameters for

MMRAN and MRAN respectively. This result, once again reiterates our observation that the better performance of DCG and MMRAN algorithm can be attributed to the adaptive shape and rotation learning of the Gaussian functions as well as the *judicious choice of initial centers* (in case of DCG). It is obvious we have achieved (*i*) more accurate convergence (*ii*) fewer basis functions, and (*iii*) fewer network parameters, and, importantly, we have a systematic method for obtaining the starting estimates.

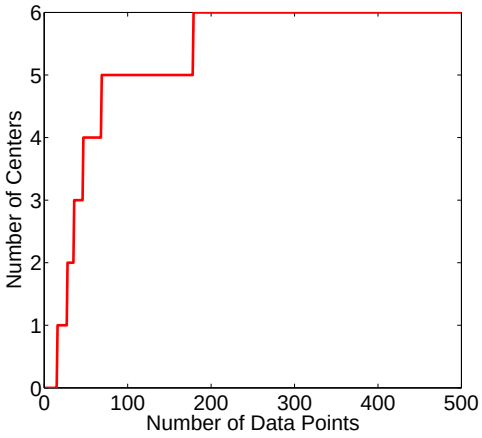
4. Test Example 4: Chaotic Time Series Prediction Problem

The effectiveness of shape and rotation parameters has also been tested with the chaotic time series generated by Mackey-Glass time delay differential equation [36]:

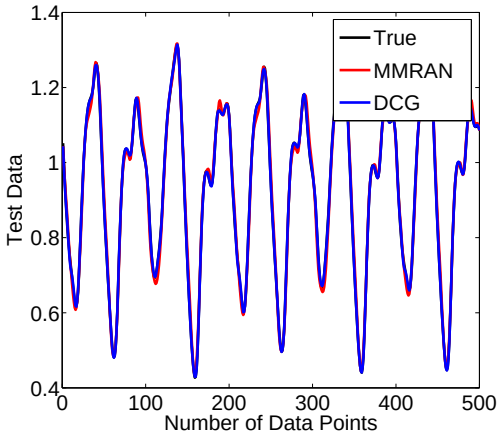
$$\frac{ds(t)}{dt} = -\beta s(t) + \alpha \frac{s(t-\tau)}{1 + s^{10}(t-\tau)} \tag{2.68}$$

This equation is extensively studied in Refs. [5,12,37,38] for its chaotic behavior and is listed as one of the benchmark problems at IEEE Neural Network Council web-site [32]. To compare directly to the previous studies [5,12,37,38], we choose the same parameter values: $\alpha = 0.2$, $\beta = 0.1$, $\tau = 17$ and $s(0) = 1.2$. Further, to generate the training and testing data set, the time series Eq. (2.68) is integrated by using the fourth-order Runge-Kutta method to find the numerical solution. This data set can be found in the file mgdata.dat belonging to the FUZZY LOGIC TOOLBOX OF MATLAB 7 and at IEEE Neural Network Council web-site [32].

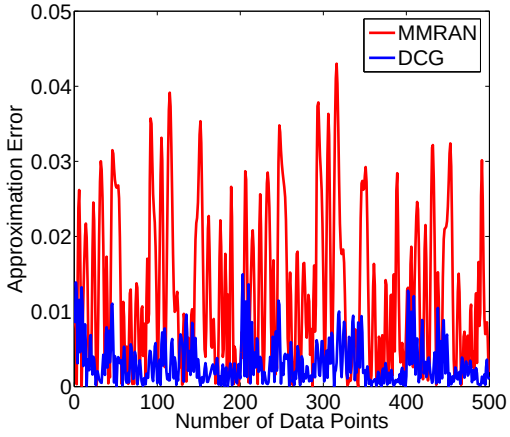
Once again, to study the effectiveness of introducing shape and rotation parameters only, we used modified MRAN algorithm and DCG algorithm to perform a short-term prediction of this chaotic time series. We predict the value of $s(t+6)$ from the current value $s(t)$ and the past values $s(t-6)$, $s(t-12)$ and $s(t-18)$. Like in previous studies [5, 12, 37, 38], the first 500 data-set values are used for network



(a) Number of centers vs data points



(b) Test data approximation result



(c) Test data approximation error

Fig. 9. Simulation results for test example 4.

Table VII. Various tuning parameters for modified MRAN algorithm for test example 4.

Algorithm	ϵ_{max}	ϵ_{min}	γ	e_{min}	$e_{r_{min}}$	κ	p_0	R	N_w	S_w	δ
MMRAN	2	0.5	0.66	10^{-5}	10^{-4}	0.27	1	10^{-1}	10^2	10^3	10^{-4}

training while the remaining 500 values are used for testing purposes. The different tuning parameters for the modified MRAN algorithm are given in Table VII. For the DCG approximation purposes, the input space is divided into $2 \times 2 \times 2 \times 2$ grid giving us freedom to choose a maximum of 32 radial basis functions. However, the final network structure required only 4 neurons to achieve approximation errors less than 5%. We mention that due to the availability of a small number of training data set examples, we used the Levenberg-Marquardt [29] algorithm to efficiently optimize the DCG network.

Fig. 9(a) shows the MMRAN network growth with the number of training data set examples while Figs. 9(b) and 9(c) show the plots for approximated test data and approximation test data error respectively. From these plots, we can conclude that the MMRAN algorithm requires only 6 Gaussian centers to learn the behavior of the system accurately as compared to 29 and 81 Gaussian centers used in Refs. [12] and [5] respectively. In terms of the total number of learning parameters, the MMRAN algorithm is also preferable as compared to the MRAN and the RAN algorithms. For the MMRAN algorithm, we need to learn only $6 \times 15 = 90$ parameters as compared to 174 parameters required for the MRAN algorithm. In the case of the DCG algorithm,

the number of Gaussian centers required was reduced even further to only 4 while the total number of learned parameters reduced to 60 as compared to 90 in case of the MMRAN algorithm and 174 for the standard MRAN algorithm. In Ref. [38], Table IX compares the various algorithms presented in the literature in terms of their root mean squared error (RMSE) for this particular problem. Here, in Table VIII, we present comparative results for MMRAN, DCG and five other algorithms. The direct comparison of MRAN and MMRAN results reveals the fact that inclusion of the shape and rotation parameters greatly enhance the approximation accuracy while significantly reducing the number of parameters required to define the RBF network for a *particular algorithm*. It should be also noted that both the DCG and MMRAN algorithm performed very well as compared to all other algorithms for this particular example, in terms of both smallness of the RMS error and the number of free network parameters.

5. Test Example 5: Benchmark Against the On-line Structural Adaptive Hybrid Learning (ONSAHL) Algorithm

In this section, we present a comparison of the MMRAN and DCG algorithms with the On-line Structural Adaptive Hybrid Learning (ONSAHL) learning algorithm on a nonlinear system identification problem from Ref. [21]. The ONSAHL algorithm uses a Direct Linear Feedthrough Radial Basis Function (DLF-RBF) network and an error sensitive cluster algorithm to determine automatically the number of RBF neurons, and to adapt their center positions, their widths and the output layer weights. This algorithm, however, does not include shape and rotation parameters. The nonlinear dynamical system is described by following difference equation and is borrowed from

Table VIII. Comparative results for Mackey-Glass chaotic time series prediction problem.

Algorithm	Network Architecture	RMS Error	Number of Network Parameters
MRAN	4-29-1	0.035	174
Modified MRAN	4-6-1	0.0164	90
DCG	4-4-1	0.004	60
Genetic Algorithm + Fuzzy Logic [38]	$9 \times 9 \times 9 \times 9$	0.0379	6633
Pomares 2000 [39]	$3 \times 3 \times 3 \times 3$	0.0058	101
Pomares 2003 [38]	4-14-1	0.0045	84
Pomares 2003 [38]	4-20-1	0.0029	120

Table IX. Various tuning parameters for modified MRAN algorithm for test example 5.

Algorithm	ϵ_{max}	ϵ_{min}	γ	e_{min}	$e_{r_{min}}$	κ	p_0	q_0	R	N_w	S_w	δ
MMRAN	2	0.9	0.99	10^{-2}	10^{-2}	0.7	1	0	1	500	5000	10^{-4}

Ref. [21].

$$y(n) = \frac{29}{40}\sin\left(\frac{16u(n-1)+8y(n-1)}{3+4u(n-1)^2+4y(n-1)^2}\right)+\frac{2}{10}\left(u(n-1)+y(n-1)\right)+\epsilon(n) \tag{2.69}$$

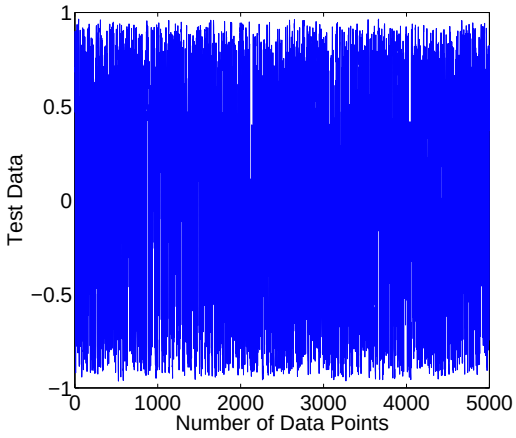
Like in Ref. [21] $\epsilon(n)$ denotes a Gaussian white noise sequence with zero mean and a variance of 0.0093. A random signal uniformly distributed in the interval $[-1, 1]$ is used for the excitation $u(n)$ in the system of Eq. (2.69). The network input vector $\mathbf{x}$ is assumed to consist of $y(n-1)$ and $u(n-1)$ while network output vector consists of $y(n)$. Eq. (2.69) is simulated with zero initial conditions to generate response data for $10,000$ integer time steps. Out of these $10,000$ data points, the first 5000 are used for training purpose while the remaining 5000 points are used for testing purpose. Fig. 10(a) shows the plot of true test data.

In this case, several MRAN tuning parameters are given in Table IX. For DCG approximation purposes, the input space is divided into 2×2 grid giving us a freedom to choose maximum 8 radial basis functions. However, final network structure consists of only 6 neurons to have approximation errors less than 5%. For comparison

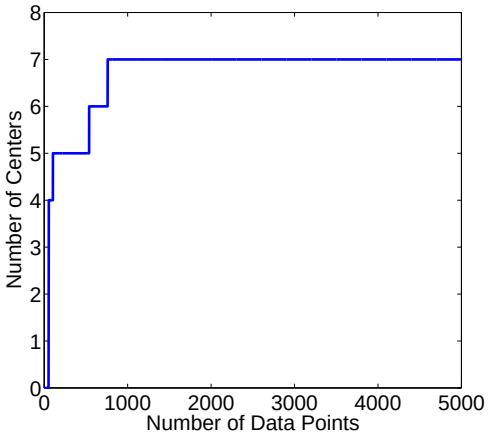
purposes, we also use the same error criteria as defined in Ref. [21].

$$I_d(n) = \frac{1}{50}\sum_{j=0}^{49} |y(n-j) - \hat{y}(n-j)| \tag{2.70}$$

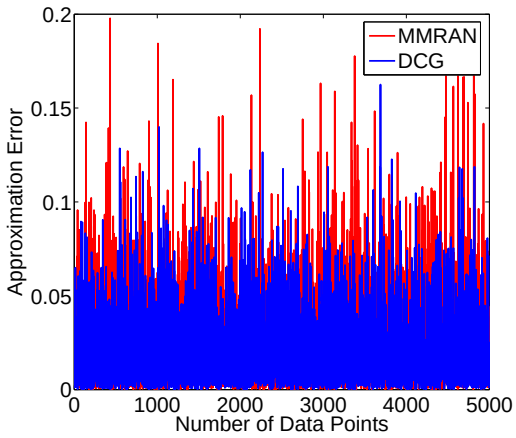
Fig. 10(b) shows the plot of MMRAN network growth with the number of training data points while Figs. 10(c) and 10(d) shows the plot of absolute approximation error and incremental $I_d(n)$ respectively. In Ref. [12], standard MRAN algorithm is employed for system identification purposes using 11 neurons while the ONSAHL algorithm is employed using 23 neurons. From the results presented in Ref. [12], it is clear that MRAN uses a smaller number of neurons as compared to the ONSAHL algorithm to accurately represent the given dynamical system. From Fig. 10(b), it is clear that number of neurons required to identify the discrete-time system accurately further reduces to 7 from 11 if shape and rotation adaptation of the Gaussian RBF is incorporated in MRAN algorithm. However, in terms of the total number of learning parameters there is a reduction of only 2 parameters if we include the shape and rotation parameters in the MRAN algorithm. From these plots, we can also conclude that the DCG algorithm requires only 6 Gaussian centers to learn the behavior of the system accurately as compared to 23 and 11 Gaussian centers used in Refs. [21] and [12] respectively. In terms of the total number of learning parameters, the DCG algorithm is again preferable. For DCG, we need to learn only 36 parameters as compared to 42 and 44 parameters for MMRAN and MRAN respectively. Finally, Table X summarizes the comparison results in terms of approximation error and number of free network parameters. These results, once again reiterates our observation and support the conclusion that the better performance of the DCG and MMRAN algorithms can be attributed to the inclusion of shape and rotation optimization of Gaussian functions as well as the optimization of their centers and



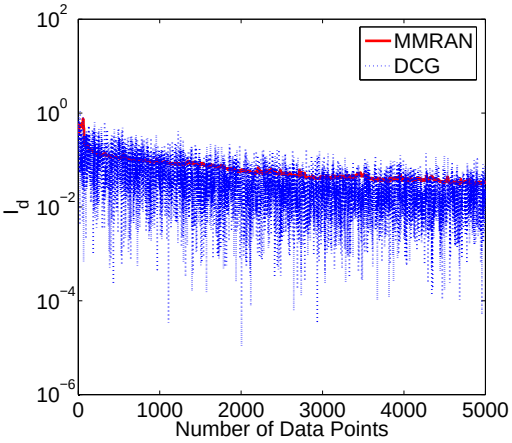
(a) True test data



(b) Number of centers vs data points



(c) Absolute test data set approximation error



(d) Incremental $I_d(n)$

Fig. 10. Simulation results for test example 5.

Table X. Comparative results for test example 5.

Algorithm	Network Architecture	Mean $I_d(n)$	Number of Network Parameters
Modified MRAN	2-7-1	0.0260	42
DCG	2-6-1	0.0209	36
Std. MRAN	2-11-1	0.0489	44
ONSAHL	2-23-1	0.0539	115

spreads. These dramatic advantages, taken with the previous four problems results provide compelling evidence for the merits of the shape and rotation optimization of the Gaussian basis functions as well as a directed connectivity graph algorithm to initialize estimates for these parameters.

F. Concluding Remarks

A direction dependent RBFN learning algorithm has been developed to obtain a minimal RBF network. New approaches are introduced and tested on variety of examples

from a variety of disciplines such as continuous function approximation, dynamic system modeling and system identification, nonlinear signal processing and time series prediction. In all of these diverse test problems, the proposed two algorithms are found to produce more compact RBF networks with the same or smaller errors as compared to many existing methods. The results are of direct utility in addressing the "curse of dimensionality" and frequent redundancy of neural network approximation.

The results presented in this chapter serve to illustrate the usefulness of shape and rotation optimization of the Gaussian basis functions as well as a directed connectivity graph algorithm to initialize estimates for these parameters. The shape and rotation optimization of the Gaussian functions not only helps us in approximating the complex surfaces better but also helps in greatly reducing the numbers of hidden units. We believe that the concept of shape and rotation optimization can be incorporated into many existing learning algorithms to very significantly enhance their performance without much difficulty. This fact was illustrated by our modification of a conventional MRAN learning algorithm. However, much research is required to extend and optimize the methodology for general multi-resolution approximations in high dimensional spaces. Finally, we mention that proving the minimality of RBF network (using any learning algorithm for that matter) is an open problem in the field of approximation theory and the word "minimal" in the chapter only signifies that, we have sought a minimum parameter representation and no more compact network apparently exists in the literature for all the test problems and the test data considered in this chapter. Finally, we fully appreciate the truth that results from any test are difficult to extrapolate, however, testing the new algorithm on five benchmark problems and providing comparisons to the most obvious five competing algorithms does provide compelling evidence and a basis for optimism.

CHAPTER III

GLOBAL LOCAL ORTHOGONAL POLYNOMIAL MAPPING (GLO-MAP) IN N-DIMENSIONS: APPLICATIONS TO INPUT-OUTPUT APPROXIMATION

A. Introduction

In the previous chapter, we have shown that the learning of shape and orientation parameters of a basis function significantly improves the approximation capability of a Gaussian basis function. This intuitively comfortable fact was illustrated by considering a variety of examples from a variety of disciplines such as continuous function approximation, dynamic system modeling and system identification, nonlinear signal processing and time series prediction. Although, the RBF learning algorithms, presented in Chapter II, are shown to work very well for different test examples, there remains several issues about the complexity and convergence of the RBF model. Also, use of RBF networks for dynamic system identification problem generally leads to a non-affine control problem due to their inherent nonlinear and complex structure, which is not desirable for controller design purposes. Besides this, the various network parameters which describe a RBF network appear nonlinearly in final network structure and necessitate the use of a nonlinear estimation algorithm to find the best estimates of these parameters from input-output data. The nonlinear RBF model is global, this has both advantages and disadvantages, but ultimately for very high dimensioned problems, it is likely defeated by the curse of high dimensionality (computation burden and convergence difficulties, mainly). Although successes have been many, the computational cost associated with learning these parameters and the convergence of nonlinear estimation algorithm remain obstacles that limits applicability to problems of low to moderate dimensionality.

In this chapter, we seek to design an efficient and robust modeling algorithm that can be utilized for a large number of different engineering applications while considering the above mentioned disadvantages of the best existing methods. A key motivation underlying these developments is to establish a more general, rigorous and computationally attractive way to construct a family of local approximations. The main idea discussed is a weighting function technique [40, 41] that generates a global family of overlapping preliminary approximations whose centroids of validity lie on at the vertices of an N-dimensional pseudo-grid. These preliminary approximations are constructed so that each represents accurately the behavior in a local sub-domain centered on a typical vertex in the grid. These sub-domains, where the preliminary approximations are valid, generally overlap and the overlapping approximations are averaged over the overlapped volume to determine final local approximations. A novel averaging method is presented that ensures these final approximations are globally piecewise continuous with adjacent approximations determined in an analogous averaging process, to some prescribed order of partial differentiation. The continuity conditions are enforced by using a unique set of weighting functions in the averaging process, without constraining the preliminary approximations being averaged. The weight functions are designed to guarantee the global continuity conditions while retaining *near complete freedom on the selection of the generating local approximations.* Further, it is shown that if the preliminary local approximations are chosen as linear combinations of a set of basis functions orthogonal with respect to the weight functions, then many advantages can be realized in terms of model complexity, computational cost and conditioning of the approximation problem. Construction of a new set of orthogonal polynomials, and several properties of these functions are novel results presented in this chapter. Finally, several applications from various diverse fields are considered to show the approximation capability of the proposed algorithm.

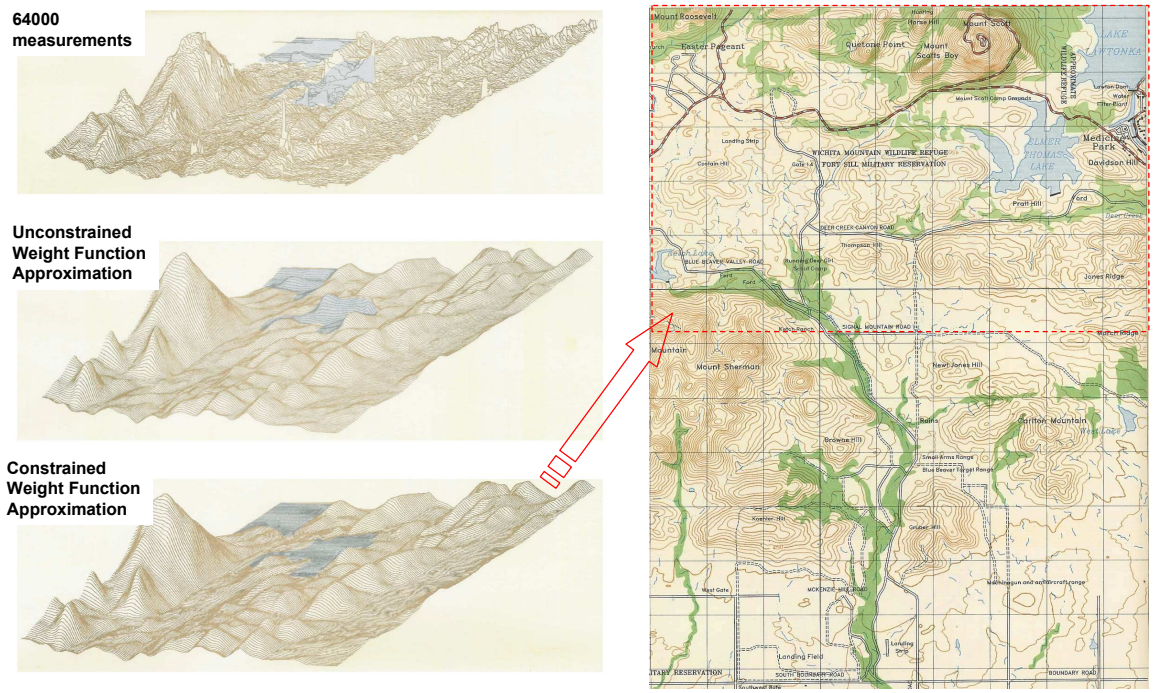


Fig. 11. Approximation of irregular functions in two dimensions.

The chapter is organized as follows. First, we introduce and illustrate the basic ideas underlying the proposed algorithm, followed by systematic development of the algorithm. Finally, the proposed approach is validated by considering different engineering applications. In Chapter IV, we provide a strong set of theoretical justification that proves the probabilistic truth that the GLO-MAP process is unbiased and the covariance of the averaged approximations are smaller than the generating approximations.

B. Basic Ideas

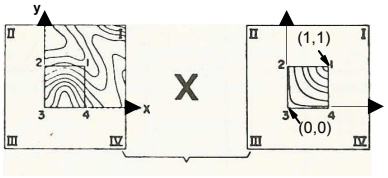
To motivate the results in this chapter, consider Fig. 11. Here we have $64,000$ noisy measurements of a irregular function $F(x, y)$. These happen to be stereo ray intersection measurements from correlation of stereo images of topography near Ft. Sill,

Oklahoma [42]; however, they could be measurements of any complicated, irregular function for which a single global algebraic expression would likely be intractable. Suppose that it is desired to obtain a smooth, least square approximation of this function, perhaps with additional constraints imposed (e.g., in this case, the stereo correlation measurement process fails reliably over water, so the large spurious noise spikes over lakes Latonka and Elmer Thomas, where reliable stereo correlation is not possible, but can be replaced by a constraint that the lake surface be a known elevation). In lieu of a single global and necessarily complicated function, it is desired to represent the function using a family of simpler local approximations. Such local approximations would be much more attractive basis for local analysis. Alternatively, one may think of the local approximations as Taylor series approximations (each evaluated at a local expansion point on a grid), or as any local approximations obtained from local measurements. However, if the local approximations are introduced without taking particular care, they will virtually certainly disagree in the value estimated for $F(x, y)$ and the derivatives thereof at any arbitrary point, although the discrepancies may be small. In other words, global continuity is not assured, unless we introduce methodology to guarantee the desired continuity properties. These challenges are compounded in higher dimensions, if usual local approximation approaches are used. It is desired to determine a piecewise continuous global family of local least squares approximations, while having freedom to vary the nature (e.g., mathematical basis functions and degrees of freedom) of the local approximations to reflect possibly large variations in the roughness of $F(x, y)$. While we are introducing the ideas in the setting of a data-fitting problem in a two dimensional space, the results are shown later in this dissertation to be of much broader utility, and to generalize fully to approximation in an N dimensional space, including opening a door to a flexible new method for solving high dimensional partial differential equations.

Preliminary Approximations:

Weight functions:

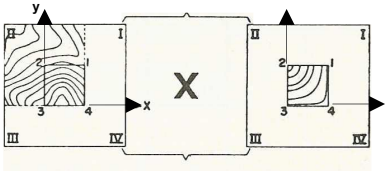
$F_{11}(x,y)$



$w_{11}(x,y) = x^2(3-2x)y^2(3-2y)$

+

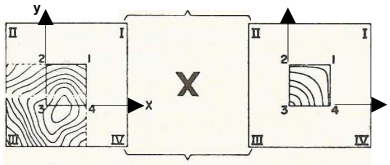
$F_{01}(x,y)$



$w_{01}(x,y) = w_{11}(1-x,y)$

+

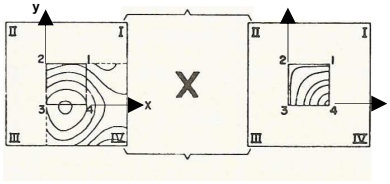
$F_{00}(x,y)$



$w_{00}(x,y) = w_{11}(1-x,1-y)$

+

$F_{10}(x,y)$



$w_{10}(x,y) = w_{11}(x,1-y)$

=

Final Approximation:

$$\bar{F}(x,y) = \sum_{i=0}^{1}\sum_{j=0}^{1} w_{ij}(x,y)F_{ij}(x,y)$$

valid over $\{0 \le x \le 1, 0 \le y \le 1\}$

Weight functions are a partition of unity:

$$\sum_{i=0}^{1}\sum_{j=0}^{1} w_{ij}(x,y) = 1$$

Fig. 12. Qualitative representation of the averaging process in two dimensions.

With reference to Fig. 12, we summarize some features of the weighting function approach to approximation in two dimensions. We prove these qualitative statements later in this chapter and in Chapter IV. From Fig. 12, we introduce several qualitative observations: Notice the attractive properties of the weight functions: At any of the four vertices, we see the weight function (associated with the function whose centroid of validity is a given vertex) is unity, while the other three weight functions are zero at that vertex. Notice further that the weight functions have a qualitative bell shape, but fair into a square base, the zero contour being the boundary opposite (e.g., 2-3-4) to the vertex (e.g., point 1) where the weight has a unit value. We will show that the four overlapping weight functions constitute a *partition of unity*, they add to unity everywhere in the overlapping unit region (which guarantees an unbiased approximation). Furthermore, notice that along any boundary, only the two weight functions associated with the two approximations centered at the end points of that boundary are non-zero along that boundary, while the other two weight functions are zero (the partial derivatives of the other two weight functions are also along this boundaries). These continuity arguments on the averaged approximation of the function can be extended readily to corresponding properties on their partial derivatives: The averaged approximation osculate in value and partial derivatives with the four preliminary approximations at their corresponding vertices, and the function and both partial derivatives along any boundary are a weighted average of the corresponding two functions associated with the end point of that boundary. Collectively, these observations lead to rigorous piecewise continuity of the averaged approximations, while leaving the user free to choose any preliminary local approximations desired or needed. These qualitative observations will be developed systematically in the subsequent sections and extended rigorously to approximation with arbitrary order continuity in an N dimensional space.

C. Approximation in 1-, 2- and N- Dimensions Using Weighting Functions

The essential ideas can be introduced rigorously in a one-dimensional piecewise approximation problem. The notations are developed for the one-dimensional problem such that the generalization is most straightforward. With reference to Fig. 13, we discuss the one-dimensional problem. An arbitrary set of knots (vertices) $\left\{{}^1X,\ {}^2X,\ \cdots,\ {}^KX,\ \cdots\right\}$ are introduced at a uniform distance h apart; a non- dimensionalization of x is introduced as a local coordinate $-1 \leq {}^Ix \triangleq (X - {}^IX)/h \leq 1$; centered on the I^{th} vertex $X = {}^IX$. The local weighted average approximation is introduced as

$$\bar{F}_I(X) = w({}^Ix)F_I(X) + w({}^{I+1}x)F_{I+1}(X),\ \text{for } 0 \leq {}^I\ x < 1 \tag{3.1}$$

where the weighting functions $w(x)$ used to average (blend) the two adjacent preliminary local approximations $\{F_I(X),\ F_{I+1}(X)\}$ are as yet un-specified. We prefer that the preliminary approximations $\{F_1(X),\ F_2(X),\ \cdots,\ F_K(X),\ \cdots\}$ *be left completely arbitrary*, so long as they are smooth and represent the local behavior of $F(X)$ well. As developed in Reference [43], the weight function can be selected to guarantee that the averaged approximation $\bar{F}(X)$ osculates with $F_I(X)$ in value and first derivative as $X \to {}^IX$, and likewise $\bar{F}(X)$ osculates with $F_{I+1}(X)$ in value and first derivative as $X \to {}^{I+1}X$. Notice that the shifted weight functions add to unity, as they must for an unbiased estimate, e.g., $w({}^Ix) + w({}^Ix - 1) = 1$, or $w({}^Ix - 1) = 1 - w({}^Ix)$ Observe that ${}^{I+1}x = {}^Ix - 1$, so if $0 \leq {}^Ix \leq 1$, $-1 \leq {}^{I+1}x = {}^Ix - 1 \leq 0$. Notice also the first derivative of the average of Eq. (3.1) at an arbitrary point is

$$\frac{d\bar{F}_I(X)}{dx} = w({}^Ix)\frac{dF_I(X)}{dx} + w({}^{I+1}x)\frac{dF_{I+1}(X)}{dx} + \frac{dw({}^Ix)}{dx}F_I(X) + \frac{dw({}^{I+1}x)}{dx}F_{I+1}(X) \tag{3.2}$$

Thus the requirement that the weighted average approximation (3.1) form a continuous global valid model leads to following boundary conditions on yet to be defined weighting functions:

$$\text{at } x=0: \begin{cases} w(0)=1 \\ \frac{dw(x)}{dx}\Big|_{x=0}=0 \end{cases}, \text{ at } x=1: \begin{cases} w(1)=0 \\ \frac{dw(x)}{dx}\Big|_{x=1}=0 \end{cases} \tag{3.3}$$

With these boundary conditions, the first term of Eq. (3.1) reduces to $F_I(X)$ as ${}^I x \rightarrow 0$ and likewise, only the first term of Eq. (3.2) contributes as ${}^I x \rightarrow 0$. Analogous osculation arguments hold at the right end of the interval. In general the requirement that the weighted average approximation in Eq. (3.1) form an m^{th}-order continuous globally valid model and additional requirement of unbiased approximation leads to the following boundary value problem that uniquely defines the necessary weighting functions:

1. The first derivative of the weighting function must have an m^{th}-order osculation with $w(0)=1$ at the centroid of its respective local approximation.

$$\begin{gathered} w(0)=1 \\ \tfrac{d^k w}{dx^k}|_{x=0}=0 \quad k=0,1,\cdots,m \end{gathered} \tag{3.4}$$

2. The weighting function must have an $(m+1)^{th}$-order zero at the centroid of its neighboring local approximation.

$$\begin{gathered} w(1)=0 \\ \tfrac{d^k w}{dx^k}|_{x=1}=0 \quad k=0,1,\cdots,m \end{gathered} \tag{3.5}$$

3. The sum of two neighboring weighting functions must be unity over the entire closed interval between their corresponding adjacent local functional approxi-

mations.

$$w({}^{I}x) + w({}^{I}x - 1) = 1 \quad \forall\ x,\ -1 \leq x \leq 1 \tag{3.6}$$

It should be noted that the first two boundary conditions are sufficient to ensure that the global function reduces exactly to the local approximations at their centroids, not only in their value but in their first m partial derivatives. If weighting function is assumed to be polynomial in an independent variable, x, then adopting the procedure listed in Ref. [44] and summarized in Appendix A, the lowest order weight function (for $m = 1$) can be shown to be simply:

$$w(x) = \left\{ \begin{array}{c} 1 - x^2(3 + 2x),\ \ -1 \leq x < 0 \\ 1 - x^2(3 - 2x),\ 0 \leq x \leq 1 \end{array} \right\} = 1 - x^2(3 - 2|x|) \tag{3.7}$$

These are the functions plotted in Fig. 13. It should be noted that the weight function given by Eq. (3.7) is a non-negative continuous functions defined on a locally compact subset of approximation space. To be more precise, the weight functions obtained by solving the boundary value problem have following properties which result in a meshless approximation and interpolation algorithm.

1. The domain of weight function, w is a compact space.

2. $w(x) > 0, \forall\ x \in (-1,\ 1)$.

3. $w(x) = 0,\ |x| \geq 1$.

4. $w(x)$ is a monotonically decreasing function of $x, \forall\ x \in (-1,\ 1)$.

In the event that discrete measurements of $F(X)$ are available, the preliminary approximations $\{F_1(X),\ F_2(X),\ \cdots,\ F_K(X),\ \cdots\}$ are fit to data subsets in the $\Delta X = \pm h$ regions centered on $\left\{ \begin{array}{cccc} {}^{1}X & {}^{2}X & \cdots & {}^{K}X \quad \cdots \end{array} \right\}$. It is evident that the final approximation on each interval is the average of overlapping weighted least

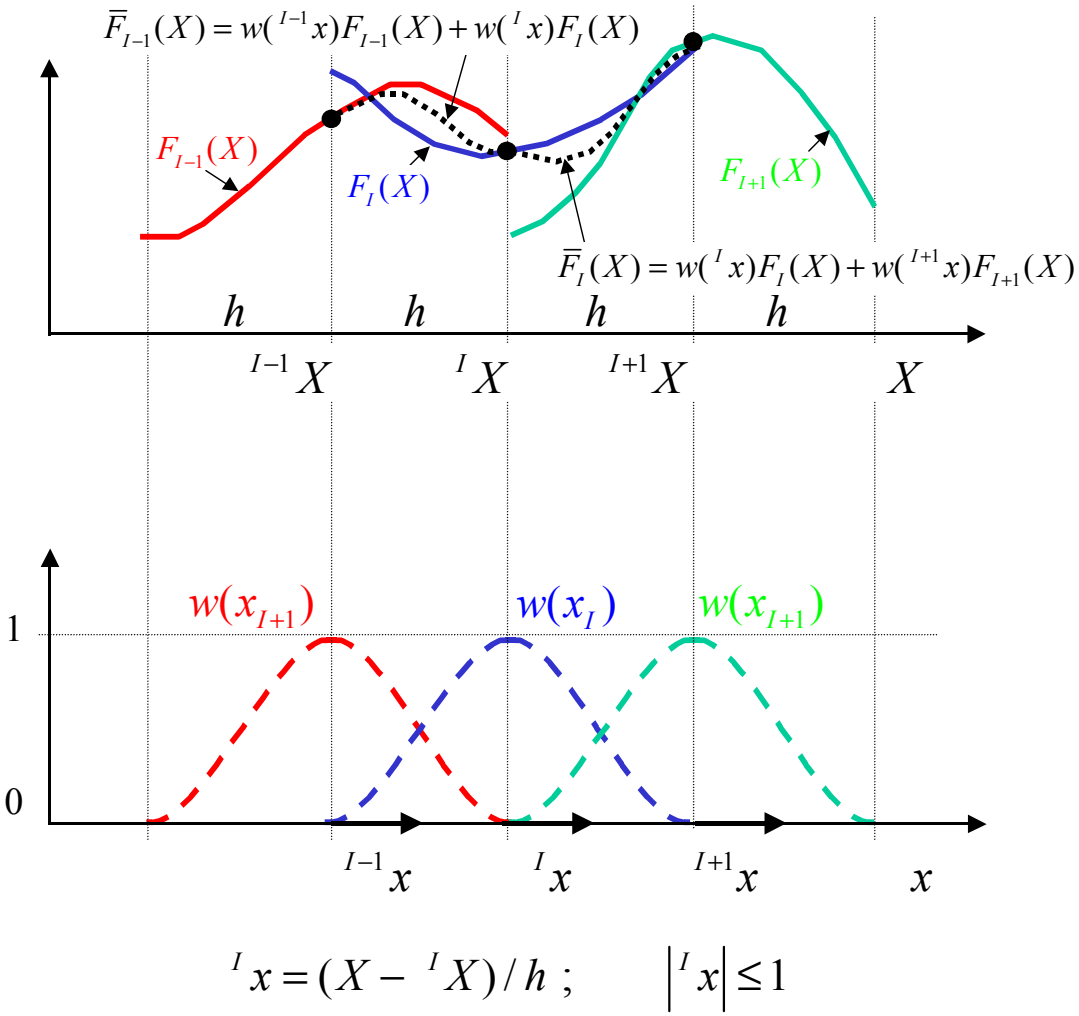


Fig. 13. Weighting function approximation of a one-dimensional function.

square approximations, fit to shifted data lying within $\pm h$ of the vertices. For equally precise measurements of $F(X)$, the least square process should use the same weight functions of Eq. (3.3). If the measurements are made with unequal expected precision, then the statistically justified weights should be scaled using the weights of Eq. (3.7). Note the qualitative justification: "If one least square fit is good, the average of two should be better." In Chapter IV, we prove the probabilistic veracity of this qualitative observation. Observe that simply through choosing the judicious weight functions of Eq. (3.7) we are guaranteed global piecewise continuity for all possible continuous local approximations $\{F_1(X), F_2(X), \cdots, F_K(X), \cdots\}$. One retains the freedom to vary the degree of the local approximations, as needed, to fit the local behavior of $F(X)$, and rely upon the weight functions to enforce continuity.

A most important characteristic of the weighting function averaging process is that it generalizes fully to N dimension without as severe a 'curse of dimensionality' that accompanies generalizations of virtually all known analysis methods to higher dimensions. The generalization to 2-Dimensions is amazingly straightforward. Introduce notation for the local approximations $\{F_{11}(X_1, X_2), \cdots, F_{I_1 I_2}(X_1, X_2), \cdots\}$ constructed such that they are valid over $(2h) \times (2h)$ regions centered on the vertices $\{(\ {}^1X_1,\ {}^1X_2), (\ {}^1X_1,\ {}^2X_2), \cdots, (\ {}^{I_1}X_1,\ {}^{I_2}X_2), \cdots\}$. Given four contiguous vertices:

$$\begin{array}{ll} (\ {}^{I_1}X_1,\ {}^{I_2+1}X_2 = {}^{I_2}X_2 + h) & (\ {}^{I_1+1}X_1 = {}^{I_1}X_1 + h,\ {}^{I_2+1}X_2 = {}^{I_2}X_2 + h) \\ (\ {}^{I_1}X_1,\ {}^{I_2}X_2) & (\ {}^{I_1+1}X_1 = {}^{I_1}X_1 + h,\ {}^{I_2}X_2) \end{array} \tag{3.8}$$

The corresponding four preliminary approximations are valid in the $(2h) \times (2h)$ regions centered at the contiguous four nodes are denoted:

$$\begin{array}{ll} F_{I_1, I_2+1}(X_1, X_2) & F_{I_1+1, I_2+1}(X_1, X_2) \\ F_{I_1, I_2}(X_1, X_2) & F_{I_1+1, I_2}(X_1, X_2) \end{array} \tag{3.9}$$

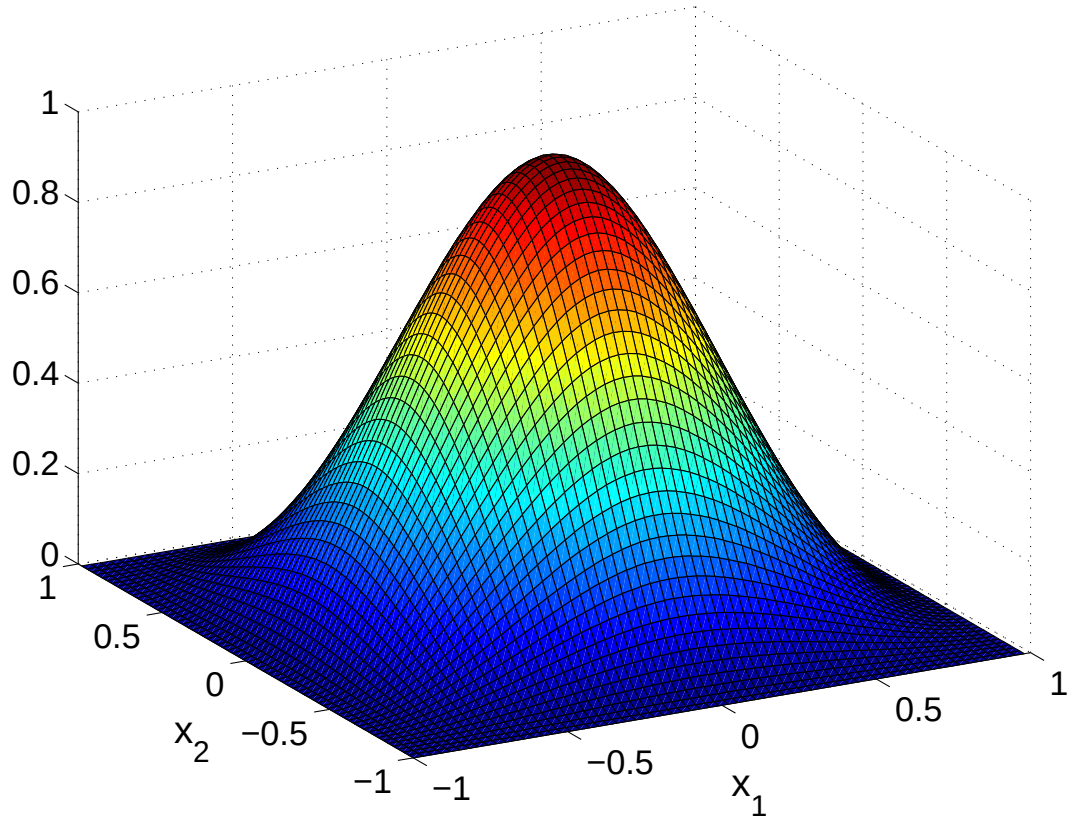


Fig. 14. Weighting function $w_{0,0}(x_1, x_2)$ for two-dimensional approximation.

The final averaged approximation valid within the $h \times h$ region bounded by the four vertices of Eq. (3.8) is given by

$$\bar{F}_{I_1,I_2}(X_1,X_2) = \sum_{i_1=0}^{1}\sum_{i_2=0}^{1} w_{i_1,i_2}({}^{I_1+i_1}x_1, {}^{I_2+i_2}x_2) F_{I_1+i_1,I_2+i_2}(X_1,X_2) \tag{3.10}$$

where, it can be verified that choosing the weight functions as the product of one dimensional weight functions as

$$w_{i_1,i_2}({}^{I_1+i_1}x_1, {}^{I_2+i_2}x_2) = w({}^{I_1+i_1}x_1) w({}^{I_2+i_2}x_2) \tag{3.11}$$

then these functions are a *partition of unity* that satisfy

$$\sum_{i_1=0}^{1}\sum_{i_2=0}^{1} w_{i_1 i_2}({}^{I_1+i_1}x_1, {}^{I_2+i_2}x_N) = 1 \tag{3.12}$$

to give an un-biased average. Note that if we use a common origin (the lower left vertex) for all four weight functions, then the one centered on the origin (for $m = 1$) is (see Fig. 14):

$$\begin{aligned} w_{0,0}(x_1,x_2) &= [1 - x_1^2(3 \mp 2x_1)][1 - x_2^2(3 \mp 2x_2)] \\ &\quad \text{the minus (plus) sign is for } x_i > 0 \ (x_i < 0) \text{ or} \\ &\equiv [1 - x_1^2(3 - 2|x_1|)][1 - x_2^2(3 - 2|x_2|)] \end{aligned} \tag{3.13}$$

The remaining three weight functions are simply obtained by translating this function to the other three vertices as:

$$\begin{aligned} w_{1,0}(x_1,x_2) &= w_{0,0}(x_1 - 1, x_2) \\ w_{0,1}(x_1,x_2) &= w_{0,0}(x_1, x_2 - 1) \\ w_{1,1}(x_1,x_2) &= w_{0,0}(x_1 - 1, x_2 - 1) \end{aligned} \tag{3.14}$$

These four overlapping weight functions are shown in Fig. 15. The central unit square of Fig. 15 is the focus of this figure, it is the region in which the final averaged

approximation of Eq. (3.10) is valid. The process can be shifted by one unit cell in any direction and continuity arguments will lead to the conclusion that the adjacent final averaged approximations match in value and both partial derivatives along their common boundaries.

We see the weight function of Fig. 12, from Refs. [43, 44] is obtained to within the obvious notation changes. The reason for adopting the above notations is that the generalization to N-dimensions follows easily from the above pattern.

The N-dimensional generalization of Eqs. (3.10) and (3.11) are:

$$\begin{aligned}\bar{F}_{I_1,\cdots,I_N}(X_1,\cdots,X_N) &= \sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_N=0}^{1}\big(w_{i_1,\cdots,i_N}({}^{I_1+i_1}x_1,\cdots,{}^{I_N+i_N}x_N)\\ &\quad F_{I_1+i_1,\cdots,I_N+i_N}(X_1,\cdots,X_N)\big)\end{aligned} \tag{3.15}$$

and

$$w_{i_1,i_2,\cdots,i_N}({}^{I_1+i_1}x_1,\cdots,{}^{I_N+i_N}x_N)=\prod_{i=1}^{N}w({}^{I_i+i_i}x_i) \tag{3.16}$$

$$\sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_N=0}^{1}w_{i_1,\cdots,i_N}({}^{I_1+i_1}x_1,\cdots,{}^{I_N+i_N}x_N)=1 \tag{3.17}$$

The partition of unity constraint of Eq. (3.17) is required for an unbiased average in Eq. (3.15). Further, it can be verified that the unbiased average requirement of Eq. (3.17) is satisfied everywhere in the hypercube where averaged final approximation $\bar{F}_{I_1\cdots,I_N}(X_1,\cdots,X_N)$ of Eq. (3.15) is valid.

The above approximation approach, and minor variations of it, has been used in a wide variety of modeling problems, including mathematical modeling of topography, the earth's gravity field, the focal plane distortions of star cameras, modeling the input/output behavior of a synthetic jet actuators, and many other problems in approximation theory, geophysics, engineering, and applied science [see Refs. [8, 40, 43–48]]. We note that that it is relatively straightforward to accommodate non-uniform meshes

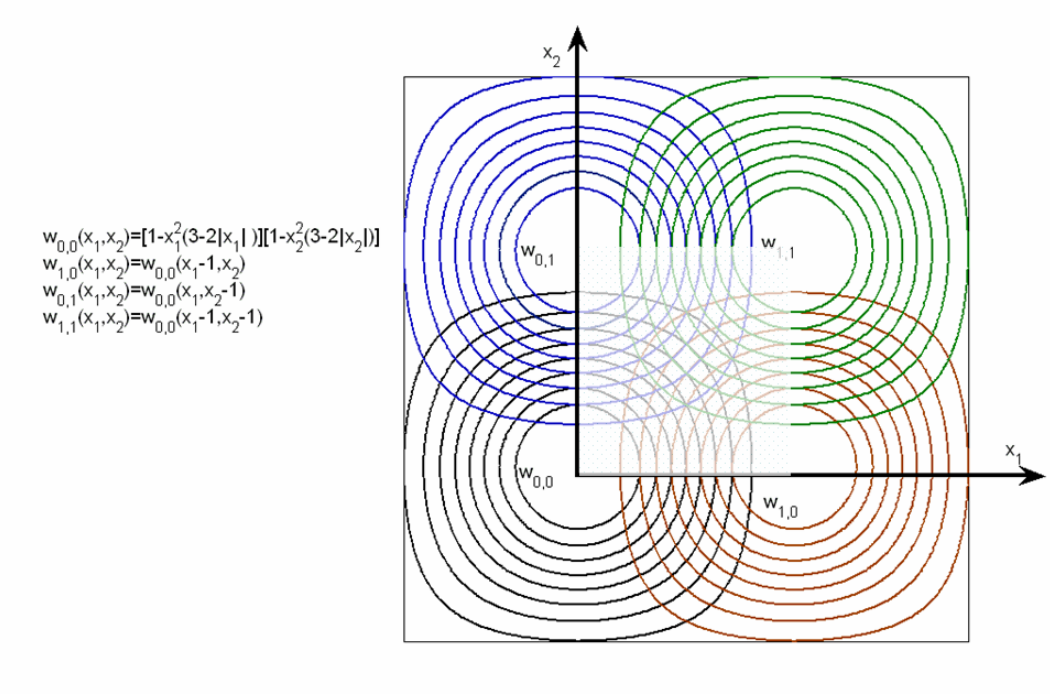


Fig. 15. Four contiguous, overlapping weighting functions for two-dimensional approximation.

but this case is not addressed in the present chapter to avoid notional complexity. While the weighting function approach has much in common with finite element methods, notice the distinction: Whereas conventional FEM methods interpolate nodal values of some distributed quantity into the continuous domain of the finite elements, this weighting function approach instead *averages overlapping local approximations in such a way that piecewise continuity is achieved*, with the user free to choose the local approximations. The degree of the local approximations can be adaptively modified to enhance convergence. Finally, it must be pointed out that one major drawback of the conventional FEM based approach is the generation of a mesh for higher dimensional spaces. However, the use of specially designed weighting functions results in a meshless techniques to alleviate some of the problems related to generating meshes for high dimensioned systems.

The weight functions given above [e.g., Eqs. (3.7), (3.16)] guarantee first order continuity. The generalized weight functions that guarantee arbitrary order continuity are given in Table XI. Only the weight function centered at the origin is tabulated, the other $2^N - 1$ weight functions are obtained by simply shifting the function using the origin translations to the other $2^N - 1$ vertices of the hypercube, analogous to Eqs. (3.14), e.g., using the $2^N - 1$ origin translations:

$$\{(0,0,0,\ldots,0,0,1),(0,0,0,\ldots,0,1,0),\ldots(1,1,1,\ldots,1,1,1)\} \tag{3.18}$$

The weight functions for the first three orders of continuity, for one and two dimensional approximation, are shown in Figs. 16 and 17, respectively.

Table XI. Weight functions for higher order continuity.

order of piecewise continuity	Weight Function: $w_{0,0,\cdots 0}(x_1, x_2, \cdots, x_N) = \prod_{i=1}^{N} w(x_i)$ $w(x),$ for all $x \in \{-1 \leq x \leq 1\},\ y \stackrel{\Delta}{=} \lvert x \rvert$
0	$w(x) = 1 - y$
1	$w(x) = 1 - y^2(3 - 2y)$
2	$w(x) = 1 - y^3(10 - 15y + 6y^2)$
3	$w(x) = 1 - y^4(35 - 84y + 70y^2 - 20y^3)$
$\vdots$	$\vdots$
m	$w(x) = 1 - y^{m+1} \left\{ \frac{(2m+1)!(-1)^m}{(m!)^2} \sum_{k=0}^{m} \frac{(-1)^k}{2m-k+1} \binom{m}{k} y^{m-r} \right\}$

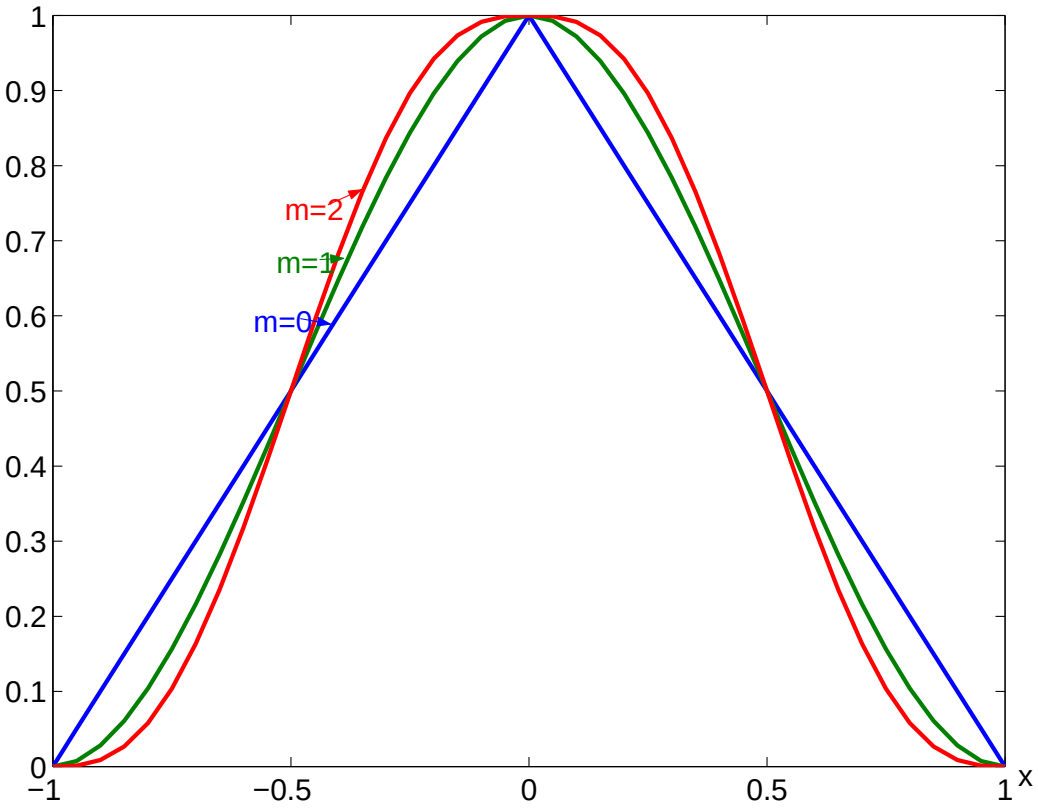


Fig. 16. 1-D weighting functions for various degrees of piecewise continuity.

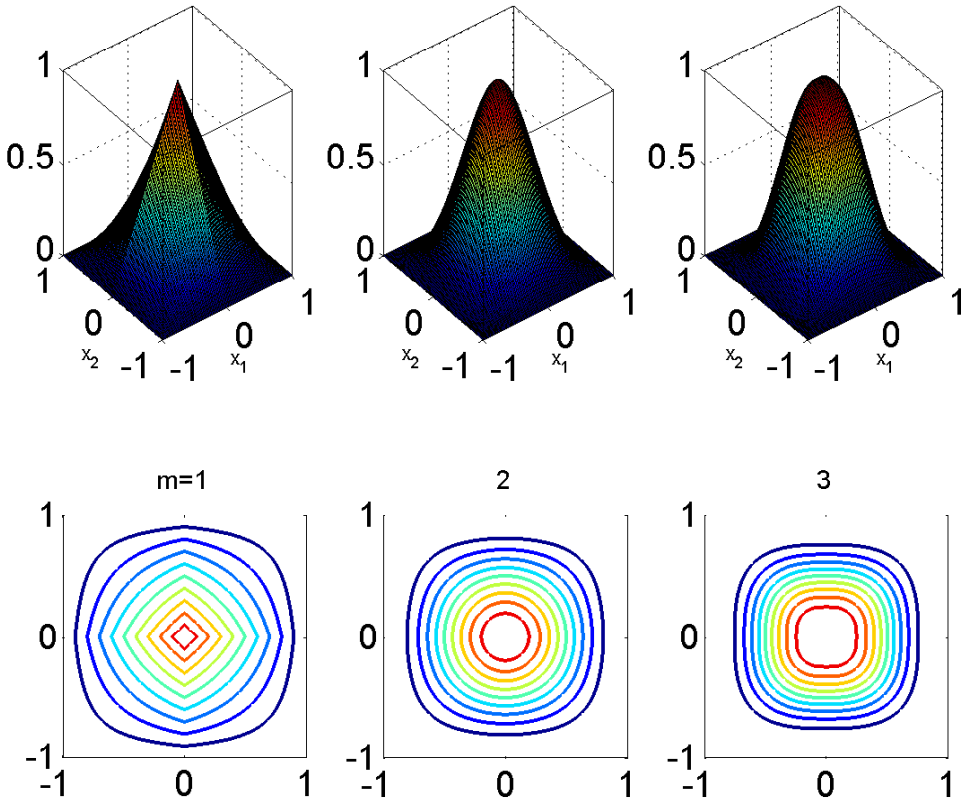


Fig. 17. 2-D weighting functions for various degrees of piecewise continuity.

D. Orthogonal Approximation in 1-, 2- and N-Dimensional Spaces

In previous section, we discussed a novel weighting function interpolation and approximation technique to blend arbitrary smooth overlapping local functional approximations. The weight functions are designed to guarantee the global continuity conditions while retaining near complete freedom on the selection of the generating local approximations. Of course the key to success of the proposed method depends upon the approximation capability of the local functions. There are infinitely many ways to specify good preliminary local approximations averaged in Eq. (3.15). However, guided by Weierstrass approximation theorem [49–52], we know one fundamental and attractive choice is polynomial basis functions to approximate continuous functions on a compact space, to within an approximation error ϵ.

$$F(X) = \sum_i a_i \phi_i(X) + \epsilon = \mathbf{a}^T \mathbf{\Phi}(X) + \epsilon \tag{3.19}$$

where, $\mathbf{\Phi}(.)$ is an infinite dimensional vector of linearly independent polynomial functions and $\mathbf{a}$ is a vector of Fourier coefficients corresponding to polynomial functions. However, according to the following theorem, the continuous function, $F(.)$ can be approximated by a set of orthogonal polynomials with a countable number of terms, instead of infinite terms.

Theorem 1. *Every nontrivial inner-product space has an orthonormal polynomial basis and further if* $\{\phi_i\}$ *is such an orthonormal basis then at most a countable number of Fourier coefficients,* $< F, \phi_i >$ *are non-zero. More generally,* $\Phi(.)$ *is any complete set of basis functions.*

Proof. Let us define a set $S_n = \{i \in \mathcal{I} : | < F, \phi_i > | > 1/n\}$. Here, $\mathcal{I}$ denotes an uncountable index set and should not be confused with the set of integers. Note, to prove this theorem, one just need to show that S_n is a finite set. Now, if $f = \sum_{j \in S_n} <$

$F, \phi_j > \phi_j$ is the orthogonal projection of F onto the subspace, $\mathcal{U} = span[\phi_j : j \in S_n]$ then by the Pythagorean Law:

$$\|F\|^2 = \|(F - f) + f\|^2 = \|F - f\|^2 + \|f\|^2 \geq \|f\|^2 = \sum_{j \in S_n} \| < F, \phi_j > \phi_j\|^2.$$

As ϕ_i is an element of the orthonormal basis, i.e., $\|\phi_i\| = 1$, the above expression reduces to

$$\|F\|^2 \geq \sum_{j \in S_n} | < F, \phi_j > |^2 \geq \sum_{j \in S_n} 1/n^2 = card(S_n)/n^2.$$

Now, as $\|F\| < \infty$ hence $card(S_n) < \infty$, i.e., S_n is a finite set. □

Therefore, this theorem motivates one to choose $\mathbf{\Phi}(.)$ as a finite dimensional vector of orthogonal polynomials. Besides this, the practical consequences of using orthogonal basis functions are enormous. Fourier coefficients of each preliminary approximations can be efficiently computed from ratios of inner-products, avoiding any matrix inversion. Furthermore, Fourier coefficients corresponding to each basis function are independent of each other and so inclusion of new basis function in basis vector does not require us to re-solve for previously computed Fourier coefficients. In this section, we illustrate the procedure of computing the preliminary approximations by using orthogonal basis functions.

1. One Dimensional Case

Consider the approximation of a one variable function $F(X)$. Suppose we are using the weighting function method as illustrated in Fig. 13. The preliminary approximations , while arbitrary, in particular could be chosen to minimize the least square criterion

$$J = \frac{1}{2} \int_{-1}^{1} w(x)[F(X) - F_I(X)]^2 dx \tag{3.20}$$

Furthermore, we consider the case that $F_I(X)$ is a linear combination of a any prescribed set of linearly independent basis functions $\{\phi_0(x), \phi_1(x), \cdots, \phi_n(x)\}$ as

$$F_I(X) = \sum_{i=0}^{n} a_i \phi_i(x); \; X = {}^{I}X + hx \tag{3.21}$$

The least square criterion, making use of Eq. (3.20) can be written as

$$J = J_0 - c^T a + \frac{1}{2} a^T M a \tag{3.22}$$

where

$$J_0 = \frac{1}{2} \int_{-1}^{1} w(x) F^2(x) dx \equiv \frac{1}{2} < F(x), F(x) > \tag{3.23}$$

$$\begin{aligned} c^T &= \left\{ \int_{-1}^{1} w(x)F(x)\phi_0(x)dx \quad \int_{-1}^{1} w(x)F(x)\phi_1(x)dx \quad \cdots \quad \int_{-1}^{1} w(x)F(x)\phi_n(x)dx \right\} \\ &\equiv \left\{ < F(x), \phi_0(x) > \quad < F(x), \phi_1(x) > \quad \cdots \quad < F(x), \phi_n(x) > \right\} \end{aligned} \tag{3.24}$$

$$M = \begin{bmatrix} \mu_{00} & \mu_{01} & \cdots & \mu_{0n} \\ \mu_{01} & \mu_{11} & \cdots & \mu_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{0n} & \mu_{1n} & \cdots & \mu_{nn} \end{bmatrix} = M^T \tag{3.25}$$

$$a \equiv \left\{ a_0 \quad a_1 \quad \cdots \quad a_n \right\}^T ; \; \mu_{ij} =< \phi_i, \phi_j > \equiv \int_{-1}^{1} w(x)\phi_i(x)\phi_j(x)dx \tag{3.26}$$

Observe that minimization of Eq.(3.22) gives the optimum (minimum integral least square fit error) coefficients as

$$a = M^{-1} c \tag{3.27}$$

While Eq. (3.27) holds for an arbitrary set of linearly independent basis functions,

for the special case that the basis functions satisfy the orthogonality condition

$$< \phi_i(x), \phi_j(x) > \equiv \int_{-1}^{1} w(x)\phi_i(x)\phi_j(x)dx = k_i\delta_{ij},\ k_i \triangleq \mu_{ii} = \int_{-1}^{1} w(x)\phi_i^2(x)dx, \quad (3.28)$$

the least square solution of Eq. (3.27), as a consequence of the diagonal M matrix, simplifies to the simple uncoupled result to compute the Fourier coefficients:

$$a_i = \frac{< F(x), \phi_i(x) >}{k_i},\ i = 1, 2, \cdots, n \quad (3.29)$$

Thus, if we can construct basis functions orthogonal with respect to the particular weight functions of Table XI, we enjoy the usual advantages that flow from approximation of orthogonal functions but now in a global/local approximation setting. We consider the special case of $m = 1$; the construction of the corresponding orthogonal basis functions requires the Gramm-Schmidt process. Using the methods of the Appendix B, it can be verified that the basis functions given in Table XII satisfy the orthogonality conditions of Eq. (3.28). Note that c_n in Table XII is determined so that $\phi_n(x)$ satisfies the normalization, $|\phi_n(\pm 1)| = 1$. The first four orthogonal functions are plotted in Fig. 18.

2. Two Dimensional Case

Consider the approximation of a two variable function $F(X_1, X_2)$. The typical preliminary local approximations $F_{IJ}(X_1, X_2)$, while arbitrary, in particular could be chosen to minimize the least square criterion

$$J = \frac{1}{2}\int_{-1}^{1}\int_{-1}^{1} w(x_1, x_2)[F(X_1, X_2) - F_{IJ}(X_1, X_2)]^2 dx_1 dx_2 \quad (3.30)$$

Furthermore, we consider the case that $F_{IJ}(X_1, X_2)$ is chosen as a linear combination of a prescribed set of linearly independent basis functions $\{\phi_{ij}(x)\};\ i = 1, 2, ...n; j =$

Table XII. One dimensional basis functions orthogonal with respect to the weight function $(x) = 1 - x^2(3 - 2|x|)$.

degree	Basis Functions, $\phi_j(x)$
0	1
1	x
2	$(-2 + 15x^2)/13$
3	$(- 9x \ + \ 28 \ x^3)/19$
$\vdots$	$\vdots$
n	$\phi_n(x) = \frac{1}{c_n}\left[x^n - \sum_{j=0}^{n-1} \frac{<x^n,\phi_j(x)>}{<\phi_j(x),\phi_j(x)>}\phi_j(x)\right]$

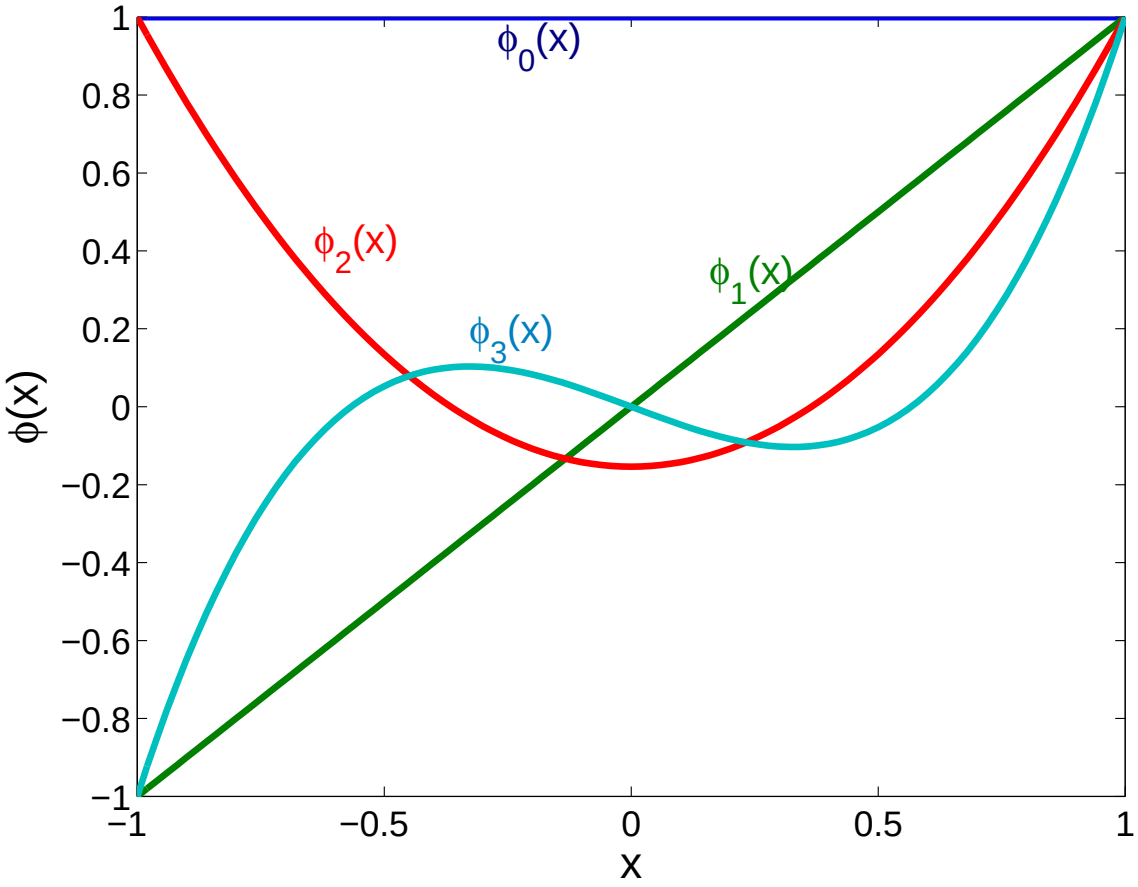


Fig. 18. One dimensional orthogonal basis functions.

$1, 2, ...n.$ as

$$F_{IJ}(X_1, X_2) = \sum_{i=0}^{n}\sum_{j=0}^{n} a_{ij}\phi_{ij}(x_1, x_2);\ X_1 = {}^{I}X_1 + hx_1,\ X_2 = {}^{J}X_2 + hx_2 \tag{3.31}$$

In particular, consider the multiplicative structure for the weight function

$$w(x_1, x_2) = [1 - x_1^2(3 - 2|x_1|)][1 - x_2^2(3 - 2|x_2|] \tag{3.32}$$

and the corresponding two dimensional basis functions

$$\phi_{ij}(x_1, x_2) = \phi_i(x_1)\phi_j(x_2) \tag{3.33}$$

where we choose the one-dimensional basis functions $\phi_i(x)$ from Table XII that are orthogonal with respect to $w(x) = 1 - x^2(3 - 2|x|)$. Introducing the inner product notation:

$$< \alpha(x_1, x_2), \beta(x_1, x_2) > \stackrel{\Delta}{=} \int_{-1}^{1}\int_{-1}^{1} w(\xi_1, \xi_2)\alpha(\xi_1, \xi_2)\beta(\xi_1, \xi_2)d\xi_1 d\xi_2 \tag{3.34}$$

As a consequence of the orthogonality $\phi_i(x)$ from Table XII , the choice of Eqs. (3.32), (3.33) and the definition of Eq. (3.34), it is evident that the functions of Eqs. (3.32) are orthogonal, because

$$\begin{aligned} < \phi_{ij}(x_1, x_2), \phi_{lm}(x_1, x_2) > &\stackrel{\Delta}{=} \int_{-1}^{1}\int_{-1}^{1} w(\xi_1)w(\xi_2)\phi_{ij}(\xi_1, \xi_2), \phi_{lm}(\xi_1, \xi_2)d\xi_1 d\xi_2 \\ &= \underbrace{\int_{-1}^{1} w(\xi_1)\phi_i(\xi_1)\phi_l(\xi_1)d\xi_1}_{k_i\delta_{il}} \underbrace{\int_{-1}^{1} w(\xi_2)\phi_j(\xi_2)\phi_m(\xi_2)d\xi_2}_{k_j\delta_{jm}} \\ &= k_i\delta_{il}k_j\delta_{jm} \end{aligned} \tag{3.35}$$

As a consequence of orthogonality, it follows that the Fourier coefficients for

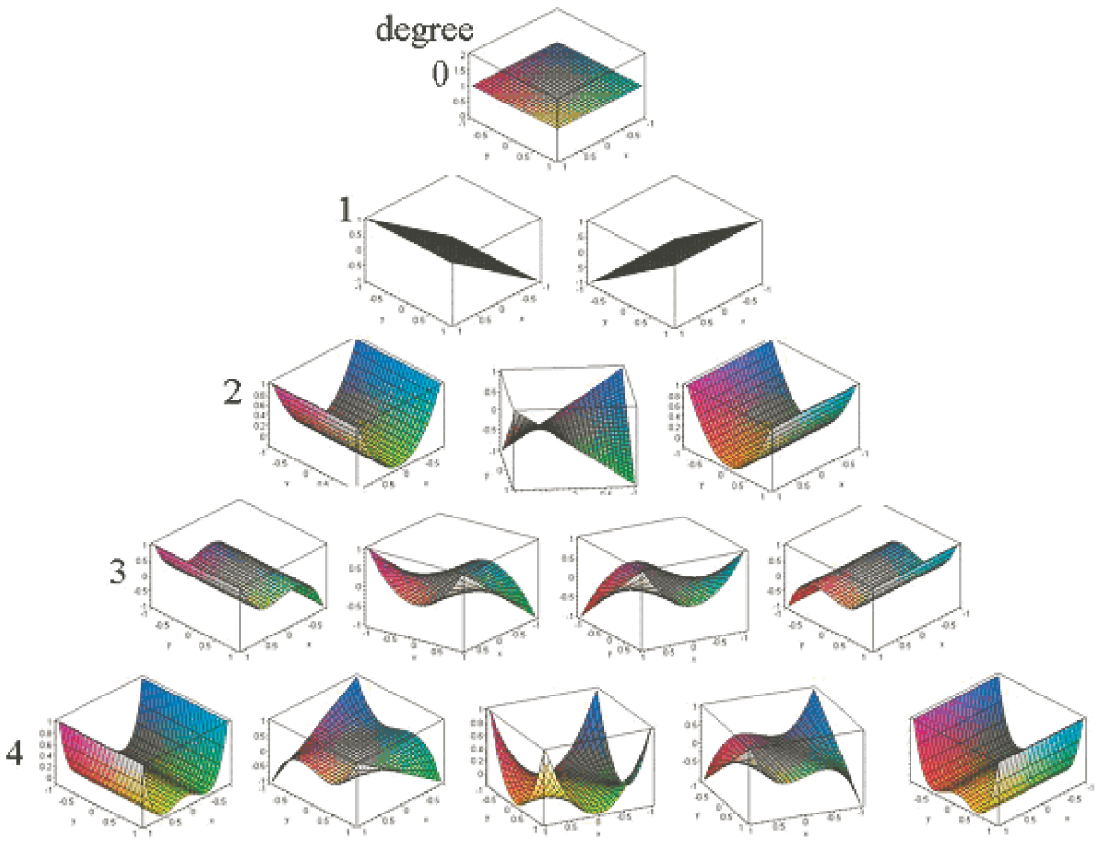


Fig. 19. Two dimensional orthogonal basis functions.

two-dimensional approximations are:

$$a_{ij} = \frac{< \phi_{ij}(x_1, x_2), F(X_1, X_2) >}{< \phi_{ij}(x_1, x_2), \phi_{ij}(x_1, x_2) >} = \frac{< \phi_{ij}(x_1, x_2), F(X_1, X_2) >}{k_i k_j} \tag{3.36}$$

The first five sets (degrees zero through four) of the two dimensional orthogonal polynomials of Eq. (3.33) are shown in Fig. 19.

3. N- Dimensional Case

Consider the approximation of a function $F(X_1, X_2, \cdots, X_N)$ of N variables. The preliminary local approximations $F_{I_1 \cdots I_N} F(X_1, X_2, \cdots, X_N)$, while arbitrary, in particular could be chosen to minimize the least square criterion

$$J = \frac{1}{2} \int_{-1}^{1} \int_{-1}^{1} \cdots \int_{-1}^{1} w(x_1, \cdots, x_N)[F(X_1, \cdots, X_N) - F_{I_1 \cdots I_N}(X_1, \cdots, X_N)]^2 dx_1 \cdots dx_N \tag{3.37}$$

Furthermore, we consider the case that $F_{I_1 \cdots I_N} F(X_1, X_2, \cdots, X_N)$ is chosen as a linear combination of a prescribed set of linearly independent basis functions $\{\phi_{i_1 \cdots i_N}(x_1, \cdots, x_N)\}$ as

$$F_{I_1 \cdots I_N}(X_1, \cdots, X_N) = \sum_{i_1=0}^{n} \cdots \sum_{i_N=0}^{n} a_{i_1 \cdots i_N} \phi_{i_1 \cdots i_N}(x_1, \cdots, x_N) \tag{3.38}$$

where, the transformation from the local non-dimensional coordinates $(x_1, \cdots, x_N)$ to the global coordinates $(X_1, \cdots, X_N)$ is:

$$X_1 = {}^{I_1}X_1 + hx_1, \cdots, X_N = {}^{I_N}X_N + hx_N \tag{3.39}$$

In particular, consider the continued product structure for the weight function

$$w(x_1, \cdots, x_N) = [1 - x_1^2(3 - 2|x_1|)] \cdots [1 - x_N^2(3 - 2|x_N|] = \prod_{i=0}^{N} [1 - x_i^2(3 - 2|x_i|)] \tag{3.40}$$

and similarly for the basis functions

$$\phi_{i_1 \cdots i_N}(x_1, \cdots, x_N) = \prod_{i=0}^{N} \phi_i(x_i) \tag{3.41}$$

where we choose the one-dimensional basis functions $\phi_i(x)$ from Table XII that are orthogonal with respect to $w(x) = 1 - x^2(3 - 2|x|)$.

Introducing the inner product notation:

$$< \alpha(x_1, \cdots, x_N), \beta(x_1, \cdots, x_N) > \stackrel{\Delta}{=} \int_{-1}^{1} \cdots \int_{-1}^{1} w(\xi_1, \cdots, \xi_N)\alpha(\xi_1, \cdots, \xi_N)\beta(\xi_1, \cdots, \xi_N)d\xi_1 \cdots d\xi_N \tag{3.42}$$

As a consequence of the orthogonality $\phi_i(x)$ from Table XII , the choice of Eqs. (3.37), (3.40), and (3.41) it is evident that the N-dimensional functions of Eqs. (3.38) are orthogonal, because

$$\begin{aligned} & < \phi_{i_1 \cdots i_N}(x_1, \cdots x_N), \phi_{j_1 \cdots j_N}(x_1, \cdots x_N) > \\ = & \int_{-1}^{1} \cdots \int_{-1}^{1} \left(w(\xi_1) \ldots w(\xi_N)\phi_{i_1 \cdots i_N}(\xi_1, \cdots, \xi_N)\phi_{j_1 \cdots j_N}(\xi_1, \cdots, \xi_N)\right) d\xi_1 \cdots d\xi_N \\ = & [k_{i_1} k_{i_2} \cdots k_{i_N}][\delta_{i_1 j_1}\delta_{i_2 j_2} \cdots \delta_{i_N j_N}] \end{aligned} \tag{3.43}$$

As a consequence of orthogonality, it follows that the least square amplitudes are the Fourier coefficients,

$$a_{i_1 \cdots i_N} = \frac{< \phi_{i_1 \cdots i_N}(x_1, \cdots, x_N), F(X_1, , \cdots, X_N) >}{\prod_{j=1}^{N} k_j} \tag{3.44}$$

Remarkably, the weight functions, basis functions, and orthogonality conditions for N-dimensional approximation are generated directly from the one dimensional results. Furthermore, we arrive at a computationally efficient piecewise continuous approximation in N dimensions with the added benefits that the local approximations are linear combinations of basis functions orthogonal to the same weight function used in averaging the overlapping approximations. Clearly, the first order piecewise continuity implicit in the above developments is "promoted" from first to m^{th} order continuity by simply choosing the appropriate weight function from Table XI.

E. Algorithm Implementation

In this section, the step by step implementation of the Global-Local Orthogonal Mapping (GLO-MAP) algorithm is discussed. Attention here is upon the hyper surface approximation when unevenly spaced discrete measured data are available whereas the above developments are for the case of continuous measurements. The main steps of the GLO-MAP algorithm are as follows:

1. Choose a set of sequential neighboring points, ${}^{I}X$, arbitrary in number and location. These points serve as the *centroids of validity* for the local functional approximation, F_I. The density and location of these points depends upon numerous factors like location and density of available measurement data and desired degree of approximation.

2. Choose set of basis functions based on computational efficiency or a priori knowledge of the nature of the given input-output data to approximate $F(X)$ in local neighborhood of a centroid of validity, ${}^{I}X$. The local neighborhood of a centroid is defined in a such a way that the number of measurement points in local neighborhood are at least equal to the number of basis functions used to approximate local behavior in that particular local domain. Generally, the sizing of the local neighborhood is dictated by the support or domain of the weight functions discussed in section C. One attractive choice for the basis functions is the orthogonal polynomial basis functions as discussed in the previous section.

3. Determine the Fourier coefficients corresponding to each local approximation. For the approximation of a given continuous functional form or from dense discretely measurable functions, the Fourier coefficients can be computed by numerically evaluating the integral expression in Eq. (3.44) using standard

numerical integration algorithms [53]. In case of the GLO-MAP algorithm the numerical integration procedure can be summarized as below:

(a) Determine Gaussian quadrature points, $\mathbf{X}_G$, in unit hypercube.

(b) For each quadrature point, determine the measurement points $\mathbf{X}_i$, for which weight function has non-zero value.

(c) Determine function value at quadrature point, $F(\mathbf{X}_G)$ can be taken as a weighted average of function value at $\mathbf{X}_i$.

$$F(\mathbf{X}_G) = \frac{1}{\sum_i w(\mathbf{X}_i)} \sum_i F(\mathbf{X}_i) w(\mathbf{X}_i) \tag{3.45}$$

(d) Evaluate the numerical integral in Eq. (3.44)

It should be noted that Gaussian quadrature points are the same for each local approximation and can be pre-computed. However, the numerical evaluation of integral expression of several variables, over regions with dimension greater than one, is not easy. As a rule of thumb, the number of function evaluations needed to sample an N-dimensional space increases as the N^{th} power of the number needed to do a one-dimensional integral, resulting in increased computational cost associated with numerical integration proportion to the N^{th} power. To avoid these difficulties, one can construct the orthogonal basis functions which satisfies exactly the discrete orthogonality condition, using the hypergeometric difference equation and the procedure given in Ref. [54]

$$\text{Discrete Orthogonality Condition:} \sum_i \phi_n(X_i) w(X_i) \phi_m(X_i) = k_{mn}^2 \delta_{mn} \tag{3.46}$$

However, the major drawback of constructing discrete orthogonal polynomials is that their functional form changes for each local approximation depending upon the number of measurements available in each local neighborhood. In

future work, we anticipate that we will develop procedures for constructing discrete orthogonal polynomials using the weight functions derived in this chapter. More generally, and especially for a general not-necessarily dense set of discrete measurements, conventional SVD or linear least squares algorithms can be employed to construct the local approximations. It should be noted that the freedom to select local approximations affords a new level of flexibility in numerical methods for the class of problems under consideration.

4. The final step of the GLO-MAP algorithm is the use of weighting functions to merge the local approximations into a single m^{th} order continuous functional model. This is accomplished by using the weight functions listed in Table XI and Eqs. (3.15)-(3.17). Note, this step is the most important feature of the GLO-MAP algorithm as it reduces the systematic error introduced due to the neglected interaction between different local models by blending overlapping local approximations into a global one.

Note, as usual, that the size h of the local neighborhood is an important factor which affects the overall accuracy and computational cost of the GLO-MAP algorithm. To find the optimal value of this parameter, analogous to mesh refinement in the FEM method, one can construct a multi-resolution algorithm which iteratively refines the local neighborhoods until introduction of more local neighborhoods does not bring any improvement in the learning of input-output mapping. The major steps involved in this algorithm are depicted in Fig. 20.

1. Sequential Version of the GLO-MAP Algorithm

There are many engineering application problems which needs to be solved in an iterative manner in real-time by successively approximating the input-output data. Many

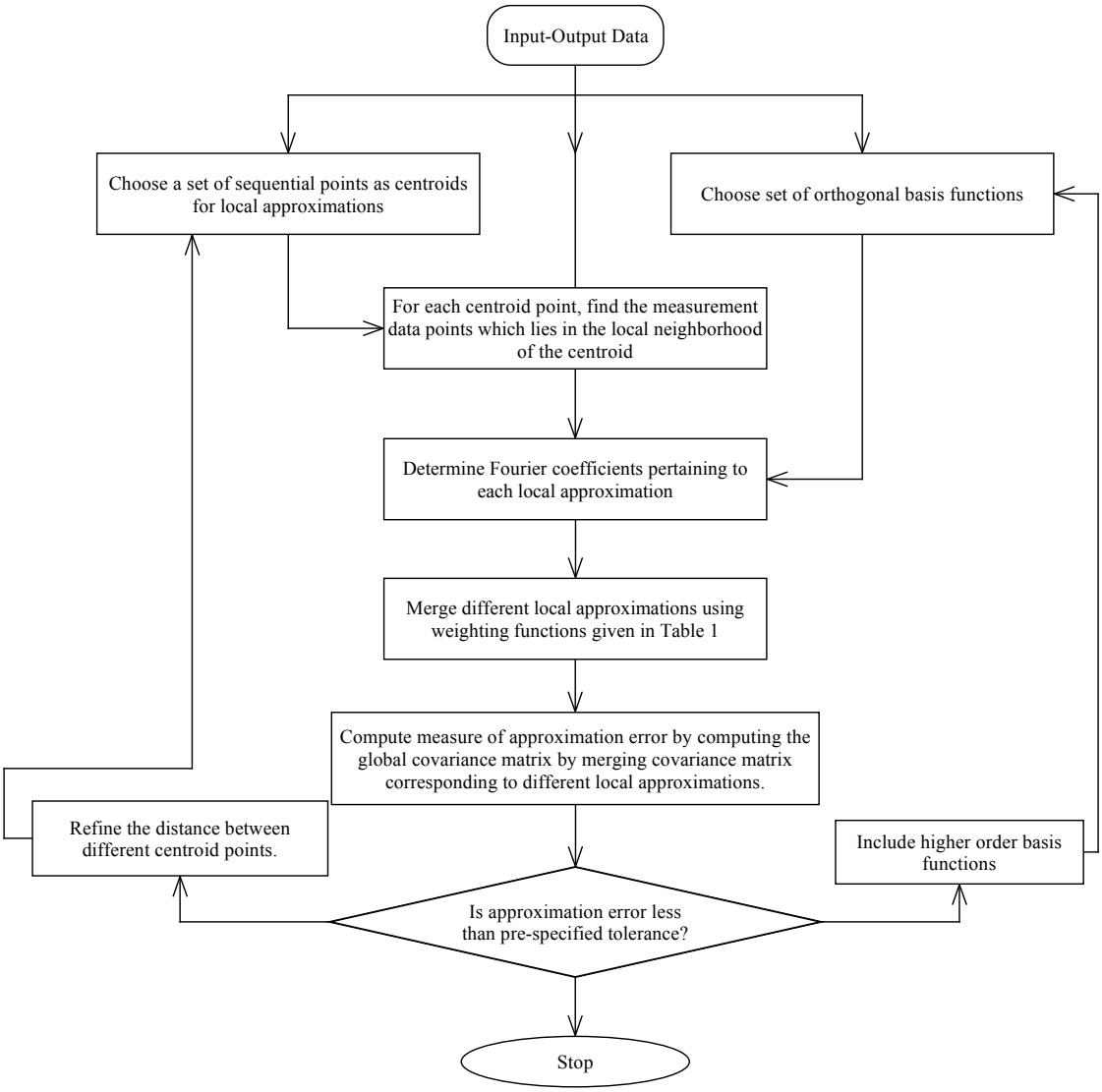


Fig. 20. Flowchart for the GLO-MAP based multi-resolution algorithm.

recursive approximation algorithms are presented in the literature but the Kalman filter [29] is one of the most widely used and powerful tools for recursive estimation problems. We note that the Kalman filter algorithm is very attractive for the problem at hand as it can also be used to update off-line a priori learned GLO-MAP network parameters in real time whenever new measurements are available. However, the main challenge associated with the use of the Kalman filter in the GLO-MAP algorithm is the dynamic state vector i.e. the components of state vector of Kalman filter changes with every measurement data depending upon the location of measurement data relative to centroids of validity of local approximations. Actually, the total number of unknowns for the GLO-MAP network is Mn coefficients of different local approximations. Here, n is the number of basis functions associated with each local approximation and M is the total number of these local approximations. However, when a new measurement is available, we just need to update $2^N n$ unknowns depending upon the location of measurement data since only the neighboring 2^N local approximations (associated with the 2^N vertices of the hypercube containing the new measurement)will have non-zero contribution in final global map obtained by merging different local approximations using Eq. (3.15). The main steps involved in the implementation of sequential version of the GLO-MAP algorithm are as follows:

1. Depending upon prior knowledge of the input space, choose a set of sequential points, ${}^{I}X$, which serve as the centroids of validity for the local functional approximations, F_I. The density and location of these points depend upon numerous factors like location and density of available measurement data and desired degree of approximation.

2. Choose a set of basis functions (preferably orthogonal functions) to approximate $F(X)$ in the local neighborhood of a centroid of validity, ${}^{I}X$. Initialize the

coefficients of each basis functions to be zero and associated covariance matrix ($\mathbf{P_G}$) to be identity times a large number.

3. Given a new measurement data point, $(\mathbf{X}, F(\mathbf{X}))$, find the neighboring 2^N centroids such that weight function associated with these 2^N centroids have non-zero value at the measurement point.

   (a) Include the coefficients of the basis functions associated with these centroids in the local algebraic Kalman filter state vector, denoted by $\mathbf{x}$. Let $\mathbf{x}_G$ denote the global super-set of coefficients and $\mathbf{M}$ be a *selection matrix* consisting of zeros and ones that satisfies $\mathbf{x} = \mathbf{M}\mathbf{x}_G$.

   (b) Extract rows and columns of $\mathbf{P}_G$ corresponding to the coefficients associated with these 2^N centroids and denote them by $\mathbf{P}_k^- = \mathbf{M}\mathbf{P}_G\mathbf{M}^T$.

4. Use the following equations to compute the local Kalman gain and update state vector, $\mathbf{x}$, and the associated covariance matrix, $\mathbf{P}$.

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}(F(\mathbf{X}) - \mathbf{\Phi}(.)\mathbf{x}_k^-) \tag{3.47}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}\mathbf{\Phi}(.))\mathbf{P}_k^- \tag{3.48}$$

$$\mathbf{K} = \mathbf{P}_k^-\mathbf{\Phi}(.)^T(\mathbf{\Phi}(\mathbf{X})\mathbf{P}_k^-\mathbf{\Phi}(.)^T + \mathbf{R}_k)^{-1} \tag{3.49}$$

where, superscript $^-$ and $^+$ denote the value of variables before and after updating various unknown using given measurement data respectively. Subscript k denotes the centroid number associated with the k^{th} approximation and varies from 1 to 2^N. Further, matrix $\mathbf{\Phi}$ and $\mathbf{R}$ are given by following equations:

$$\mathbf{\Phi}_k = \begin{bmatrix} \phi_1(\mathbf{X}_k) & \cdots & \phi_N(\mathbf{X}_k) \end{bmatrix} \tag{3.50}$$

$$\mathbf{R}_k = \sigma w(\mathbf{X}_k) \tag{3.51}$$

where, $\mathbf{X}_k$ denotes the local coordinates of measurement point assuming origin at k^{th} neighboring centroid and σ denotes the variance of the measurement data.

5. Update rows and columns of the global covariance matrix, $\mathbf{P}_G$ and coefficients associated with each local approximation.

6. Once the value of $\mathbf{P}_G$ is less than a pre-specified tolerance, use appropriate weighting functions to merge various local approximations into a single m^{th} order continuous functional model. This is accomplished by using the weight functions listed in Table XI and Eqs. (3.15)-(3.17), centered at the 2^N vertices of the local volume containing the point $\mathbf{x}$.

F. Illustrative Engineering Applications

The approximation algorithm presented in this chapter has been tested on a variety of engineering applications. In this section, we present four sets of results from these studies: (*i*) an analytical test case for function approximation, (*ii*) a dynamical System identification from wind tunnel testing of synthetic jet actuation wing, (*iii*) a vibrating Space Based Radar (SBR) antenna surface approximation, and (*iv*) an approximation of the "pork-chop" surface for a family of Lambert's problem solution.

1. Function Approximation

The test case for the analytical function approximation is constructed by using the following surface [33].

$$\begin{aligned} f(X_1, X_2) &= \frac{10}{(X_2 - X_1^2)^2 + (1 - X_1)^2 + 1} + \frac{5}{(X_2 - 8)^2 + (5 - X_1)^2 + 1} \\ &+ \frac{5}{(X_2 - 8)^2 + (8 - X_1)^2 + 1} \end{aligned} \tag{3.52}$$

Figs. 21(a) and 21(b) show the true surface and contour plots for the function given by Eq. (3.52). According to our experience, this particular function has many important features such as sharp ridge line that is very difficult to learn with existing function approximation algorithms with reasonable number of nodes.

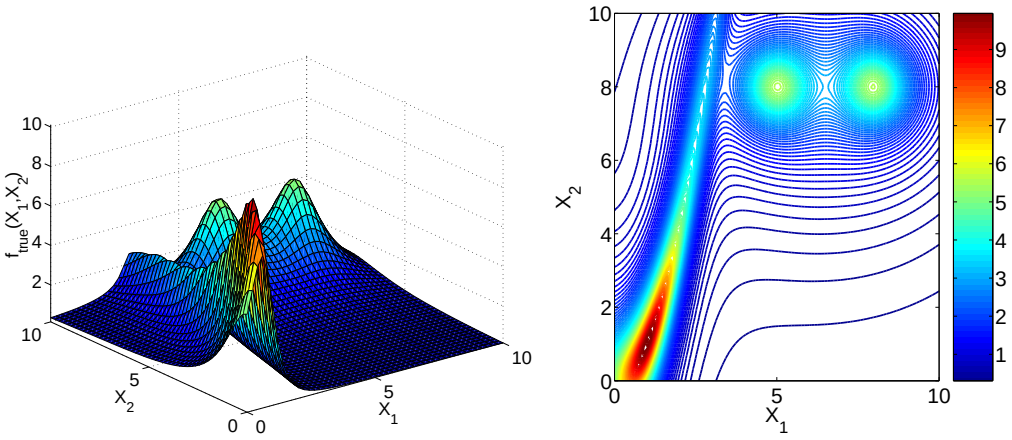
To approximate the function given in Eq. (3.52), the whole input region is divided into a set of finite element cells, defined with cartesian coordinates, X_1 and X_2. Therefore, 10×10 modeling region can be divided into different number of cells depending upon cell length. For example, the whole input region can be modeled by a total of 16 cells of dimension 1.2766×1.2766 or by a total of 576 cells of dimension 0.4082×0.4082. Further, to obtain preliminary approximations for a particular cell, as described in section D, two test cases are considered. In first test case, the continuous functional expression given by Eq. (3.52) is used to obtain the coefficients of the preliminary local approximations while in second test case a discrete measurement data is used to compute preliminary approximations. The discrete measurement data set is generated by taking 100 random samples over the interval [0-10, 0-10] for both X_1 and X_2, giving total 10^4 measurements.

The local approximation of analytical function $\hat{f}(x_1, x_2)$, for a particular cell is modeled by orthogonal polynomials of the form:

$$\hat{f}(x_1, x_2) = \sum_i \sum_j a_{ij} \phi_i(x_1, x_2) \phi_j(x_1, x_2), \; i + j \leq 2 \tag{3.53}$$

The orthogonal functions (listed in table XII), ϕ_i and ϕ_j, are chosen in such a way that the degree of $\hat{f}(x_1, x_2)$ is always less than or equal to 2. Further, x_1 and x_2 denote the local cell coordinates defined as below:

$$x_1 = 2(X_1 - X_{1_m})/X_{1_{cell}} \quad x_2 = 2(X_2 - X_{2_m})/X_{2_{cell}} \tag{3.54}$$



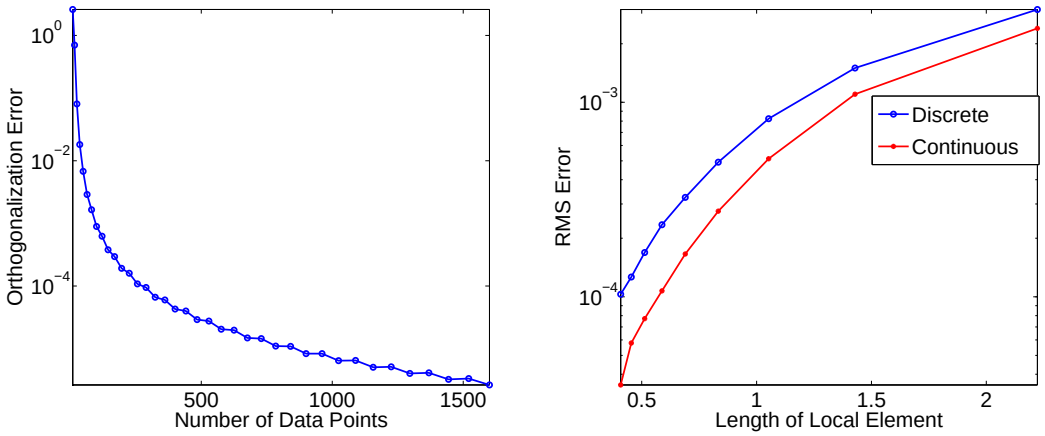
(a) True surface (b) True contours

Fig. 21. Test surface and contour plots of Eq. (3.52)

where, (X_{1_m}, X_{2_m}) and $X_{1_{cell}} \times X_{2_{cell}}$ represent the centroid and dimensions of a particular cell respectively.

Since ϕ_i and ϕ_j are chosen to be orthogonal polynomial functions therefore, unknown coefficients a_{ij} can be determined from Eq. (3.36). Due to complex nature of the function in Eq. (3.52), the various integral expressions in Eq. (3.36) are computed by numerical integration as explained in section E. The total number of Gauss quadrature points required for numerical integration are decided by checking the orthogonality condition of Eq. (3.35) i.e. $< \phi_{ij}(x_1, x_2), \phi_{kl}(x_1, x_2) > \neq k_i\delta_{ik}k_j\delta_{jl}$. Fig. 22(a) shows the plot of orthogonalization error, defined as $|| < \phi_{ij}(x_1, x_2), \phi_{kl}(x_1, x_2) > - k_i\delta_{ik}k_j\delta_{jl}||$, versus number of Gauss quadrature points in a particular cell. As expected, the orthogonalization error decreases quickly as the number of quadrature points inside a particular cell increases. To generate this plot, we decided to use a total of 100 (10 in each direction) quadrature points.

Now, in the first test case, the analytical expression given by Eq. (3.52) is used

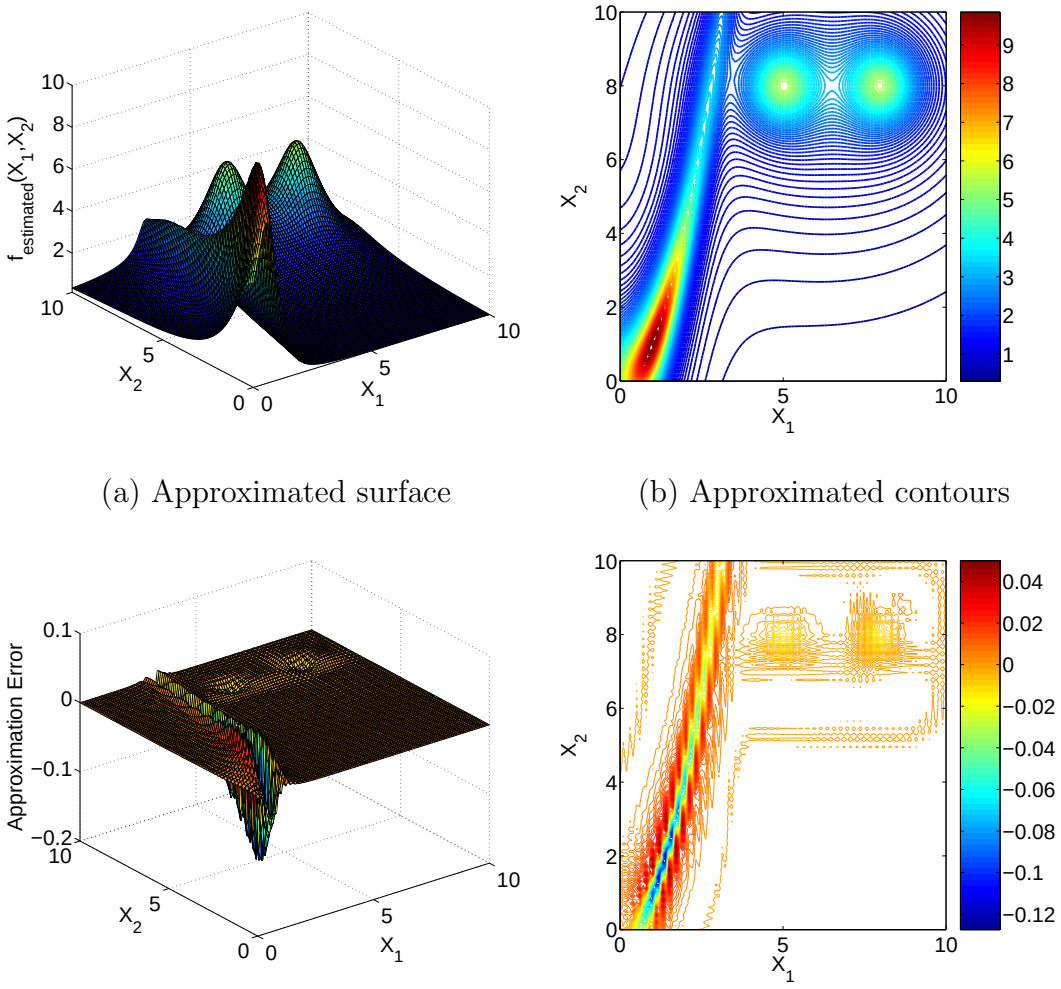


(a) Orthogonalization error vs number of data points

(b) Approximation error vs grid size

Fig. 22. Error Analysis.

to obtain the integrand values at different quadrature points while in second case the integrand values were obtained by weighting average procedure as discussed in section E. As discussed earlier, the approximation error depends upon the grid size, therefore, it was decided to study the Root Mean Square (RMS) approximation error as a function of cell size for a fixed order of polynomials. Fig. 22(b) shows the plot of root mean square approximation error versus cell size for both the test cases. As expected, the root mean square error decreases for both the test cases as cell size decreases. Due to the fact that as cell size decreases, the local behavior of the unknown function can be approximated more accurately and first order weighting function interpolation becomes more accurate. Further, it is also apparent from this figure that the RMS error for the first test case is less than the second test case. This is because in second test case the integrand values are obtained by interpolating the available measurement values in the local neighborhood of a particular quadrature point while in first test case the analytical function expression is used to compute numerical integrals. It



(a) Approximated surface

(b) Approximated contours

(c) Error surface

(d) Error contours

Fig. 23. The GLO-MAP approximation results for analytical function of Eq. (3.52)

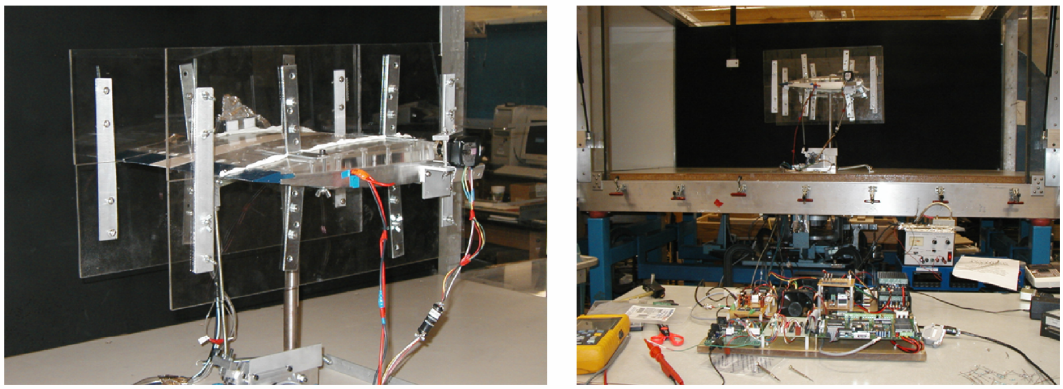
should be also noted that the approximation accuracy for either test case is bounded by the orthogonalization error which in this case is $O(10^{-5})$. These results provide a basis for optimism regarding the practical utility of this approach.

Finally, Figs. 23(a) and 23(b) show surface and contour plots of the approximated surface whereas Figs. 23(c) and 23(d) shows plots of the approximation error surface and the error contours respectively. Not surprising - the largest errors occur along the knife edge of the sharp ridge - experimentation indicted we can reduce the maximum error to any tolerance dictated by the feature sharpness and data spacing. These results corresponds to a total of 625 cells. From these figures, it is clear that we are able to learn the analytical function given in Eq.(3.52) very well with worst case relative approximation errors less than 2%. Of course, it is evident using dense measurements along with either a smaller h or higher degree local approximations, we can make the errors as small as desired.

We also mention that for any standard interpolation algorithm to represent the surface data shown in Fig. 21(a), the interpolation matrix consists of n^2 real numbers or $n(n+1)/2$ real numbers. For example, if radial basis functions are used, we require $O(n^3)$ Floating Point Operations (FLOPs) to solve the associated system of equations. For a surface with 10^4 data points, one needs $O(10^{12})$ FLOPs, which is obviously impractical for many practical purposes. Further for any global representation, one needs to worry about the possible rank deficiency of the large interpolation matrix to solve the system of equations. However, the GLO-MAP algorithm, proposed here, greatly reduces the overall number of basis functions, requires no large matrix inverse, improves the surface approximation accuracy, and provides a feasible path to achieve any desired precision by appropriate refinements. We now consider a diverse set of applications to obtain further insight.

2. Synthetic Jet Actuator Modeling

There is a significant thrust in aerospace industry to develop advanced technologies that would enable adaptive, intelligent, shape controllable micro and macro structures, for advanced aircraft and space systems. These designs involve precise control of the shape of the structures with micro and macro level manipulations (actuation). Synthetic jet actuators [55] (SJA) represent an alternative to reconfigurable wings that adaptively shapes the flow field and pressure field around a fixed wing and are one of such devices being researched for active flow control that enable enhanced performance of conventional aerodynamic surfaces at high angles of attack and these technologies may lead to full replacement of hinged control surfaces thereby achieving hingeless control. Active flow control can be achieved by embedding sensors and actuators at micro scales on an aerodynamic structure. The desired force and moment profiles are achieved by impinging a jet of air to alter the flowfield using these actuators and thereby creating a desired pressure distribution over the structure. The distinguishing feature of synthetic jet actuation modeling problem is that the relationship between input and output variables is poorly modeled and is nonlinear in nature. Further, un-steady flow effects make it impossible to capture the physics fully from static experiments. The issue at hand is to derive comprehensive mathematical models that capture the input output behavior of SJA so that one can derive automatic control laws that can command desired lift and moment profiles. While the conventional modeling approaches evolve to handle these problems, one can pursue non parametric, multi-resolution, adaptive input/output modeling approaches to capture macro static and dynamic models directly from experiments. However, the large data sets need to be replaced by a consistent multi-dimensioned approximation, consistent with the accuracy of the measurement. In this section, we show the appli-



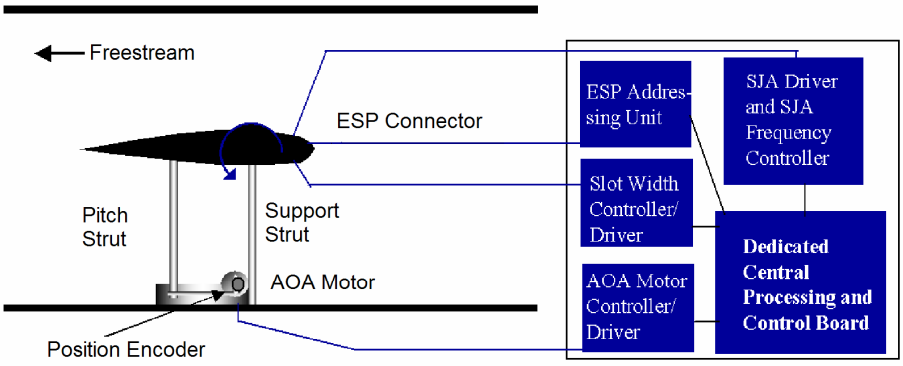


Fig. 24. Hingeless control-dedicated experimental setup for synthetic jet actuation wing.

cation of the GLO-MAP algorithm to learn the mapping between the synthetic jet actuation parameters (frequency, direction, etc. for each actuator) and the resulting aerodynamic lift, drag, and moment. These results show the effectiveness of the GLO-MAP algorithm presented in this chapter to learn the nonlinear input-output mapping for the synthetic jet actuation wing.

a. Experimental Set up

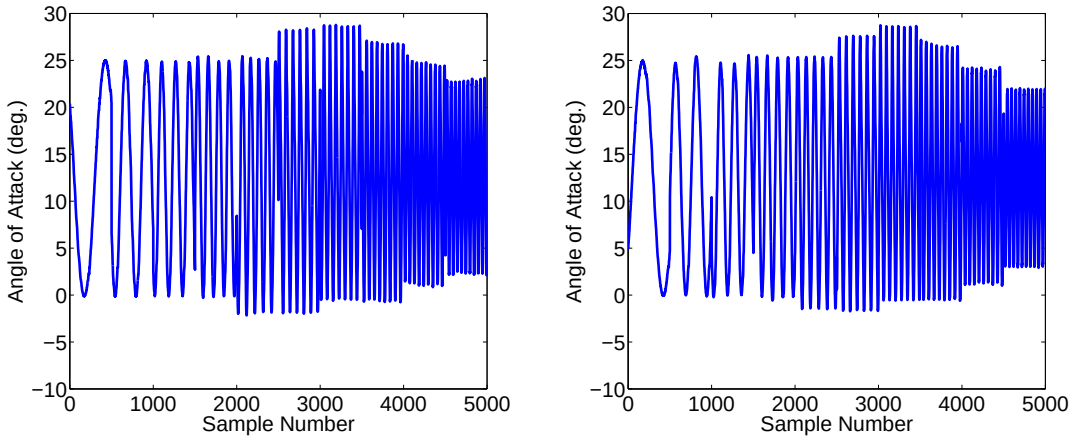
A Hingeless-Control-Dedicated experimental setup has been developed. As part of the initial effort, a stand-alone control unit has been developed that controls all of

the wing's and SJA's parameters and variables. The setup is installed in the $3' \times 4'$ wind tunnel of the Texas A&M Aerospace Engineering Department (Fig. 24). The test wing profile for the dynamic pitch test of the synthetic jet actuator is a NACA 0015 airfoil. This shape was chosen due to the ease with which the wing could be manufactured and the available interior space for accommodating the synthetic jet actuator (SJA).

Experimental evidence [47, 55] suggests that a SJA, mounted such that its jet exit tangentially to the surface, has minimal effect on the global wing aerodynamics at low to moderate angles of attack. The primary effect of the jet is at high angles of attack when separation is present over the upper wing surface. In this case, the increased mixing associated with the action of a synthetic jet, delays or suppresses flow separation. As such, the effect of the actuator is in the non-linear post stall domain. To learn this nonlinear nature of SJA experiments were conducted with the control-dedicated setup shown in Fig. 24. The wing Angle of Attack (AOA) is controlled by the following reference signal.

1. Oscillation type: sinusoidal Oscillation magnitude: 12.5°.
2. Oscillation offset (mean AOA): 12.5°
3. Oscillation frequency: from 0.2Hz to 20Hz.

In other words, the AOA of airfoil is forced to oscillate from 0° to 25° at a given frequency (see Fig. 25). The experimental data collected were the time histories of the pressure distribution on the wing surface (at 32 locations). The data was also integrated to generate the time histories of the lift coefficient and the pitching moment coefficient. Data was collected with the SJA on and with the SJA off (i.e. with and without active flow control). All the experimental data were taken for 5 sec at a 100 Hz sampling rate.



(a) Angle of attack variation without SJA.

(b) Angle of attack variation with sja actuation frequency of 60 Hz.

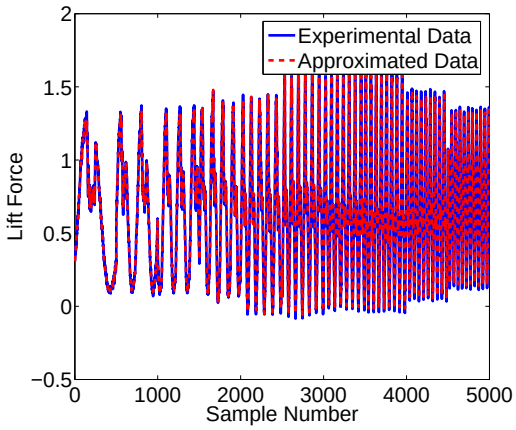
Fig. 25. Angle of attack variation.

The experiments described above were performed at a free-stream velocity of $25m/sec$. From the surface pressure measurements, the lift and pitching moment coefficients were calculated via integration. As the unknown SJA model is known to be dynamic in nature so SJA wing lift force and pitching moment coefficients are modeled by first order system i.e. they are assumed to be function of current and previous time states (angle of attack).

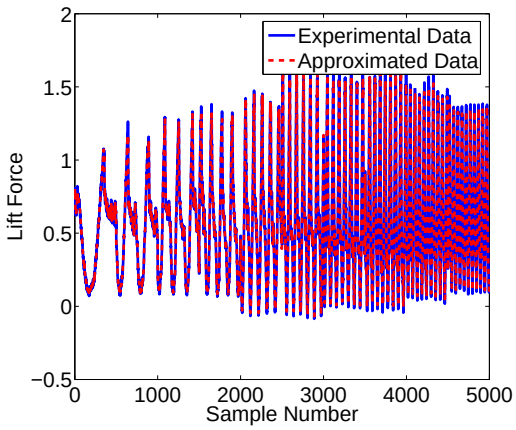
$$C_L(t_k) = C_L(\alpha_{t_k}, C_L(t_{k-1})) \tag{3.55}$$

$$C_M(t_k) = C_M(\alpha_{t_k}, C_M(t_{k-1})) \tag{3.56}$$

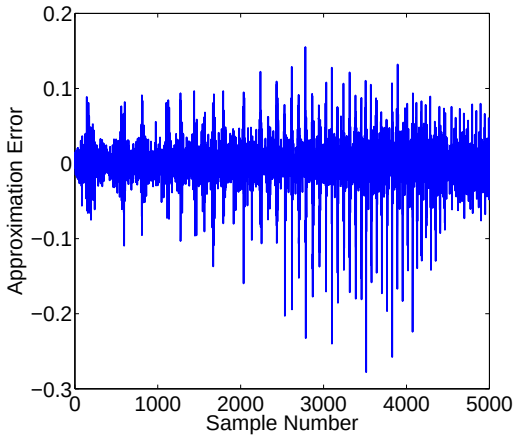
In this case, the moment and lift data is grided based upon the time interval as described in the previous section. To approximate the dynamics in a particular time



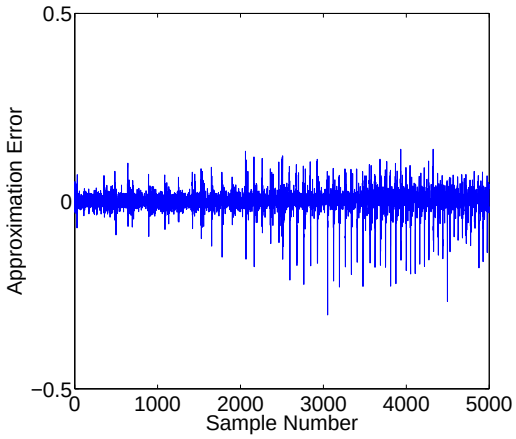
(a) 0 Hz Actuation frequency



(b) 60 Hz Approximation frequency

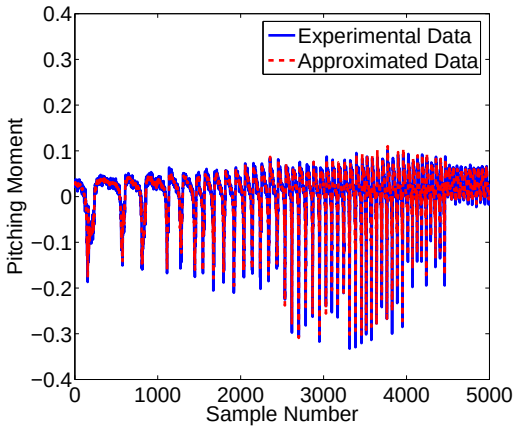


(c) Approximation error for 0 hz actuation frequency

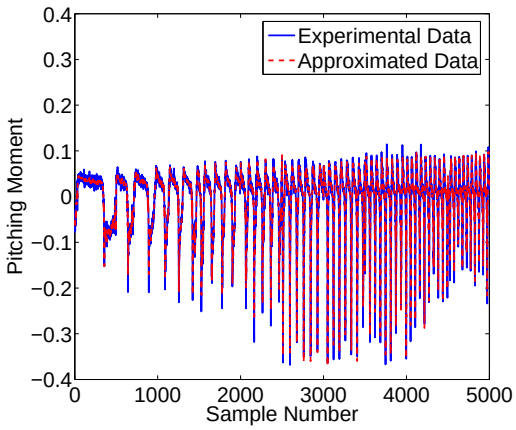


(d) Approximation error for 60 hz actuation frequency

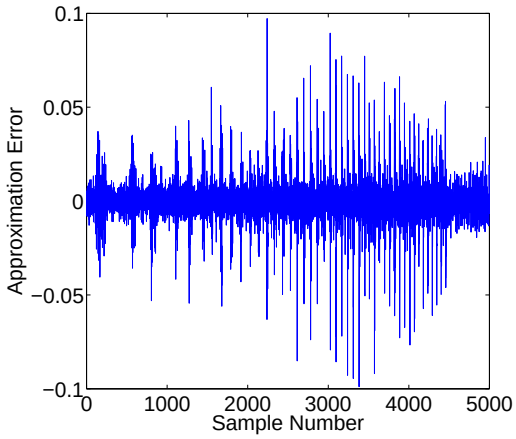
Fig. 26. Lift force approximation results.



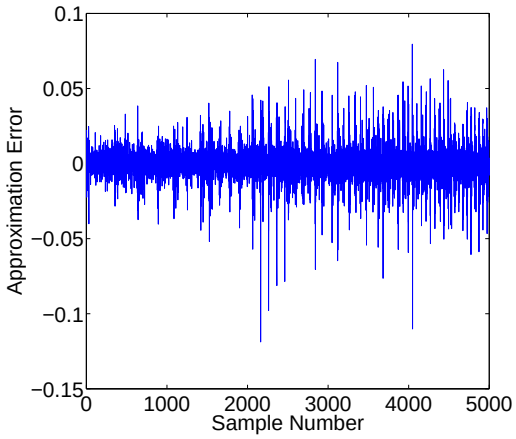
(a) 0 Hz Actuation frequency



(b) 60 Hz Approximation frequency



(c) Approximation error for 0 hz actuation frequency



(d) Approximation error for 60 hz actuation frequency

Fig. 27. Pitching moment approximation results.

interval the orthogonal functions, ϕ_{ij}, listed in table XII are used.

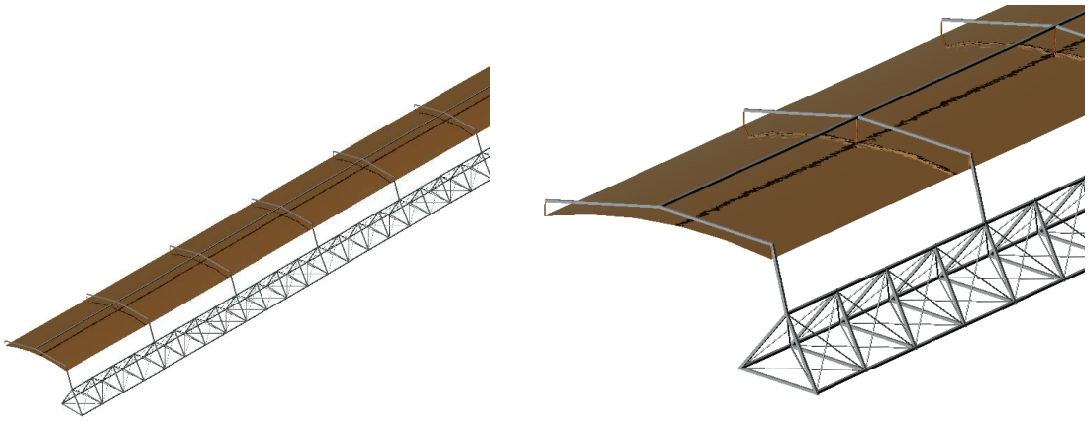
$$\begin{aligned} C_L(t_k) &= \sum_i \sum_j a_{ij}\phi_i(\alpha(t_k))\phi_j(C_L(t_{k-1})),\ i+j \leq 2 \\ C_M(t_k) &= \sum_i \sum_j a_{ij}\phi_i(\alpha(t_k))\phi_j(C_M(t_{k-1})),\ i+j \leq 2 \end{aligned} \tag{3.57}$$

Figs. 26(a) and 26(b) show the measured and approximated lift coefficient for zero and 60 Hz jet actuation frequency respectively with time interval size of 25. Figs. 26(c) and 26(d) show the corresponding approximation error plots. From these figures, it is clear that we are able to learn the nonlinear relationship between lift coefficient and angle of attack with and without SJA on.

Similarly, Figs. 27(a) and 27(b) show the measured and GLO-MAP approximated pitching moment coefficient for zero and 60 Hz jet actuation frequency respectively. Figs. 27(c) and 27(d) show the corresponding approximation error plots. From these figures, it is clear that we are able to learn the nonlinear relationship between moment coefficient and angle of attack (with and without SJA being turned on) very well within experimental accuracy.

3. Space Based Radar (SBR) Antenna Shape Approximation

Space Based Radar systems envisioned for the future may be a constellation of large spacecraft that provide persistent real-time information of ground activities through the identification and tracking of moving targets, high-resolution synthetic aperture radar imaging, and collection of high-resolution terrain information. The accuracy of the information obtain from SBR systems depend upon many parameters like the geometric shape of the antenna, permittivities of the media through which radar wave is traveling, etc. Therefore the characteristics of the scattered wave received by the SBR antenna for a given frequency depend on the surface and geometric parameters



(a) NASTRAN SBR antenna model consists of 7 panels

(b) Close-up of one panel

Fig. 28. NASTRAN model of SBR antenna.

of the radar. Therefore, to apply necessary corrections for scattering of radar waves, the precise knowledge of the instantaneous SBR antenna shape becomes a necessity. However, excitation of flexible dynamics mode by frequent pointing maneuvers makes shape estimation difficult. While a variety of surface models can be employed to model the instantaneous shape, we consider the case that the surface is measured at discrete points using a metrology sensor system and a smooth least square approximation is desired. The objective of this section is to evaluate the GLO-MAP methodology, developed in this chapter, as a candidate approach to estimate the real time SBR antenna shape.

For simulation purposes, the SBR antenna dynamics is modeled in NASTRAN [56]. The antenna model consists of total 7 panels as shown in Fig. 28. To construct the shape of antenna it is assumed that measurements of 50 points are available along a given cross section with the help of some vision sensor. Further, such 40 cross section measurements are assumed to be available along the length of the antenna at

a particular time with a sampling frequency of $10Hz$. Thus 2000 measurements are available every 0.1 seconds. Further, true measurements are corrupted by Gaussian white noise of standard deviation of $1cm$. To make the shape estimation problem more interesting, the shape of antenna is assumed to vary with respect to both in spatial coordinates and time. NASTRAN is used to generate mass, $\mathbf{M}$, and stiffness, $\mathbf{K}$, matrices for the antenna structure and coordinate transformation matrix, $\mathbf{T}$, to transform the modal coordinates to physical coordinates i.e. deflections along each axis.

$$\text{Modal Equations: } \mathbf{M}\ddot{\boldsymbol{\eta}} + \mathbf{K}\boldsymbol{\eta} = 0 \tag{3.58}$$

$$\text{Transformation to Physical Coordinates: } \mathbf{y} = \mathbf{T}\boldsymbol{\eta} \tag{3.59}$$

where, $\boldsymbol{\eta}$ and $\mathbf{y}$ represent modal and physical coordinates respectively. First, 10 modes are considered to generate the measurement data. Further, the NASTRAN generated FEM model with 6000 degrees of freedom was simulated in MATLAB [57] environment to generate the true measurement data for 20 seconds at $10Hz$ frequency.

To approximate the SBR antenna shape at a particular time, the measurement data is modeled using a total of 64 finite element cells 4 along each cartesian coordinates, X, Y and Z. Now, a continuous approximation, of SBR antenna shape, for a particular cell is generated via a least-square procedure as listed in section D.

The SBR antenna shape at time t for a particular cell is modeled by orthogonal polynomials (given in Table XII) of antenna shape at time t_0:

$$\hat{x}(t) = \sum_i \sum_j \sum_k a_{ijk}\phi_i(x(t_0))\phi_j(y(t_0))\phi_k(z(t_0)),\ i+j+k \leq 2 \tag{3.60}$$

$$\hat{y}(t) = \sum_l \sum_m \sum_n a_{lmn}\phi_l(x(t_0))\phi_m(y(t_0))\phi_n(z(t_0)),\ l+m+n \leq 2 \tag{3.61}$$

$$\tag{3.62}$$

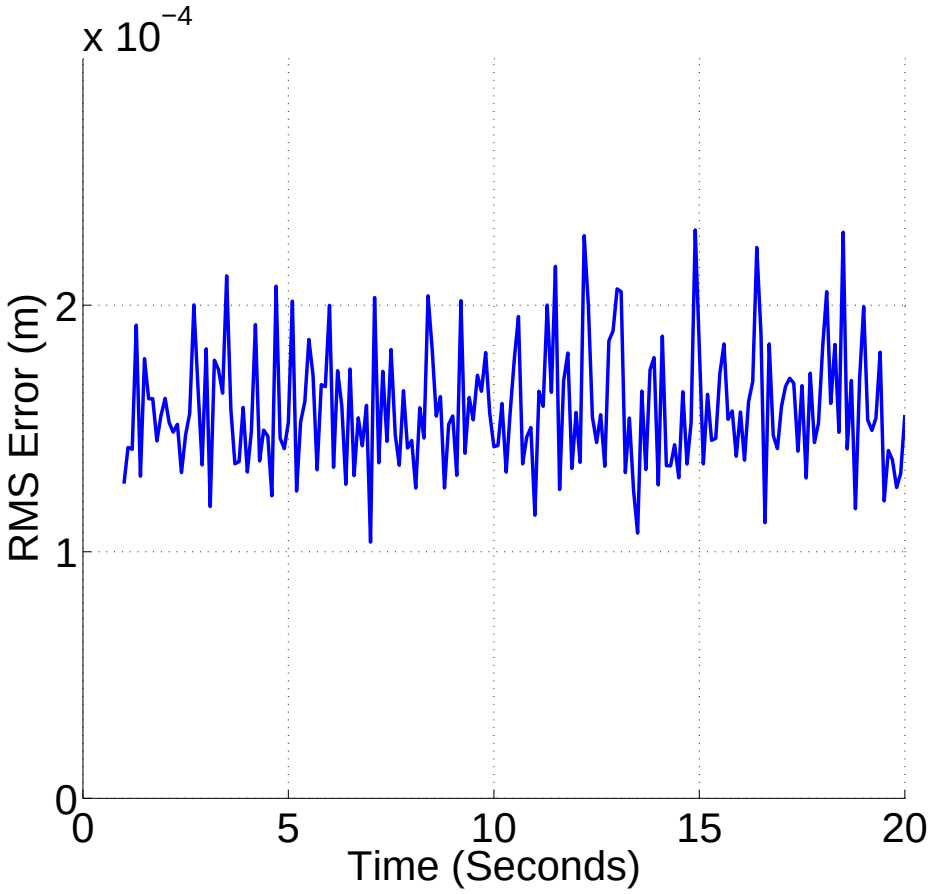
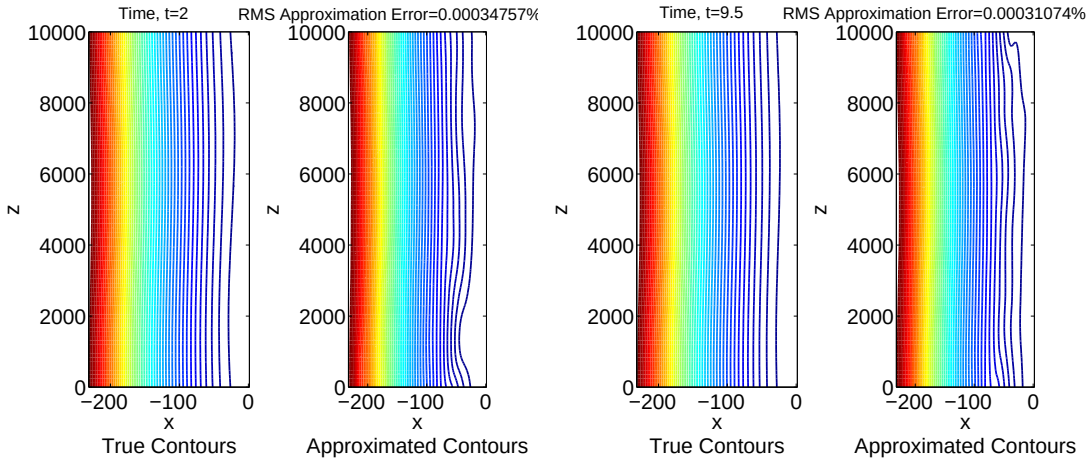


Fig. 29. RMS approximation error for SBR antenna shape approximation.



(a) Time, t=2 seconds

(b) Time, t=9.5 seconds

(c) Time, t=13 seconds

(d) Time, t=20 seconds

Fig. 30. Space based radar antenna simulation results: true and approximated contour plots.

It should be noticed that, x, y, and z denote the non-dimensional local cell coordinates defined below:

$$x = 2(X - X_m)/X_{cell} \quad y = 2(Y - Y_m)/Y_{cell} \quad z = 2(Z - Z_m)/Z_{cell} \tag{3.63}$$

where, (X_m, Y_m, Z_m) and $X_{cell} \times Y_{cell} \times Z_{cell}$ represent the centroid and dimensions of a particular cell respectively. To recursively learn the local approximations at each measurement time, vision sensor measurements are processed sequentially. Initially, all Fourier coefficients are assumed to be zero and the corresponding covariance matrix initialized to 10^6 times identity matrix.

Further, the true antenna shape is simulated by considering 80 points along each cross-section and 80 such cross-sections thus giving rise to total 6400 test points at each time instant. The first order weighting function is used to blend adjacent local approximations. Fig. 29 shows the plot of RMS approximation error at each time instant. While, Fig. 30 shows the contour plots for instantaneous antenna shape. From these figures, it is clear that mean RMS approximation error for X and Y coordinate are even less than half a percent at all time intervals. Therefore, we can conclude that we are able to learn the SBR antenna shape precisely even in presence of measurement errors. Finally, we mention that, the simulated antenna shape is just representative and may be a poor approximation of the actual flexible dynamics. In Ref. [58], we use the GLO-MAP algorithm to approximate the flexible body dynamics instead of modeling just the instantaneous shape measurements.

4. Porkchop Plots Approximation for Mission to Near Earth Objects (NEOs)

Near-Earth Objects (NEOs) are asteroids, comets and large meteoroids whose orbit intersects Earth's orbit and which may therefore pose a collision danger to Earth. In terms of orbital elements, NEOs are heavenly bodies with their perihelion distance

less than 1.3AU [1] [59]. Further, NEOs are divided into different groups depending upon many factors like their orbit size and closest approach to Earth. Out of many near Earth objects there is immense interest in visiting near Earth asteroids due to their size and proximity. There are total 701 known near Earth asteroids which are supposed to make threatening close approach (0.05AU) to Earth [59]. Therefore, information on their structural and chemical compositions is important to make intelligent choices in case of Earth threatening trajectory. In the last one decade, many space missions (NEAR, Deep Impact, STARDUST, MUSES etc.) were launched to study various near Earth Objects. For any interplanetary mission design, developing porkchop plots is the most important thing to do. As name suggested porkchop plots are porkchop-shaped contour plots that display trade space of launch and arrival dates and helps mission designer to find minimum energy transfer between two planetary objects. Actually, the porkchop plot represents a family of numerical solutions to the two point boundary value problem known as Lambert's problem [60]. The solution to Lambert's problem gives us a pair of launch and arrival dates that represents a single, unique interplanetary trajectory. So porkchop plots are generated by solving the Lambert problem for the Earth to NEO transfer over the range of launch and arrival dates. Later porkchop plots are used to identify a small region of launch and arrival dates with minimum energy requirements.

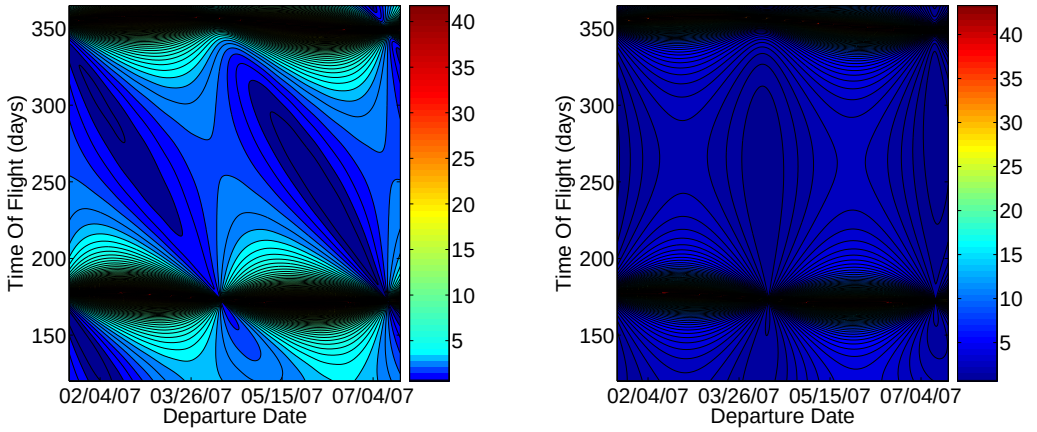
There are many algorithms listed in the literature to solve the classic Lambert's problem using the two-body model. [60, 61]. However, for a mission to near Earth object, one should obtain a solution to the generalized, perturbed Lambert's problem using general n-body model. This solution is generally based on the use of pre determined reference orbits, by using the two-body model as the first guess and defin-

[1] $1AU \approx 150$ million KM

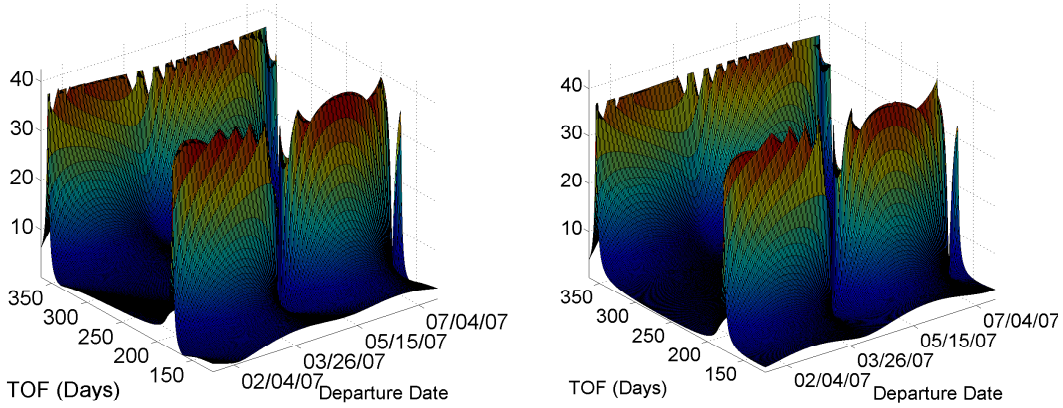
ing the effect of other bodies as perturbation to these reference orbits. Due to high sensitivity of the perturbed solution to the first guess (reference orbits obtained by using two-body model) one needs to solve and store many reference orbits for different values of position vectors of targeting bodies and time of flight. Further, to obtain perturbed solution accurately using these reference orbits as first guess, it is desired to represent these reference orbits as a function of departure date and time of flight. The objective of this section is to apply the GLO-MAP methodology to approximate the porkchop plots for a mission from Earth to asteroid 2003-$YN107$.

Asteroid 2003-$YN107$ is a quasi-moon of the Earth in a neighboring solar orbit with time period of 362.264 days and made its closest approach to Earth (0.03 AU) in June 2005 [59]. To obtain the porkchop plots for this specific mission, first we solve the Lambert's problem using two-body model. The details of the solution to Lambert's problem are beyond the scope of this dissertation and can be found in Refs. [60, 61].

The Lambert algorithm was used to iteratively solve for the initial launch velocity at a prescribed departure date to reach a target asteroid at a prescribed date. By sweeping the departure and arrival dates, the Lambert algorithm can generate a dense family of solutions. The converged departure and arrival data set can be displayed in "porkchop" surface plots to be used in mission analysis. Approximating these surfaces enables interpolation of points between the Lambert algorithm points. To approximate the porkchop plots, the measurement data (departure and arrival velocities) are generated by solving the Lambert's problem considering launch window between January 2007 and January 2008 with the interval of 1 day. Further, the Time of Flight (TOF) is varied between 120-365 days giving rise to total $365 \times 246 = 89790$ measurement points for approximation purposes. We should mention that to solve Lambert's problem, the ephemeris data for Earth and asteroid

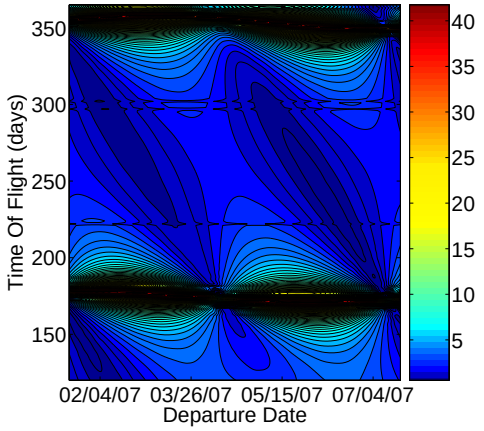


(a) True porkchop plot for departure ΔV_∞ (b) True porkchop plot for arrival ΔV_∞

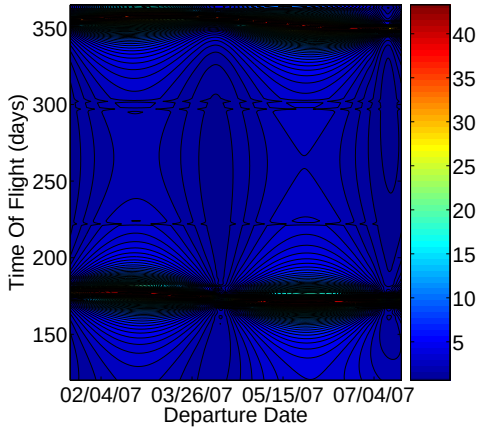


(c) True surface plot for departure ΔV_∞ (d) True surface plot for arrival ΔV_∞

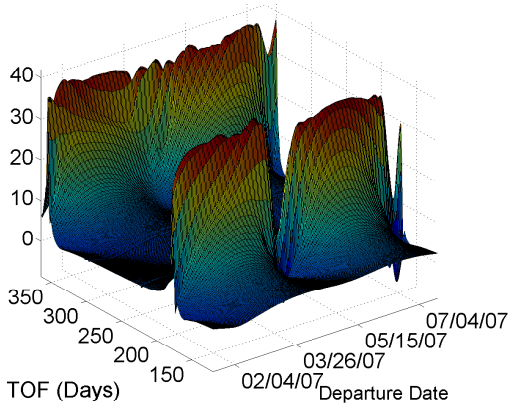
Fig. 31. True departure and arrival ΔV_∞ plots for a mission to the asteroid 2003-$YN107$.



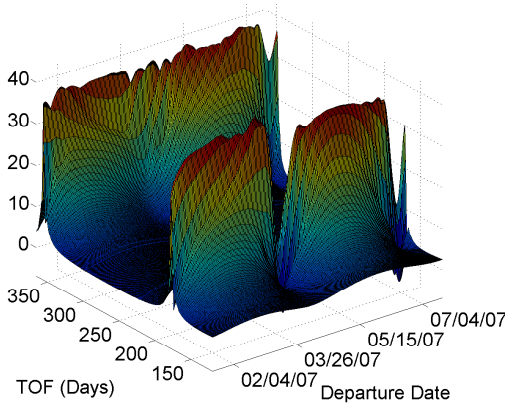
(a) Approximated porkchop plot for departure ΔV_∞



(b) Approximated porkchop plot for arrival ΔV_∞

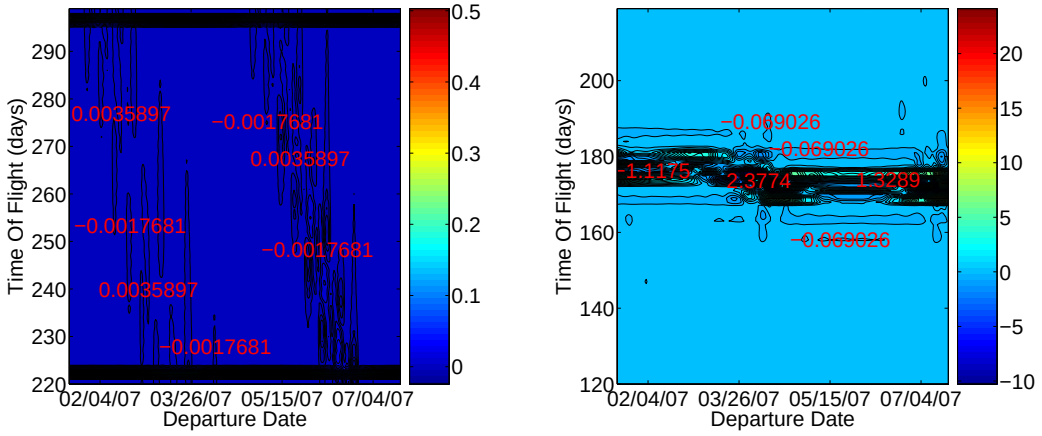


(c) Approximated surface plot for departure ΔV_∞



(d) Approximated surface plot for arrival ΔV_∞

Fig. 32. Approximated departure and arrival ΔV_∞ plots for a mission to the asteroid 2003-$YN107$.



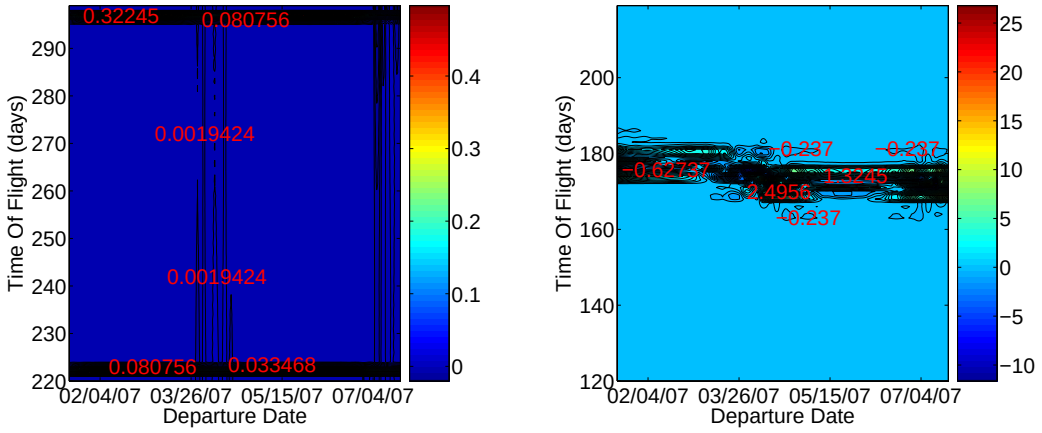
(a) Error contour plot for $220 \leq TOF \leq 300$

(b) Error contour plot for $TOF < 220$

(c) Error contour plot for $TOF > 300$

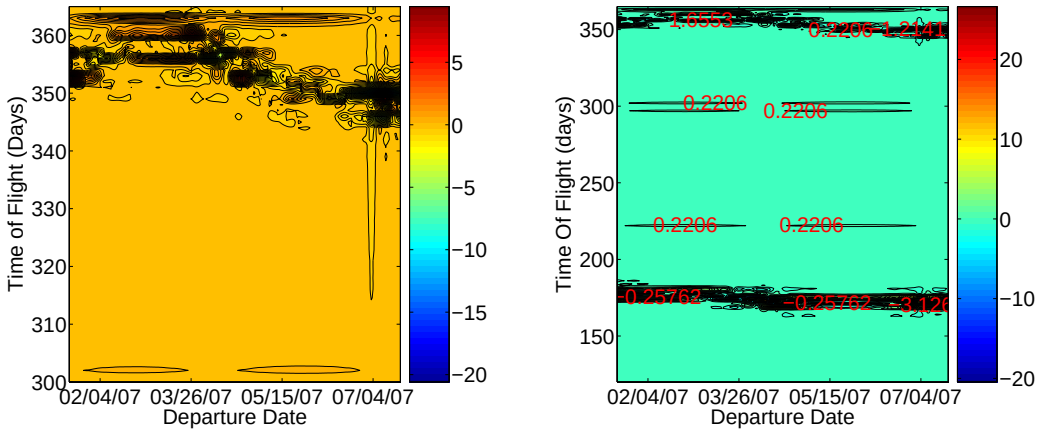
(d) Error contour plot for whole domain

Fig. 33. Departure ΔV_∞ approximation results.



(a) Error contour plot for $220 \leq TOF \leq 300$

(b) Error contour plot for $TOF < 220$



(c) Error contour plot for $TOF > 300$

(d) Error contour plot for whole domain

Fig. 34. Arrival ΔV_∞ approximation results.

2003-$YN107$ is obtained from NASA's near Earth object program site [59]. Figs. 31(a) and 31(b) show the porkchop plots for departure and arrival velocities respectively whereas Figs. 31(c) and 31(d) show the corresponding surface plots. It should be mentioned that sharp peaks in solution makes the problem of approximating these solutions extremely difficult. To obtain better approximation accuracy, we use coarse grid (25×25) in the region where departure and arrival velocity profile is comparatively smooth i.e. $220 \leq TOF \leq 300$ days and relatively finer grid (20×20) in the region where we have sharp peaks i.e. $TOF < 220$ days and $TOF > 300$ days. Further, to approximate the measurement data in each grid cell the orthogonal basis functions (listed in Table XII) up to quadratic terms in TOF and departure date are used. To compute the value of approximated departure and arrival velocity for given departure date and TOF, we use first order weighting function to blend different local approximations. Figs. 32(a) and 32(b) show the approximated porkchop plots for departure and arrival velocities respectively whereas Figs. 31(c) and 31(d) show the corresponding approximated surface plots. Further, Figs. 33 and 34 show the plots of error contours for departure and arrival velocity for different values of TOF. From these figures, we can conclude that we are able to approximate the porkchop plots for this particular mission with worst case errors less than 0.05km/sec.

Finally, we also mention that to store the GLO-MAP approximation, one just need $8.(625 + 2.400) = 11400$ real numbers as compared to $3 \times 89790 = 269370$ real number for original measurement data. So the use of the GLO-MAP algorithm reduces the storage space by a order of magnitude less in this particular case. Beside this, it should be noted that the GLO-MAP algorithm allows us to obtain the porkchop plots at any desired resolution without negotiating with approximation accuracy.

G. Concluding Remarks

We have presented a general methodology for input/output mapping in N dimensions. The method averages overlapping local preliminary approximations whose centroids of validity lie at the vertices of a user specified, generally non-uniform, N dimensional grid. The averaging makes use of a special class of weight functions that guarantee a prescribed degree of piecewise continuity and osculation with the preliminary approximations at their centroids of validity. The preliminary approximations can be chosen arbitrarily to take advantage of prior knowledge of a particular problem. Alternatively, the preliminary approximations can be chosen as linear combinations of any complete set of linearly independent basis functions. A particularly attractive choice is shown to be polynomial basis functions that are orthogonal with respect to the weight functions of the averaging process. We constructed these new orthogonal polynomials using a Gramm-Schmidt process. The result is an new method for orthogonal function local approximation with an associated averaging process giving a global piecewise continuous approximation. Further, the new approach is tested on several examples from a variety of disciplines such as continuous function approximation, dynamic system modeling and system identification. The results are of direct utility in addressing the "curse of dimensionality" and frequent redundancy of neural network approximation. The broad generality of the method, together with a number of examples presented provides a strong basis for optimism for the importance and practical utility of these ideas.

CHAPTER IV

MULTI-RESOLUTION ALGORITHM

A. Introduction

A key question regarding the proper selection of an approximation algorithm is "How irregular is the input-output map?" A global best fit of the input-output map should be sufficient if the slope of the input-output map is smooth globally without large local variations in the space-time frequency context. In the presence of irregular localized features, a multi-resolution based learning algorithm may be required to take care of local and as well as global complexity of the input-output map. In the previous chapter, we have advocated the use of the GLO-MAP algorithm for input-output data approximation and have claimed (without any proof) that the GLO-MAP algorithm is a multi-resolution approximation algorithm. However, we still need to clarify "What do we mean by a multi-resolution algorithm?" Further, in the preceding chapter, we advocated the use of orthogonal polynomial basis functions to obtain preliminary local approximations in the GLO-MAP algorithm. However, the issues regarding the approximation capabilities of orthogonal polynomials and the conditions under which the whole GLO-MAP process converges, need to be addressed more formally.

In this chapter, our main aim is to address these issues rather broadly and prove that the GLO-MAP algorithm qualifies as a multi-resolution algorithm. In addition, some results are presented that provide insight on the approximation ability and other probabilistic properties of the GLO-MAP approximation. Finally, an adaptive, hybrid multi-resolution approximation algorithm is presented based on the RBFN and the GLO-MAP learning algorithms. The particular approximation algorithm uses the RBFN for global approximation and the GLO-MAP algorithm to locally

refine the global models obtained by RBFN while maintaining global continuity and computational efficiency.

The structure of this chapter is as follows: in the next section, a formal definition of a multi-resolution approximation is given followed by the discussion on the multi-resolution attributes of the GLO-MAP approximation. Next, an error analysis for the GLO-MAP algorithm is presented and finally, an adaptive, multi-resolution approximation algorithm comprising of the RBFN and the GLO-MAP algorithm is discussed for general input-output mapping.

B. Multi-Resolution Learning Algorithm

As name suggests, multi-resolution approximation can be defined as a mathematical process of hierarchically decomposing the input-output approximation to capture both macroscopic and microscopic features of the system behavior.

The unknown function underlying any given input-output data can be considered as consisting of high frequency local input/output variation details superimposed on the comparatively low frequency smooth background. More than two levels of granularity (resolution) will be required in a general setting. The term "resolution" can be defined as the scale to measure the details of the input-output data that can not be discerned. At a given resolution, the input-output data is approximated by ignoring all variations below that scale. As name suggests, the term multi-resolution refers to the simultaneous presence of different resolutions [62]. Therefore, multi-resolution approximation can be defined as a mathematical process of hierarchically decomposing the input-output approximation. At each stage, finer details are added to the coarser description, providing a successively better approximation to the input-output data. Eventually when the resolution goes to infinity, we would expect to

approach the exact smooth function underlying any given input-output data. The term "multi-resolution" enjoys wide popular use in the wavelet analysis as wavelets allow a function to be described in terms of a coarse overall shape, plus details that range from broad to narrow [63]. Similarly, we can view the space of functions that are square integrable as composed of a sequence of subspaces W_k and V_j , such that the approximation at resolution level j is in V_j and the higher frequency details are in W_k. This brings us to the following formal definition of the multi-resolution approximation:

Multi-Resolution Approximation. *A sequence* $\{V_j\}_{j\in\mathbb{Z}}$ *of closed subspaces is a multi-resolution approximation if the following* 6 *properties are satisfied:*

1. $\forall j \in \mathbb{Z},\ V_j \subset V_{j+1}$
2. $\lim_{j\to-\infty} V_j = \cap_{-\infty}^{\infty} V_j = \{0\}$
3. $\lim_{j\to\infty} V_j = \overline{(\cup_{-\infty}^{\infty} V_j)} = \mathcal{L}^2(\mathbb{R})$
4. $\forall j \in \mathbb{Z},\ f(t) \in V_j \Leftrightarrow f(2t) \in V_{j+1}$
5. $\forall\ (k) \in \mathbb{Z},\ f(t) \in V_0 \Leftrightarrow f(t-k) \in V_0$
6. *There exists a function* $\theta(t)$*, called the scaling function, such that* $\{\theta(t-k)\}$ *is an orthonormal basis of* V_0*.*

where, $\mathbb{Z}$ *denotes an index set for resolution index* j*,* $\cap_{-\infty}^{\infty} V_j$ *denotes the intersection of all possible subspaces* V_j *and* $\overline{(\cup_{-\infty}^{\infty} V_j)}$ *represents the closure of the union of all possible subspaces* V_j*.*

According to the above definition of Multi-Resolution Approximation (MRA), the starting point for the MRA analysis is the decomposition of function space, V_0 into a sequence of subspaces, V_j. Now, the first condition implies that the subspace

V_j be contained in all the higher subspaces. Literally, it means that information contained at level j must be included in the information at a higher resolution which is a reasonable requirement. The second condition corresponds to the fact that as resolution gets coarser and coarser, the approximation becomes more crude, and in the limit $j \to -\infty$, we should get a constant function which can only be a zero function due to square integrable constraint. The third condition is the opposite of the second condition and states that as the resolution is increased, more details are included in the approximation and in the limit $j \to \infty$, we should get back the entire space $\mathcal{L}^2(\mathbb{R})$. The fourth condition is equivalent to scale or dilation invariance of space V_j while fifth condition corresponds to translation and dilation invariance. The sixth and final condition guarantees the existence of a orthonormal basis for V_j. Note, if $\theta(t-k)$ form an orthonormal basis for V_0 then by scale and translation invariance $\theta_{jk} = 2^{\frac{j}{2}}\theta(2^j t - k)$ forms an orthonormal basis for V_j [62, 63].

Based upon the definition of the *Multi-Resolution Approximation*, it is easy to conclude that the GLO-MAP orthogonal polynomial forms a multi-resolution analysis. To prove it formally, we state the following proposition:

Proposition 1. *Let V_0 be the space consist of all functions $\phi_n(t)$ that are orthogonal polynomials used in the GLO-MAP approximation of degree at most n valid over interval $[k-1, k+1]$ with $n-1$ continuous derivatives for $n > 0$ i.e. $\phi_n(t) \in \mathcal{C}^n$. Further, assume that V_j is the space of orthogonal polynomials of the GLO-MAP approximation over the interval $\left[\frac{k-1}{2^j}, \frac{k+1}{2^j}\right]$. Now, our claim is that this collection of spaces forms a multi-resolution approximation of the square integrable function space, $\mathcal{L}^2(\mathbb{R})$.*

Proof. From the definition of V_j, it is easy to see that as j increases V_j grows and in the limit $j \to \infty$, $V_j \to \mathcal{L}^2(\mathbb{R})$. Similarly, as j decreases subspace V_j shrinks in

size and in the limit $j \to -\infty$, V_j shrinks to zero. Also, $V_0 \subset V_1$ because if $\phi_n(t)$ is a polynomial of degree n over the interval $[k-1, k+1]$ then $\phi_n(2t)$ is clearly a polynomial of degree n over the interval $[\frac{k-1}{2}, \frac{k+1}{2}]$. Similarly $V_j \subset V_{j+1}$, $\forall j$. Further, condition 5 follows from the fact that dilation of a polynomial of degree n is also a polynomial of degree n. Note, if we choose $\theta_n(t) = \phi_n(t)$ then there is no doubt that that translates of $\theta_n(t)$ forms a basis for V_0 which may not be orthogonal. However, following the procedure listed in Refs. [64, 65], one can also find the orthogonal basis $\theta_n(t)$ □

From the Proposition 1, we can conclude that the GLO-MAP approximation algorithm is indeed a multi-resolution approximation algorithm. We mention that the subspaces V_j form a nested sequence that provides successively better approximations to $\mathcal{L}^2(\mathbb{R})$. Further, the scaling function, $\theta(t)$ generates all the orthogonal basis functions for each space $V_j(t)$. Note that the Fourier transform of $\theta(t)$ leads to the corresponding orthogonal basis functions in the frequency domain.

The multi-resolution analysis is an important property to have for any approximation algorithm. Whether one is compressing satellite images, trying to solve PDE's, modeling an irregular function, there is broad interest in multi-resolution analysis. The Finite Element Methods (FEM), Spline approximations and Wavelets are most commonly used multi-resolution algorithms. Although, the general strategy of all the multi-resolution algorithms is similar but due to some additional properties specific to each algorithm, some algorithms are more suited for some specific applications. For example, the wavelets are more suited for image processing and the FEM are generally accepted as being more effective for solving PDE's. These approximation methodologies are shown to be compatible with a wide variety of disciplines such as continuous function approximation, dynamic system modeling, time series prediction,

and, image processing. The multi-resolution properties of these algorithm have led to broadly useful approximation approaches that have good local approximation properties for any given input-output data. However, it is not possible to form conformity, i.e., inter-element continuity without having independent local approximations. The GLO-MAP algorithm, however, offers rigorous means to construct piecewise global approximations out of any given system of local approximation without sacrificing the approximation properties. In fact, as is shown subsequently in this chapter, the final GLO-MAP approximations are unbiased approximations that are generally superior (smaller variance) to the generating local approximations. In addition, the GLO-MAP algorithm offers an easy way to include any a priori and analytical information available about the unknown input-output mapping. Another prominent issue with various multi-resolution algorithms is their generalization to high dimensional input space. For example, the FEM and the wavelet algorithms are applicable only to moderate dimensional problems (generally, 2 or 3 dimension). In Chapter III, the GLO-MAP algorithm is extended rigorously for approximation with arbitrary order continuity in a general N dimensional space. In summary, the freedom to vary in a general way the resolution (e.g., degrees of freedom) of the local approximations and the generalizations to an N dimensional space are the unique characteristics of the GLO-MAP algorithm which distinguish it from other multi-resolution algorithms. These truths are established in the present chapter and are consistent with the numerical studies throughout this dissertation.

C. Choice of Basis Function and Approximation Error

The central difficulty in learning any given input-output data lies in choosing appropriate basis functions. There are infinitely many choices for the basis functions

such as polynomials, trigonometric functions, radial basis functions etc. Against the backdrop of these choices, Stone-Weierstrass theorem gives one of the most remarkable results in the field of approximation theory, stating that there exists a sequence of polynomials that converge uniformly to any prescribed continuous function on a compact interval. This theorem was first stated by Weierstrass for polynomial approximations in 1-D spaces [49] and was, later modified by Stone to generalize it for polynomial approximation in compact 2-D spaces [50–52]. For a general compact space, this theorem can be generalized to N-dimensions as follows [66]:

Stone-Weierstrass Approximation Theorem. *Let X be a compact Hausdorff space and $C[X]$ be a space of continuous functions on X. Then the set of polynomials in N variables form a dense set in $C[X]$.*

As a consequence of this theorem, we can approximate any continuous function on a compact interval with polynomial series having a sufficient number of terms. We mention that this theorem is the main theoretical justification behind using polynomial basis functions for preliminary approximations over compact interval defined by the support of specially designed weight functions in the GLO-MAP algorithm. Besides many advantages (like numerical conditioning and computational cost, as discussed in Chapter III) of using orthogonal polynomials for preliminary approximations, one very important property of the approximation of a continuous function f on a compact interval $[a, b]$ by orthogonal polynomials is that approximation errors vanishes in at least $n+1$ points of (a, b), where, n is the degree of the approximation. We formally state and prove this property as follows:

Theorem 2. *Let $\{\phi_i\}$ be a set of orthogonal polynomials with respect to weight function, w, over compact interval $[a, b]$. Where, subscript i denotes the degree of the polynomial. Let $\hat{f}$ denote the least square approximation of a continuous function f*

using orthogonal polynomials ϕ_i:

$$\hat{f} = \sum_{i=0}^{n} a_i \phi_i \tag{4.1}$$

Then $f - \hat{f}$ changes sign or vanish identically at least $n+1$ times in open interval (a, b).

Proof. The proof of this theorem follows from the most important characteristic of the least square approximation according to which residual error, $e = f - \hat{f}$, of the least square solution is orthogonal to the range space spanned by basis functions $\{\phi_i\}$. Now, to prove that e must change sign at least $n+1$ times in (a, b), we first show that e must change sign at least once and then we prove the rest by contradiction arguments.

Note, as e is orthogonal to $\phi_0 = 1$, therefore, $\langle e, 1\rangle = 0$. Thus, if $e \neq 0$, then it is obvious that e must change sign at least once in (a, b). Now, assume that e changes sign fewer than $n+1$ times and $x_1 < x_2 < \cdots < x_m$ be the points where, e changes sign. In each interval (a, x_1), (x_1, x_2), $\cdots$, (x_k, b), e does not changes sign but has opposite signs in the neighboring intervals. As a consequence of this, we can define a polynomial function, $P(x) = \prod_{i=1}^{m}(x - x_i)$, of degree m with following condition:

$$\langle e(x), P(x)\rangle \neq 0 \tag{4.2}$$

However, $P(x)$ being a polynomial of degree $m < n$ can be written as a linear combination of $\phi_0, \phi_1, \cdots, \phi_n$ and is therefore orthogonal to e i.e. $\langle e(x), P(x)\rangle = 0$. This is a contradiction of Eq. (4.2) and therefore, e must change sign at least $n+1$ times in the interval (a, b) □

As a consequence of this theorem, if we use orthogonal polynomials for preliminary approximations of the GLO-MAP algorithm, then they must interpolate a

continuous functions exactly ($e = 0$) at $n+1$ points in the local domain. Note, the key point of the proof of Theorem 2 lies in the fact that residual error e is orthogonal to the range space spanned by basis functions $\{\phi_i\}$. Now, we state the following Lemma according to which the residual error of the GLO-MAP process, even after carrying out the averaging process, is orthogonal to the range space spanned by orthogonal basis functions $\phi_i(x)$.

Lemma 4. *Let $\{\phi_i\}$ be a set of orthogonal polynomials used in the GLO-MAP algorithm with respect to the weight function, w, over interval $[-1, 1]$. Where, subscript i denotes the degree of the polynomial. Let $\hat{f}$ denote the GLO-MAP approximation of a continuous function f over the interval $[a, b]$*

$$\hat{f} = \sum_{i=1}^{m} w_i \hat{f}_i \tag{4.3}$$

where, m is the total number of local approximations, $\hat{f}_i$ is the local least square approximation in the i^{th} interval and w_i is the specially designed GLO-MAP weight function associated with the i^{th} interval. Now, if we define residual error $e = f - \hat{f}$ then e is orthogonal to the range space spanned by the basis functions ϕ_i under the norm induced by $\sum_{i=1}^{m} w_i(x) = 1$.

Proof. Note, as each local approximation $\hat{f}_i$ denotes the least square approximation over i^{th} interval, therefore, the following holds:

$$\int_{x_{il}}^{x_{iu}} (f(x) - \hat{f}_i(x, x_i)) w_i(x, x_i) \phi_j(x) dx = \int_{a}^{b} (f(x) - \hat{f}_i(x, x_i)) w_i(x, x_i) \phi_j(x) dx = 0,$$
$$j = 1, 2, \cdots, n \tag{4.4}$$

where, x_{il} and x_{iu} denote the lower and upper limits of the i^{th} local interval. Now,

let us consider the residual error over the whole interval $[a, b]$

$$\int_a^b (f(x) - \hat{f}(x))\phi_j(x)dx = \int_a^b f(x)\phi_j(x)dx - \sum_{i=1}^{m} \int_a^b w_i(x, x_i)\hat{f}_i(x, x_i)\phi_j(x)dx, \quad j = 1, 2, \cdots, n \quad (4.5)$$

Further, from Eqs. (4.4) and (4.5), we have:

$$\int_a^b (f(x) - \hat{f})\phi_j dx = \int_a^b f(x)\phi_j(x)dx - \sum_{i=1}^{m} \int_a^b w_i(x, x_i)f(x)\phi_j(x)dx, \quad j = 1, 2, \cdots, n \quad (4.6)$$

Now, as the GLO-MAP weight functions satisfy the *partition of unity* paradigm, therefore, Eq. 4.6 reduces to

$$\int_a^b (f(x) - \hat{f})\phi_j dx = \int_a^b f(x)\phi_j(x)dx - \int_a^b f(x)\phi_j(x)dx = 0, \ j = 1, 2, \cdots, n \quad (4.7)$$

Hence, the GLO-MAP residual error e is orthogonal to range space spanned by the basis functions ϕ_i. □

Although, we have shown that the GLO-MAP residual error e is orthogonal to the range space spanned by the basis functions ϕ_i but Theorem 2 does not hold for the final GLO-MAP approximation. This is due to the fact that the orthogonality condition of the basis functions and orthogonality of the residual error, e to the range space are not induced by same norm. Also, obviously, the final GLO-MAP approximation polynomials are of higher degree than the generating polynomials, since the weight functions themselves are of degree $m + 2$ where m is the degree of piecewise continuity desired.

We have discussed earlier that according to the Stone-Weierstrass theorem, we can approximate any continuous functions over a compact interval with infinite poly-

nomial series. However, it is intractable in a practical applications to approximate a function with an infinite term polynomial series because such a problem will have far too many parameters to determine from limited number of observations. According to Theorem 1 in Chapter III, the Fourier coefficients converge to zeros as the number of orthogonal polynomial functions approaches infinity. In other words, one needs only a countable number of orthogonal basis functions to approximate a bounded continuous function to a prescribed resolution. Practically, we can use a finite series polynomial to locally approximate any given continuous function, as discussed in Chapter III. To account for the errors introduced due to the truncation of infinite series polynomial, we state following theorem which basically gives us a bound for approximation error using any (up to) degree n polynomial basis functions.

Theorem 3. *Let f be an $n+1$ times differentiable function over the compact interval $[a, b]$ i.e. $f \in \mathcal{C}^{n+1}[a, b]$ and $\hat{f}$ denotes the approximation of the unknown function f using a complete set of polynomial basis functions up to degree n. Further, let $x_i,\ i = 0, 1, 2, \cdots, n$ are $n + 1$ interpolation points in the compact interval $[a, b]$ then we have the following approximation error equation:*

$$e \triangleq f(x) - \hat{f}(x) = (x - x_0)(x - x_1) \cdots (x - x_n) \frac{f^{n+1}(\xi(x))}{(n + 1)!} \tag{4.8}$$

where, $\xi(x) = \xi(x_0, x_1, \cdots, x_n)$

Proof. First, note that if $x = x_i,\ i = 0, 1, 2, \cdots, n$ then the error expression of Eq. (4.8) is trivial. Therefore, we assume that $x \neq x_i$ and define following function

$$F(t) \triangleq f(t) - \hat{f}(t) - \frac{f(x) - \hat{f}(x)}{(x - x_0)(x - x_1) \cdots (x - x_n)} (t - x_0)(t - x_1) \cdots (t - x_n) \tag{4.9}$$

Now, it is apparent that F has $n + 2$ zeros, namely, $x_0, x_1, \cdots, x_n, x$. Now, according to Rolle's theorem [66] F^{n+1} has at least one zero, $\xi = \xi(x, x_0, x_1, \cdots, x_n)$. This

implies that

$$F^{n+1}(\xi(x)) = f^{n+1}(\xi(x)) - \hat{f}^{n+1}(\xi(x)) - \frac{f(x) - \hat{f}(x)}{(x-x_0)(x-x_1)\cdots(x-x_n)}(n+1)! = 0$$

and hence we prove that $e(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{n+1}(\xi(x))$ □

Note, in the case of the interpolation problem, the definition of $x_i,\ i = 1, 2, \cdots, n$ is straightforward and in the case of the Least-Square approximation, Theorem 2 guarantees the existence of these points, if one uses orthogonal basis functions. Further, if $f^{n+1}(.)$ is bounded by a number M, then Eq. (4.8) can be replaced by the following inequality:

$$e(x) \leq \frac{(b-a)^{n+1}}{(n+1)!} M \tag{4.10}$$

From the above Eq. (4.10), it is clear that when $n \to \infty$, $e(x) \to 0$.

Further, we state following theorem to have a measure of net approximation error after merging different local approximations using the GLO-MAP algorithm.

Theorem 4. *Let f be a continuous function over N-dimensional space $\Omega \subset \mathcal{R}^N$. Let $\{w_i\}$ be a set of specially designed GLO-MAP weight functions with compact support Ω_i satisfying*

1. *$\{\Omega_i\}$ is an open cover for Ω.*
2. *$\sum_i w_i = 1$ on Ω.*
3. *$\|w_i\|_\infty \leq 1$.*
4. *$\|\nabla w_i\|_\infty \leq \frac{1}{h_i}$.*

where, h_i denotes the size of the i^{th} sub-domain Ω_i. Assume that $\hat{f}_i$ denotes the

approximation of f on sub-domain Ω_i such that

$$\|f - \hat{f}_i\|_{\mathcal{L}_2(\Omega_i)} \leq e_{1_i} \tag{4.11}$$

$$\|\nabla(f - \hat{f}_i)\|_{\mathcal{L}_2(\Omega_i)} \leq e_{2_i} \tag{4.12}$$

Then the GLO-MAP approximation $\hat{f} = \sum_i w_i \hat{f}_i$ satisfies following error bounds:

$$\|f - \hat{f}\|_{\mathcal{L}_2(\Omega)} \leq 2^{\frac{N}{2}} \left(\sum_i e_{1_i}^2 \right)^{\frac{1}{2}} \tag{4.13}$$

$$\|\nabla(f - \hat{f})\|_{\mathcal{L}_2(\Omega)} \leq 2^{\frac{N+1}{2}} \left(\sum_i \frac{e_{1_i}^2}{h_i^2} + \sum_i e_{2_i}^2 \right)^{\frac{1}{2}} \tag{4.14}$$

Proof. Since the weight functions w_i form a partition of unity over Ω, we have

$$f = 1.f = \sum_i w_i f \tag{4.15}$$

Substituting for f from Eq. (4.15) in Eq. (4.13), we get

$$\|f - \hat{f}\|^2_{\mathcal{L}_2(\Omega)} = \| \sum_i w_i (f - \hat{f}_i) \|^2_{\mathcal{L}_2(\Omega)} \tag{4.16}$$

Since, at any point $\mathbf{x} \in \Omega$ only 2^N local approximations overlap, the summation terms in Eq. (4.16) also contain at most 2^N terms for any $x \in \Omega$. Thus, we have:

$$\|f - \hat{f}\|^2_{\mathcal{L}_2(\Omega)} \leq 2^N \sum_i \|w_i (f - \hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)} \tag{4.17}$$

Further, making use of the fact that the support of weight function w_i is Ω_i, we have

$$\begin{aligned} \|f - \hat{f}\|^2_{\mathcal{L}_2(\Omega)} &\leq 2^N \sum_i \|w_i (f - \hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)} \\ &\leq 2^N \sum_i \|w_i (f - \hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)_i} \\ &\leq 2^N \sum_i 1.e_{1_i}^2 = 2^N \sum_i e_{1_i}^2 \end{aligned} \tag{4.18}$$

This proves the estimates of Eq. (4.13). To show the estimates of Eq. (4.14), let us consider $\|\nabla(f-\hat{f})\|^2_{\mathcal{L}_2(\Omega)}$

$$\begin{aligned}
\|\nabla(f-\hat{f})\|^2_{\mathcal{L}_2(\Omega)} &= \|\nabla(f-\hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)} \\
&\leq \|\nabla\sum_i w_i(f-\hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)} \\
&\leq 2\|\sum_i \nabla w_i(f-\hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)} + 2\|\sum_i w_i\nabla(f-\hat{f}_i)\|^2_{\mathcal{L}_2(\Omega)} \\
&\leq 2^{N+1}\sum_i \|\nabla w_i(f-\hat{f}_i)\|^2_{\mathcal{L}_2(\Omega_i)} + 2^{N+1}\sum_i \|w_i\nabla(f-\hat{f}_i)\|^2_{\mathcal{L}_2(\Omega_i)} \\
&\leq 2^{N+1}\sum_i \left(\frac{1}{h_i^2}e_{1_i}^2 + e_{2_i}^2\right)
\end{aligned}$$

which proves the theorem. □

We mention that the local approximation error bounds of Eqs. (4.11) and (4.12) are given by the Theorem 3. Although Theorem 4 quantifies the approximation error of the GLO-MAP algorithm, it does not provide any information about the effect of the measurement error on the net approximation error. To quantify the effect of measurement error on the net approximation error, an alternative, probabilistic approach is presented in the next section.

1. Probabilistic Analysis of the GLO-MAP Algorithm

In estimating unknown parameters from a statistical model, one is interested in how the estimates deviate from the true value of the parameter. The deviations generally come from two sources.

1. Random Error: The source of this error is the random noise present in measurement data.

2. Bias or Systematic Error: Bias is the difference between the average value of the estimates from the true value.

The difference between the estimation algorithm bias error and the random error is that the estimate bias can typically be reduced by increasing the measurement data size while the random error can not be reduced arbitrarily. In this section, we quantify these errors for the GLO-MAP algorithm using a statistical approach and discuss some other statistical properties of the GLO-MAP algorithm.

To quantify the effect of discretization and measurement error, let us consider a collection of m data points $(\mathbf{x}_i, \tilde{y}_i)$ which are to be approximated by polynomial basis functions, $\phi_i(\mathbf{x}),\ i = 1, 2, \cdots, n$.

$$\tilde{\mathbf{y}} = \mathbf{H}\mathbf{a} + \boldsymbol{\nu} \tag{4.19}$$

where, $\tilde{\mathbf{y}}$ is a $m \times 1$ vector of measurement points $\tilde{y}_i$, $\mathbf{H}$ is a $m \times n$ matrix with $H_{ij} = \phi_j(\mathbf{x}_i)$ and $\mathbf{a}$ is a $n \times 1$ vector of Fourier coefficients. Further, $\boldsymbol{\nu}$ represents the measurement error modeled by the zero mean Gaussian white noise process with error covariance matrix, $\mathbf{R}$.

To find the least square estimates of the Fourier coefficient vector $\hat{\mathbf{a}}$, we define following loss function:

$$J = \frac{1}{2}(\tilde{\mathbf{y}} - \mathbf{H}\mathbf{a})^T \mathbf{W}\mathbf{R}^{-1}(\tilde{\mathbf{y}} - \mathbf{H}\mathbf{a}) \tag{4.20}$$

where, $\mathbf{W}$ is a $m \times m$ positive-definite weight matrix. Now, carrying out the least square procedure as listed in Ref. [29], we get the following equation for $\hat{\mathbf{a}}$

$$\hat{\mathbf{a}} = \left(\mathbf{H}^T \mathbf{W}\mathbf{R}^{-1}\mathbf{H}\right)^{-1} \mathbf{H}^T \mathbf{W}\mathbf{R}^{-1}\tilde{\mathbf{y}} \tag{4.21}$$

If basis functions ϕ_i are assumed to be orthogonal relative to the weight function $w(\mathbf{x})$ over the range of $\mathbf{x}$, then following equation holds:

$$\int_V \phi_i(\mathbf{x})\phi_j(\mathbf{x})w(\mathbf{x})dV = k_{ij}\delta_{ij} \tag{4.22}$$

We mention that in case of the GLO-MAP preliminary approximations, V denotes the volume of N-D hypercube. If we weight measurement points in accordance with the weight function $w(\mathbf{x}_i)$, then the weight matrix $\mathbf{W}$ is given by

$$\mathbf{W} = diag(w(\mathbf{x}_1), w(\mathbf{x}_2), \cdots, w(\mathbf{x}_m)) \tag{4.23}$$

Further, if measurement error covariance matrix is also assumed to be a scalar times identity matrix $\mathbf{R} = \sigma^2 \mathbf{I}_{m\times m}$ then,

$$\hat{\mathbf{W}} \triangleq \mathbf{W}\mathbf{R}^{-1} = diag\left(\frac{w(\mathbf{x}_1)}{\sigma^2}, \frac{w(\mathbf{x}_2)}{\sigma^2}, \cdots, \frac{w(\mathbf{x}_m)}{\sigma^2}\right) \tag{4.24}$$

Further, if measurement data points are independent and randomly selected throughout the range of $\mathbf{x}$, then as the number of data points increases, $\mathbf{A} \triangleq \left(\mathbf{H}^T\mathbf{W}\mathbf{R}^{-1}\mathbf{H}\right)$ approaches a diagonal matrix due to the orthogonality condition of Eq. (4.22)

$$\mathbf{A}_{kl} = \lim_{m\to\infty}\left(\sum_{i=1}^{m}\frac{w(\mathbf{x}_i)}{\sigma^2}\phi_k(\mathbf{x}_i)\phi_l(\mathbf{x}_i)\right) = \int_V \frac{w(\mathbf{x})}{\sigma^2}\phi_k(\mathbf{x})\phi_l(\mathbf{x})dV \tag{4.25}$$

$$= \frac{k_{kl}}{\sigma^2}\delta_{kl} \approx \frac{1}{\sigma^2}\sum_{i=1}^{m} w(\mathbf{x}_i)\phi_k(\mathbf{x}_i)\phi_l(\mathbf{x}_i)\delta_{kl} \tag{4.26}$$

If orthogonality of the basis functions is maintained then the diagonal structure of the matrix $\mathbf{A}$ results in following uncoupled equations for Fourier coefficients:

$$a_j = \frac{1}{k_{jj}}\sum_{i=1}^{m} w(\mathbf{x}_i)\phi_j(\mathbf{x}_i)\tilde{y}_i \tag{4.27}$$

Note, if one uses Eq. (4.21) to compute the Fourier coefficient vector then total $O(n^3m)$ floating point operations are required while only $O(m)$ floating point operations are required to compute the Fourier coefficients vector using Eq. (4.27). However, in case of discrete data the orthogonality condition of Eq. (4.22) is merely true and in that case the Fourier coefficient vector can be evaluated by assuming

matrix $\mathbf{A}$ to be a diagonal matrix:

$$a_j = \frac{1}{\sum_{i=1}^{m} w(\mathbf{x}_i)\phi_j^2(\mathbf{x}_i)} \sum_{i=1}^{m} w(\mathbf{x}_i)\phi_j(\mathbf{x}_i)\tilde{y}_i \tag{4.28}$$

Note, even in this case, one needs $O(mn)$ floating point operations to evaluate the Fourier coefficients which is far less than that are required if one uses Eq. (4.21) for non-orthogonal basis functions. Finally, we mention that the continuous orthogonal basis functions $\phi_i(.)$ do not generally satisfy the discrete orthogonality condition in case of discrete data set and results in a fully populated $\mathbf{A}$ matrix rather than a diagonal one. Fig. 35 shows the plot of orthogonalization error, $e = (\|\mathbf{A}^{-1} - [\sum_{i=1}^{m} w(\mathbf{x}_i)\phi_j^2(\mathbf{x}_i)]^{-1}\|)$ versus number of data points, m for the continuous case orthogonal basis functions of Table XII up to degree 2. From this figure, it is clear that $\mathbf{A}$ is generally a diagonally dominant matrix and the assumption of a diagonal $\mathbf{A}$ matrix in Eq. (4.28) is valid with a good accuracy as the number of measurements increase. Now, using Eq. (4.21), it is straightforward to show that $\mathbf{A}^{-1}$ denotes the Fourier coefficients error covariance matrix, $\mathbf{P}_{aa}$, given by the following equation:

$$P_{aa_{ij}} = \frac{\sigma^2}{k_{ij}}\delta_{ij} \approx \sigma^2 \left[\sum_{k=1}^{m} w(\mathbf{x}_i)\phi_i(\mathbf{x}_k)\phi_j(\mathbf{x}_k)\delta_{ij}\right]^{-1} \tag{4.29}$$

$\mathbf{P}_{aa}$ is a diagonally dominant matrix and may be approximated without computing the more expansive least square approximation. Further, the measurement estimate error covariance matrix, $\mathbf{P}_{yy}$, can be written as:

$$\mathbf{P}_{yy} = E[(\mathbf{y} - \hat{\mathbf{y}})(\mathbf{y} - \hat{\mathbf{y}}^T)] = \mathbf{H}\mathbf{P}_{aa}\mathbf{H}^T \tag{4.30}$$

Note, $\mathbf{P}_{yy}$ and $\mathbf{P}_{aa}$ denote measurement estimate and Fourier coefficient error covariance matrices, respectively, for any local approximation of the GLO-MAP algorithm. Further, in this chapter, we use subscript $_l$ to denote error covariance matrices for

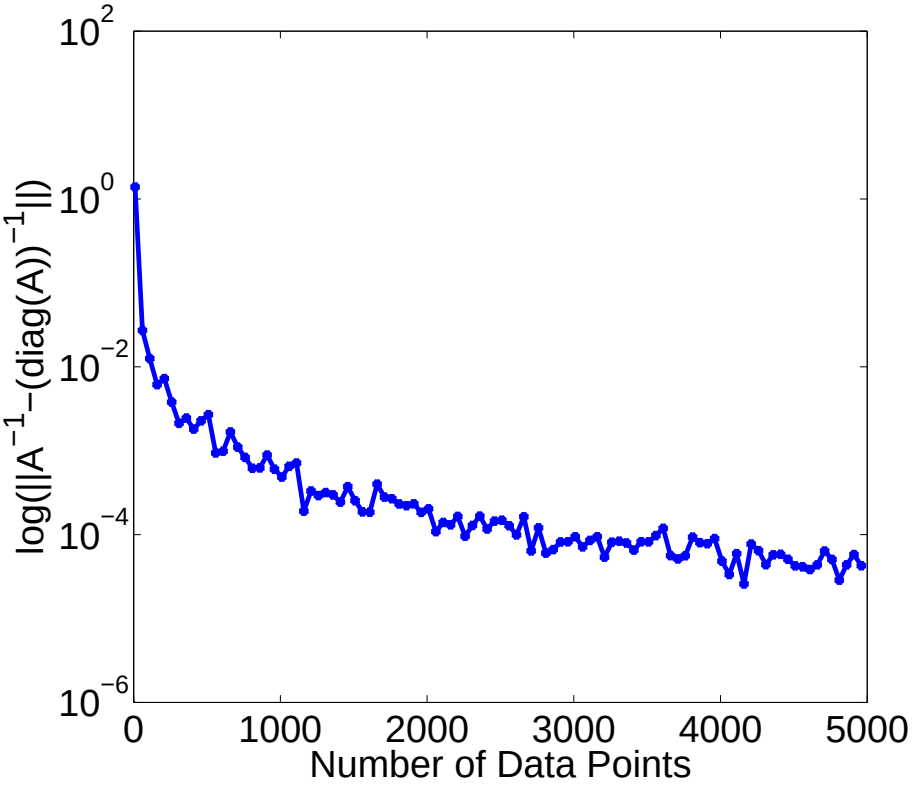


Fig. 35. Orthogonalization error for basis functions, ϕ_i of Table XII.

the l^{th} local approximation.

Now, according to the GLO-MAP procedure listed in Chapter III, different local approximations are merged together using specially designed weight functions to obtain a desired order piecewise continuous global estimates:

$$\hat{y}(\mathbf{x}) = \sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l)\hat{y}_l(\mathbf{x}_l) \tag{4.31}$$

where, M is the total number of local approximations and $\hat{y}_l = \boldsymbol{\phi}^T \mathbf{a}_l$ denotes l^{th} local approximation obtained by the least-square process. So, Eq. (4.31) can be re-written as:

$$\begin{aligned} \hat{y}(\mathbf{x}) &= \sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l)\mathbf{a}_l^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{x}_l) = \sum_{l=1}^{M} \bar{\mathbf{a}}_l^T \boldsymbol{\phi}(.) && (4.32) \\ &= \boldsymbol{\Phi}^T \bar{\mathbf{a}} && (4.33) \end{aligned}$$

where,

$$\begin{aligned} \boldsymbol{\Phi} &= \{\boldsymbol{\phi}(\mathbf{x}, \mathbf{x}_1), \cdots, \boldsymbol{\phi}(\mathbf{x}, \mathbf{x}_M)\}^T && (4.34) \\ \bar{\mathbf{a}} &= \{\bar{\mathbf{a}}_1, \bar{\mathbf{a}}_2, \cdots, \bar{\mathbf{a}}_M\}^T && (4.35) \end{aligned}$$

Now, using the linear error propagation theory, we can write:

$$\mathbf{P}_{yy_G} = \boldsymbol{\Phi}^T \mathbf{P}_{\bar{a}\bar{a}} \boldsymbol{\Phi} \tag{4.36}$$

where, $\mathbf{P}_{yy_G}$ denotes the global measurement estimate error covariance matrix and $\mathbf{P}_{\bar{a}\bar{a}} = \mathcal{W}\mathbf{P}_{aa_G}\mathcal{W}$, with

$$\mathcal{W} = \begin{bmatrix} w(\mathbf{x}, \mathbf{x}_1)\mathbf{I}_{n\times n} & & \\ & \ddots & \\ & & w(\mathbf{x}, \mathbf{x}_M)\mathbf{I}_{n\times n} \end{bmatrix} \tag{4.37}$$

$$\mathbf{P}_{aa_G} = \begin{bmatrix} \mathbf{P}_{aa_1} & & \\ & \ddots & \\ & & \mathbf{P}_{aa_M} \end{bmatrix} \tag{4.38}$$

Note, $\mathcal{W}$ is a diagonal matrix with all entries less than or equal to one. As a consequence of this, $\mathbf{P}_{\bar{a}\bar{a}}$ can be regarded as a contraction mapping of $\mathbf{P}_{aa_G}$ i.e. we can write:

$$\|\mathbf{P}_{\bar{a}\bar{a}}\| \leq \mathbf{P}_{aa_G} \tag{4.39}$$

Further, let us define global measurement error covariance matrix, $\mathbf{P}_{yy}$ without the averaging process of the GLO-MAP algorithm:

$$\mathbf{P}_{yy} = \mathbf{\Phi}^T\mathbf{P}_{\bar{a}\bar{a}}\mathbf{\Phi} \tag{4.40}$$

Now, from Eqs. (4.36), (4.39) and (4.40), we can conclude that

$$\|\mathbf{P}_{yy_G}\| \leq \|\mathbf{P}_{yy}\| \tag{4.41}$$

Note, Eq. (4.41) provides a quantitative justification for qualitative observation made earlier in Chapter III: "If one least square fit is good, the average of two should be better."

Finally, let us take expected value of Eq. (4.31)

$$E[\hat{y}(\mathbf{x})] = E[\sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l)\hat{y}_l(\mathbf{x}_l)] \tag{4.42}$$

As local approximations $y_l(\mathbf{x}_l)$ are obtained by carrying out the least-square process, therefore, as a consequence of un-biased property of the least square estimator, we have:

$$E[\hat{y}_l(\mathbf{x}_l)] = y(\mathbf{x}) \tag{4.43}$$

Now, substitution of Eq. (4.43) in Eq. (4.42) leads to the following equation for the expected value of the GLO-MAP approximation:

$$E[\hat{y}(\mathbf{x})] = \sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l) E[\hat{y}_l(\mathbf{x})] \tag{4.44}$$

$$= \sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l) y(\mathbf{x}) \tag{4.45}$$

Now, making use of the fact that weight functions $w(\mathbf{x}, \mathbf{x}_l)$ forms partition of unity i.e. $\sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l) = 1$, we get:

$$E[\hat{y}(\mathbf{x})] = \sum_{l=1}^{M} w(\mathbf{x}, \mathbf{x}_l) y(\mathbf{x}) = y(\mathbf{x}) \tag{4.46}$$

which proves that the GLO-MAP algorithm is a un-biased estimator which is an important property to have as in practice, unbiased estimators are rare. This truth, together with the covariance result of Eq. (4.41) provide a strong probabilistic justification of the GLO-MAP algorithm, to augment the attractive localization and piecewise continuity features.

D. Adaptive Multi-Resolution Algorithm

In this section, we present an efficient multi-resolution learning algorithm to approximate a general unknown input-output map. The main steps involved in formulating this multiresolution learning algorithm are described as follows:

1. Given input-output data, find a simple global model which captures the global

complexity at least in a coarse manner.

2. Refine the global model learned in the previous step until the desired approximation accuracy is achieved. To refine the global model we can introduce the local models based upon some heuristic without altering the global model.

3. To add these local models, use "model mismatch heuristic" i.e. add local models in the region where current model errors are more than desired accuracy.

4. Select the basis functions to describe the local models and learn their parameters using weighted statistics of local training data.

5. If the approximation errors are still large then either change the local basis functions or introduce more local models.

This whole process is repeated until introduction of more local models do not bring any improvement in the learning of input-output mapping.

To learn the global model, we use a two layer ANN with RBF activation functions (discussed in greater detail in Chapter II). The main feature of the proposed learning algorithm for the RBF based ANN is the judicious choice for locating and shaping the RBFs via a Directed Connectivity Graph approach. This approach allows a priori adaptive sizing of the network and zeroth order network pruning. In addition, it provides direction dependent scaling and rotation of basis functions for maximal trend sensing and minimal parameter representations. Adaptation of the network parameters is done to account for online tuning, given additional measurements. To gain high resolution, the input-output data is further represented by using a family of simpler local approximations, in addition to the global RBF approximation. This is done, because the RBF approach may be defeated by the curse of dimensionality if a highly irregular, high dimensioned system is to be approximated with high precision.

The most important step in implementing the multi-resolution algorithm is to learn the local models without altering the global approximation of the input-output map. Therefore, to learn local models, we propose the use of the GLO-MAP algorithm. The main feature of the GLO-MAP process is a weighting function technique that locally averages a family of overlapping preliminary approximations as corrections to a given global model and having a complete freedom on choosing preliminary local approximations. We introduce local models based upon the statistics of the global approximation residuals map. The regions where statistical measures of the errors (e.g. mean and standard variation) are larger than prescribed tolerances, the GLO-MAP process can be used to reduce approximation errors to achieve the desired resolution. To get an idea of the statistical error, the error analysis presented in the previous section can be used.

E. Numerical Results

To show the effectiveness of the proposed multi-resolution algorithm, we consider the problem of focal plane calibration of a vision sensor.

1. Calibration of Vision Sensors

Vision based sensors have found immense applications not only in aerospace industry but manufacturing inspection and assembly. Star tracker cameras and vision based sensors are primarily used to determine a spacecraft's attitude and position. *However, no sensor is perfect !.* In order to achieve high precision information from these sensors, those systematic effects which tend to introduce error in the information must be accounted for. These effects can include lens distortion and instrument aging. A lot of learning algorithms have been presented in literature to learn the focal plane

distortion map. A detailed overview of calibration of CCD cameras (digital cameras) can be found in Refs. [67, 68]. These papers provide a description of the various distortion mechanisms, and review means for which these distortion mechanisms can be accounted.

The first step in the calibration process is to hypothesize an observation model for the vision sensor. This is usually based on the physical insight regarding the particular sensor. For camera like sensors, the following collinearity equations are used to model the projection from object space to image space as a function of the attitude of the object:

$$x_i = -f\frac{C_{11}r_{x_i} + C_{12}r_{y_i} + C_{13}r_{z_i}}{C_{31}r_{x_i} + C_{32}r_{y_i} + C_{33}r_{z_i}} + x_0,\ i = 1, 2, \cdots, N \tag{4.47}$$

$$y_i = -f\frac{C_{21}r_{x_i} + C_{22}r_{y_i} + C_{23}r_{z_i}}{C_{31}r_{x_i} + C_{32}r_{y_i} + C_{33}r_{z_i}} + y_0,\ i = 1, 2, \cdots, N \tag{4.48}$$

where, C_{ij} are the unknown elements of attitude matrix $\mathbf{C}$ associated to the orientation of the image plane with respect to some reference plane, f is known focal length, (x_i, y_i) are the known image space measurements for the i^{th} line of sight, $(r_{x_i}, r_{y_i}, r_{z_i})$ are the known object space direction components of the i^{th} line of sight and N is the total number of measurements. x_0 and y_0 refers to the principal point offset. Generally, the focal plane calibration process is divided into two major parts:

1. Calibration of principal point offset (x_0, y_0) and focal length (f).

2. Calibration of the non-ideal focal plane image distortions due to all other effects (lens distortions, misalignment, detector alignment, etc.).

The implicit pin-hole camera model is not exact so we need to find the best effective estimates of principal point offset (x_0, y_0) and focal length (f). However, the principal point offset is obviously correlated with the inertial pointing of the boresight. In our earlier work [30, 48], we proposed the "attitude independent" approach (essentially,

based upon interstar angle measurements) to eliminate this difficulty. While this approach leads to reduced observability of (x_0, y_0), we find redundant measurement are sufficient to determine good estimates for (x_0, y_0) and f. Beside this, we need one attitude independent algorithm to identify the objects in the image plane. In Ref. [69], we presented a non-dimensional star identification algorithm for spacecraft attitude determination problem using star camera to identify the stars without any attitude knowledge. For any focal plane calibration algorithm to work, the un-calibrated sensor's errors must be sufficiently small so that the non-dimensional star identification algorithm works reliably. After the first calibration is achieved, our studies indicate that any of the several star identification algorithms work reliably. While the "how to get started" issue is important, we choose not to add to this discussion in this dissertation, and implicitly assume that the Eqs. (4.47) and (4.48) are sufficiently precise with the initial estimates of (x_0, y_0) and f. In this chapter, we only demonstrate the application of multi-resolution approximation procedure discussed in the previous section to learn higher order image distortion effects.

2. Simulation and Results

To demonstrate the effectiveness of the multi-resolution learning algorithm, an $8^\circ \times 8^\circ$ Field of View (FOV) star camera is simulated by using the pinhole camera model dictated by Eqs. (4.47) and (4.48) with principal point offset of $x_0 = 0.75mm$ and $y_0 = 0.25mm$. The focal length of the star camera is assumed to be $64.2964mm$.

For simulation purposes, the spacecraft is assumed to be in a low Earth orbit tumbling with following angular velocity about the sensor axis aligned to z-axis of the spacecraft body frame.

$$\omega = \{ \ \omega_0 sin(\omega_0 t) \quad \omega_0 cos(\omega_0 t) \quad \omega_0 \ \}, \omega_0 = 10^{-3} rad/sec \tag{4.49}$$

Assuming star camera frequency to be $1Hz$, the star data is generated for $2hr$ motion of the spacecraft.

The true lens distortion is assumed to be given by following models [68]

$$\mathbf{\Phi} \;=\; \left\{ \begin{array}{cccc} r & r^2 & r^3 & r^4 \end{array} \right\};\; \delta x = x\mathbf{\Phi}^T\mathbf{a} \;\&\; \delta y = y\mathbf{\Phi}^T\mathbf{b} \tag{4.50}$$

where,

$$r = \sqrt{x^2 + y^2} \tag{4.51}$$

To learn the distortion map, we need some measure of the measurement error that can be used to model the focal plane distortion map. Further, this model of the distortion map can be used to correct measurements. The best estimate of attitude and cataloged vectors are "run through" Eqs. (4.47) and (4.48) to predict δx_i, δy_i, the differences from measurements. Initially, the attitude, (C_{ij}), in Eqs. (4.47) and (4.48) was perturbed (not only by measurement errors, but also by calibration errors) because no distortion calibration has been applied on the first pass. We mention that after the first approximate calibration, δx and δy estimated distortions should be added to correct the measured x_i, y_i in Eqs. (4.47) and (4.48) before forming line of sight vectors that are used to estimate the attitude matrix $\mathbf{C}$. In Ref. [48], we have shown how the second order calibration perturbations of $\mathbf{C}$ gets reduced as δx, δy gets better. We mention that the reason for the convergence of calibration process is that the moderate sized calibration errors perturb the "rigid body" attitude estimate, but the residual errors in measurements minus the prediction of Eqs. (4.47) and (4.48) still have most of the high order distortion effects. For simulation purposes, the net attitude error due to residual calibration and sensor noise is sought to be $10\mu rad$ or smaller.

Fig. 36 shows the surface plot of true distortion map given by Eq. (4.50) with

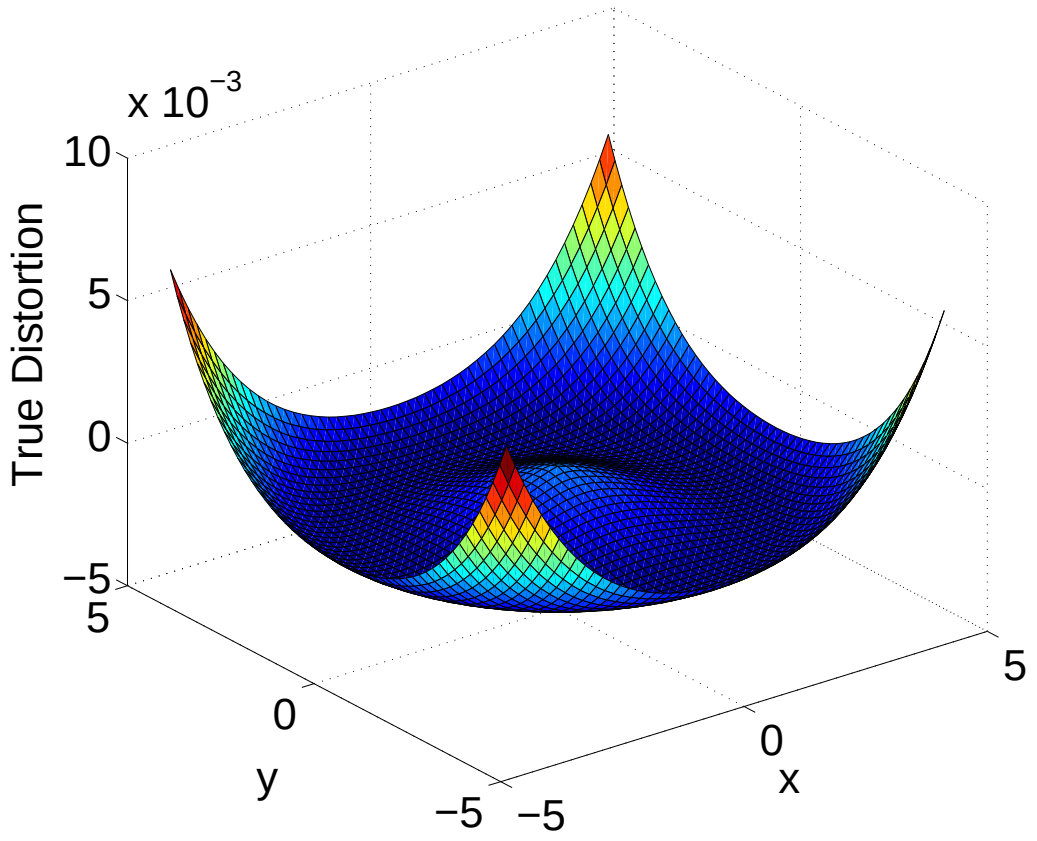


Fig. 36. True distortion map.

following value of **a**:

$$\mathbf{a} = \left\{ \begin{array}{cccc} 5e-04 & -5.0e-04 & 8e-04 & -8.0e-04 \end{array} \right\}^T \tag{4.52}$$

From this figure, it is clear that distortion surface amounts to calibration errors of the order of 10^{-3} radians. We seek to reduce these errors to the order of 10μ radians by using the multi-resolution algorithm discussed in the previous section. In next section we present the global approximation result using the DCG algorithm as discussed in

chapter II followed by local approximation (based upon "model mismatch heuristic") results using the GLO-MAP algorithm.

3. DCG Approximation Result

In this section, we present the global approximation result for the distortion map shown in Fig. 36 using the DCG algorithm as discussed in Chapter II. To approximate the distortion map given by Eq. (4.50), the input region is divided into a total of 4 square regions (2 in each direction). Then according to the procedure listed in Chapter II, we generate a directed connectivity graph of the local maxima and minima in each sub-region that finally add up to 8 radial basis functions to have approximation errors of the order of $O(10^{-3})$.

Fig. 37 shows the approximation error for the training set. From this figure, it is clear that the DCG learned RBF network is able to approximate the distortion map with a very good accuracy using only 8 radial basis functions. The DCG algorithm is tested upon uniformly distributed points in the focal plane. Fig. 38(a) shows the approximated distortion map learned by the DCG algorithm whereas Fig. 38(b) shows the approximation error surface. From these results, it is clear that although the DCG approach has done a good job in learning the shape of the distortion map and reducing the errors by about one order of magnitude, the approximated map still has some large amplitude errors.

4. Local Approximation Results

In the previous section, we presented the global approximation results for the distortion map given by Eq. (4.50). In this section, we present the results which show how the multi-resolution based learning algorithm improves the global approximation.

From the results presented in the previous section, it is clear that with only a

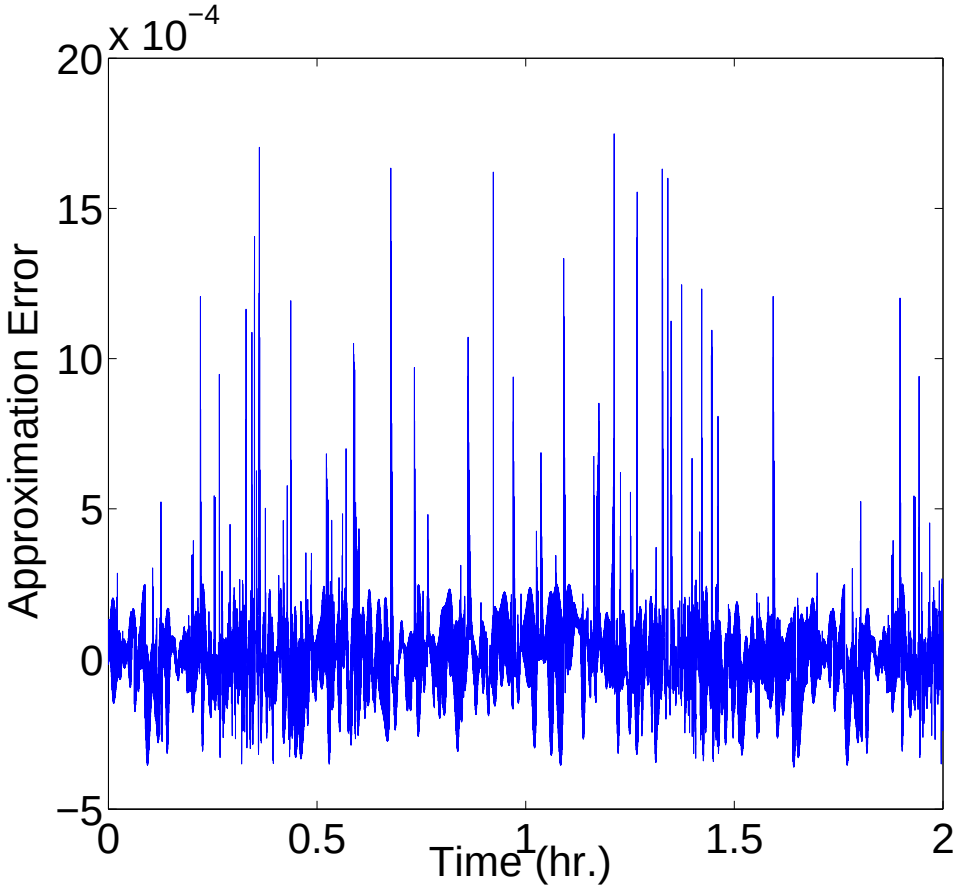
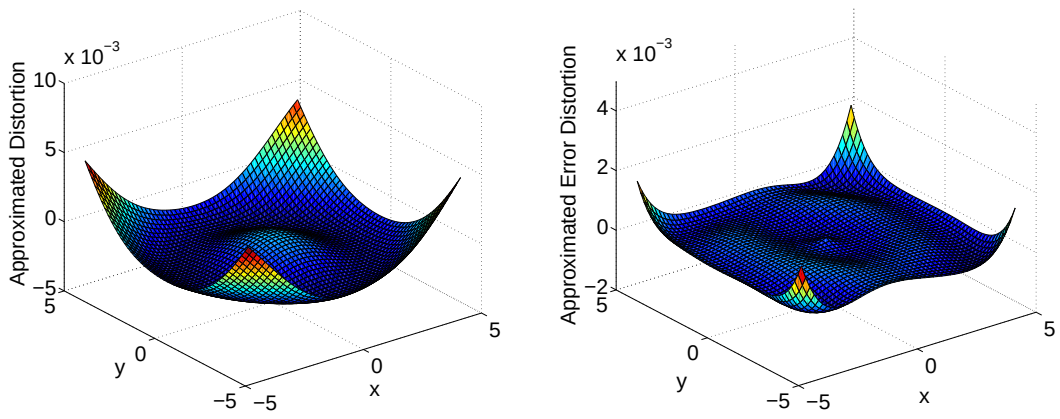


Fig. 37. Global approximation results using the DCG algorithm for the training data set.



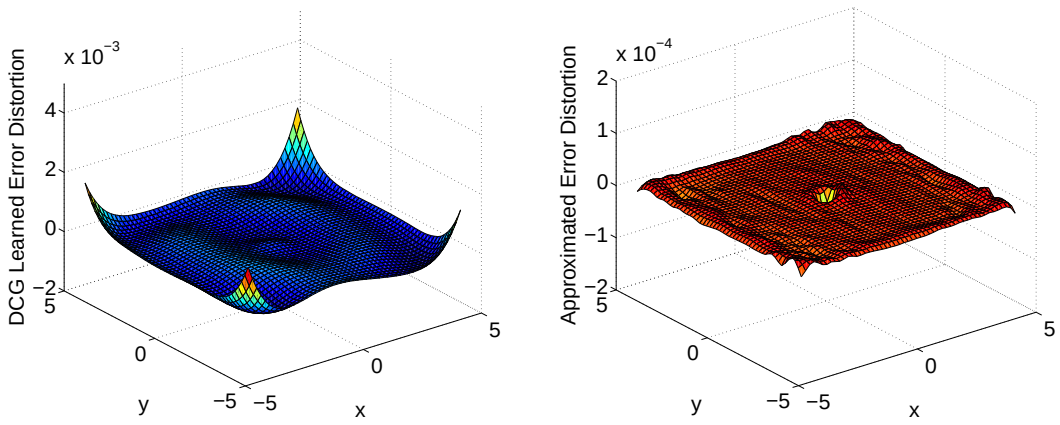
(a) The DCG approximated test surface (b) The DCG approximation error surface

Fig. 38. Global approximation results using the DCG algorithm for the test data set.

global DCG based RBFN approximation, we are able to learn the distortion map with an accuracy of the order of $O(10^{-3})$. To achieve the desired accuracy of the order of $O(10^{-5})$, we invoke the multi-resolution GLO-MAP algorithm to correct the approximation error surfaces from the DCG algorithm.

Fig. 39(a) shows the DCG approximation error surface corrected by the GLO-MAP algorithm whereas Fig. 39(b) show the net approximation error surfaces. From these figures, it is clear that with the help of local approximation using the GLO-MAP algorithm, we can further reduce the global approximation errors by two orders of magnitude. We mention that for the GLO-MAP process, we also have 2×2 grid so we have a total of 4 local approximations.

From these results, we can conclude that the multi-resolution based local approximation helps us in having a flexible and adaptive calibration process that does not rely on simply guessing a distortion map.



(a) The DCG approximation error surface learned by the GLO-MAP algorithm

(b) Net approximation error surface using the DCG and the GLO-MAP algorithms

Fig. 39. Multi-resolution approximation results.

F. Concluding Remarks

In this chapter we have made a transition from discussing numerical results of Chapter III to numerical analysis. In earlier chapters simulation results are used to validate the GLO-MAP algorithm, however, in this chapter, we have discussed multi-resolution approximation capability and various other properties of the GLO-MAP algorithm in detail. We have shown that GLO-MAP residual errors are orthogonal to the range space spanned by basis function ϕ_i. Beside this, approximation error bounds are computed for continuous function approximation followed by the discussion on the discretization issue. The freedom to vary in a general way the resolution (e.g., degrees of freedom) of the local approximations, un-biased estimates, and generalization to an N dimensional space are some of the prominent characteristics of the GLO-MAP algorithm. Finally, an efficient adaptive learning algorithm is developed which not only has the global approximation capability of the ANN but also has the multi-

resolution capability of the GLO-MAP algorithm. Computational experiments are conducted to evaluate the utility of the developed multi-resolution algorithm and simulation results does provide compelling evidence and a basis for optimism.

CHAPTER V

ROBUST NONLINEAR SYSTEM IDENTIFICATION ALGORITHMS USING AN ORTHOGONAL POLYNOMIAL NETWORK

A. Introduction

System IDentification (SysID) is the term associated with the estimation and validation of mathematical models of physical phenomena from measured input-output data. SysID is a most fundamental step in virtually all disciplines of science and engineering. Dynamical system models are used for the design and analysis of complex technical systems. Various classical and modern controller design techniques, such as the Linear Quadratic Regulator (LQR), and Lyapunov controller require a dynamical model between the control variables and the system output. Of course dynamical models can frequently be constructed from first principles, but it is also of vital importance that they can be approximated directly from measurements.

In the last five decades, mathematical system identification theory has evolved into a powerful scientific tool of wide applicability. However, the most mature part of the theory deals with linear systems using well established techniques of linear algebra and the theory of ordinary differential or difference equations. In contrast to this, the nonlinear system identification problem is still treated mostly on a system by system basis. In this chapter, our main interest is to present a general nonlinear system identification technique that can be applied for large flexible space structures.

This chapter is written with four main objectives. The first and most important objective is to present a novel robust nonlinear system identification method using the GLO-MAP algorithm. The second objective of this chapter is to present adaptive learning algorithms to adjust in real time the parameters of the GLO-MAP model.

The learning algorithm proposed in this chapter is inspired by recent developments in adaptive control [70, 71]. The third objective of this chapter is to compare the proposed algorithm with some existing identification algorithms like the Eigensystem Realization Algorithm [72] (ERA) considering applications involving modeling of large flexible space structures. The fourth and final objective of this chapter is to set down a theoretical framework including all assumptions, that guarantee the stability of the algorithm. Because these theoretical results have very few companion results in the existing nonlinear system identification theory, special care is taken to clearly state all the assumptions and develop theoretical conditions for stability.

The structure of this chapter is as follows: first the system identification problem is introduced followed by a brief review of some existing system identification algorithms. We give special attention to the Eigensystem Realization Algorithm (ERA) because of its broad utility and numerical robustness for linear and near linear systems. Then, two different robust system identification algorithms are introduced using the GLO-MAP algorithm [41], discussed in Chapter III. Finally, the proposed algorithms are validated and compared by simulating test cases concerned with large space structure applications.

B. Problem Statement and Background

Let us consider a nonlinear system described by the following differential and algebraic equations:

$$\dot{\mathbf{x}}(t) = f\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{5.1}$$

$$\mathbf{y}(t) = g\left(\mathbf{x}(t), \mathbf{u}(t)\right) \tag{5.2}$$

where $\mathbf{x} \in \mathcal{R}^n$ and $\mathbf{u} \in \mathcal{R}^p$ represent state and control vectors respectively, and $\mathbf{y} \in \mathcal{R}^m$ represents a vector of system outputs at time t. The discrete equivalent of this system is the following nonlinear difference equations:

$$\mathbf{x}_k = f_d(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \tag{5.3}$$

$$\mathbf{y}_k = g_d(\mathbf{x}_k, \mathbf{u}_k) \tag{5.4}$$

Now, if the functions, $f(.)$, $g(.)$, $f_d(.)$ and $g_d(.)$ are unknown, then the system identification problem can be formally stated as follows:

Definition of the System Identification Problem. *Identify a mathematical model which when subject to the actual input vector,* $\mathbf{u}$*, an output estimate* $\hat{\mathbf{y}}$ *is produced which approximates the actual system output,* $\mathbf{y}$*, such that*

$$\|\mathbf{y} - \hat{\mathbf{y}}\| \leq \epsilon \tag{5.5}$$

Here, $\|.\| : \mathcal{R}^m \rightarrow \mathcal{R}$ represents a suitable norm on the system output space, $\mathbb{Y}$, and ϵ dictates the desired accuracy of the system identification problem. In other words, the system identification problem corresponds to finding a model whose outputs are as close as desired to the true system outputs when the same input is applied to both. Therefore, the system identification problem can also be regarded as the identification of a continuous map from system input space to system output space [73]. Consequently the problem of approximating a continuous functional arises in the system identification problem. The output at any given time is considered as a function of the input signal, which is a function of time. Implicitly, we hope the input-output data approximated is sufficiently rich that the model will be accurate and useful for other purposes such as controlling the system.

Various system identification algorithms are described in the literature for input-

output mapping [31, 73–78]. The main computational tool employed by these algorithms is the process of Least Squares Estimation (LSE) frequently implemented using the Singular Value Decomposition (SVD). The LSE method along with SVD result in numerical robustness under very weak assumptions on the persistency of excitation of the inputs. In the past few decades, Artificial Neural Networks (ANNs) have emerged as a powerful set of tools in the areas of pattern classification, time series analysis, signal processing, dynamical system modeling and control. The emergence of the ANN can be attributed to the fact that these network models are frequently able to learn behavior when traditional modeling is very difficult to generalize. However, the optimal number of hidden units, perceptrons, depends upon many factors, like the ability of the chosen basis functions to approximate the given system's behavior, the number of data points, the signal to noise ratio, the complexity of the learning algorithms, etc. Narendra et al. [73, 75] have proposed different models that utilize two-layered neural networks with sigmoid functions as activation functions for system identification. In those papers, the output signal at any time is considered a function of finitely many samples of the input and output signals. The different ANN parameters are estimated using a back-propagation algorithm [3]. A key issue arises because if one fixes the architecture and activation functions, a given ANN's ability to approximate a given system's behavior can be deduced only after the learning process is over. Adaptation of the network architecture, not simply adjusting weights, has emerged as the key to convergence reliability and accuracy.

In Chapter II, an adaptive RBFN architecture making use of the Directed Connectivity Graph (DCG) algorithm is introduced and used for many system identification problems. It has been shown that the adaptive nature of the network improves significantly the performance of the algorithm in terms of accuracy and number of free network parameters, in comparison to many traditional algorithms. However, like

other traditional ANN algorithms, the DCG algorithm also treats the system identification problem as the identification of a continuous map from system input space to system output space. As a consequence of this, the performance of these algorithms decreases drastically as the dimension of the system output vector increases. To make this point more clear, consider a problem of active control of a flexible space structure. To derive a control law, a model of the system dynamics from the control variable, $\mathbf{u}(t)$ to the system output, $\mathbf{y}(t)$ is desired. Generally, the system output vector consists of surface distortion measurements at various spatial points, $O(10^3)$ which are measured by sensors like strain gauges, stereo vision systems, LIDAR, etc. Therefore, if one seeks a dynamic continuous map between the system output and input vectors then the dimension of such a map can be as large as number of measurements, i.e., $O(10^3)$. However, the dimension of the hidden states corresponding to the true system corresponds to the number of dynamic structural modes of interest which are typically on the order of 10 to 30. So, a system identification algorithm is desired that can approximate the system output well, while keeping the dimension of dynamic map as low as possible. To deal with this problem, various model reduction techniques are often adopted [79] for approximating high order dynamic models by simpler, lower order models. The most popular method for model reduction is Proper Orthogonal Decomposition (POD) [79], also known as the Principal Component Analysis (PCA). However, in model reduction, one would like to preserve properties of the original model, such as stability and physically important dynamical mode shapes. However, as POD uses second-order statistics for model reduction, it sometimes de-emphasizes infrequent events which can be dynamically very important.

In the next section, two novel nonlinear system identification algorithms are introduced which make use of the classical Eigensystem Realization Algorithm (ERA) [72] and the recently developed Global-Local Orthogonal Polynomial Mapping (GLO-

MAP) [41] network to deal with the issues of nonlinearity and high dimensioned output vector in an efficient manner.

C. Novel System Identification Algorithm

In the previous section, issues concerning the inability of various system identification algorithms to handle the "curse of dimensionality" are discussed. Here, two novel system identification algorithms are presented which not only has the approximation ability of the ANN but also has the model reduction ability of algorithms like POD.

The basic idea of both the proposed algorithms is to split the identification process into two steps: linear system identification followed by the nonlinear system identification process. The linear system identification process not only helps in designing an estimator to estimate hidden dynamic states from the measurement data, but also gives an approximate dimension for the reduced order dynamical model for the hidden state vector. It implicitly defines a transformed state space that is physically motivated to capture the best linear representation of the system input-output behavior. We elect to retain this linear transformation, as the starting point for a perturbation to account for the nonlinear departure from this best linear model. The use of the linear system system to establish a desired order state space model is an attractive feature of the new algorithms which also helps in dealing with the "curse of dimensionality".

The first step of both algorithms is to identify a linear dynamical system from the time history of input-output data. Referring to Figs. 40 and 41, let the best linear model ("realization") be written as:

$$\dot{\mathbf{x}}_l = \mathbf{A}_l\mathbf{x}_l + \mathbf{B}_l\mathbf{u} \tag{5.6}$$

$$\mathbf{y}_l = \mathbf{C}_l\mathbf{x}_l + \mathbf{D}_l\mathbf{u} \tag{5.7}$$

Here, $\mathbf{x}_l \in \mathcal{R}^n$ is a hidden state vector corresponding to the best linear approximation of given input-output data. While $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{C}_l$, $\mathbf{D}_l$ are not unique, the underlying input-output map is, and the $\mathbf{A}_l$, $\mathbf{B}_l$, $\mathbf{C}_l$, $\mathbf{D}_l$ realization from the ERA can be robustly computed, including the dimension n of the state space. The accuracy of the output vector, $\mathbf{y}_l$, in approximating the true system output data, $\mathbf{y}$, depends upon the nonlinearities involved. Further, two different sequential algorithms are proposed to apply the correction to the best identified linear system.

1. In the first approach, the linear state dynamical model of Eq. (5.6) is perturbed by a nonlinear term to learn the difference between the linear propagated state vector, $\mathbf{x}_l$, and the best estimate of state vector, $\mathbf{x}_b$.

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}_l\hat{\mathbf{x}} + \hat{\mathbf{B}}\mathbf{u} + \mathbf{g}(\hat{\mathbf{x}}) \tag{5.8}$$

$$\hat{\mathbf{y}} = \mathbf{C}_l\hat{\mathbf{x}} + \mathbf{D}_l\mathbf{u} \tag{5.9}$$

   Here, $\mathbf{g}(\hat{\mathbf{x}})$ is a vector of unknown nonlinearities which can be learned by conventional ANN methods. Notice we have made $\mathbf{g}$ depend on $\hat{\mathbf{x}}$, the same reduced order state vector that resulted from the best fitting linear system. To find the best estimates of the hidden dynamic state vector, $\mathbf{x}_b$, from given system output data, $\mathbf{y}$, an efficient estimator such as an algebraic Kalman filter can be designed using measurement model of Eq. (5.9). The main steps of this algorithm are illustrated in Fig. 40 and we denote this approach by short form SysID 1. For the purpose of this chapter, $\mathbf{g}(.)$ represents the system nonlinearities not captured by the linear model and are modeled by using the GLO-MAP process of Chapter III. Note, the performance of this approach depends upon the dimensionality of the hidden state vector $\mathbf{x}$ and the frequency at which hidden state vector estimates can be obtained.

2. In the second approach, we propose the design of a Kalman filter using the best known linear model followed by the nonlinear transformation of the estimated output data to compensate for their deviation from true output data $\mathbf{y}$.

$$\hat{\mathbf{y}}_l = \mathbf{C}_l\hat{\mathbf{x}}_l + \mathbf{D}_l\mathbf{u} \tag{5.10}$$

$$\hat{\mathbf{y}} = \hat{\mathbf{y}}_l + \mathbf{h}(\mathbf{x}_p) \tag{5.11}$$

Here, $\hat{\mathbf{x}}_l$ represents the state output of linear Kalman filter. Note, the design of Kalman filter helps us in reducing the propagation error arising due to system nonlinearities if output data $\mathbf{y}$ is available at a reasonable frequency. Further, $\mathbf{h}(.)$ represents the system nonlinearities not captured by the linear model and can be modeled by traditional ANN algorithms. To keep the dimensionality of the nonlinear transformation $\mathbf{h}(.)$ to be low, we introduce a new variable $\mathbf{x}_p$. The dummy variable $\mathbf{x}_p$ can be regarded as a physical variable associated with the problem in hand not necessarily be same as hidden state vector $\mathbf{x}_l$. For example, in case of the modeling of flexible space structure, the system output vector consists of surface distortion measurements at various spatial points, therefore, the dummy variable $\mathbf{x}_p$ can consist of cartesian coordinates (x, y, z). In other words, the surface distortions can be modeled as a function of cartesian coordinates. We find, in many problems, a physically motivated low-dimensioned $\mathbf{x}_p$ can be chosen which leads to an accurate nonlinear input-output map. The overall architecture of this algorithm is illustrated in Fig. 41. For the purpose of this chapter, $\mathbf{h}(.)$ is modeled by using the GLO-MAP process of Chapter III and short form SysID 2 is used to describe this approach. Note, the performance of this approach depends upon the number of measurement points available to learn $\mathbf{h}(.)$ and the frequency at which measurement data is

available.

We mention that, ideally, a combination of both the approaches is desired for efficient and accurate modeling of the dynamical systems. However, the use of the first approach is recommended if the dimensionality of the hidden state vector $\mathbf{x}_l$ is low because as the dimensionality of the hidden state vector $\mathbf{x}_l$ increases the number of terms required to model $\mathbf{g}(.)$ increases exponentially. Further, the use of the second approach is highly desirable for the modeling of flexible space structure when the number of participating modes are large in number O(10-30). We mention that the system-identification problem as stated in this section does not deal with the issue of uniqueness of the mathematical model. This issue can not be dealt with theoretically in general for nonlinear system, but this criticism is not unique for our approach. We find that this theoretical deficiency is usually not an obstacle to practical progress; we note that the main practical objective of the system identification process is the generation of workable mathematical model for technical analysis. Finally, the convergence of both the algorithms is an important issue but will be studied later in this chapter after discussing each step in detail.

1. Linear System Identification

As discussed in the previous section, linear system identification plays an important role in the success of both the nonlinear system identification algorithms. The linear system identification process not only helps in designing an estimator to estimate hidden dynamic states from sensor noise corrupted measurement data, but also gives a desired order dynamical model for the hidden state vector. A large class of linear system identification methods [76–78] are addressed in the literature to estimate hidden state variables along with the dynamical model from given input-output data.

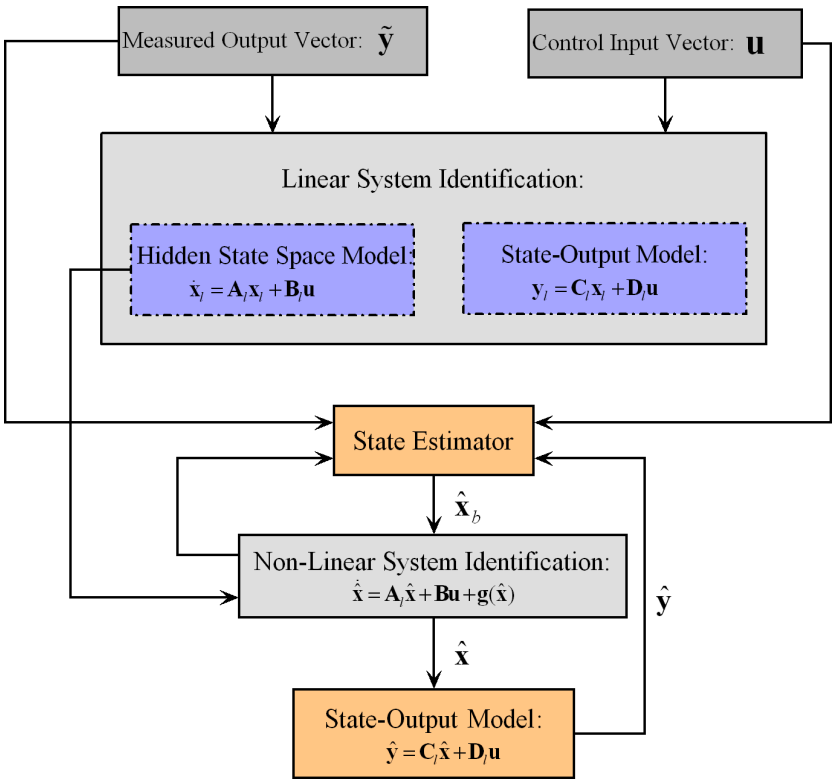


Fig. 40. Overall architecture of the first proposed system identification algorithm.

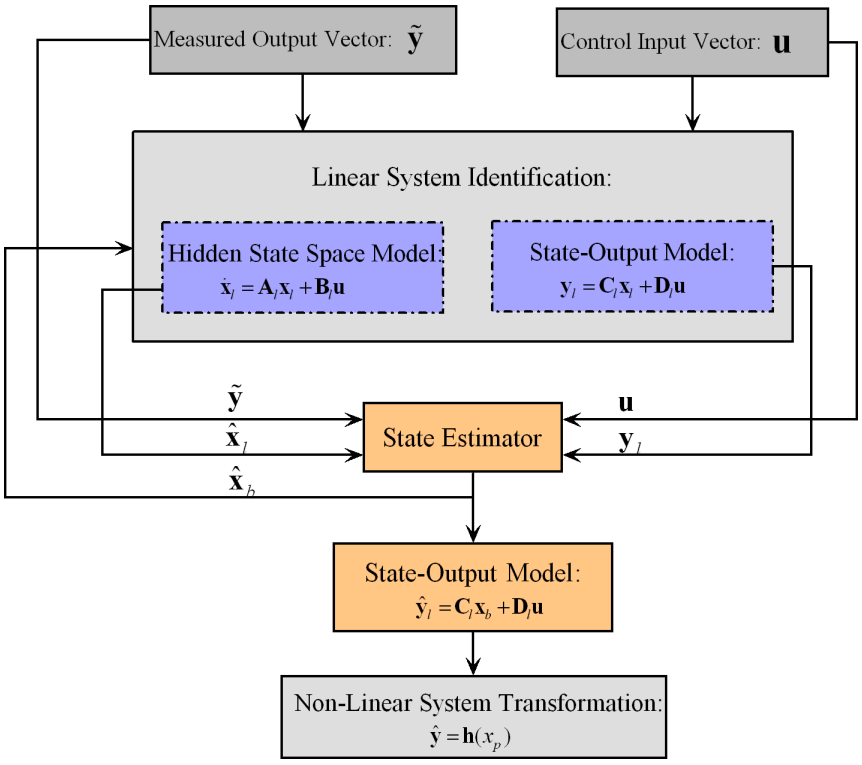


Fig. 41. Overall architecture of the second proposed system identification algorithm.

However, we believe the ERA and the Observer/Kalman filter IDentification (OKID) algorithms [72,78] are among the most popular for the dynamic system modeling and have been successfully used in various system identification problems for structural analysis. The first realization, i.e., ERA, has been found to be particularly robust and useful for structural dynamic systems where one is interested in identifying the dominate mode, eigenvalues, and modal shapes; these may be identified directly by this approach. The ability to identify only the modes actually participating in the measured behavior of the system helps in dramatically reducing the order of the system and thus implicitly dealing with the "curse of dimensionality". The ERA algorithm is recommended for the linear system identification module in both the algorithms(see Figs. 40 and 41), however, any other linear system identification algorithm can be used instead of the ERA. In this section, the main steps of the ERA algorithm are briefly discussed and more details can be found in Ref. [72].

1. The first step of the ERA method is to form the Hankel matrix from the measurement outputs $\mathbf{Y}(t_k)$, according to the following expression.

$$\mathbf{H}_{rs}(k-1) = \begin{bmatrix} \mathbf{Y}(k) & \mathbf{Y}(k+t_1) & \cdots & \mathbf{Y}(k+t_{s-1}) \\ \mathbf{Y}(j_1+k) & \mathbf{Y}(j_1+k+t_1) & \cdots & \mathbf{Y}(j_1+k+t_{s-1}) \\ \vdots & \vdots & & \vdots \\ \mathbf{Y}(j_r+k) & \mathbf{Y}(j_r+k+t_1) & \cdots & \mathbf{Y}(j_r+k+t_{s-1}) \end{bmatrix} \tag{5.12}$$

   Further, it can be easily shown that the Hankel matrix expression generalizes to the following factored expression:

$$\mathbf{H}_{rs}(k) = \mathbf{V}_r \mathbf{A}^k \mathbf{W}_s \tag{5.13}$$

where $\mathbf{V}_r$ is the observability matrix given by following expression

$$\mathbf{V}_r = \begin{bmatrix} \mathbf{C} \\ \mathbf{C}\mathbf{A}^{j1} \\ \vdots \\ \mathbf{C}\mathbf{A}^{j_r-1} \end{bmatrix} \tag{5.14}$$

whereas $\mathbf{W}_s$ is a controllability or disturbability matrix, given by following equations depending upon the control input.

$$\text{Impulse Response (IR): } \mathbf{W}_s = \begin{bmatrix} \mathbf{B} & \mathbf{A}^{t1}\mathbf{B} & \cdots & \mathbf{A}^{t_s-1}\mathbf{B} \end{bmatrix} \tag{5.15}$$

$$\text{Initial State Response (ISR): } \mathbf{W}_s = \begin{bmatrix} \mathbf{B} & \mathbf{A}^{t1}\mathbf{X}_0 & \cdots & \mathbf{A}^{t_s-1}\mathbf{X}_0 \end{bmatrix} \tag{5.16}$$

2. In the second step, a matrix $\mathbf{H}^\sharp$ is desired such that following is true:

$$\mathbf{W}_s\mathbf{H}^\sharp\mathbf{V}_r = \mathbf{I}_n \tag{5.17}$$

A general solution for $\mathbf{H}^\sharp$ is found by the Singular Value Decomposition (SVD) of $\mathbf{H}_{rs}(0) = \mathbf{P}\mathbf{D}\mathbf{Q}^T$:

$$\mathbf{H}^\sharp = \mathbf{Q}\mathbf{D}^{-1}\mathbf{P}^T \tag{5.18}$$

3. Finally, after some algebraic manipulations, the following relationship for $\mathbf{Y}(k+1)$ is obtained:

$$\mathbf{Y}(k+1) = \underbrace{\mathbf{E}_p^T\mathbf{P}\mathbf{D}^{1/2}}\underbrace{\left[\mathbf{D}^{-1/2}\mathbf{P}^T\mathbf{H}_{rs}(1)\mathbf{Q}\mathbf{D}^{-1/2}\right]^k}\underbrace{\mathbf{D}^{1/2}\mathbf{Q}^T\mathbf{E}_m} \tag{5.19}$$

where, $\mathbf{E}_k^T = \begin{bmatrix} \mathbf{I}_k & \mathbf{O}_k & \cdots & \mathbf{O}_k \end{bmatrix}$. Comparing this relationship with Eqs. (5.15) and (5.16), the following expressions for matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are obtained for the case of Impulse Response (IR) and Initial State Response (ISR),

respectively:

$$\text{IR:}\quad \mathbf{A} = \mathbf{D}^{-1/2}\mathbf{P}^T\mathbf{H}_{rs}(1)\mathbf{Q}\mathbf{D}^{-1/2} \quad \mathbf{B} = \mathbf{D}^{1/2}\mathbf{Q}^T\mathbf{E}_m \quad \mathbf{C} = \mathbf{E}_p^T\mathbf{P}\mathbf{D}^{1/2}$$

$$\text{ISR:}\quad \mathbf{A} = \mathbf{D}^{-1/2}\mathbf{P}^T\mathbf{H}_{rs}(1)\mathbf{Q}\mathbf{D}^{-1/2} \quad \mathbf{X}_0 = \mathbf{D}^{1/2}\mathbf{Q}^T\mathbf{E}_m \quad \mathbf{C} = \mathbf{E}_p^T\mathbf{P}\mathbf{D}^{1/2}$$

Now, let the estimated state matrix $\mathbf{A}$ be of order n and have a complete set of linearly independent eigenvectors $(\psi_1, \psi_2, \cdots, \psi_n)$ with corresponding eigenvalues $(\lambda_1, \lambda_2, \cdots, \lambda_n)$ which are not necessarily distinct. Define $\mathbf{\Lambda}$ as the diagonal matrix of eigenvalues and $\mathbf{\Psi}$ as the matrix whose columns are the eigenvectors. Then the minimum state-realization can be transformed to a minimum modal-realization. The diagonal matrix $\mathbf{\Lambda}$ contains the information about modal damping rates and damped natural frequencies, which are simply the real and imaginary parts of the eigenvalues, after transformation from discrete to continuous-time domain via $\Lambda_c = ln(\Lambda)/\delta t$. The columns of the matrix $\mathbf{\Psi}^{-1}\mathbf{B}$ define the initial modal amplitudes, or information that indicates how effective a particular input is at exciting each mode. The columns of the matrix $\mathbf{C}$ define the transformation from modal coordinates to the physical coordinates, i.e., system outputs.

2. State Variable Estimation

The hidden state variable estimation is an important step in both the system identification algorithms given the input-output data and the best learned linear system. Among various estimation algorithm listed in the literature, the Kalman filter [26] is the most widely used for dynamical state identification.

Kalman filtering is a modern (since 1960) development in the field of estimation [29, 30] although it has its roots as far back as in Gauss' work in the 1800's. The only qualitative difference between the Kalman filter and the sequential version

of the Gaussian least squares is that the Kalman filter uses a dynamical model of the plant to propagate the state estimates and the corresponding error covariance matrix between two sets of measurements. In this section, Kalman filter algorithm is described to find the best estimate of the hidden state variable, $\hat{\mathbf{x}}_b$ for proposed nonlinear system identification algorithms.

The various steps involved in the estimation of hidden state variables using Kalman filter are listed as:

1. **Propagation**: This step involves the propagation of the estimated hidden state variable $\hat{\mathbf{x}}$ and its corresponding state error covariance matrix $\mathbf{P}_x$ using the best known dynamical system and the corresponding Ricatti equation:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{G}\mathbf{w} \tag{5.20}$$

$$\dot{\mathbf{P}}_x = \mathbf{P}_x\mathbf{A} + \mathbf{A}^T\mathbf{P}_x + \mathbf{G}\mathbf{Q}\mathbf{G} \tag{5.21}$$

   Here, $\mathbf{w}$ represent the process noise vector modeled as Gaussian white noise with known covariance matrix $\mathbf{Q}$. It should be mentioned that Eq. (5.20) represents the best known differential equation for the evolution of hidden state variable $\mathbf{x}$. Eqs. (5.6) and (5.8) are used for hidden state propagation in system identification approaches, SysID 2 and SysID 1, respectively.

2. **Update**: Given the measurement vector, $\tilde{\mathbf{y}}$, at any time, t, the algebraic relationship between the state vector $\mathbf{x}$ and the system output vector $\mathbf{y}$ is used to update the propagated estimates of the unknown hidden state vector $\mathbf{x}^-$ and the corresponding error covariance matrix $\mathbf{P}_x^-$:

$$\tilde{\mathbf{y}} = \mathbf{H}\mathbf{x} + \mathbf{D}\mathbf{u} + \boldsymbol{\nu} \tag{5.22}$$

   where $\boldsymbol{\nu}$ denotes the measurement noise vector modeled as Gaussian white noise

with known covariance matrix $\mathbf{R}$. The following expression can be derived for the state vector estimates using the least square criteria described in Ref. [29]:

$$\begin{aligned} \mathbf{K} &= \mathbf{P}_x^- \mathbf{H}^T \left(\mathbf{H} \mathbf{P}_x^- \mathbf{H}^T + \mathbf{R} \right)^{-1} \\ \hat{\mathbf{x}} &= \hat{\mathbf{x}}^- + \mathbf{K} \left(\tilde{\mathbf{y}} - \hat{\mathbf{y}} \right) \\ \mathbf{P}_x &= \left(\mathbf{I} - \mathbf{K}\mathbf{H} \right) \mathbf{P}_x^- \end{aligned}$$

The main assumption in using the above expression is *complete observability* of the state vector $\mathbf{x}$ from Eq. (5.22). In other words, to estimate $\mathbf{x}$ using Eq. (5.22), the matrix $\mathbf{H}$ should have its rank equal to the dimension of $\mathbf{x}$ i.e. n. It should be mentioned that Eq. (5.7) is used as a counterpart to Eq. (5.22) for both the system identification algorithms described in Section C.

D. Nonlinear System Identification Algorithm

In previous sections, we have introduced two nonlinear system identification algorithms. However, we have not discussed, in detail, the procedure to learn the nonlinear terms in both the algorithms. In this section, first, an adaptive learning algorithm is described to update the linear dynamic model using the hidden state estimates as measurements in case of SysID 1 followed by the description of the learning algorithm for SysID 2.

1. Learning Algorithm for SysID 1

The learning algorithm for SysID 1 is based upon the recently developed GLO-MAP network and uses Lyapunov's stability theorem [80] to determine the update laws for different parameters of the GLO-MAP network.

Consider the perturbed linear dynamic model, where $\mathbf{g}(\mathbf{x})$ is a vector of nonlinear

terms.

$$\dot{\mathbf{x}} = \mathbf{A}_l\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{g}(\mathbf{x}) \tag{5.23}$$

$$\mathbf{y} = \mathbf{C}_l\mathbf{x} + \mathbf{D}_l\mathbf{u} \tag{5.24}$$

Here, $\mathbf{A}_l \in \mathcal{R}^{n\times n}$ is a known hurwitz matrix and $\mathbf{B} \in \mathcal{R}^{n\times p}$ is a control effectiveness matrix. It should be noted that the matrix $\mathbf{A}_l$ is chosen in such a way that it captures the modal frequencies of the interest and can be obtained by the ERA or any other linear system identification algorithm as described in section C.

The time history estimates of the hidden state vector $\mathbf{x}$ can be obtained by using the procedure described in section C, so the system identification problem can be re-defined as:

System Identification Problem. *Given the time history estimates of the state vector* $\mathbf{x}(t)$ *and control variable,* $\mathbf{u}(t)$*, find estimates of the unknown nonlinearity vector* $\mathbf{g}(.)$ *and control effectiveness matrix* $\mathbf{B}$*.*

Further, if $\mathbf{g}(.)$ is assumed to be a continuous function in $\mathbf{x}$ then according to Weierstrass approximation theorem [49,50], $\mathbf{g}(.)$ can be approximated arbitrarily close by any set of complete functions, including a polynomial series.

$$\mathbf{g}(\mathbf{x}) = \mathbf{C}^T\mathbf{\Phi}(\mathbf{x}) + \epsilon \tag{5.25}$$

where, $\mathbf{\Phi}(.)$ is an infinite dimensional vector of polynomial functions, $\mathbf{C}$ is a matrix of Fourier coefficients corresponding to polynomial functions, and ϵ denotes the residual approximation error. However, as a consequence of Theorem 1, $\mathbf{\Phi}(.)$ can be chosen as a finite dimensional vector of orthogonal polynomials. Therefore, $\mathbf{C} \in \mathcal{R}^{N\times n}$ is a matrix of Fourier coefficients corresponding to these orthogonal polynomial functions.

Now, substituting Eq. (6.16) in Eq. (5.23) yields:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_l\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{C}^T\mathbf{\Phi}(\mathbf{x}) + \epsilon \tag{5.26}$$

But the Fourier coefficient matrix $\mathbf{C}$ is unknown so we write an estimate equation

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}_l\hat{\mathbf{x}}(t) + \hat{\mathbf{B}}\mathbf{u}(t) + \hat{\mathbf{C}}^T\mathbf{\Phi}(\mathbf{x}) \tag{5.27}$$

Let us define $\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$, which leads to the following expression:

$$\begin{aligned} \dot{\mathbf{e}}(t) &= \mathbf{A}_l\mathbf{e} + (\mathbf{B} - \hat{\mathbf{B}})\mathbf{u}(t) + (\mathbf{C} - \hat{\mathbf{C}})^T\mathbf{\Phi}(\mathbf{x}) + \epsilon & (5.28)\\ &= \mathbf{A}_l\mathbf{e} + \tilde{\mathbf{B}}\mathbf{u}(t) + \tilde{\mathbf{C}}^T\mathbf{\Phi}(\mathbf{x}) + \epsilon & (5.29) \end{aligned}$$

Now, to find adaptation laws for the unknown parameters, we consider the following Lyapunov function:

$$V = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{1}{2}Tr(\tilde{\mathbf{B}}\Gamma_1\tilde{\mathbf{B}}^T) + \frac{1}{2}Tr(\tilde{\mathbf{C}}^T\Gamma_2\tilde{\mathbf{C}}) \tag{5.30}$$

where, $\mathbf{P}$ is a positive definite symmetric matrix. Now taking the time derivative of V leads to the following equation:

$$\begin{aligned} \dot{V} &= \frac{1}{2}\mathbf{e}^T\underbrace{\left(\mathbf{P}\mathbf{A}_l + \mathbf{A}_l^T\mathbf{P}\right)}_{-\mathbf{Q}}\mathbf{e} + \mathbf{e}^T\mathbf{P}\left(\tilde{\mathbf{B}}\mathbf{u}(t) + \tilde{\mathbf{C}}^T\mathbf{\Phi}(\mathbf{x}) + \epsilon\right) + Tr\left(\tilde{\mathbf{B}}\Gamma_1\dot{\tilde{\mathbf{B}}}^T\right)\\ &\quad + Tr\left(\tilde{\mathbf{C}}^T\Gamma_2\dot{\tilde{\mathbf{C}}}\right) & (5.31)\\ &= -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} + Tr\left(\tilde{\mathbf{B}}\left[\Gamma_1\dot{\tilde{\mathbf{B}}}^T + \mathbf{u}\mathbf{e}^T\mathbf{P}\right]\right) + Tr\left(\tilde{\mathbf{C}}^T\left[\Gamma_2\dot{\tilde{\mathbf{C}}} + \mathbf{\Phi}(\mathbf{x})\mathbf{e}^T\mathbf{P}\right]\right)\\ &\quad + \mathbf{e}^T\mathbf{P}\epsilon & (5.32) \end{aligned}$$

Note, here $\mathbf{Q} \in \mathcal{R}^{n\times n}$ is a positive definite matrix which satisfies the following algebraic Ricatti equation

$$\mathbf{P}\mathbf{A}_l + \mathbf{A}_l^T\mathbf{P} = -\mathbf{Q} \tag{5.33}$$

Now, if following adaptation laws are chosen for $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$,

$$\dot{\tilde{\mathbf{B}}}^T = -\dot{\hat{\mathbf{B}}}^T = -\Gamma_1^{-1}\mathbf{u}\mathbf{e}^T\mathbf{P} \tag{5.34}$$

$$\dot{\tilde{\mathbf{C}}} = -\dot{\hat{\mathbf{C}}} = -\Gamma_2^{-1}\mathbf{\Phi}(\mathbf{x})\mathbf{e}^T\mathbf{P} \tag{5.35}$$

then $\dot{V}$ reduces to:

$$\dot{V} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} + \epsilon^T\mathbf{P}\mathbf{e} \tag{5.36}$$

$$\Rightarrow \dot{V} \leq -\frac{1}{2}|\lambda_{min}(\mathbf{Q})|\|\mathbf{e}\|^2 + \|\epsilon\|\|\mathbf{P}\|\|\mathbf{e}\| \tag{5.37}$$

Note:

- When $\epsilon = 0$, i.e., there are no approximation errors, we have following expression for $\dot{V}$:

  $$\dot{V} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} \leq 0 \tag{5.38}$$

  Now, the convergence of tracking residual $\mathbf{e}$ follows from the assumption that $\mathbf{e} \in L_\infty$, i.e., both $\mathbf{x}$ and $\hat{\mathbf{x}}$ are bounded signals. Further, from the integral of $\dot{V}$, it can be easily shown that $\mathbf{e} \in L_2 \cap L_\infty$ and therefore, from Barbalat's Lemma [70] $\mathbf{e} \to 0$ as $t \to \infty$, which in turn leads to $\tilde{\mathbf{B}} \to 0$ and $\tilde{\mathbf{C}} \to 0$ based on Eqs. (5.34) and (5.35). We mention that although $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$ approaches 0 but their convergence to corresponding true values is not guaranteed without the satisfaction of the *persistence of excitation* condition [70].

- For a given level of the tracking errors $\mathbf{e}$, we can only conclude bounded stability, as long as the approximation error ϵ satisfies the following bound:

  $$\|\epsilon\| \leq \frac{|\lambda_{min}(\mathbf{Q})|\|\mathbf{e}\|}{2\|\mathbf{P}\|} = \epsilon_{ub} \tag{5.39}$$

  The above inequality gives us a upper bound on the approximation error ϵ to guarantee the bounded stability of system identification error $\mathbf{e}$. Recall that,

Theorem 3 provides us a lower bound on approximation errors which can be used to find the conservative estimate on the number of polynomial basis functions required to approximate $\mathbf{g}(.)$ so that Eq. (5.39) is satisfied. However, if the inequality in Eq. (5.39) is violated then $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$ may drift to infinity with time. To accommodate these one can set upper bounds $\mathcal{B}$ and $\mathcal{C}$ on $\|\hat{\mathbf{B}}\|$ and $\|\hat{\mathbf{C}}\|$, respectively. Thus the modified adaptation laws are:

$$\dot{\hat{\mathbf{B}}}^T = \begin{cases} -\Gamma_1^{-1}\mathbf{u}\mathbf{e}^T\mathbf{P}, & \text{if } \|\hat{\mathbf{B}}\| \leq \mathcal{B} \\ 0, & \text{otherwise} \end{cases} \tag{5.40}$$

$$\dot{\hat{\mathbf{C}}} = \begin{cases} -\Gamma_2^{-1}\mathbf{\Phi}(\mathbf{x})\mathbf{e}^T\mathbf{P}, & \text{if } \|\hat{\mathbf{C}}\| \leq \mathcal{C} \\ 0, & \text{otherwise} \end{cases} \tag{5.41}$$

According to the above modified update laws $\dot{V}$ is always negative semi-definite and stability arguments are same as in case of $\epsilon = 0$. However, in the case where the bound in Eq. (5.39) is violated, the estimates of $\hat{\mathbf{B}}$, $\hat{\mathbf{C}}$ and $\mathbf{e}$ may increase as $\dot{V} > 0$ but all the quantities are still bounded due to the adaptation law in Eqs. (5.40) and (5.41).

It should be noticed that Eq. (5.39) reiterates the importance of accurate approximation of nonlinear function $\mathbf{g}(.)$. According to Stone-Weierstrass's approximation theorem as $N \to \infty$, that the approximation error $\epsilon \to 0$ over a compact Hausdroff space. However, in practice this is not possible as these adaptation laws are based upon the assumption that all the parameters of the network can be optimized simultaneously. The global nature of the continuous map, $\mathbf{g}(.)$, can lead to globally-optimal network parameters which adequately minimize the approximation error but not to desired level. An alternative to global learning is local learning using local weight functions. The local learning algorithms involve estimation of network parameters

using the observations in the local neighborhood of the operating point. Generally, the sizing of the local neighborhood is dictated by the support or domain of the weight functions. In Chapter III, an approximation method is presented that enables a piecewise continuous approximation in a n-dimensional space using orthogonal polynomials and specially designed weight functions for overlapping the approximations in contiguous overlapping local regions to obtain the desired order of global continuity. Further, in Chapter IV, we have shown that the introduction of local models and averaging of different local approximations improves the approximation accuracy for a continuous map. In the next section, those results will be extended for the dynamical system identification case so that the approximation error ϵ can be significantly reduced. The adaptive nature of this approximation approach can essentially guarantee a small ϵ, if low noise measurement density in space and time is available.

a. Adaption Law Derivation Using The GLO-MAP Network

As discussed in Chapter III, the main idea of the GLO-MAP algorithm is a weighting function technique that generates a global family of overlapping preliminary approximations whose centroids of validity lie on the vertices of an n-dimensional grid, with vertices separated by a uniform step h. These preliminary approximations are constructed so they represent the behavior in local hypercubes with a volume $(2h)^n$ centered on a typical vertex in the grid. A novel averaging process is developed in Ref. [41, 43] to determine a piecewise continuous global family of local least squares approximations, while having the freedom to vary the nature (e.g., degrees of freedom) of the local approximations. The continuity conditions are enforced by using a unique set of weighting functions in the averaging process. The weight functions are designed to guarantee the global continuity conditions while retaining near complete freedom on the selection of the generating local approximations.

In Fig. 12, several qualitative observations regarding the weighting function approach are illustrated. One critical attractive property of the weight functions is that they add to unity everywhere in the overlapping unit region, i.e., they form a partition of unity. Notice further that the weight functions have a qualitative bell shape, but fairing into a square base, the zero contour being the boundary opposite (e.g., 2-3-4) to the vertex (e.g., point 1) where the weight has a unit value. Furthermore, notice that along any boundary, only the two weight functions associated with the two approximations centered at the end points of that boundary are non-zero along that boundary, while the other two weight functions are zero (the partial derivatives of the other two weight functions are also zero along this boundaries). These continuity arguments on the averaged approximation of the function can be extended readily to corresponding properties on their partial derivatives: The averaged approximation osculate in value and partial derivatives with the four preliminary approximations at their corresponding vertices, and the function and both partial derivatives along any boundary are a weighted average of the corresponding two functions associated with the end point of that boundary and their partial derivatives are likewise an average of the partial derivatives of the functions at the end point of that boundary. Collectively, these observations lead to rigorous piecewise continuity of the averaged approximations, while leaving the user free to choose any Chapter III, these qualitative observations are developed systematically and extended rigorously to approximation with arbitrary order continuity in an n dimensional space. In general, the final approximation in any hypercube is obtained by averaging 2^n overlapping approximations centered at the vertices of that local hypercube.

To illustrate this approach let us first assume $n = 2$ and $g(x_1, x_2) : \mathcal{R}^2 \rightarrow \mathcal{R}^2$ is a continuous function which can be approximated by the GLO-MAP process according

to the following equation:

$$\begin{aligned}\mathbf{g}(x_1,x_2) &= w_{0,0}(x_1^{I_1},x_2^{I_2})\mathbf{g}_{I_1,I_2}(x_1,x_2)+w_{0,1}(x_1^{I_1},x_2^{I_2+1})\mathbf{g}_{I_1,I_2+1}(x_1,x_2)+\cdots \\ &\quad w_{1,0}(x_1^{I_1+1},x_2^{I_2})\mathbf{g}_{I_1+1,I_2}(x_1,x_2)+w_{1,1}(x_1^{I_1+1},x_2^{I_2+1})\mathbf{g}_{I_1+1,I_2+1}(x_1,x_2) \\ &= \underbrace{\begin{bmatrix}\mathbf{g}_{I_1,I_2}(.) & \cdots & \mathbf{g}_{I_1+1,I_2+1}(.)\end{bmatrix}}_{\mathbf{F}_{2\times 4}} \underbrace{\begin{Bmatrix} w_{0,0}(x_1^{I_1},x_2^{I_2}) \\ w_{0,1}(x_1^{I_1},x_2^{I_2+1}) \\ w_{1,0}(x_1^{I_1+1},x_2^{I_2}) \\ w_{1,1}(x_1^{I_1+1},x_2^{I_2+1})\end{Bmatrix}}_{\mathbf{W}_{4\times 1}}\end{aligned} \tag{5.42}$$

where $x_i^I = \frac{x_i - X_i^I}{h}$ is a local coordinate and X_i^I denotes grid point coordinates. Also, the weight function are chosen such that these functions form a partition of unity so that they satisfy:

$$\sum_{i_1=0}^{1}\sum_{i_2=0}^{1} w_{i_1 i_2}(\,{}^{I_1+i_1}x_1,{}^{I_2+i_2}x_N) = 1 \tag{5.43}$$

Further, the local approximations, $\mathbf{f}_{I_1,I_2}(x_1,x_2)$, can be approximated by a set of orthogonal basis functions, $\mathbf{\Phi}$ as follows:

$$\mathbf{g}_{I_1,I_2}(x_1,x_2) = \mathbf{c}_{I_1,I_2}\boldsymbol{\phi}(x_1^{I_1},x_2^{I_2}) \tag{5.44}$$

Now, making use of Eq. (5.44) the matrix $\mathbf{F}$ in Eq. (5.42) can be rewritten as:

$$\begin{aligned}\mathbf{F} &= \underbrace{\begin{bmatrix}\mathbf{c}_{I_1,I_2} & \cdots & \mathbf{c}_{I_1+1,I_2+1}\end{bmatrix}}_{\mathbf{C}_{2\times 4N}} \\ &\quad \underbrace{\begin{bmatrix} \boldsymbol{\phi}(x_1^{I_1},x_2^{I_2}) & O_{N\times 1} & \cdots & O_{N\times 1} \\ O_{N\times 1} & \boldsymbol{\phi}(x_1^{I_1},x_2^{I_2+1}) & O_{N\times 1} & \vdots \\ \vdots & O_{N\times 1} & \boldsymbol{\phi}(x_1^{I_1+1},x_2^{I_2}) & O_{N\times 1} \\ O_{N\times 1} & O_{N\times 1} & O_{N\times 1} & \boldsymbol{\phi}(x_1^{I_1+1},x_2^{I_2+1})\end{bmatrix}}_{\mathbf{\Phi}(.)_{4N\times 4}}\end{aligned} \tag{5.45}$$

So, Eq. (5.42) reduces to:

$$\mathbf{g}(x_1, x_2) = \mathbf{C}\mathbf{\Phi}(.)\mathbf{W} \tag{5.46}$$

Now, using the approximation for $\mathbf{g}(.)$ given by Eq. (5.46), Eq. (5.23) reduces to:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_l\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{C}\underbrace{\mathbf{\Phi}(.)\mathbf{W}}_{\mathbf{\Psi}(.)} + \epsilon \tag{5.47}$$

Once again, the Fourier coefficient matrix $\mathbf{C}$ and control effectiveness matrix $\mathbf{B}$ are unknown and one can write:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}_l\hat{\mathbf{x}}(t) + \hat{\mathbf{B}}\mathbf{u}(t) + \hat{\mathbf{C}}\mathbf{\Psi}(.) \tag{5.48}$$

Let us define $\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ and time derivative of $\mathbf{e}(t)$ can be written as:

$$\begin{aligned} \dot{\mathbf{e}}(t) &= \mathbf{A}_l\mathbf{e} + (\mathbf{B} - \hat{\mathbf{B}})\mathbf{u}(t) + (\mathbf{C} - \hat{\mathbf{C}})\mathbf{\Psi}(.) + \epsilon & (5.49)\\ &= \mathbf{A}_l\mathbf{e} + \tilde{\mathbf{B}}\mathbf{u}(t) + \tilde{\mathbf{C}}\mathbf{\Psi}(.) + \epsilon & (5.50) \end{aligned}$$

Now to find adaptation laws for unknown parameters, let us consider following Lyapunov function:

$$V = \frac{1}{2}\mathbf{e}^T\mathbf{P}\mathbf{e} + \frac{1}{2}Tr(\tilde{\mathbf{B}}\Gamma_1\tilde{\mathbf{B}}^T) + \frac{1}{2}Tr(\tilde{\mathbf{C}}\Gamma_2\tilde{\mathbf{C}}^T) \tag{5.51}$$

where, $\mathbf{P}$ is a positive definite symmetric matrix. Now taking time derivative of V leads to following Eq.:

$$\begin{aligned} \dot{V} &= \frac{1}{2}\mathbf{e}^T\underbrace{\left(\mathbf{P}\mathbf{A}_l + \mathbf{A}_l^T\mathbf{P}\right)}_{-\mathbf{Q}}\mathbf{e} + \mathbf{e}^T\mathbf{P}\left(\tilde{\mathbf{B}}\mathbf{u}(t) + \tilde{\mathbf{C}}\mathbf{\Psi}(.) + \epsilon\right) + Tr\left(\tilde{\mathbf{B}}\Gamma_1\dot{\tilde{\mathbf{B}}}^T\right)\\ &\quad + Tr\left(\tilde{\mathbf{C}}\Gamma_2\dot{\tilde{\mathbf{C}}}^T\right)\\ &= -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} + Tr\left(\tilde{\mathbf{B}}\left[\Gamma_1\dot{\tilde{\mathbf{B}}}^T + \mathbf{u}\mathbf{e}^T\mathbf{P}\right]\right) + Tr\left(\tilde{\mathbf{C}}\left[\Gamma_2\dot{\tilde{\mathbf{C}}}^T + \mathbf{\Psi}(.)\mathbf{e}^T\mathbf{P}\right]\right)\\ &\quad + \mathbf{e}^T\mathbf{P}\epsilon \end{aligned} \tag{5.52}$$

Therefore, if following adaptation laws are chosen for $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$,

$$\dot{\tilde{\mathbf{B}}}^T = -\Gamma_1^{-1}\mathbf{u}\mathbf{e}^T\mathbf{P} \tag{5.53}$$

$$\dot{\tilde{\mathbf{C}}}^T = -\Gamma_2^{-1}\boldsymbol{\Psi}(\mathbf{x})\mathbf{e}^T\mathbf{P} \tag{5.54}$$

then $\dot{V}$ reduces to:

$$\dot{V} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}\mathbf{e} + \epsilon^T\mathbf{P}\mathbf{e} \tag{5.55}$$

$$\Rightarrow \dot{V} \leq -\frac{1}{2}|\lambda_{min}(\mathbf{Q})|||\mathbf{e}||^2 + \|\epsilon\|\|\mathbf{P}\|\|\mathbf{e}\| \tag{5.56}$$

Therefore, $\dot{V}$ is negative definite if $\|\epsilon\| \leq \frac{|\lambda_{min}(\mathbf{Q})|\|\mathbf{e}\|}{2\|\mathbf{P}\|}$. The bounded stability of the tracking residual $\mathbf{e}$ follows from the same arguments as outlined in the last section.

The adaptation laws presented in this chapter do not guarantee the convergence of the unknown control effectiveness matrices $\mathbf{B}$ and Fourier coefficients $\mathbf{C}$ to their true values but ensure that the parameter estimation errors are bounded. The convergence of unknown parameters to their true value can only be guaranteed by satisfying the persistence of excitation conditions [70].

The generalization of Eq. (5.42) is:

$$\begin{aligned}\mathbf{g}(X_1, \cdots, X_N) = \sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_N=0}^{1}\big(w_{i_1,\cdots,i_N}({}^{I_1+i_1}x_1, \cdots, {}^{I_N+i_N}x_N) \\ \mathbf{g}_{I_1+i_1,\cdots,I_N+i_N}(X_1, \cdots, X_N)\big)\end{aligned} \tag{5.57}$$

However, the expression for the adaptation laws for the Fourier coefficients and control effectiveness matrix remains same except that now matrix $\mathbf{C}$ in Eq. (5.54) consists of the coefficients of 2^n neighboring approximations depending upon the value of $\mathbf{x}$.

Finally, we mention that the state vector, $\mathbf{x}$ is generally unknown so the best available estimates of state vector, $\hat{\mathbf{x}}$ are used as state measurements. These estimates can be obtained by using the Kalman filter algorithm along with the best known

linear system, as discussed in section C. The convergence of the nonlinear system identification algorithm, SysID 1 can be proved under reasonable set of assumptions, which are captured in the following theorem:

Theorem 5. *If the linear system described by Eqs. (5.6) and (5.7) is fully observable and tuning parameters* $\mathbf{P}$ *and* $\mathbf{Q}$ *are chosen in such a way that* $\dot{V}$ *described by Eq. (5.55) is negative definite then* $\|\hat{\mathbf{y}} - \mathbf{y}\| \rightarrow \|\mathbf{C}_l(\epsilon_1 + e_{lb})\|$. *Where,* ϵ_1 *represents the hidden state estimation accuracy and* $e_{lb} = 2\frac{\|\epsilon\|\|\mathbf{P}\|}{|\lambda_{min}(\mathbf{Q})|}$.

Proof. The observability of the identified linear dynamic system guarantees that the hidden state $\mathbf{x}$ can be detected from the given output $\mathbf{y}$. In other words, the Kalman filter estimate $\hat{\mathbf{x}}_b$ can be obtained in such a way that

$$\|\mathbf{x} - \hat{\mathbf{x}}_b\| = \epsilon_1 \tag{5.58}$$

Now, we just need to show that $\hat{\mathbf{x}}$ asymptotically converges to $\hat{\mathbf{x}}_b$. Eq. (5.55) tells us the important qualitative truth: if the system can be modeled as we have modeled it in Eq. (5.47), then asymptotic convergence is assured. However, in presence of modeling errors the tracking errors will converge to the following value, as discussed in the previous section:

$$||\mathbf{e}|| \rightarrow 2\frac{\|\epsilon\|\|\mathbf{P}\|}{|\lambda_{min}(\mathbf{Q})|} = e_{lb} \tag{5.59}$$

This means that state identification error $\mathbf{e}$ is always bounded by e_{lb} which further implies that

$$\begin{aligned} \|\mathbf{y} - \hat{\mathbf{y}}\| &= \|\mathbf{C}_l(\mathbf{x} - \hat{\mathbf{x}})\| \\ &= \|\mathbf{C}_l(\underbrace{\mathbf{x} - \hat{\mathbf{x}}_b}_{\epsilon_1} + \hat{\mathbf{x}}_b - \hat{\mathbf{x}})\| \\ &\rightarrow \|\mathbf{C}_l(\epsilon_1 + e_{lb})\| \end{aligned}$$

□

The above theorem gives us the lower bound for the system identification errors which can be reduced to a desired tolerance by the judicious selection of various tuning parameters.

2. Learning Algorithm for SysID 2

In the previous section, we have described the learning algorithm for the nonlinear system identification algorithm, SysID 1. The performance of the SysID 1 depends upon the dimensionality of the hidden state vector $\mathbf{x}$. As the dimensionality of state vector $\mathbf{x}$ increases the number of terms required to approximate nonlinear function $\mathbf{g}(.)$ in Eq. (5.23) increases exponentially. For example, total 6 basis functions are required for second order approximation of $\mathbf{g}(.)$ in 2-D while the same number shoots to 66 for 10-D state vector. In case of the GLO-MAP approximation this number can rise even more depending upon the number of local approximations involved. This unexpected increase in number of parameters makes SysID 1 undesirable for the identification of structural mechanics systems where the number of participating modes runs to 10-20.

In section C, we have described an alternate approach (SysID 2) to tackle the issue of dimensionality of the state vector $\mathbf{x}$. The overall architecture of the system identification algorithm, SysID 2, is depicted in Fig. 41. Once again, the basic idea of SysID 2 is to split the system identification process into linear and nonlinear identification processes. The linear system identification process is same as in the case of SysID 1 and the main difference lies in the nonlinear identification process. In SysID 1, the linear dynamical model is perturbed by nonlinear term $\mathbf{g}(.)$ to compensate for error arising due to system nonlinearities whereas in SysID 2 the nonlinear transformation of best estimates of output vector is suggested to compensate for unknown system nonlinearity effects.

Consider the perturbed linear dynamical model, where $\mathbf{h}(\mathbf{x}_p)$ is a vector of nonlinear terms.

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}_l\mathbf{x} + \mathbf{B}\mathbf{u} \tag{5.60}$$

$$\mathbf{y} = \hat{\mathbf{y}}_l + \mathbf{h}(\mathbf{x}_p) \tag{5.61}$$

with

$$\hat{\mathbf{y}}_l = \mathbf{C}_l\hat{\mathbf{x}} + \mathbf{D}_l\mathbf{u} \tag{5.62}$$

Here, $\mathbf{A}_l \in \mathcal{R}^{n\times n}$ is a known Hurwitz matrix and $\mathbf{B} \in \mathcal{R}^{n\times p}$ is a control effectiveness matrix. Once again, we mention that the matrix $\mathbf{A}_l$ is chosen in such a way that it captures the modal frequencies of the interest and can be obtained by the ERA or any other linear system identification algorithm as described in section C. The vector $\mathbf{x}_p \in \mathcal{R}^s$ is a dummy variable which can be chosen as a physical variable associated with the problem in hand not necessarily be same as hidden state vector $\mathbf{x}_l$. For example, in case of the modeling of flexible space structure, the system output vector consists of surface distortion measurements at various spatial points, therefore, the dummy variable $\mathbf{x}_p$ can consist of cartesian coordinates (x, y, z). In other words, the surface distortions can be modeled as a function of cartesian coordinates.

The time history estimates of the estimated linear output vector $\hat{\mathbf{y}}_l$ can be obtained by using the procedure described in section C, so the system identification problem can be re-defined as:

System Identification Problem. *Given the time history estimates of the state vector* $\mathbf{x}(t)$ *and control variable,* $\mathbf{u}(t)$*, find estimates of the unknown nonlinearity vector* $\mathbf{h}(.)$*.*

Now, due to obvious reasons, discussed in Chapters III and IV, we use the GLO-

MAP algorithm to approximate $\mathbf{h}(.)$:

$$\begin{aligned}\mathbf{h}(X_1,\cdots,X_s) &= \sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_s=0}^{1}\big(w_{i_1,\cdots,i_s}({}^{I_1+i_1}x_1,\cdots,{}^{I_s+i_s}x_s)\\ &\quad \mathbf{h}_{I_1+i_1,\cdots,I_s+i_s}(X_1,\cdots,X_s)\big)\end{aligned} \tag{5.63}$$

where $x_i^I = \frac{x_i - X_i^I}{h}$ is a local coordinate and X_i^I denotes grid point coordinates. Also, the weight function are chosen such that these functions are a partition of unity so that they satisfy:

$$\sum_{i_1=0}^{1}\sum_{i_2=0}^{1}\cdots\sum_{i_s=0}^{1} w_{i_1,\cdots,i_s}({}^{I_1+i_1}x_1,\cdots,{}^{I_s+i_s}x_s) = 1 \tag{5.64}$$

Further, the local approximations, $\mathbf{h}_{I_1+i_1,\cdots,I_s+i_s}(.)$, can be approximated by a set of orthogonal basis functions, $\mathbf{\Phi}$ as discussed in Chapter III.

Finally, we mention that the convergence of SysID 2 follows form the guaranteed convergence of Kalman filter and the GLO-MAP algorithm. Further, the system identification error can be reduced to a desired tolerance by the judicious selection of number of local approximations and degree of basis functions for each local approximation.

E. Numerical Simulation

The proposed nonlinear system identification algorithms are tested on a variety of test cases mainly concerned with large space structures. In this section, some results from these studies are presented.

1. Dynamic System Identification of Large Space Antenna

Space Based Radar (SBR) systems envisioned for the future may be a constellation of spacecraft that provide persistent real-time radar images of the Earth environment

through the identification and tracking of moving targets, high-resolution synthetic aperture radar imaging, and collection of high-resolution terrain information. The accuracy of the information obtained from the SBR system depend upon many parameters like the geometric shape of the antenna, permittivities of the media through which radar wave is traveling, etc. and our ability to compensate in real-time implicitly depends on the accuracy of system identification. Therefore the characteristics of the scattered wave received by the SBR antenna for a given frequency depend on the surface and geometric parameters of the radar. To apply necessary corrections for scattering of radar waves, the precise knowledge of the SBR antenna becomes a necessity. However, the transient excitation of the flexible dynamics mode necessitated by the need to slew the antenna makes the shape estimation problem more difficult. While a variety of surface models can be employed to model the instantaneous shape, we consider the case that the surface is measured at discrete points and a dynamical model for shape estimation is desired. The objective of this section is to apply the system identification methodologies, developed in this chapter, to estimate the real time SBR antenna shape using only the discrete time measurements of the antenna surface.

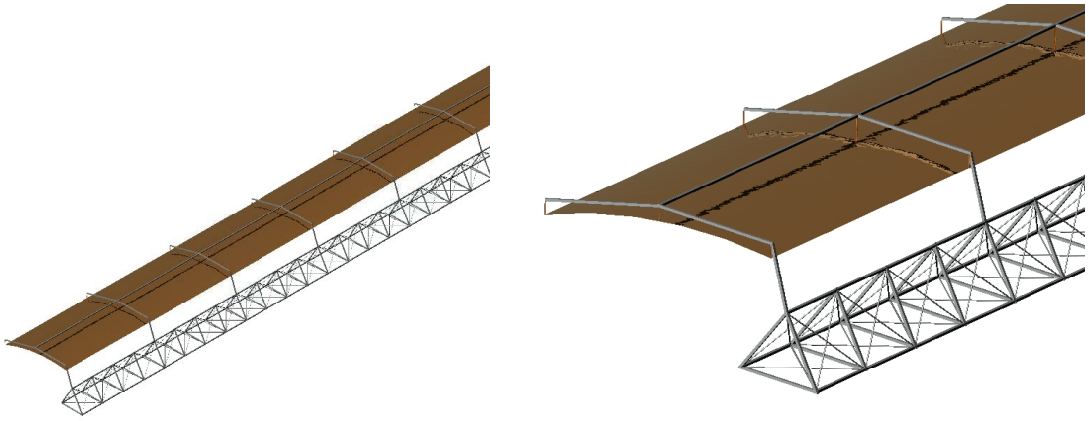
For simulation purposes the SBR antenna geometry is modeled in NASTRAN [56]. The antenna model consists of total 7 panels as shown in Fig. 42. Each panel is assumed to be $100m$ long in length and $200 \times 250m^2$ in area. It is assumed that shape deflections measurements are available at uniformly distributed 1500 spatial points at each time instant. NASTRAN is used to generate mass, $\mathbf{M}$, and stiffness, $\mathbf{K}$, matrices for the antenna structure and coordinate transformation matrix, $\mathbf{T}$, to transform the

modal coordinates to physical coordinates i.e. deflections along each axis.

$$\text{Modal Equations: } \mathbf{M}\ddot{\boldsymbol{\eta}} + \mathbf{K}\boldsymbol{\eta} = 0 \tag{5.65}$$

$$\text{Transformation to Physical Coord.: } \mathbf{y} = \mathbf{T}\boldsymbol{\eta} \tag{5.66}$$

where, $\boldsymbol{\eta}$ and $\mathbf{y}$ represent modal and physical coordinates respectively. The order of the FEM model was $1500 \times 3 = 4500$. If one tries to use traditional ANN method to find a continuous map between system output and input space then the order of such a model will be equal to the order of FEM model i.e. 4500 (which is not desirable in terms of computational efficiency!) However, order reduction methods can be used to reduce the dimension of the model state space to 10-30. These equations, augmented with artificial damping and nonlinearities, are simulated using the MATLAB [57] environment to generate the measurement data for 50 seconds at $10Hz$ frequency. For the purpose of this chapter, radial basis functions are used to simulate artificial nonlinearity with random magnitude and center.



(a) NASTRAN SBR antenna model consists of 7 panels

(b) Close-up of one panel

Fig. 42. NASTRAN model of the SBR antenna.

To test the effectiveness of both the system identification algorithm, we consider two test cases. In the first test case, the measurement data is generated by exciting first two modes while in the second test case, first five modes are excited to generate measurement data. Now, according to the procedure listed in section C, first, the ERA algorithm is used to generate linear dynamic model for the SBR antenna model. As expected, the ERA system gives us 4^{th} and 10^{th} order linear dynamic model for first and second test case, respectively. Finally, the nonlinear system identification algorithms are used to refine the linear model learned by the ERA algorithm.

Fig. 43(a) shows some of the true simulated measurements for various points on the antenna surface corresponding to the first test case and Fig. 43(b) shows the relative output error plots corresponding to the ERA identified model. We mention that relative output error e_y is defined as below:

$$e_y = \frac{\|\mathbf{y}_{True} - \mathbf{y}_{Est}\|}{\|\mathbf{y}_{True}\|} \tag{5.67}$$

From these plots, it is clear that although ERA is able to capture the 2 modes of interest, there is significant error in estimating the true nonlinear output.

To model the effects of nonlinearities involved in the true dynamic model, the SysID-1 algorithm is employed according to the procedure listed in section a. Only one element is used to grid the estimated modal data according to Eq. (3.15) giving rise to total 16 local approximations. The orthogonal polynomial functions used to model the nonlinear function, $\mathbf{g}(.)$ are listed in Table XII.

$$\mathbf{g}_{I_1}(\mathbf{x}_l) = \mathbf{C}^T \mathbf{\Phi}(\mathbf{x}_l) \tag{5.68}$$

where, $\mathbf{x}_l \in \mathcal{R}^4$ consists of the ERA identified modal coordinates. We mention that vector $\mathbf{\Phi}$ consists of only second and higher order terms in $\mathbf{x}_l$ to have the same linear dynamics as identified by the ERA algorithm. Therefore, $\mathbf{\Phi} \in \mathcal{R}^{10}$ and $\mathbf{C}$ is a 4×10

matrix of unknown Fourier coefficients. As total 16 local approximations are used to identify nonlinear function $\mathbf{g}(.)$, therefore, total $16 \times 10 = 160$ Fourier coefficients are required to be estimated. Initially, $\mathbf{C}$ is assumed to be a zero matrix and is adapted by using Eq. (5.54). Fig. 43(c) shows the adaptation plot of some of the Fourier coefficients.

In case of the SysID 2 algorithm, the unknown nonlinear function $\mathbf{h}(.)$ is approximated by the GLO-MAP algorithm and dummy variable $\mathbf{x}_p$ is assumed to consist of cartesian coordinates (x, y, z). To approximate the SBR antenna shape at a particular time, the measurement data is modeled using a total of 64 finite element cells 4 along each cartesian coordinates, X, Y and Z. Now, a continuous approximation, of SBR antenna shape, for a particular cell is generated via a least-square procedure as listed in Chapter III

$$\hat{x}(x_l, y_l, z_l, t) = \sum_i \sum_j \sum_k a_{ijk} \phi_i(x_l) \phi_j(y_l) \phi_k(z_l), \; i + j + k \leq 2 \tag{5.69}$$

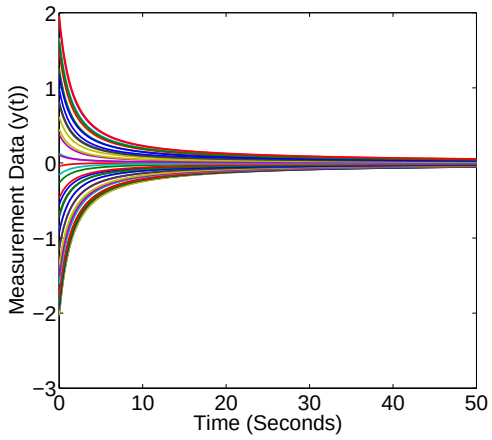
$$\hat{y}(x_l, y_l, z_l, t) = \sum_l \sum_m \sum_n b_{lmn} \phi_l(x_l) \phi_m(y_l) \phi_n(z_l), \; l + m + n \leq 2 \tag{5.70}$$

$$\hat{z}(x_l, y_l, z_l, t) = \sum_p \sum_q \sum_r c_{pqr} \phi_l(x_l) \phi_m(y_l) \phi_n(z_l), \; l + m + n \leq 2 \tag{5.71}$$

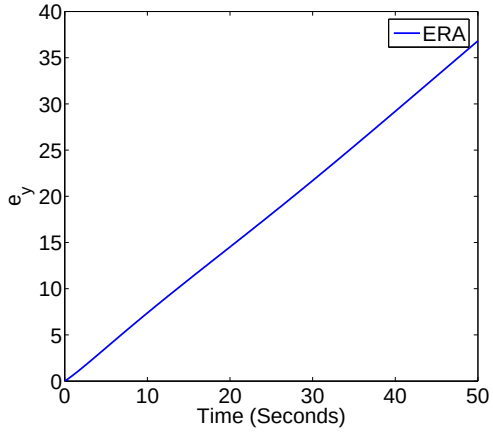
Here, (x_l, y_l, z_l) denote the local cartesian coordinates of a point predicted by using linear system identified by ERA algorithm. To learn the local approximations at each time, vision sensor measurements are processed sequentially. Initially, all Fourier coefficients are assumed to be zero and the corresponding covariance matrix initialized to 10^6 times identity matrix.

Fig. 43(d) shows the relative output error plots for both the system identification algorithms. From these plots, it is clear that the use of nonlinear system identification algorithm reduces the estimation error by at least two orders of magnitude.

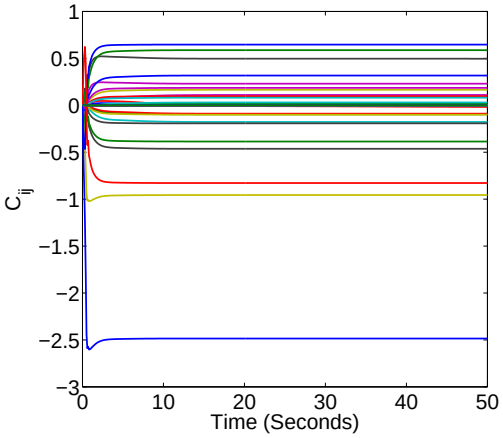
Further, Fig. 44(a) shows some of the true simulated measurements for various



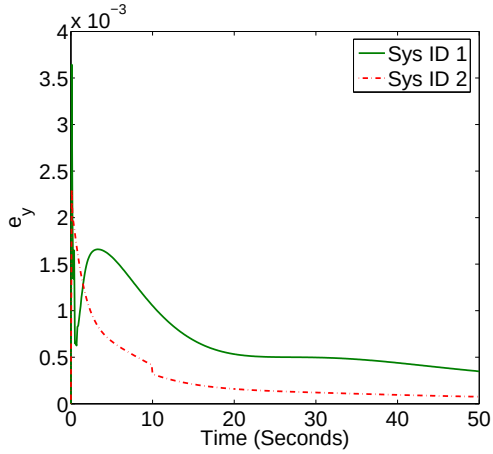
(a) Measured system output vector



(b) Relative error between the measured and the ERA estimated system outputs

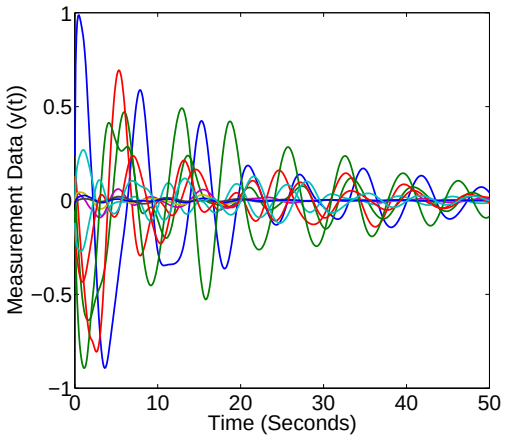


(c) Time history of various Fourier coefficients in case of the SysID 1 algorithm

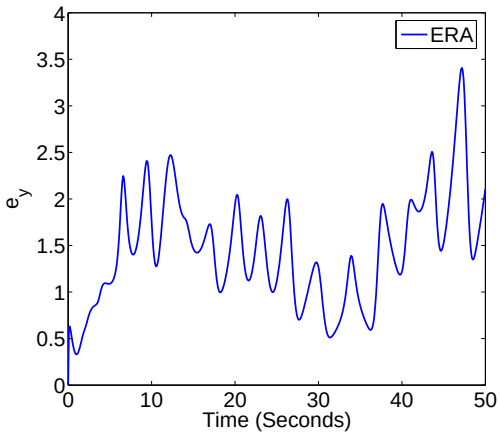


(d) Relative error between the measured and the identified nonlinear system outputs

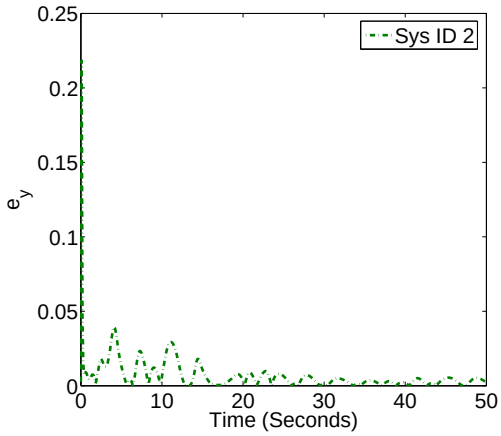
Fig. 43. Non-linear system identification results for the test case 1.



(a) Measured system output vector



(b) Relative error between the measured and the ERA estimated system outputs



(c) Relative error between the measured and the identified nonlinear system outputs

Fig. 44. Non-linear system identification results for the test case 2.

points on the antenna surface corresponding to the second test case and Fig. 44(b) shows the relative output error plots corresponding to the ERA identified model. From these plots, once again it is clear that although ERA is able to capture all 5 modes of interest, there is significant error in estimating the true nonlinear output.

As the dimension of linear state vector is 10, therefore, one needs total $2^{10} \times 55 = 56,320$ Fourier coefficients to learn the nonlinear function $\mathbf{g}(.)$ using only one grid element. This high increase in number of Fourier coefficients makes the use of SysID 1 highly inefficient. Therefore, in the second test case, only SysID 2 algorithm is used to refine the linear model learned by the ERA algorithm. Once again, the unknown nonlinear function $\mathbf{h}(.)$ is approximated by the GLO-MAP algorithm and dummy variable $\mathbf{x}_p$ is assumed to consist of cartesian coordinates (x, y, z). To approximate the SBR antenna shape at a particular time, the measurement data is modeled using a total of 64 finite element cells 4 along each cartesian coordinates, X, Y and Z.

Fig. 44(c) shows the relative output error plots for the SysID 2 system identification algorithms. Once again, the use of nonlinear system identification algorithm reduces the estimation error by at least two order of magnitude.

For the sake of simulations, it is implicitly assumed that actual surface deformation is sufficiently smooth and relatively sparse set of measurements can provide support for the needed surface estimates. Of course, validating this assumption will be crucial in real applications and we did not attempt this. Finally, we mention that the simulation results present in this section provides a basis for optimism regarding the utility of both the algorithm. However, the effect of measurement data frequency and sensor noise needs to be considered before making strong conclusion about the utility of these algorithms.

F. Concluding Remarks

A general methodology for non-linear system identification is presented in this chapter. The method splits the nonlinear system identification process into two parts: 1)Linear system identification using ERA and 2) Nonlinear system identification using the GLO-MAP. We use the ERA - determined linear system state variable transformation to reduce nonlinear system state space dimensionality. We have found this to work well in the examples we have studied but there is no theoretical guarantee that this approach to order reduction for a general nonlinear system will give the best order reduction. The GLO-MAP algorithm is used to learn the nonlinear correction term in hidden state dynamical model and nonlinear transformation of measurement data for SysID 1 and SysID 2, respectively. A particularly attractive choice to model the nonlinear term is shown to be polynomial basis functions that are orthogonal with respect to the weight functions of the averaging process of the GLO-MAP algorithm. The adaption laws for different parameters of the GLO-MAP network are derived by using Lyapunov's analysis in case of the SysID 1. The convergence of both the algorithms is supported by a thorough analysis and demonstrated in the numerical study. The broad generality of the method, together with simulation results provide a strong basis for optimism for the practical importance of these ideas. However, there remains an issue of uniqueness of the learned mathematical model, but again, we do not believe these open theoretical questions limit the usefulness of this approach for most engineering problems. We simply point out that proceeding with caution in the absence of theoretical justification is a necessary leap of faith until more theoretical progress can be made.

CHAPTER VI

MESHLESS FINITE ELEMENT METHODS

A. Introduction

The classical Finite Element Method (FEM) is a very promising approach to find the solution of Partial Differential Equations (PDE). In the classical FEM, the approximate solution to PDE is obtained by the discretization of the spatial domain into volume/area elements. The local approximations for each element are obtained by the use of polynomial basis functions which interpolate the solution and satisfy additional constraints like exact interpolation at nodal points and inter-element continuity conditions. Generally, the polynomial degree p is fixed which is dictated by the element type and the number of nodal points per element. The success of classical FEM depends upon the approximation ability of the polynomial basis functions and further improvement in the approximated solution can be achieved only by refining the mesh size, h. In most of the cases the degree of these basis functions is less than or equal to 2. For example, in case of the triangular mesh with three nodes per element one can only use degree one polynomials for interpolation to satisfy necessary continuity requirements. Further, in some problems the use of non-polynomial basis functions may be desirable to achieve better accuracy. For example, in case of the Helmholtz equation, the solution is known to be highly oscillatory in nature and therefore, it may be desirable to use non-polynomial basis functions to approximate the exact solution. However, even though the analytical knowledge about local behavior of exact solution is available there are no convenient means to incorporate this knowledge in the conventional FEM solution. Also, the reliance of the conventional FEM on a mesh is not well suited to problems involving discontinuities and moving domain.

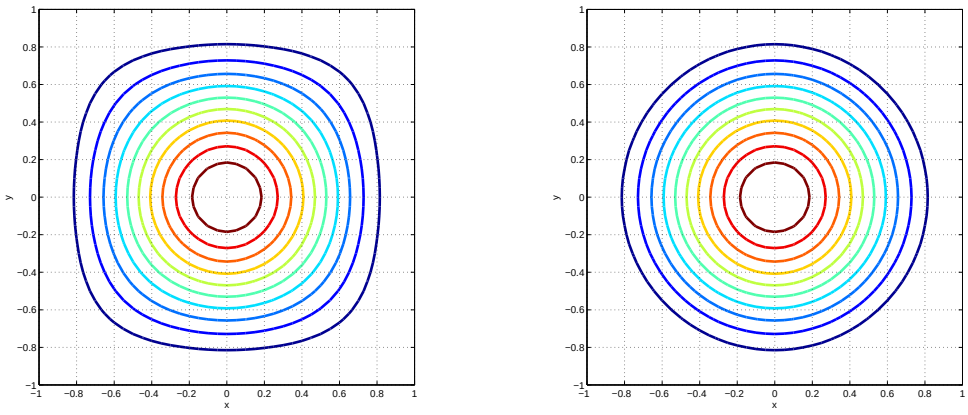
Generally, to deal with moving domains and discontinuities in the conventional FEM methods, the original mesh is regenerated in each step of the evolution so that mesh lines are in accordance with moving domain and discontinuity. However, this strategy of re-meshing at each stage can introduce numerous difficulties such as the need to project the solution between meshes in successive stages of the problem, complexity in the computer program and not to mention the computational burden associated with a large number of re-meshing.

The main objective of the meshless methods is to construct the PDE solution entirely in terms of nodes in the absence of element connectivity. In the meshless FEM, nodal points can be added easily to the part of the domain where the solution is (expected to be) poor. In addition, since meshless methods use a non-element interpolation technique and a functional basis, therefore, the solution and its derivatives may be found directly where they are needed without interpolation errors. This makes the meshless FEM more flexible than the conventional FEM and a powerful tool to solve large classes of problems which are very awkward with mesh based conventional FEM.

Although the meshless methods are greatly developed recently, the research efforts in this field have a long history [44,81]. The main proposals which follow meshless FEM concepts are the Smoothed Particle Hydrodynamics Method (SPH), [81], the Element Free Galerkin Method (EFGM) [82],the Reproducing Kernel Particle (RKP) method [83], the Meshless Petrov Galerkin (MLPG) method [84], the Partition of Unity Finite Element Method (PUFEM) [85] and the hp-cloud method [86]. An overview of various meshless methods can be found in Ref. [87]. Most of these methods except the PUFEM make use of the Moving Least Squares Approximation (MLSA) technique to find the expression for different local approximations. For the MLSA based approach, the local shape functions are constructed with the help of

methods from data fitting followed by the Galerkin discretization process to set up a linear system of equations. Finally, these systems of equations are solved for the solution value at specified nodal points. We mention that although successes have been many by using the MLSA based meshless methods but there are many drawbacks of such an approach not to mention the computational cost associated with these methods. Alternative to the MLSA is a *partition of unity* approach where the Galerkin discretization process is directly used to find the local shape function instead of using some data fitting process followed by the Galerkin discretization. The main advantage of the PUFEM over the MLSA approach is that the PUFEM approach results in a continuous approximation of the exact solution while the MLSA just provides the solution value at specified nodal points and interpolation process is required to obtain the solution value at any point other than the specified nodal points.

A common feature of all meshless methods is a weight function which is used to define the domain of integration for a particular node. The specially designed weight functions are positive functions with compact support which dictates the domain of integration for a particular nodal point. We mention that the domain of integration defines the local region over which local weak form associated with particular node is valid and it is analogous to the element space in the conventional FEM. The most commonly used sub-domains are circular, rectangular or elliptical in shape. Depending upon the input argument to weight functions different shapes can be achieved for domain of integration. For example, in Fig. 45(a), a rectangular shape is obtained by tensor product of 1-D weight functions while in Fig. 45(b) circular shape is obtained by selecting the input argument of the 1-D weight function to be radial distance of the point from origin or nodal point in question. In case of the PUFEM, these weight functions needs to satisfy extra constraint of *partition of unity*. Notice that the GLOMAP weight functions (Table XI) are positive functions with compact support and



(a) $w(x,y) = w_1(x) \times w_1(y)$ (b) $w(x,y) = w_1(r),\ r = \sqrt{(x^2+y^2)}$

Fig. 45. Different shapes for domain of integration.

form a partition of unity. As a consequence of this, they can be used in various meshless methods. The main advantage of using the GLO-MAP weight functions is that they are polynomial in nature (whereas most PUFEM weight function are not simple polynomials) and further if one use the polynomial functions orthogonal to the GLO-MAP weight function to locally approximate the solution, then many numerical integrals can be evaluated accurately and easily.

In this chapter, attention is focused on the use of the GLO-MAP algorithm along with the Galerkin discretization process to solve PDEs in an efficient manner. Modifications of the standard MLPG and PUFEM methods are proposed using the GLO-MAP algorithm. We mention that the novel averaging process of the GLO-MAP algorithm differentiate it advantageously from their conventional counterparts.

The structure of this chapter is as follows: first, the MLSA based Meshless Petrov Galerkin method is described followed by the modification of this method using the GLO-MAP algorithm. Next, the use of the GLO-MAP algorithm is described in

context with the PUFEM approach. Finally, numerical studies are performed to compare the performance of various algorithms proposed in this chapter.

B. MLPG-Moving Least Square Based Approach

In this section the main characteristics of the Meshless Petrov Galerkin (MLPG) algorithm are discussed and one should refer to Refs. [84,88] for more detail discussion on this method.

Let us consider following linear PDE to be solved over global domain Ω with boundary Γ

$$\mathcal{L}u = f \tag{6.1}$$

and following boundary conditions

$$u = \bar{u} \text{ on } \Gamma_u \tag{6.2}$$

$$\nabla u.\hat{\mathbf{n}} = \bar{q} \text{ on } \Gamma_q \tag{6.3}$$

where $\mathcal{L}$ is the general differential operator, u is the unknown function to be solved and f is the forcing term. Further, Γ_u and Γ_q are parts of the global boundary Γ where Dirichlet and Neumann boundary conditions are imposed, respectively. Finally, $\hat{\mathbf{n}}$ is the outward normal vector.

Like in any FEM method, we approximate the unknown function u as $\hat{u}$ and write a generalized local weak form of Eq. (6.1) over a local sub-domain Ω_x:

$$\int_{\Omega_x} [\mathcal{L}\hat{u} - f]v_x d\Omega + \alpha \int_{\Gamma_{xu}} [\hat{u} - \bar{u}]v_x d\Gamma + \beta \int_{\Gamma_{xq}} [q - \bar{q}]v_x d\Gamma = 0 \tag{6.4}$$

where, v_x is the test function associated with nodal point $\mathbf{x}$ and has a compact support $\Omega_x \subset \Omega$ also known as the domain of integration associated with nodal point $\mathbf{x}$ as shown in Fig. 46. The choice of the test function v_x determines the shape and size

of the local domain Ω_x. Further, Γ_{xu} and Γ_{xq} are the boundary parts of the sub-domain Ω_x over which Dirichlet and Neumann boundary conditions are imposed i.e. $\Gamma_{xu} = \Gamma_x \cap \Gamma_u$ and $\Gamma_{xq} = \Gamma_x \cap \Gamma_q$. α and β are penalty parameters used to impose the Dirichlet and Neumann boundary conditions, respectively.

The approximation $\hat{u}$ of the unknown function, u, can be written as:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \psi_i(\mathbf{x})\hat{u}_i \tag{6.5}$$

where N is the total number of nodal points used to discretize the domain Ω, $\hat{u}_i$ is the solution value at the i^{th} nodal point and $\psi_i(.)$ is the shape function associated with the i^{th} nodal point with compact support Ω_i. The support Ω_i of the shape function ψ_i is known as the domain of definition in the literature [84] and is shown in Fig. 46. Finally, the substitution of Eq. (6.5) in Eq. (6.4) leads to the following set of linear equations in unknown variables $\hat{u}_i$:

$$\mathbf{K}\hat{\mathbf{u}} = \mathbf{f} \tag{6.6}$$

where, $\mathbf{K}$ and $\mathbf{f}$ are given as:

$$K_{ij} = \int_{\Omega_x} \mathcal{L}\psi_j(\mathbf{x}) v_{x_j} d\Omega + \alpha \int_{\Gamma_{xu}} \psi_j(\mathbf{x}) v_{x_i} d\Gamma + \beta \int_{\Gamma_{xq}} \nabla\psi_j(\mathbf{x}) v_{x_i} d\Gamma \tag{6.7}$$

$$f_i = \int_{\Omega_x} f v_{x_i} d\Omega + \alpha \int_{\Gamma_{xu}} \bar{u} v_{x_i} d\Gamma + \beta \int_{\Gamma_{xq}} \bar{q} v_{x_i} d\Gamma \tag{6.8}$$

To obtain the expression for the shape function $\psi_i(.)$, the Moving Least Square (MLS) fitting algorithm is adopted which is also known as the local regression algorithm in the literature. The moving least square approximation u^h of the unknown function u is written as:

$$u^h(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x})\mathbf{a}(\mathbf{x}) \tag{6.9}$$

where, $\boldsymbol{\phi} \in \mathcal{R}^m$ is a vector of the complete set of basis functions ϕ_i and $\mathbf{a} \in \mathcal{R}^m$ is a vector of corresponding Fourier coefficients a_i which are function of $\mathbf{x}$ instead of being constant as in the conventional Gaussian least square approach. Furthermore, a set of nodal points $\{\mathbf{x}_i\}_{i=1,\cdots,M}$ are considered in the neighborhood Ω_M of $\mathbf{x}$ and the coefficient vector $\mathbf{a}(\mathbf{x})$ is obtained by minimizing the mean square error.

$$J = \sum_{i=1}^{M} w_i(\mathbf{x}, \mathbf{x}_i) \left(\boldsymbol{\phi}^T(\mathbf{x}_i)\mathbf{a}(\mathbf{x}) - u_i^h\right)^2 \tag{6.10}$$

Here, u_i^h is the value of unknown function u at points $\mathbf{x}_i$ for which we want to solve. w_i is a weight function associated with i^{th} node such that $w_i(\mathbf{x}, \mathbf{x}_i) > 0$. Beside positivity the weight function w_i also satisfies following properties:

1. The domain Ω_i of weight function w_i is a compact sub-space of Ω.

2. w_i is a monotonically decreasing function in $\mathbf{x}$.

3. As $\mathbf{x} \to 0$, $w_i \to \delta$.

According to the definition of weight function w_i, the domain of definition, Ω_m, of a point $\mathbf{x}$ is defined as collection of points for which $w_i(\mathbf{x}, \mathbf{x}_i) \geq 0,\ i = 1, 2, \cdots, M$. In other words, Ω_M can be defined as union of sub-domains, Ω_i i.e. $\Omega_M = \bigcup_{i=1}^{M} \Omega_i$

Generally, a Gaussian weight function of the following form is used:

$$w_i(\mathbf{x}) = \begin{cases} \frac{e^{-(\|\mathbf{x}-\mathbf{x}_i\|/c)^2} - e^{-(r_i/c)^2}}{1-e^{-(r_i/c)^2}}, & \|\mathbf{x} - \mathbf{x}_i\| \leq r_i \\ 0, & \|\mathbf{x} - \mathbf{x}_i\| > r_i \end{cases} \tag{6.11}$$

where r_i and c are parameters which dictate the size of domain of definition, Ω_m. As another possibility, a spline weight function is used for the MLS approximation in

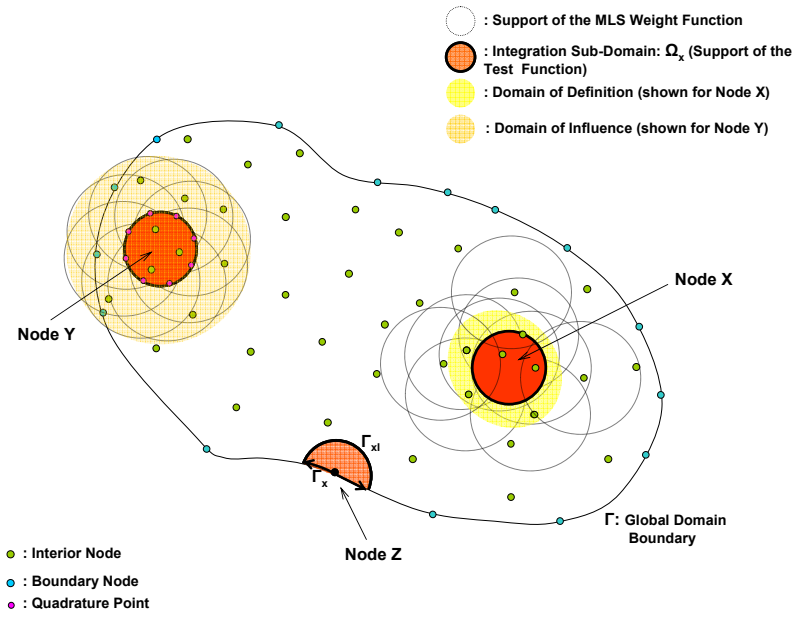


Fig. 46. Illustration of the domain of definition, domain of influence and domain of integration for the MLPG algorithm.

Ref. [84].

$$w_i(\mathbf{x}) = \begin{cases} 1 - 6\left(\frac{d_i}{r_i}\right)^2 + 8\left(\frac{d_i}{r_i}\right)^3 - 3\left(\frac{d_i}{r_i}\right)^4, & d_i = \|\mathbf{x} - \mathbf{x}_i\| \leq r_i \\ 0, & d_i > r_i \end{cases} \tag{6.12}$$

The first order optimality condition for the loss function of Eq. (6.10) results in the following set of linear equations for $\mathbf{a}(\mathbf{x})$

$$\mathbf{a}(\mathbf{x}) = (\underbrace{\mathbf{\Phi}^T\mathbf{W}(\mathbf{x})\mathbf{\Phi}}_{\mathbf{A}(\mathbf{x})})^{-1}\mathbf{\Phi}^T\mathbf{W}(\mathbf{x})\mathbf{u}^h \tag{6.13}$$

where $\mathbf{u}^h \in \mathcal{R}^M$ is a vector with entries u_i^h and the matrices $\mathbf{\Phi}$ and $\mathbf{W}$ are given by

$$\mathbf{\Phi}_{ij} = \phi_j(\mathbf{x}_i) \tag{6.14}$$

$$\mathbf{W}_{ij} = w_i(\mathbf{x})\delta_{ij} \tag{6.15}$$

The necessary condition for the MLS solution to exist is that the rank of the matrix $\mathbf{\Phi}$ should be at least m. As a consequence of this the domain of definition Ω_M should consist of at least m nodal points. Now, the MLS approximated solution u^h can also be expressed as:

$$u^h(\mathbf{x}) = \sum_{i=1}^{n} \psi_i(\mathbf{x})u_i \tag{6.16}$$

where the shape function ψ_i is given by

$$\psi_i(\mathbf{x}) = \begin{cases} \sum_{i=1}^{m} \phi_j(\mathbf{x})(\mathbf{A}^{-1}(\mathbf{x})\mathbf{\Phi}\mathbf{W}(\mathbf{x}))_{ji} & w_i(\mathbf{x}) > 0 \\ 0 & w_i(\mathbf{x}) = 0 \end{cases} \tag{6.17}$$

Note that the shape function $\psi_i(\mathbf{x})$ vanishes at the nodal points where weight function $w_i(\mathbf{x}) = 0$. The continuity of the shape function $\psi(.)$ depends upon the continuity of the weight function w_i and basis functions $\boldsymbol{\phi}(.)$. Generally, the basis functions are chosen as m^{th} order polynomial functions in $\mathbf{x}$, however, one has the freedom to

choose any set of basis functions depending upon the problem in hand. Note, for $m = 1$, the shape function $\psi(\mathbf{x})$ is given by the following expression and are known as the Shepard function:

$$\psi_i(\mathbf{x}) = \frac{w_i(\mathbf{x})}{\sum_{i=1}^{M} w_i(\mathbf{x})} \tag{6.18}$$

So the continuity of the Shepard function depends solely upon the continuity of the weight functions. In Table XI of Chapter III, weight functions are listed that guarantee arbitrary order continuity and satisfy all requirements of the MLS weight function. The weight functions for first four orders of continuity and their first derivatives, for 2-D approximations, are shown in Figs. 47, 48 and 49, respectively. The main advantage of using the GLO-MAP weight functions is that they are polynomial in nature and further if one use the polynomial functions orthogonal to the GLO-MAP weight function (Table XII) then the shape function can be evaluated accurately and easily.

1. Poisson Equation

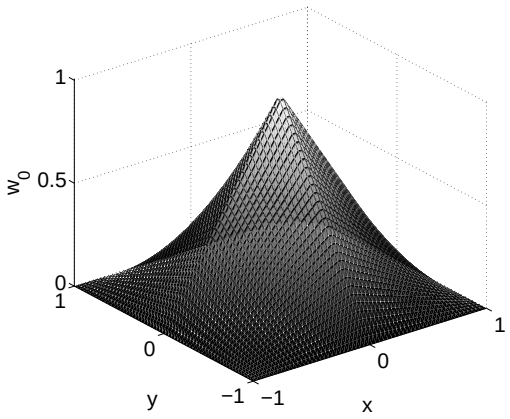
To illustrate the whole procedure of the MLPG approach, we consider Poisson equation in 2-D space.

$$\nabla^2 u = f \text{ in } \Omega \tag{6.19}$$

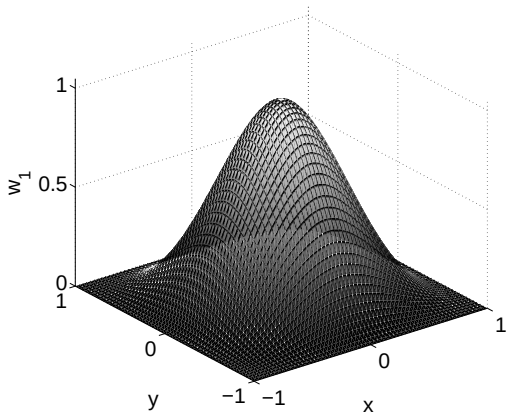
$$u = \bar{u} \text{ on } \Gamma_u \tag{6.20}$$

$$u_{,n} = \bar{q} \text{ on } \Gamma_q \tag{6.21}$$

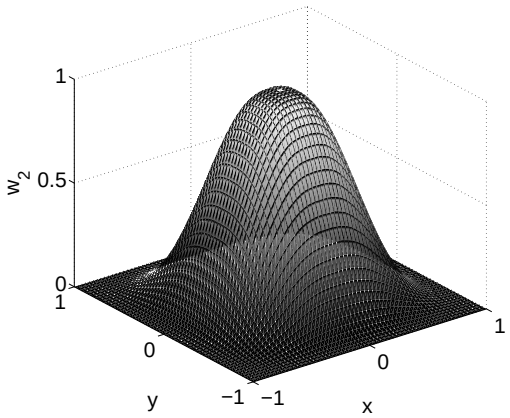
where $\nabla(.) = [\frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2}](.)$ is the Laplace operator and n is the direction normal to the boundary of the domain. Analogous to Eq. (6.4), we write a generalized local



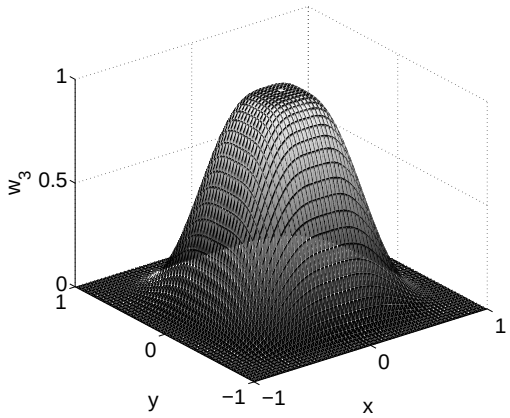
(a) $w_0(x, y) = w_0(x) \times w_0(y)$



(b) $w_1(x, y) = w_1(x) \times w_1(y)$

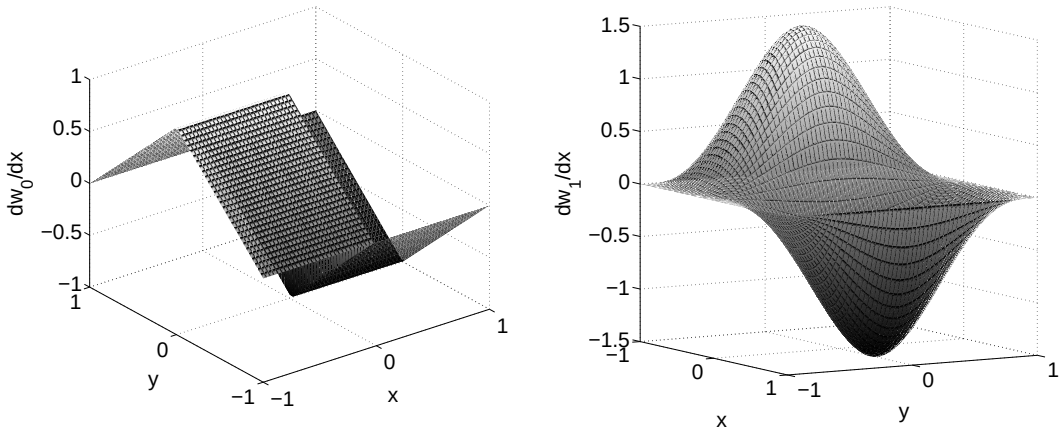


(c) $w_2(x, y) = w_2(x) \times w_2(y)$

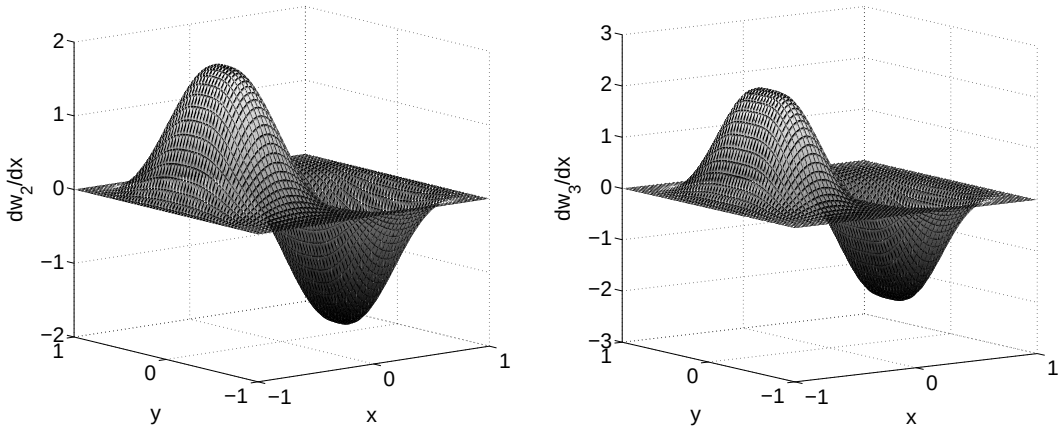


(d) $w_3(x, y) = w_3(x) \times w_3(y)$

Fig. 47. Weight functions for first four order of continuity.

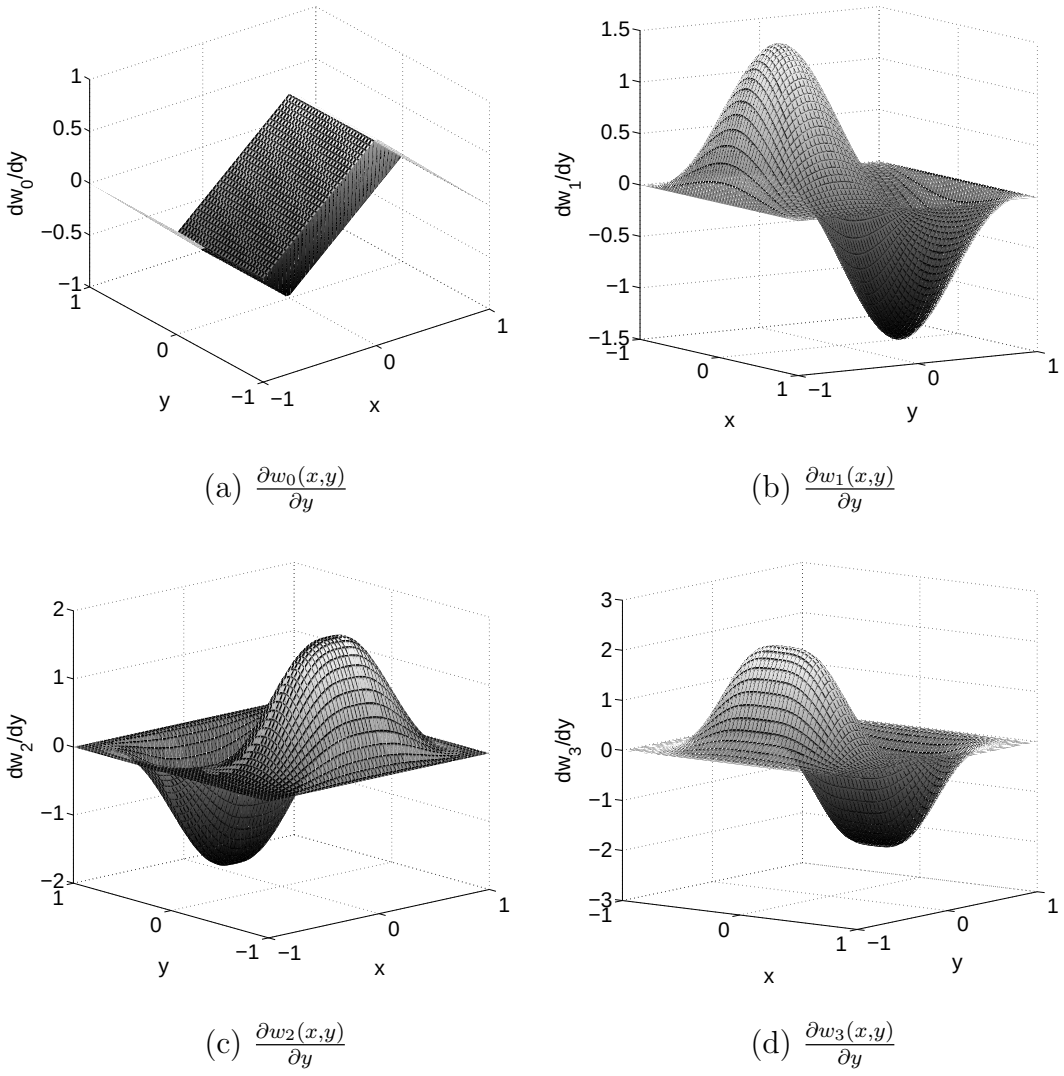


(a) $\frac{\partial w_0(x,y)}{\partial x}$ (b) $\frac{\partial w_1(x,y)}{\partial x}$



(c) $\frac{\partial w_2(x,y)}{\partial x}$ (d) $\frac{\partial w_3(x,y)}{\partial x}$

Fig. 48. First derivative of weight functions w.r.t. x for first four order of continuity.



(a) $\frac{\partial w_0(x,y)}{\partial y}$ (b) $\frac{\partial w_1(x,y)}{\partial y}$

(c) $\frac{\partial w_2(x,y)}{\partial y}$ (d) $\frac{\partial w_3(x,y)}{\partial y}$

Fig. 49. First derivative of weight functions w.r.t. y for first four order of continuity.

weak form of for the Poisson Equation over local sub-domain Ω_x as:

$$\int_{\Omega_s} (\nabla^2 \hat{u} - f) v_x d\Omega - \alpha \int_{\Gamma_{xu}} (\hat{u} - \bar{u}) v_x d\Gamma = 0 \tag{6.22}$$

where $\hat{u}$ is the trial function, v_x is the test function associated with nodal point x and Γ_{xu} is a part of the boundary Γ_u. Note that second term in Eq. (6.22) is introduced to impose the essential boundary condition. According to Eq. (6.22), trial function should be at least twice differentiable i.e. $\hat{u} \in \mathcal{C}^2$ while the test function v_x should be a continuous function i.e. $v_x \in \mathcal{C}^0$. However, using the fact that $(\nabla^2 \hat{u}) v_x = (\hat{u}_{,i} v)_{x,i} - \hat{u}_{,i} v_{x,i}$ and the divergence theorem, we can re-write Eq. (6.26) such that both trial function and test function are at least once differentiable i.e. $v_x, \hat{u} \in \mathcal{C}^1$.

$$\int_{\partial\Omega_x} \hat{u}_{,i} n_i v_x d\Gamma - \int_{\Omega_x} (\hat{u}_{,i} v_{x,i} + f v_x) d\Omega - \alpha \int_{\Gamma_{xu}} (\hat{u} - \bar{u}) v_x d\Gamma = 0 \tag{6.23}$$

Here, $\partial\Omega_x$ is the boundary of Ω_x which can be divided into three parts:

$$\partial\Omega_x = \Gamma_{xu} + \Gamma_{xq} + \Gamma_{xI} \tag{6.24}$$

where, Γ_{xI} is the part of boundary $\partial\Omega_x$ which neither intersects Γ_u nor Γ_q. Also, if we deliberately select a test function v_x such that it vanishes over the boundary of sub-domain Ω_x then the first term of Eq. (6.23) evaluated over Γ_{xI} can be simplified. This can be easily accomplished by using the GLO-MAP weight function of Table XI as the test function. As mentioned earlier, the domain of the test function v_x determines the domain Ω_x over which various integral expressions of Eq. (6.23) should be evaluated. Here, we choose sub-domain Ω_x to be a square centered at nodal point x. As test function v_x should be at least $\mathcal{C}^1$, therefore, we choose test function to be the 2^{nd}

order GLO-MAP weight function as listed in Table XI.

$$\begin{aligned} v_x(x,y) &= w_2(x) \times w_2(y) \\ &= \left[1 - \frac{x}{h}^3 \left(10 - 15\frac{x}{h} + 6(\frac{x}{h})^2\right)\right] \left[1 - \frac{y}{h}^3 \left(10 - 15\frac{y}{h} + 6(\frac{y}{h})^2\right)\right] \end{aligned} \quad (6.25)$$

Here, h is one half of the side of the square domain Ω_x. Now, using the fact that test function vanishes over Γ_{xI} the Eq. (6.23) reduces to

$$\int_{\Omega_x} (\hat{u}_{,i} v_{x,i}) d\Omega + \alpha \int_{\Gamma_{xu}} \hat{u} v_x d\Gamma - \int_{\Gamma_{xu}} q v_x d\Gamma = \int_{\Gamma_{xq}} \bar{q} v_x d\Gamma + \alpha \int_{\Gamma_{xu}} \bar{u} v_x d\Gamma - \int_{\Omega_x} f v_x d\Omega \quad (6.26)$$

To obtain the algebraic equations from Eq. (6.26), the MLS approximation of Eq. (6.16) is used to approximate the trial function $\hat{u}$. To find the expression for shape function, $\psi(.)$, we use the 2^{nd} order GLO-MAP weight function, given by Eq. (6.25), and, polynomial basis functions up to 2^{nd} degree in both x and y. The polynomial functions for the MLS approximation are designed to be orthogonal to the MLS weight function and are shown in Fig. 50. Finally, Substitution of Eq. (6.16) into Eq. (6.26) for all nodes leads to the following system of linear equations:

$$\mathbf{K}\mathbf{u} = \mathbf{f} \quad (6.27)$$

where, the entries of the "stiffness" matrix $\mathbf{K}$ and the "load" vector, $\mathbf{f}$ are given as:

$$\mathbf{K}_{ij} = \int_{\Omega_x} \psi_{j,k} v_{x,k}(\mathbf{x}, \mathbf{x}_i) d\Omega + \alpha \int_{\Gamma_{xu}} \psi_j v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma - \int_{\Gamma_{xu}} \psi_{j,n} v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma \quad (6.28)$$

$$\mathbf{f}_i = \int_{\Gamma_{xq}} \bar{q} v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma + \alpha \int_{\Gamma_{xu}} \bar{u} v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma - \int_{\Omega_x} f v_x(\mathbf{x}, \mathbf{x}_i) d\Omega \quad (6.29)$$

Note, theoretically, as long as the union of all local domains covers the global domain i.e., $\cup\Omega_s \supset \Omega$, the equilibrium equation and the boundary conditions will be satisfied in the global domain and on its boundary, respectively. To ensure this, we choose parameter h to be the minimum distance of nodal point $\mathbf{x}$ from all other nodal points.

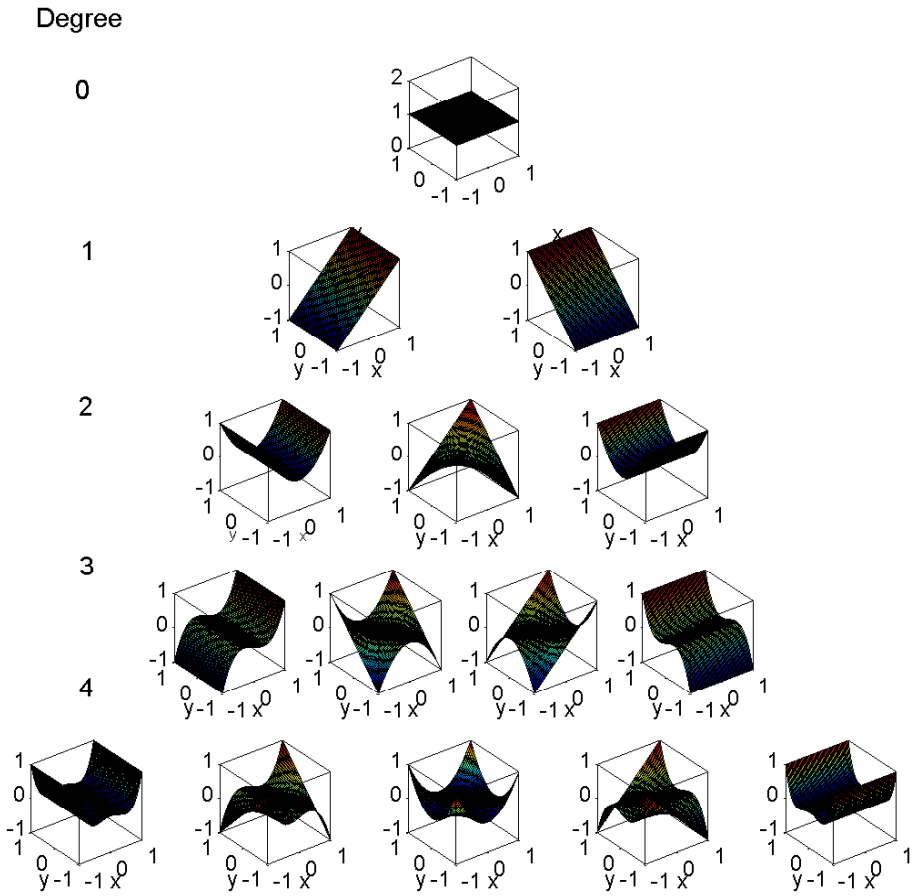


Fig. 50. Two-dimensional polynomial basis functions orthogonal to weight function given by Eq. (6.25).

Also, since we are using 2^{nd} order polynomials for the MLS approximation, therefore, we need to make sure that there are at least 6 nodal points in the domain of definition Ω_M associated with each nodal point. To ensure this, we choose the support of Ω_i to be $6h$. The implementation of the MLPG method can be carried out according to the following steps and is illustrated in Fig. 51:

1. Choose a finite number of nodes to discretize the global domain Ω and global boundary Γ.

2. Determine the local sub-domain Ω_x and its corresponding local boundary $\partial\Omega_x$ for each node.

3. Loop over all nodes located inside the global domain and at the global boundary Γ

   (a) Determine Gaussian quadrature points $\mathbf{x}_Q$ in the domain of integration Ω_x and its boundary $\partial\Omega_x$.

   (b) Loop over the quadrature points $\mathbf{x}_Q$ in the sub-domain Ω_x and on the local boundary $\partial\Omega_x$

      i. determine nodal points $\mathbf{x}_i$ such that $w_i(\mathbf{x}_Q, \mathbf{x}_i) > 0$

      ii. using the MLS approximation for trial function evaluate numerical integrals in Eqs. (6.28) and (6.29).

      iii. assemble contributions to the linear system for all nodes in $\mathbf{K}$ and $\mathbf{f}$.

   (c) End loop over quadrature points

4. End node loop.

5. Solve the linear system for the fictitious nodal values $\hat{u}_i$.

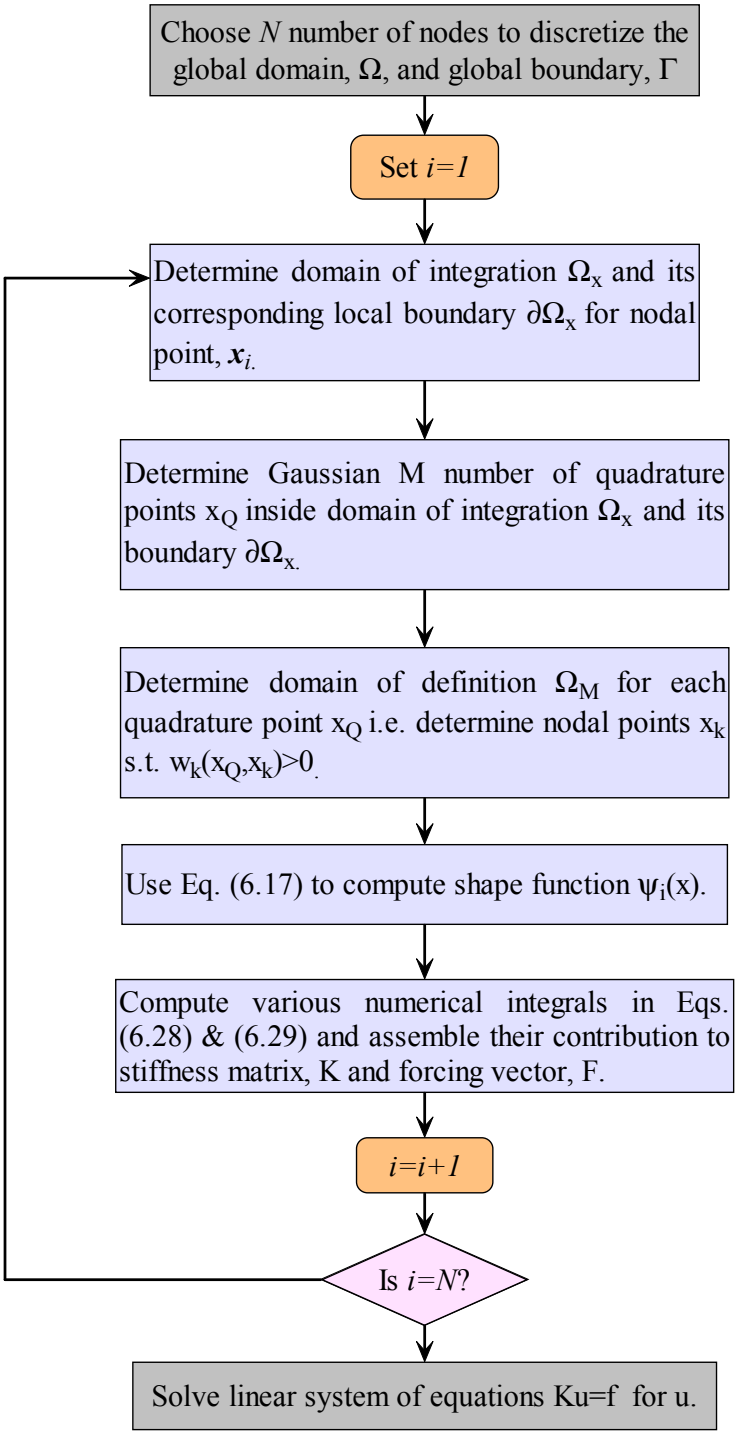


Fig. 51. Flow-chart for the MLPG algorithm.

To show the effectiveness of the method discussed in this section, we assume $f = -1$ in Eq. (6.19) and the following boundary conditions on a square domain of unit length:

$$u(x,1) = u(1,y) = 0 \tag{6.30}$$

$$\frac{\partial u}{\partial x}|_{(0,y)} = \frac{\partial u}{\partial y}|_{(x,1)} = 0 \tag{6.31}$$

The exact analytical solution to this boundary value problem is given by the following equation [89]:

$$u(x,y) = \frac{1}{2}\left[\left(1-y^2\right) + 4\sum_{n=1}^{\infty} \frac{(-1)^n \cos\alpha_n y \cosh\alpha_n x}{\alpha_n^3 \cosh\alpha_n}\right] \tag{6.32}$$

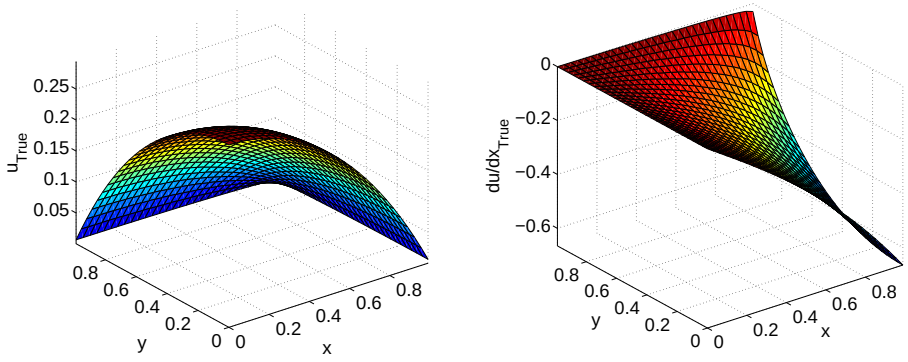
with

$$\alpha_n = \frac{1}{2}(2n-1)\pi \tag{6.33}$$

Fig. 52 shows the plots of true solution surface and various partial derivative of the true solution. The different integrals appearing in Eqs. (6.28) and (6.29) are evaluated numerically using the Gauss quadrature method. Regular meshes of different sizes are considered to study the convergence and accuracy of the method. In all the cases the computed solution is tested on a total of 2500 uniformly distributed points inside unit square. Fig. 53 shows the plot of relative error e with respect to mesh-size(h) for linear, quadratic and cubic basis functions.

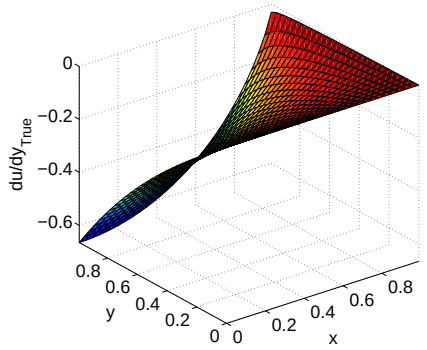
$$e = \frac{\sqrt{(u-\hat{u})^2 + (\frac{\partial u}{\partial x} - \frac{\partial \hat{u}}{\partial x})^2 + (\frac{\partial u}{\partial y} - \frac{\partial \hat{u}}{\partial y})^2}}{\sqrt{u^2 + \frac{\partial u}{\partial x}^2 + \frac{\partial u}{\partial y}^2}} \tag{6.34}$$

As expected, the relative error decreases with decrease in mesh size and increase in the order of basis functions. Also, it is clear that the MLPG converges as might be expected to reasonably accurate results for the solution and its derivatives.



(a) Exact Solution

(b) Exact $\frac{\partial u}{\partial x}$



(c) Exact $\frac{\partial u}{\partial y}$

Fig. 52. Exact solution to the Poisson's equation.

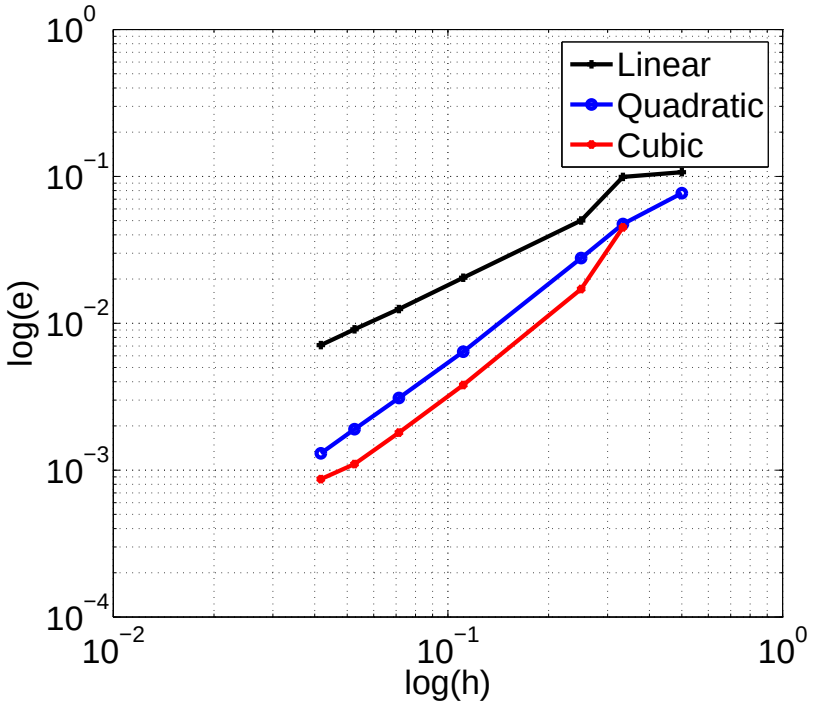


Fig. 53. Relative error for the Poisson's Equation using the MLPG method.

2. Comments on The MLPG Algorithm

Although the MLS shape function used in the MLPG algorithm reproduce the true solution at nodal points in accordance with the least square principle, however; there are several significant drawbacks associated with this approach.

1. The first disadvantage of the MLPG approach is that for each point under consideration a new linear system must be solved to find the value of the shape functions ψ_i and hence the value of the approximated solution $\hat{u}$. This is a computationally burdensome task.

2. As mentioned earlier, the smoothness of the shape functions ψ_i and hence the smoothness of the approximated solution $\hat{u}$ is directly related to the smoothness of both the basis functions and the weight functions used in the MLS approximation. As a consequence of this the approximated solution $\hat{u}$ is continuous up to an arbitrary order p over whole domain Ω if shape functions corresponding to all he nodes are continuous up to same order p. This is possible if the same set of basis functions is used in each local domain Ω_M. In other words, the basis functions of i^{th} local region can not be chosen independently from the basis function of j^{th} local region. So one can not increase the degree of approximation in a particular local region arbitrarily to reduce the approximation errors to a desired tolerance.

3. The shape functions ψ_i fails to have the selective property known as *partition of unity* i.e. $\sum_i \psi_i = 1$. Hence, u_i does not have the interpretation of nodal value of $\hat{u}$.

$$u^h(\mathbf{x}_i) \neq u_i \tag{6.35}$$

4. Even though the basis functions used to approximate u can be polynomial

in nature, there is no guarantee that shape function ϕ_i will be polynomial in nature which makes numerical integration more difficult than if all functions in the integrands were polynomials.

C. Modification of The MLPG Algorithm Using The GLO-MAP Algorithm

In the previous section, we discussed in detail the MLPG algorithm to solve the partial differential equations. One of the main disadvantage associated with MLPG algorithm is that the basis functions used for the MLS approximation in the i^{th} local region can not be chosen independently from the basis functions used in another local region without introducing discontinuity across the boundary of different local regions. Basically, the main problem is the lack of rigorous tools to merge different independent local approximations to obtain a desired order globally continuous approximation. In Chapter III, the GLO-MAP algorithm is introduced whose main attraction is a novel averaging process to determine a piecewise continuous global family of local least squares approximations, while having the freedom to vary the nature (e.g., degrees of freedom) of the local approximations. The continuity conditions are enforced by using a unique set of weighting functions (See Appendix A) in the averaging process. The weight functions are designed to guarantee the global continuity conditions while retaining near complete freedom on the selection of the generating local approximations. In this section, we propose a modification to the conventional MLPG algorithm to make use of the GLO-MAP averaging process.

Like in the previous section, let us consider a linear PDE to be solved over global domain Ω with boundary Γ

$$\mathcal{L}u = f \tag{6.36}$$

and following boundary conditions

$$u = \bar{u} \text{ on } \Gamma_u \tag{6.37}$$

$$\nabla u.\hat{\mathbf{n}} = \bar{q} \text{ on } \Gamma_q \tag{6.38}$$

Once again, let us approximate the unknown function u as $\hat{u}$ and write a generalized local weak form for Eq. (6.36) over a local sub-domain $\Omega_x \subset \Omega$ using test function v_x:

$$\int_{\Omega_x} [\mathcal{L}\hat{u} - f] v_x d\Omega + \alpha \int_{\Gamma_{xu}} [\hat{u} - \bar{u}] v_x d\Gamma + \beta \int_{\Gamma_{xq}} [q - \bar{q}] v_x d\Gamma = 0 \tag{6.39}$$

Now, introducing shape functions ψ_i, we can write the approximation $\hat{u}$ of the unknown function, u, as follows:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \psi_i(\mathbf{x}) \hat{u}_i \tag{6.40}$$

where N is the total number of nodal points used to discretize the global domain Ω and $\hat{u}_i$ is the solution value at the i^{th} nodal point. Further, the substitution of Eq. (6.40) in Eq. (6.39) leads to the following set of linear equations in unknown variables $\hat{u}_i$:

$$\mathbf{K}\hat{\mathbf{u}} = \mathbf{f} \tag{6.41}$$

where, $\mathbf{K}$ and $\mathbf{f}$ are given as:

$$K_{ij} = \int_{\Omega_x} \mathcal{L}\psi_j(\mathbf{x}) v_{x_j} d\Omega + \alpha \int_{\Gamma_{xu}} \psi_j(\mathbf{x}) v_{x_i} d\Gamma + \beta \int_{\Gamma_{xq}} \nabla\psi_j(\mathbf{x}) v_{x_i} d\Gamma \tag{6.42}$$

$$f_i = \int_{\Omega_x} f v_{x_i} d\Omega + \alpha \int_{\Gamma_{xu}} \bar{u} v_{x_i} d\Gamma + \beta \int_{\Gamma_{xq}} \bar{q} v_{x_i} d\Gamma \tag{6.43}$$

Recall that in the previous section, we use the MLS based fitting algorithm to find the expression for the shape function to be used in the MLPG algorithm. Like in the MLPG algorithm, here also we use the least square criteria to find the value of a shape

function. In case of the MLPG algorithm, one needs to solve for a new local approximation whenever one needs to find the value of a shape function. Here, we propose the concept of the GLO-MAP algorithm to keep the number of local approximations in check and further, the use of the GLO-MAP algorithm allows us to introduce higher order approximations in a particular region to decrease approximations error to a desirable tolerance.

To approximate a unknown function u using the GLO-MAP algorithm, a set of grid points $\{\bar{\mathbf{x}}_i\}_{i=1,\cdots,N_G}$, uniformly distributed over global domain Ω, are introduced. We mention that these grid points $\bar{\mathbf{x}}_i$ are different from the nodal points $\mathbf{x}_i$ on which we want to solve the given PDE. Each grid point is equipped with a set of weight functions $\{w_i(\mathbf{x})\}_{i=1,\cdots,N_G}$ and local approximations $\{F_i(\mathbf{x})\}_{i=1,\cdots,N_G}$. These local approximations are constructed so each represents the behavior in the hypercube centered on a typical vertex in the grid. These hypercubes, where local approximations are valid, generally overlap and are averaged over the overlapped volume to determine final approximations. The final weighted average approximation can be written as:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N_G} w_i(\mathbf{x}) F_i(\mathbf{x}) \tag{6.44}$$

where w_i are the weighting functions used to average (blend) the 2^n adjacent preliminary local approximations and are listed in Table XI. The weight functions are designed to guarantee the global continuity conditions while retaining near complete freedom on the selection of the local approximations F_i. The local approximation $F_i(\mathbf{x})$ associated with grid point $\bar{\mathbf{x}}_i$ can be written as a linear combination of user-defined basis functions ϕ_i:

$$F_i(\mathbf{x}, \bar{\mathbf{x}}) = \boldsymbol{\phi}_i^T(\mathbf{x}, \bar{\mathbf{x}}) \mathbf{a}_i(\bar{\mathbf{x}}) \tag{6.45}$$

Here, $\boldsymbol{\phi} \in \mathcal{R}^m$ is a vector of basis functions ϕ_i and $\mathbf{a} \in \mathcal{R}^m$ is a vector of corresponding Fourier coefficients. Note, unlike the MLS approximation the Fourier coefficient vector $\mathbf{a}$ is not a function of $\mathbf{x}$ but it depends upon the location of grid point $\bar{\mathbf{x}}$. To solve for the Fourier coefficient vector $\mathbf{a}_i$, the mean square error is minimized over the set of nodal point lying inside the support, Ω_w, of the weight function associated with i^{th} grid point.

$$J = \sum_{k=1}^{M} w_i(\mathbf{x}_k, \bar{\mathbf{x}}_i) \left(\boldsymbol{\phi}_i^T(\mathbf{x}_k, \bar{\mathbf{x}}_i)\mathbf{a}_i(\bar{\mathbf{x}}) - u_k^h \right)^2 \tag{6.46}$$

Here, u_k^h is the fictitious value of the unknown function u at $\mathbf{x}_i$ for which we want to solve. We mention that the support, Ω_w, of the weight function $w_i(.)$ is a hypercube centered at the i^{th} grid point. The first order optimality condition for the loss function, given by Eq. (6.46), results in a set of linear equations for the Fourier coefficient vector $\mathbf{a}_i$

$$\mathbf{a}_i(\bar{\mathbf{x}}) = (\underbrace{\boldsymbol{\Phi}_i^T \mathbf{W}_i \boldsymbol{\Phi}_i}_{\mathbf{A}})^{-1} \boldsymbol{\Phi}_i^T \mathbf{W}_i \mathbf{u}^h \tag{6.47}$$

where, $\mathbf{u} \in \mathcal{R}^m$ is a vector with entries u_i^h and the matrices $\boldsymbol{\Phi}_i$ and $\mathbf{W}_i$ are given by

$$\boldsymbol{\Phi}_{i_{jk}} = \phi_{i_j}(\mathbf{x}_k, \bar{\mathbf{x}}_i) \tag{6.48}$$

$$\mathbf{W}_{i_{jk}} = w_i(\mathbf{x}_k, \bar{\mathbf{x}}_i)\delta_{jk} \tag{6.49}$$

The necessary condition to solve for the Fourier coefficient vector $\mathbf{a}$ is that the rank of the matrix $\boldsymbol{\Phi}$ should be at least m. That means $M \geq m$. Now, the approximated

solution $\hat{u}$ can also be expressed as:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N_G} w_i(\mathbf{x}, \bar{\mathbf{x}}_i) F_i(\mathbf{x}, \bar{\mathbf{x}}_i) = \sum_{i=1}^{N_G} w_i(\mathbf{x}, \bar{\mathbf{x}}_i) \boldsymbol{\phi}_i^T(\mathbf{x}, \bar{\mathbf{x}}_i) \mathbf{a}_i \tag{6.50}$$

$$= \underbrace{\sum_{i=1}^{N_G} w_i(\mathbf{x}, \bar{\mathbf{x}}_i) \boldsymbol{\phi}_i^T(\mathbf{x}, \bar{\mathbf{x}}_i) \mathbf{A}_i^{-1} \boldsymbol{\Phi}_i^T \mathbf{W}_i(.)}_{\boldsymbol{\psi}_i^T} \mathbf{u}^h \tag{6.51}$$

$$= \sum_{k=1}^{N} \psi_k(\mathbf{x}) u_k \tag{6.52}$$

It should be noticed that the support of the shape function ψ_i associated with i^{th} nodal point $\mathbf{x}_i$ is equal to the union of the domains of 2^n weight functions which have non-zero value at $\mathbf{x}_i$. Like the MLPG algorithm, the continuity of the approximated solution $\hat{u}$ depends upon the continuity of the GLO-MAP weight functions w_i and basis functions $\boldsymbol{\phi}_i$. Further, the use of the weighting functions to blend different local approximations allows us to use different basis functions in different local regions. As a consequence of this, the shape function $\boldsymbol{\psi}$ and hence the approximated solution $\hat{u}(x)$ are continuous over whole global domain Ω even though different order basis functions are used to obtain different local approximations.

1. Poisson Equation

Once again, to illustrate the whole procedure, discussed in the previous section, we consider the Poisson equation in the 2-D space.

$$\nabla^2 u = f \text{ in } \Omega \tag{6.53}$$

$$u = \bar{u} \text{ on } \Gamma_u \tag{6.54}$$

$$u_{,n} = \bar{q} \text{ on } \Gamma_q \tag{6.55}$$

where $\nabla(.) = [\frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2}](.)$ is the Laplace operator and n is the direction normal to the boundary of the domain. Adopting the procedure listed in the previous section the symmetric weak form for Eq. (6.53) over local sub-domain Ω_x can be written as:

$$\int_{\partial\Omega_x} \hat{u}_{,i} n_i v_x d\Gamma - \int_{\Omega_x} (\hat{u}_{,i} v_{x,i} + f v_x) d\Omega - \alpha \int_{\Gamma_{xu}} (\hat{u} - \bar{u}) v_x d\Gamma = 0 \tag{6.56}$$

where, $\partial\Omega_x$ is the boundary of the local sub-domain Ω_x, $\hat{u}$ is the trial function, approximated by the GLO-MAP process and v_x is the test function with support equal to the local sub-domain Ω_x. Now, if we deliberately select a test function v_x such that it vanishes over the boundary of the sub-domain Ω_x then the first term of Eq. (6.56) can be simplified as discussed in the previous section and this can be easily accomplished by using the GLO-MAP weight function of Table XI as the test function. Once again, we choose test function to be the 2^{nd} order GLO-MAP weight function as listed in Table XI and sub-domain Ω_x to be a square centered at nodal point x.

$$\begin{aligned} v_x(x,y) &= w_2(x) \times w_2(y) \\ &= \left[1 - \frac{x}{h}^3 \left(10 - 15\frac{x}{h} + 6(\frac{x}{h})^2\right)\right] \left[1 - \frac{y}{h}^3 \left(10 - 15\frac{y}{h} + 6(\frac{y}{h})^2\right)\right] \end{aligned} \tag{6.57}$$

Here, h is one half of the length of the side of the square domain Ω_x. Now, using the fact that test function vanishes over the boundary of sub-domain Ω_x the Eq. (6.56) reduces to

$$\int_{\Omega_x} (\hat{u}_{,i} v_{x,i}) d\Omega + \alpha \int_{\Gamma_{xu}} \hat{u} v_x d\Gamma - \int_{\Gamma_{xu}} q v_x d\Gamma = \int_{\Gamma_{xq}} \bar{q} v_x d\Gamma + \alpha \int_{\Gamma_{xu}} \bar{u} v_x d\Gamma - \int_{\Omega_x} f v_x d\Omega \tag{6.58}$$

To obtain the algebraic equations from Eq. (6.58), the GLO-MAP approximation of Eq. (6.40) is used to approximate the trial function $\hat{u}$. To find the expression for the shape function, $\psi(.)$, we use the 2^{nd} order GLO-MAP weight function and polynomial

basis functions up to 2^{nd} order in Eq. (6.52). Finally, Substitution of Eq. (6.52) into Eq. (6.58) for all nodes leads to the following system of linear equations:

$$\mathbf{Ku} = \mathbf{f} \tag{6.59}$$

where, the entries of the "stiffness" matrix $\mathbf{K}$ and the "load" vector, $\mathbf{f}$ are given as:

$$\mathbf{K}_{ij} = \int_{\Omega_x} \psi_{j,k}(\mathbf{x}) v_{x,k}(\mathbf{x}, \mathbf{x}_i) d\Omega + \alpha \int_{\Gamma_{xu}} \psi_j v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma - \int_{\Gamma_{xu}} \psi_{j,n} v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma \tag{6.60}$$

$$\mathbf{f}_i = \int_{\Gamma_{xq}} \bar{q} v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma + \alpha \int_{\Gamma_{xu}} \bar{u} v_x(\mathbf{x}, \mathbf{x}_i) d\Gamma - \int_{\Omega_x} f v_x(\mathbf{x}, \mathbf{x}_i) d\Omega \tag{6.61}$$

Note, theoretically, as long as the union of all local domains covers the global domain i.e., $\cup\Omega_s \supset \Omega$, the equilibrium equation and the boundary conditions will be satisfied in the global domain and on its boundary, respectively. To ensure this, we choose parameter h to be equal to the minimum distance of nodal point $\mathbf{x}$ from all other nodal points. Also, as we are using 2^{nd} order polynomials for the MLS approximation, therefore, we need to make sure that there are at least 6 nodal points in the domain of definition Ω_M associated with each nodal point. To ensure this, we choose grid points to be $h/2$ distance apart. The implementation of the modified MLPG method can be carried out according to the following routine and is illustrated in Fig. 54:

1. Choose a finite number of nodes to discretize the global domain Ω and global boundary Γ.

2. Choose uniformly distributed grid points $\mathbf{x}_{g_i}$ and assign a GLO-MAP weight function with each grid point.

3. Determine the local sub-domain Ω_x and its corresponding local boundary $\partial\Omega_x$ for each node.

4. Loop over all nodes located inside the global domain and at the global boundary

Γ

(a) Determine Gaussian quadrature points $\mathbf{x}_Q$ in domain of integration Ω_x and its boundary $\partial\Omega_x$.

(b) Loop over quadrature points $\mathbf{x}_Q$ in the sub-domain Ω_x and on the local boundary $\partial\Omega_x$

i. Determine 2^n grid points $\mathbf{x}_{g_i}$ such that $w_i(\mathbf{x}_Q, \mathbf{x}_{g_i}) > 0$

ii. For each grid point $\mathbf{x}_{g_i}$ find local approximation for trial function and evaluate numerical integrals in Eqs. (6.60) and (6.61).

iii. assemble contributions to the linear system for all nodes in $\mathbf{K}$ and $\mathbf{f}$.

(c) End loop over quadrature points

5. End node loop.

6. Solve the linear system for the fictitious nodal values $\hat{u}_i$.

To consistent with the performance test for the MLPG algorithm, we choose $f = -1$ in Eq. (6.53) and the following boundary conditions on a square domain of unit length:

$$u(x,1) = u(1,y) = 0 \tag{6.62}$$

$$\frac{\partial u}{\partial x}|_{(0,y)} = \frac{\partial u}{\partial y}|_{(x,1)} = 0 \tag{6.63}$$

The exact solution to this boundary value problem is given by Eq. (6.32) and the surface plot for its various partial derivatives are shown in Fig. 52. The different integrals appearing in Eqs. (6.60) and (6.61) are evaluated numerically using the Gauss quadrature method. Uniformly distributed nodal points with different inter nodal distances are considered to study the convergence and accuracy of the method.

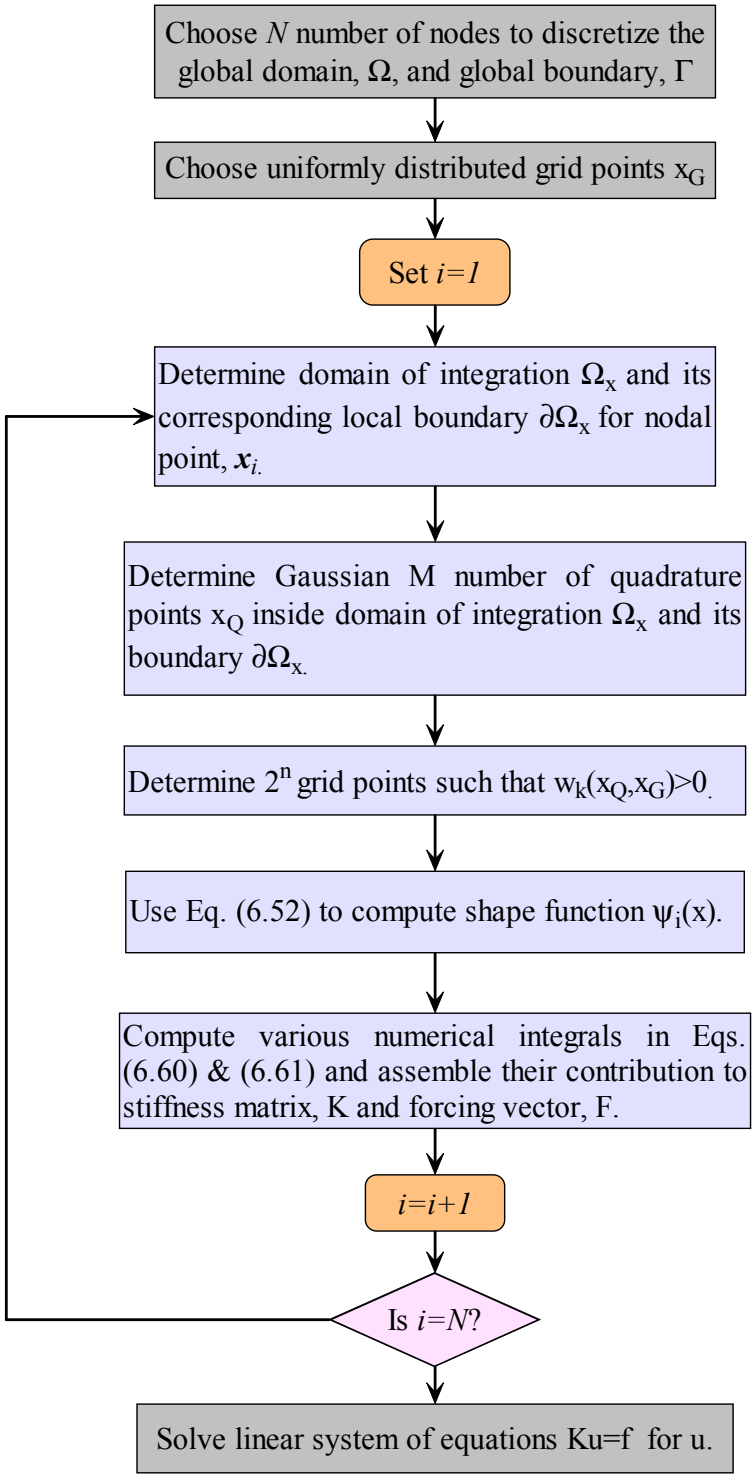


Fig. 54. Flow-chart for the modified MLPG algorithm.

Beside this, the number of grid points also plays an important role in the convergence of the modified MLPG algorithm. Ideally, one would suspect that keeping the number of grid points constant and increasing the number of nodal points should increase the approximation accuracy. However, in this case we encountered a singular stiffness matrix if we increase the number of nodal points arbitrarily while keeping the number of grid points to be constant. After experimenting with different grid sizes, it was found that if we choose the number of grid points to be half the number of total nodal points then we never encountered a singularity in the stiffness matrix. For this particular case, this is also the minimum number of grid points required to find the local approximations using quadratic basis functions. Like in the previous section, the computed solution is tested on total 2500 uniformly distributed points inside unit square. Fig. 55 shows the plot of relative error e, given by Eq. (6.34), with respect to distance between nodal points (h) for quadratic basis functions. As expected, the relative error decreases with decrease in nodal point distance h. As compared to Fig. 53, the approximations error are comparable to the ones obtained with the conventional MLPG algorithm.

Finally, we mention that although the issue of independent local approximation can be solved very easily using the GLO-MAP algorithm but the modified MLPG still suffers from other disadvantages of the MLPG algorithm, not to mention the computational cost associated with computing local approximations. To deal with other issues associated with the MLPG method we propose another algorithm which is inspired by the PUFEM algorithm as discussed in Ref. [85]. The following algorithm addresses all of the disadvantages mentioned above.

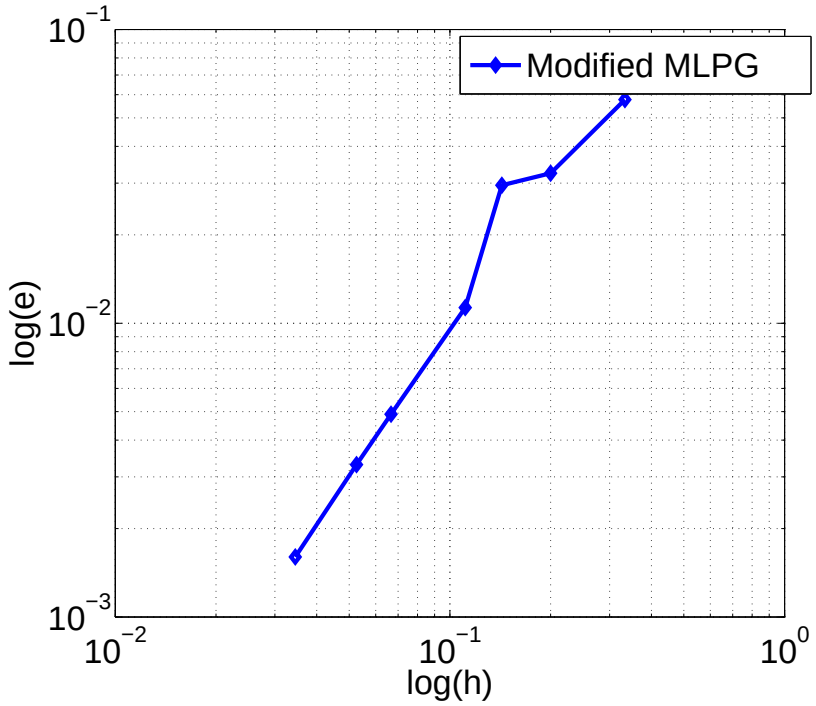


Fig. 55. Relative error for the Poisson's equation using the modified MLPG method.

D. Partition of Unity Finite Element Method

In the previous sections, we discussed the MLPG and the modified MLPG algorithms to solve partial differential equations. In both the methods, the local shape functions are constructed with the help of methods from data fitting and further, these shape functions are used in a Galerkin discretization process to set up a linear system of equations. Finally, these systems of equations need to be solved efficiently. Beside solving the final system of linear equations, one also need to solve a new system of linear algebraic equations to evaluate the shape function value at any point which not only increases the computational cost but also restricts the accuracy that can be achieved. In this section, we directly use the Galerkin discretization process to find the local shape function instead of using some data fitting process followed by the Galerkin discretization process.

The partition of unity viewpoint for the meshless FEM has been developed by Babuška and Melenk [85]. The partition of unity is a mathematical paradigm in which a domain Ω is covered by overlapping sub-domains Ω_s each of which is associated with a function f_s which is non zero over Ω_s and has the property that

$$\sum_s f_s(\mathbf{x}) = 1 \tag{6.64}$$

We mention that the partition of unity condition, given by Eq. (6.64), is identical to the zeroth order consistency condition for the functions $f_s(\mathbf{x})$, i.e. function $f_s(.)$ can reproduce constant function exactly. For example, if the functions values and the derivatives are given at the nodal points $\mathbf{x}_i$ then one can choose the function

$$\hat{u}(\mathbf{x}) = \sum_i f_i(\mathbf{x}) T_i(\mathbf{x}) \tag{6.65}$$

where $T_i(.)$ are the Taylor polynomial of function $u(\mathbf{x})$ centered about the nodal point

$\mathbf{x}_i$ and thus involve value of the function and its derivatives evelauated at the nodal point $\mathbf{x}_i$. According to Eq. (6.65), there are two ways to improve the performance of the approximation $\hat{u}(.)$: One is to improve the order of the consistency of the shape functions $f_i(.)$; the other is directly to improve the consistency orders of the polynomial $T_i(.)$. The first one is difficult to achieve due to the partition of unity constraint given by Eq. (6.64). Therefore, generally Shepard' s function ($f(\mathbf{x}) = \frac{f_i(.)}{\sum\limits_i f_i(.)}$) is used as the zeroth order shape function and the polynomial functions for the local approximation are chosen as:

$$T_i(\mathbf{x}) = \sum_{j=1}^{m} \phi_{ij}(\mathbf{x}) a_{ij} \tag{6.66}$$

where a_{ij} are the Fourier coefficients corresponding to various polynomial basis functions denoted by $\phi_{ij}(.)$. Now, the approximated function for the method of partition of unity can be written as:

$$\hat{u}(\mathbf{x}) = \sum_{i=1}^{N} \sum_{j=1}^{m} f_i(\mathbf{x}) \phi_{ij}(\mathbf{x}) a_{ij} \tag{6.67}$$

Now, if Shepard's function is used as the shape function as suggested in Ref. [85] then the consistency in the above equation depends on the order of polynomial basis functions $\phi(.)$. Further, note that the Shepard's function provides the partition of unity and hence the compact support for the local approximation. The coefficients a_{ij} are the unknowns and can be found directly by using the Galerkin discretization process instead of using a data fitting algorithm. Therefore, one needs at least m test functions per node to obtain a sufficient number of equations to determine the unknowns Fourier coefficients.

To illustrate the whole procedure, let us consider a general linear PDE given by Eq. (6.36) for which a generalized local weak form over a local sub-domain $\Omega_s \subset \Omega$

can be written as:

$$\int_{\Omega_s} [\mathcal{L}\hat{u} - f] v_s d\Omega + \alpha \int_{\Gamma_{su}} [\hat{u} - \bar{u}] v_s d\Gamma + \beta \int_{\Gamma_{sq}} [q - \bar{q}] v_s d\Gamma = 0 \tag{6.68}$$

where, v_s represents the test function for sub-domain Ω_s and $\hat{u}$ is the approximated solution for Eq. (6.36) which is valid over sub-domain Ω_s. Further, substituting for $\hat{u}$ from Eq. (6.67) in Eq. (6.68), we get:

$$\begin{aligned} \int_{\Omega_s} [\mathcal{L} \sum_{s=1}^{N} \sum_{j=1}^{m} f_s(\mathbf{x}) \phi_{sj}(\mathbf{x}) a_{sj} - f] v_s d\Omega \quad &+ \quad \alpha \int_{\Gamma_{su}} [\sum_{s=1}^{N} \sum_{j=1}^{m} f_s(\mathbf{x}) \phi_{sj}(\mathbf{x}) a_{sj} - \bar{u}] v_s d\Gamma \\ &+ \quad \beta \int_{\Gamma_{sq}} [q - \bar{q}] v_s d\Gamma = 0 \end{aligned} \tag{6.69}$$

To find unknown Fourier coefficients a_{sj}, we need at least m test functions per nodal point. Now, choosing test functions to be same as the trial functions $\{f_s \phi_{si}\}_{i=1}^{m}$, we get following set of algebraic equations for Fourier coefficients a_{si}:

$$\mathbf{K}\mathbf{a} = \mathbf{F} \tag{6.70}$$

where, $\mathbf{a}$ is a vector consisting of mN Fourier coefficients. Further, stiffness matrix $\mathbf{K}$ and forcing vector $\mathbf{F}$ are given as:

$$\begin{aligned} K_{ij} \quad &= \quad \int_{\Omega_s} \mathcal{L} f_i(\mathbf{x}) \Phi_i(\mathbf{x}) f_j \Phi_j(\mathbf{x}) d\Omega + \alpha \int_{\Gamma_{su}} f_i(\mathbf{x}) \Phi_i(\mathbf{x}) f_j \Phi_j(\mathbf{x}) d\Gamma \\ &\quad + \beta \int_{\Gamma_{sq}} \frac{\partial f_i(\mathbf{x}) \Phi_i(\mathbf{x})}{\partial n} f_j(\mathbf{x}) \Phi_j d\Gamma \end{aligned} \tag{6.71}$$

$$F_j \quad = \quad \int_{\Omega_s} f f_j \Phi_j(\mathbf{x}) d\Omega + \alpha \int_{\Gamma_{su}} \bar{u} f_j \Phi_j(\mathbf{x}) d\Gamma + \beta \int_{\Gamma_{sq}} \bar{q} f_j \Phi_j(\mathbf{x}) d\Gamma \tag{6.72}$$

where, $\Phi = \{\boldsymbol{\phi}_s\}_{1=i} N$ is a vector consisting of basis functions for each sub-domain Ω_s. It should be noticed that in this case the dimension of the stiffness matrix $\mathbf{K}$ is $Nm \times Nm$ which is m times the dimension of the stiffness matrix in case of the MLPG

algorithm. However, in case of the MLPG algorithm one needs to solve a new system of linear equations to find m Fourier coefficients for each local approximation. Also, in case of the MLPG algorithm, the computation of the solution and its derivative at a point other than nodal point also involves the solution of a system of linear algebraic equations whereas in case of the PUFEM approach there is no extra computational burden associated with these calculations once one has solved Eq (6.70) for Fourier coefficients.

We mention that the GLO-MAP weight functions are extremely attractive in the PUFEM approach as they satisfy the partition of unity condition and have a compact support as required here. Apparently the existence of these simple polynomial weight functions has not heretofore been explored in the PUFEM literature. The main advantage of using the GLO-MAP weight functions is that they are polynomial in nature and further if one uses the polynomial functions orthogonal to the GLO-MAP shape/weight function then many numerical integrals in Eqs. (6.71) and (6.72) can be evaluated accurately and easily. Besides this, they also provide an automated path to generate all of the higher order weight functions compatible with the partition of unity constraint. Furthermore, they have been fully extended to n-dimensions, so this also opens up a path to generalization of PUFEM methods to solve high dimensioned PDEs, such as the Fokker-Planck equation [90].

1. Poisson Equation

To illustrate the PUFEM ideas, we again consider the Poisson's equation in 2-D space:

$$\nabla^2 u = f \text{ in } \Omega \tag{6.73}$$

$$u = \bar{u} \text{ on } \Gamma_u \tag{6.74}$$

$$u_{,n} = \bar{q} \text{ on } \Gamma_{uq} \tag{6.75}$$

where $\nabla(.)$ is the Laplace operator and n is the direction normal to the boundary of the domain. As discussed earlier, after some algebraic manipulations and making use of the divergence theorem, we get the following weak form equation from Eq. (6.73):

$$\int_{\partial\Omega_s} \hat{u}_{,i} n_i v_s d\Gamma - \int_{\Omega_s} (\hat{u}_{,i} v_{x,i} + f v_s) d\Omega - \alpha \int_{\Gamma_{su}} (\hat{u} - \bar{u}) v_s d\Gamma = 0 \tag{6.76}$$

where, $\partial\Omega_s$ is the boundary of Ω_s, $\hat{u}$ is the trial function, approximated by the GLO-MAP process and v_s is the test function with support equal to the local sub-domain Ω_s. Further, if we deliberately divide the boundary term $\partial\Omega_s$ in three parts Γ_{su}, Γ_{sq} and Γ_{sI}. where, Γ_{su} and Γ_{sq} are the parts of global boundary on which Dirichlet and Neumann boundary conditions are defined and Γ_{sI} is the part of $\partial\Omega_s$ which do not intersect global boundary Γ. Now, if we deliberately select a test function v_s such that it vanishes over Γ_{sI} then the first term of Eq. (6.76) can be simplified as discussed previously in this Chapter. We mention that this can be easily accomplished by using the GLO-MAP weight function of Table XI as the test function. Further, the substitution of the Eq. (6.67) in Eq. (6.76) leads to the following system of linear equations:

$$\mathbf{K}\mathbf{a} = \mathbf{f} \tag{6.77}$$

where,

$$\begin{aligned} \mathbf{K}_{ij} &= \int_{\Omega_s} ((f_j \boldsymbol{\phi}^T)_{,k} v_{i,k}) d\Omega + \alpha \int_{\Gamma_{su}} (f_j \boldsymbol{\phi}^T - (f_j \boldsymbol{\phi}^T)_{,k} n_{,k}) v_i d\Gamma \\ &- \int_{L_s} (f_j \boldsymbol{\phi}^T)_{,k} v_i d\Gamma \qquad (6.78) \\ \mathbf{f}_i &= \int_{\Gamma_{sq}} \bar{q} v_i d\Gamma + \alpha \int_{\Gamma_{su}} \bar{u} v_i d\Gamma - \int_{\Omega_s} f v_i d\Omega \qquad (6.79) \end{aligned}$$

where $v_i = f_i \boldsymbol{\phi}$ is the test function associated with i^{th} node. Note that, if one uses the GLO-MAP weight functions as the partition of unity functions f_i and the corresponding set of orthogonal polynomials as basis functions ϕ_i then all integral

terms in Eqs. (6.78) and (6.79) are polynomial in nature which can be evaluated analytically without further approximation. The implementation of the PUFEM method can be carried out according to the following routine

1. Choose a finite number of nodes to discretize the global domain Ω and global boundary Γ.

2. Determine the local sub-domain Ω_x and its corresponding local boundary $\partial\Omega_x$ for each node.

3. Assign partition of unity weight function $w_i(\mathbf{x}, \mathbf{x}_i)$ and basis functions $\boldsymbol{\phi}_i$ with each node point $\mathbf{x}_i$

4. Loop over all nodes located inside the global domain and at the global boundary Γ

   (a) Determine Gaussian quadrature points $\mathbf{x}_Q$ in domain of integration Ω_x and its boundary $\partial\Omega_x$.

   (b) Loop over quadrature points $\mathbf{x}_Q$ in the sub-domain Ω_x and on the local boundary $\partial\Omega_x$

      i. Determine nodal points $\mathbf{x}_k$ such that $w_i(\mathbf{x}_i, \mathbf{x}_k) > 0$

      ii. Evaluate numerical integrals in Eqs. (6.78) and (6.79).

      iii. Assemble contributions to the linear system for all nodes in $\mathbf{K}$ and $\mathbf{f}$.

   (c) End loop over quadrature points

5. End node loop.

6. Solve the linear system for Fourier coefficients $\mathbf{a}$.

To show the effectiveness of the method discussed in this section, we assume $f = -1$ in Eq. (6.73) and the following boundary conditions on a square domain of unit length:

$$u(x,0) = u(0,y) = 0 \tag{6.80}$$

$$\frac{\partial u}{\partial x}|_{(1,y)} = \frac{\partial u}{\partial y}|_{(x,L)} = 0 \tag{6.81}$$

The exact solution to this boundary value problem is given by Eq. (6.32) and the plots of the true solution are shown in Fig. 52. The local approximations T_i are approximated by using the zeroth order ($m = 0$) and the first order ($m = 1$) weight functions listed in Table XI. Further, to study the approximation error convergence with the order of basis functions we use linear, quadratic and cubic polynomials, orthogonal to the zeroth and the first order weight functions, as basis functions. The plot of these orthogonal polynomials are shown in Figs. 56 and 57. We also consider uniformly distributed points with different inter nodal distance h to study the convergence and accuracy of the method. In all the cases the computed solution is tested on total 2500 uniformly distributed points inside unit square.

Figs. 58 and 59 show surface plots of the computed solution and its various partial derivatives using zeroth and first order weight functions, respectively. From these figures, it is clear that both zeroth and first order weight functions are able to approximate the solution accurately. However, as expected in case of the zeroth order weight function, the various partial derivative are discontinuous along the boundary of a particular sub-domain Ω_s (see Fig. 60) while the first order weight function merge different local approximations to guarantee the continuity of solution and its various first derivatives. Fig. 61 shows the plot of relative error (e) with respect to nodal distance h. As expected, the relative error decreases with decrease in nodal point

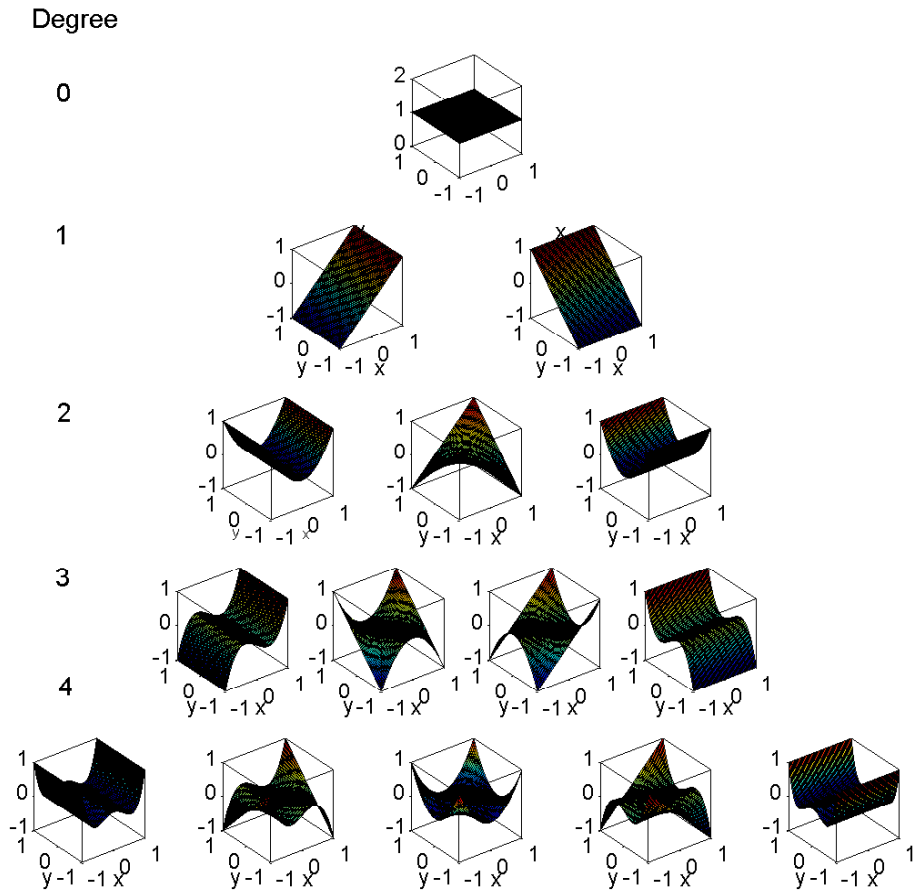


Fig. 56. Two-dimensional polynomial basis functions orthogonal to zeroth order weight function.)

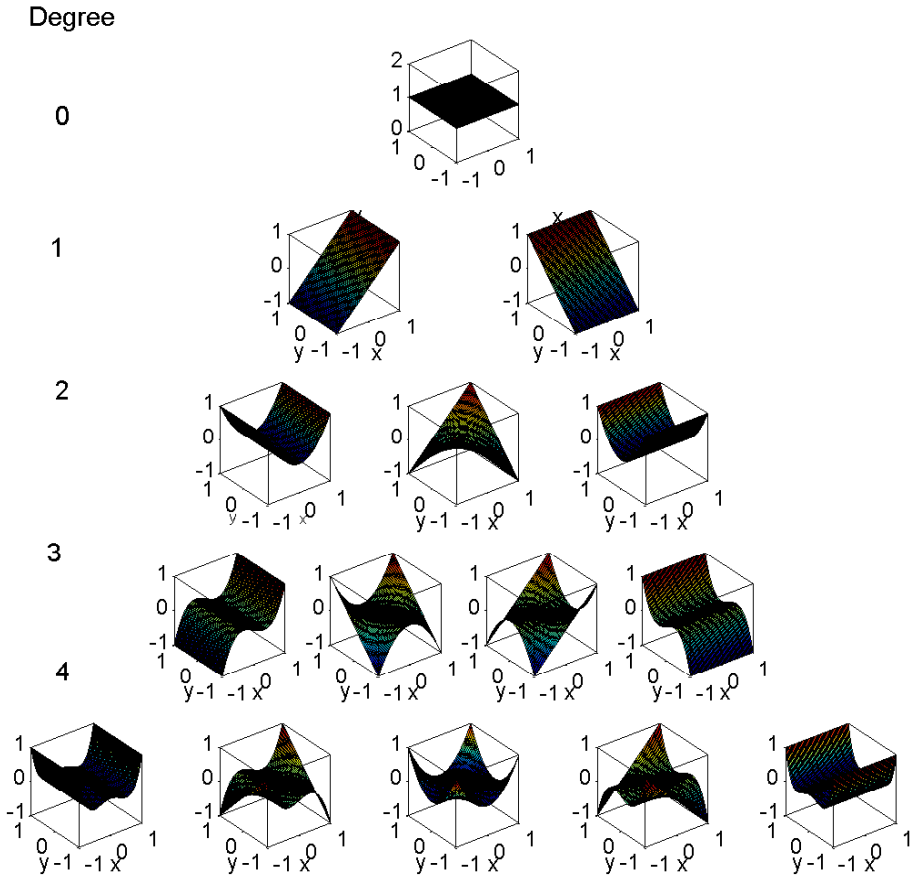
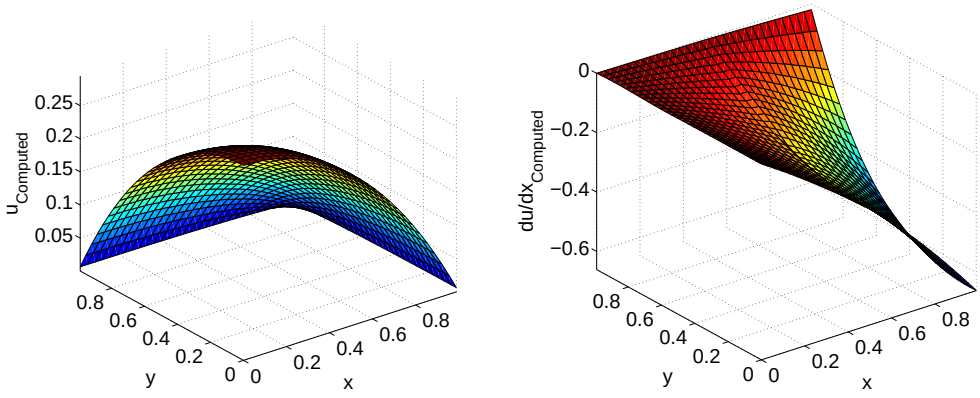
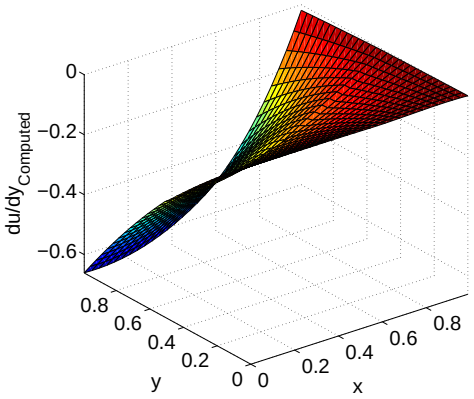


Fig. 57. Two-dimensional polynomial basis functions orthogonal to first order weight function.)



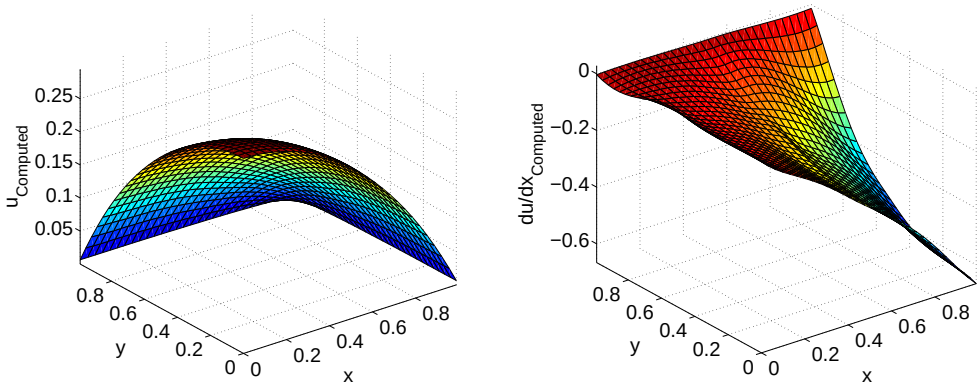
(a) Computed solution

(b) Computed $\frac{\partial u}{\partial x}$



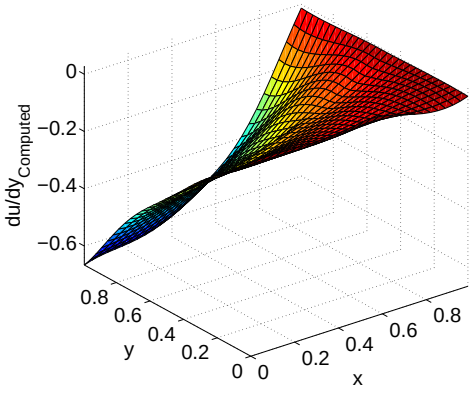
(c) Computed $\frac{\partial u}{\partial y}$

Fig. 58. Computed solution to the Poisson's equation using the zeroth order weight function.



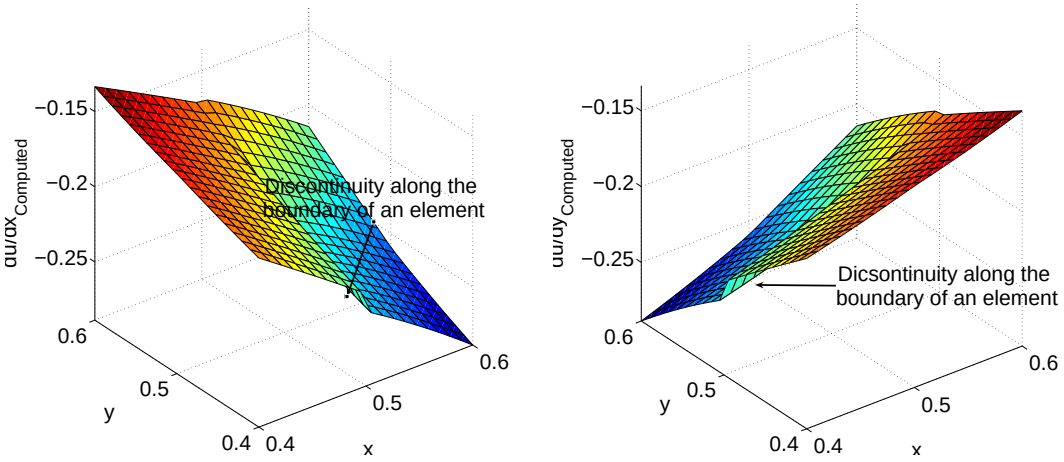
(a) Computed solution

(b) Computed $\frac{\partial u}{\partial x}$



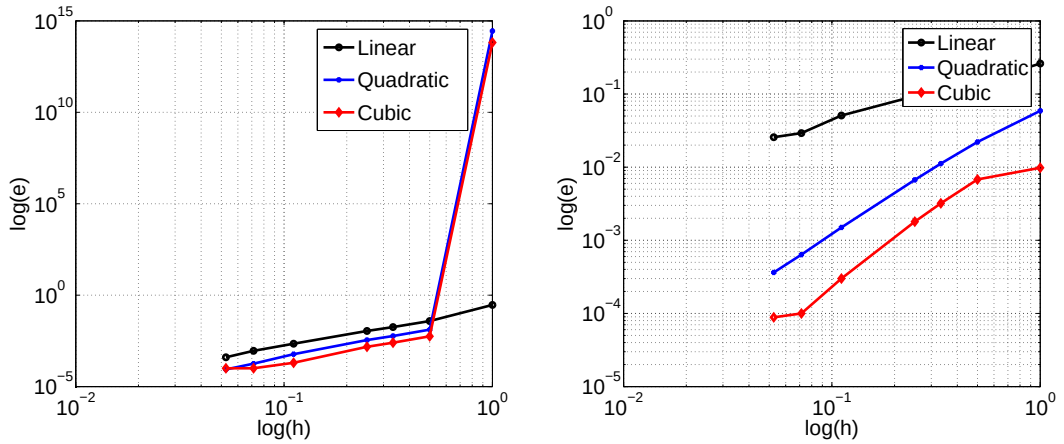
(c) Computed $\frac{\partial u}{\partial y}$

Fig. 59. Computed solution to the Poisson's equation using the first order weight function.



(a) Computed $\frac{\partial u}{\partial x}$ Solution (b) Computed $\frac{\partial u}{\partial y}$

Fig. 60. Zoomed solution using the zeroth order weight function.



(a) Relative error plot using the zeroth order weight function (b) Relative error plot using the first order weight function

Fig. 61. Relative error plot for the PUFEM method.

distance h and increase in the order of basis functions. Further, comparing results of Fig. 61 with those of Figs. 53 and 53 reveals that in case of the PUFEM algorithm, we achieve even better convergence (one order of magnitude). This is also due the fact that even though the inter nodal distance h is same for all three algorithms but the local approximations in the case of the MLPG and the modified MLPG algorithms are computed using nodal points in much larger domain to guarantee well conditioned linear system of equation for each local approximation.

E. Concluding Remarks

In this chapter, three different meshless algorithms are discussed to solve PDEs in an efficient manner. The successful implementation of all three meshless methods depends upon following three main issues:

1. The implementation of the Dirichlet and Neumann boundary condition.
2. The approximation capability of the basis functions used for local approximations.
3. The conditioning of the particular stiffness matrix created by the Galerkin discretization process.

All three algorithm impose essential boundary conditions using penalty terms. Also, all three algorithms allows the inclusion of a priori knowledge about the solution of the PDE in hand by selecting appropriate basis functions for each local approximation. However, in case of the MLPG algorithm one can not increase the order of the basis functions in one particular local region to decrease the approximation errors to a desirable tolerance. To take care of this problem, the conventional MLPG algorithm is modified by using the GLO-MAP averaging process. However, for this modified

algorithm, it was found that the condition number of the stiffness matrix is an issue for some particular configuration of nodal points. To deal with the problems associated with the MLPG and the modified MLPG algorithm, alternative PUFEM algorithms are discussed. The main problem in the implementation of the PUFEM algorithm lies in the selection of the appropriate partition of unity functions. The use of the GLO-MAP weight function is proposed to automatically select the partition of unity functions. The performance of all three algorithms is studied by considering the classical Poisson's equation. Numerical studies show that the new PUFEM algorithm performs better than both the MLPG and the modified MLPG algorithms. Finally, we mention that all three algorithms can, in principle, be extended to handle high dimensioned partial differential equations. All three approaches will be affected to a yet-to-be established degree by the "curse of dimensionality," when high-dimensioned problems are addressed. A more detailed study is required to generalize the algorithms and study their relative merits for the solution of the high dimensioned PDEs.

CHAPTER VII

CONTROL DISTRIBUTION FOR OVER ACTUATED SYSTEMS

A. Introduction

In this chapter, we consider the control distribution problem for highly over-actuated systems which arises with the development of embedded actuation systems. Control distribution is the term used for the control of over-actuated systems. In case of over actuated systems, there is redundancy the total number of actuators to achieve desired total control effort governing equations of motion of the system. For example, in the F-16 aircraft thrust vectoring is used along with conventional control surfaces (aileron, rudder and elevator) to produce six net control force and moment components in accordance with the six degree of freedom equations of motion. Generally, there are several ways to achieve the desired total control effort and control distribution is the process of distributing the total control effort among individual actuators taking into account constraints on individual actuator response and response rate.

In the last one decade there are significant advances in the fields of Micro Electro Mechanical Systems (MEMS), Nano Electro Mechanical Systems (NEMS) and nano bio systems. It is anticipated that advancements in these technologies together will lead to the development of adaptive, intelligent and shape controllable structures for future aircraft and space systems. The design of such advanced system involves control of the shape of the structures with highly redundant micro and nano level manipulations (actuation). Actuators embedded in conventional structural materials at discrete or distributed locations can be used to achieve these objectives by changing ("morphing") surface shape. Currently existing smart structure actuators are Shape Memory Alloys (SMAs), piezoelectric and electrostrictive ceramics,

electro- and magneto-rheological fluids, Synthetic Jet Actuators (SJAs) and active elastomers. Current research activities in nano technologies are aimed at engineering these functionalities into materials at molecular and atomic scales. Such systems can have quite a large number of actuators ($\sim 10^6$) which collectively produce the required control effort and lead to controller design problem in which the number of control components may exceed the degrees of freedom of the system by several order of magnitude. There is no doubt that with such massive redundancy in control variables, one can achieve precise and fault-tolerant control. However, the main challenge lies in developing control approaches that scale efficiently with a large number of control variables. Among the many practical challenges associated with the design of redundant control variables are:

1. Actuator Models: The issue at hand is to derive comprehensive mathematical models that capture the input output behavior of these actuator so that one can derive automatic control laws that can command desired shape and behavior changes. This mapping should also generate an envelope that bounds the maximum reachable control inputs.

2. Dimensionality: Number of actuators vs Degree of Freedom (DoF)

3. Numerical Conditioning: Solving for large number of control variables ($\sim 10^6$) using conventional methods generally lead to ill-conditioned numerical problem.

4. Computational Cost: The controller design must be compatible with near-real-time computing, as ultimately required.

5. Sensing and communication requirements.

In Chapters II and III, non parametric, multi-resolution, adaptive input-output modeling approaches are discussed to capture macro static and dynamic models directly

from experiments which can be used, in principle, for these embedded systems. In Ref. [47], we used the RBF based non-parametric mathematical model to learn the mapping between the SJA parameters (synthetic jet frequency, amplitude, direction, etc. for each SJA) and the resulting aerodynamic lift, drag, and moment. In this chapter, our main interest is to present a general control distribution technique that can be applied for very large scale dynamical systems. This chapter is being written with three main objectives. The first and the most important objective is to present a recursive control distribution approach using adaptive distribution functions to address the issue of dimensionality and computational efficiency. The second objective of this chapter is to establish insight on the implementation of the recursive algorithm and learning different parameters of the distribution functions. The third and final objective of this chapter is to compare the newly developed algorithm with some existing algorithms in terms of computational efficiency and distribution accuracy.

The structure of chapter is as follows: first a problem statement for the control distribution problem is introduced followed by a brief review of some existing control distribution algorithms. Then, a novel recursive control distribution algorithm is introduced and finally, the new algorithm is validated and compared by simulating various test cases.

B. Problem Statement and Background

Let us consider a general dynamical system governed by following differential and algebraic equations:

$$\dot{\mathbf{x}} = f(\mathbf{x}, t) + g(\mathbf{x})\mathbf{u} \tag{7.1}$$

$$\mathbf{y} = h(\mathbf{x}, \mathbf{u}) \tag{7.2}$$

where, $\mathbf{x} \in \mathcal{R}^n$ is the dynamical system state vector, $\mathbf{u} \in \mathcal{R}^p$ is a vector of actuator inputs and $\mathbf{y} \in \mathcal{R}^m$ is a vector of system outputs measured by various sensors on board. The control of dynamical system given by Eqs. (7.1) and (7.2) is defined as follows: *given a model of system dynamics and the desired state trajectory* $\mathbf{x}_d$*, compute the appropriate control vector that will drive the system to the desired state trajectory.* Depending upon the relative size of system state, actuator input vectors, and the controllability of the system, there are three possible outcomes to the control problem:

1. Infinite number of solutions ($p > n$): pick the best one.

2. One unique solution ($p = n$).

3. No solution ($p < n$): find best approximate solution

In this chapter, we are interested in the first case when $p > n$ i.e. over-actuated systems. Conventional linear and nonlinear control methodologies are applicable for only the second case when $p = n$ and control problem is much more complicated for over-actuated systems. To make use of recent advances in the conventional control literature, generally the control problem for the over actuated system is divided into two parts:

1. First, conventional control laws are designed specifying how much total physical control (e.g. resultant forces and moments, also known more generally as *virtual control variable*) effort is required. Equations of motions dictated by Newton's second law are used to derive these control laws and the total number of virtual control variables is generally equal to the rigid body degree of freedoms.

2. In the second step, the total control effort is distributed among individual actuators taking into account various actuator constraints. Algebraic or dynamic

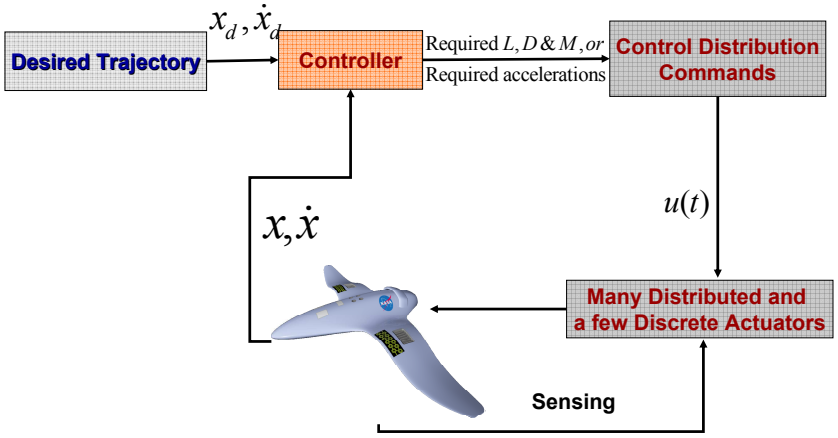


Fig. 62. Control of highly over-actuated system.

models describing the relationship between actuator input and output are used to find the set of actuator inputs that produce the resultant net virtual control.

To illustrate the procedure, let us consider the example of the control of an advanced aircraft with thousands of actuators embedded in the aircraft wing. It is well known fact that aircraft dynamics represents a 6 DoF rigid body and theoretically, one requires 3 forces and 3 moments along yaw, pitch, roll axes to control the aircraft. In the first step, these three forces and moments constitute the 6 dimensional virtual control vector $\mathbf{v}$ which can be solved by using conventional control methodologies and equation of motion governed by Newton's second law of motion. In the second step, the desired virtual control vector $\mathbf{v}$ is allocated among individual actuators to find actual control variables like voltage input to each actuator; this assume that the virtual control vector is physically possible, given the constraints on individual

actuator inputs. This two step process for control of over-actuated system is shown in Fig. 62. According to this two step control methodology the control distribution problem can be defined as follows:

Control Distribution Problem. *Given the desired profile for virtual/physical control vector* $\mathbf{v}(t) \in \mathcal{R}^q$ *find true input vector* $\mathbf{u}(t) \in \mathcal{R}^p$ *such that following is true:*

$$\mathbf{g}(\mathbf{u}(t)) = v(t) \tag{7.3}$$

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}} \tag{7.4}$$

$$\underline{\dot{\mathbf{u}}} \leq \dot{\mathbf{u}} \leq \bar{\dot{\mathbf{u}}} \tag{7.5}$$

where $\mathbf{g}(\mathbf{u}(t))$ represents the transformation from higher dimension space $\mathcal{R}^p$ to lower dimension space $\mathcal{R}^q$. Further, Eqs. (7.4) and (7.5) describes the position and rate constraint on actuator response. It should be mentioned that for successful design of the control distribution algorithm, the accurate knowledge of $\mathbf{g}(.)$ is very essential. The input-output algorithms presented in Chapters II and III can be used to learn the input-output mapping for various kind of actuators. In Ref. [47], we use the DCG algorithm to learn the input-output mapping of synthetic jet actuator directly from experiments. Generally, a linear model is desired between virtual control vector $\mathbf{v}$ and true control vector $\mathbf{u}$ such that Eq. (7.3) can be replaced by the following equation:

$$\mathbf{B}\mathbf{u}(t) = \mathbf{v}(t) \tag{7.6}$$

where $\mathbf{B} \in \mathcal{R}^{q \times p}$ is a matrix with rank q. In case of the linear control distribution problem, the solution lies on the intersection of the hyper-surface $\mathbf{B}\mathbf{u} = \mathbf{v}$ and position control hyper-box defined by Eqs. (7.4) and (7.5). We mention that Eqs. (7.3) and (7.6) are written with the assumption that actuator response is instantaneous i.e. actuator dynamics is negligible which might not be true for many actuators. For

example, as discussed in Refs. [91] the actuator dynamics is certainly present in case of SJAs. However, this assumption is valid to a degree of approximation, if the closed loop system is designed to be substantially slower than the actuator dynamics.

In Ref [92], numerous advantages of dividing the control problem in two steps have been discussed in detail. Practically, dividing the control problem in two steps allows us to exploit individual actuators to their full level without degrading the closed loop performance of the controller design in the first step. Further, actuator constraints, saturation and failure can be handled more efficiently. If one actuator saturates, and fails then another actuator may be used to make up the difference. In another words, the reconfiguration of different actuators can be performed in the event of an actuator failure, without redesigning the control law in first step. Another main advantage is that actuator utilization can be optimized independently for specific application in mind. For example, thrust vectoring can be used as auxiliary control to obtain high maneuverability. Similarly, the use of trailing edge SJAs is preferable at low Angle of Attack (AOA) while the leading edge SJAs are useful in case of flow separation.

Many algorithms [92,93] have been suggested in the literature to solve the control distribution problem for over actuated systems. The generalized inverse known as the pseudo-inverse is frequently used method to compute the solution for control distribution problem. Generalized inverse solution is obtained by minimizing 2-norm of the true control vector $\mathbf{u}$ subject to constraint given by Eq. (7.6). In the absence of any constraints on control variable $\mathbf{u}$, the explicit generalized inverse solution is given by the following equation:

$$\mathbf{u} = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{v} \tag{7.7}$$

More computationally robust minimum norm solution can be computed using SVD algorithms. In a more general approach, the following optimization problem is defined to solve the control distribution problem:

$$Minimize: \ \|\mathbf{u}\|_p \tag{7.8}$$

subject to

$$\mathbf{Bu} = \mathbf{v} \tag{7.9}$$

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}} \tag{7.10}$$

$$\underline{\dot{\mathbf{u}}} \leq \dot{\mathbf{u}} \leq \bar{\dot{\mathbf{u}}} \tag{7.11}$$

where, $\|\mathbf{u}\|_p$ is the p norm of vector $\mathbf{u}$ defined as follows:

$$\|\mathbf{u}\|_p = \left(\sum_{i=1}^{p} |u_i|^p \right)^{\frac{1}{p}} \tag{7.12}$$

Most commonly used norms are 1 and 2 norm. Although a variety of norm definitions can be used to solve the control distribution problem, the use of different norms lead to different numerical methods to solve the problem and the suitability of a given norm will be dictated by the viability of the algorithm and the physical characteristics of the resulting control distribution problem. The use of 2-norm leads to the quadratic optimization problem whereas the use of 1-norm or ∞-norm leads to the linear programming problem. Generally, active set methods [94, 95], primal and dual simplex methods [96, 97] and interior point optimization methods [98] are used to solve linear and quadratic optimization problems. Active set methods solves an optimization problem by partitioning inequality constraints into two sets: active and inactive. The inactive constraints are ignored while solving the problem whereas the active set constitutes the working set for the solution at any given step. Then,

the problem is solved by moving on the surface defined by the working set. These methods search for a solution along the edges and faces of the feasible set by solving a sequence of equality-constrained quadratic programming problems. On other hand, the conventional simplex method solves the optimization problem by searching from vertex to vertex on the boundary of the feasible polyhedron, repeatedly increasing the objective function until either an optimal solution is found, or it is established that no solution exists. In principle, the time required might be an exponential function of the number of variables, and this can happen in some contrived cases.

To deal with the problems associated with the simplex method, interior point optimization methods are used. Unlike the simplex method, interior point optimization methods do not search for the solution from vertex to vertex, but search only through the interior of the feasible region. In brief, active set and interior point methods differ from the simplex method that the solution in this case need not to be on vertices of the feasible set. Generally, active set methods are used to solve the quadratic optimization problem while simplex method is mainly used to solve linear programming problem. Quadratic optimization problems are more difficult to solve than linear programming problem, because unlike the solution to linear programming problem, the quadratic problem solution may use all variables of the problem. Therefore, the 2-norm solution tries to distribute the total control effort among all of the control inputs whereas the 1-norm solution utilizes as few control variables as possible and may lead to the saturation of many control variables. Finally, we mention that the percentage of attainable control effort using the optimization problem solution can be quite small, depending on a number of factors, including the number of control variables and the definition of the norm used in the optimization problem [99]. Also, most of these numerical methods can handle an unlimited number of variables and constraints, subject to the availability of computer time, memory and numerical con-

ditioning of the particular application. Practical experience tells us, however, that solving a large scale optimization problem is not desirable for real-time problems.

For very large scale systems, the use of a hierarchical approach, known as *daisy chain* [100], is discussed. The method of daisy chain uses the heuristic logic to divide the p control inputs into P groups $\{\mathbf{u}^1, \mathbf{u}^2, \cdots, \mathbf{u}^P\}$. Initially, the control variable in first ("primary") group $\mathbf{u}^1$ are used i.e. $\mathbf{B}^1\mathbf{u}^1 = \mathbf{v}$. If the primary control variables in $\mathbf{u}^1$ satisfies all the constraints of Eqs. (7.4)-(7.6) then the distribution is successful. Otherwise, the control variables in the secondary group $\mathbf{u}^1$ which violate constraints of Eqs. (7.4) and (7.5) are saturated and control variables in group $\mathbf{u}^2$ are used to provide the rest of the control effort i.e. control effort equal to the difference between the total control effort $\mathbf{v}$ and the control effort produced by control variables in group $\mathbf{u}^1$. This procedure is recursively repeated until desired control effort is produced or all control variables are used. The main advantage of the daisy chain method is that primary and secondary actuators can be expected to be used most frequently and the higher control groups are used only when necessary. On other hand the main disadvantage of this approach is that this procedure does not take into account the actuator constraints directly and employs the simple heuristic of saturating the actuators where they are commanded more control effort than their physical limit. Generally, saturation of actuators is not desirable for many problems. Further, for a very large scale system this approach is not desirable as the number of control variables in each group and number of groups can be quite large for such systems and make this algorithm computationally inefficient.

In Refs. [93, 101], different approaches to establish hierarchical algorithms are explored for control allocation in a large scale distributed system. The hybrid algorithm approach is based upon a *divide and conquer* method and works by breaking a high-dimensional problem into a number of smaller problems that can either be re-

duced in size or solved using optimization algorithms discussed earlier in this section. The optimization algorithms can range from discrete optimal search to continuous constrained optimization problem. The method discussed in Ref. [101] is tested for control distribution problem among hundreds of actuators to control the translation of a sheet of paper over a air-jet table. The optimal algorithm yields low errors but the time required to compute the solution varies exponentially with the number of actuators as standard optimization algorithms are used to find the discrete control variables at various scales. Therefore, even though the hierarchical algorithm discussed in Refs. [93, 101] is shown to work for reasonably large scale problems, they are not computationally efficient for very large scale problems where the number of control variables can be in millions. For example, let us consider a problem of control distribution among 10^6 actuators. In this case, even though one divides this highly redundant actuation problem into 1000 small problems then also each small problem has 1000 optimization variables to be solved for, which can be computationally very inefficient. Here, we propose the use of *distribution functions* to reduce the number of control variables by a order of magnitude. The main idea is to approximate the feasible solution set by making use of continuous functions. These functions are defined by a few parameters and spatially distribute the controls by interpolating the inputs to each discrete actuator. However, if one uses a global set of distribution functions then the number of distribution functions can be very high to have a reasonable approximation of the feasible set. Therefore, a hierarchical approach is used to improve the accuracy of the feasible set and as well as to deal with the issue of high dimensionality.

In this chapter, our main focus is to design an efficient control distribution algorithm to generate commands to a highly redundant system in real-time. To deal with the issues of high dimensionality, a hierarchical approach is proposed which makes

use of specially designed distribution functions for control distribution purposes. In rest of this chapter, the concept of distribution functions and the detailed description of the proposed hierarchical algorithm are discussed.

C. Control Distribution Functions

In the previous section, a brief introduction to various control distribution algorithms is given. Theoretically, each algorithm has the ability to handle an unlimited number of control variables subject to the availability of time, processing power and memory. However, in case of a very large scale distribution problem, most of these algorithms fail to compute the solution given practical limitations on processing power, time and memory. Basically, the main problem is the lack of a tool to reduce the dimensionality of the problem to the desired order so that the given problem can be solved with a modest computation burden, we introduce the idea of distribution functions to approximate the control effort.

Let us consider a general problem of distributing virtual/physical control vector $\mathbf{v}(t) \in \mathcal{R}^q$ among true input vector $\mathbf{u}(t) \in \mathcal{R}^p$ such that following is true:

$$\mathbf{B}\mathbf{u}(t) = v(t) \tag{7.13}$$

$$\underline{\mathbf{u}} \leq \mathbf{u} \leq \bar{\mathbf{u}} \tag{7.14}$$

$$\underline{\dot{\mathbf{u}}} \leq \dot{\mathbf{u}} \leq \bar{\dot{\mathbf{u}}} \tag{7.15}$$

where, $\mathbf{v}(t)$ denotes the relevant physical forces which depend on actual displacement and velocity error vectors and $\mathbf{u}(t)$ is a vector of actual control inputs. As mentioned earlier, we neglect the actuator dynamics assuming that actuator dynamics is much faster than the closed loop dynamics of the system under consideration. Further, rate constraints given by Eq. (7.15) can be converted into position constraints by

approximating $\dot{\mathbf{u}}(t)$ by first order finite difference approach:

$$\underline{\dot{\mathbf{u}}} \leq \frac{\mathbf{u}(t) - \mathbf{u}(t^-)}{\Delta t} \leq \bar{\dot{\mathbf{u}}} \tag{7.16}$$

$$\underline{\dot{\mathbf{u}}}\Delta t + \mathbf{u}(t^-) \leq \mathbf{u}(t) \leq \bar{\dot{\mathbf{u}}}\Delta t + \mathbf{u}(t^-) \tag{7.17}$$

where, $t^- = t - \Delta t$. As a consequence of this, the control distribution problem can be considered with constraint on actuator response $\mathbf{u}(t)$ only.

$$\mathbf{u}_l(t) \leq \mathbf{u}(t) \leq \mathbf{u}_u(t) \tag{7.18}$$

where,

$$\mathbf{u}_l(t) = \min\left(\underline{\mathbf{u}}, \underline{\dot{\mathbf{u}}}\Delta t + \mathbf{u}(t_1)\right) \tag{7.19}$$

$$\mathbf{u}_u(t) = \max\left(\bar{\mathbf{u}}, \bar{\dot{\mathbf{u}}}\Delta t + \mathbf{u}(t_1)\right) \tag{7.20}$$

Now, the optimization problem to solve the control distribution problem can be re-defined as follows:

$$\min_{\Omega(t)} : \|\mathbf{u}\|_p \tag{7.21}$$

subject to

$$\mathbf{B}\mathbf{u}(t) = \mathbf{v}(t) \tag{7.22}$$

$$\mathbf{u}_l(t) \leq \mathbf{u}(t) \leq \mathbf{u}_u(t) \tag{7.23}$$

where, $\Omega(t)$ denotes the set of feasible solutions and depends upon the total physical/virtual control effort $\mathbf{v}(t)$. Now, assuming that all actuators are spatially distributed over some surface and there exists a set of distribution functions $\boldsymbol{\phi}(\mathbf{x}) = \{\phi_i(\mathbf{x})\}$ such that

$$span(\boldsymbol{\phi}(\mathbf{x})) = \Omega(t) \tag{7.24}$$

where, $\mathbf{x} \in \mathcal{R}^M$ is a vector of spatial coordinates. We mention that generally M is

equal to 2 or 3. As a consequence of Eq. (7.24), any feasible solution $\mathbf{u}(t)$ can be approximated as a linear combination of distribution functions $\phi_i(\mathbf{x})$.

$$u(\mathbf{x},t) = \sum_{i=1}^{N} a_i(t)\boldsymbol{\phi}_i(\mathbf{x}) \tag{7.25}$$

where $\mathbf{a}(t) \in \mathcal{R}^N$ is a vector consist of amplitudes of various distribution functions $\boldsymbol{\phi}_i \in \mathcal{R}^N$. Now, the true control vector $\mathbf{u}$ can be written as:

$$\mathbf{u} = \boldsymbol{\Phi}(\mathbf{x})\mathbf{a} \tag{7.26}$$

where,

$$\boldsymbol{\Phi} = \begin{bmatrix} \phi_1(\mathbf{x}_1) & \phi_2(\mathbf{x}_1) & \cdots & \phi_N(\mathbf{x}_1) \\ \phi_1(\mathbf{x}_2) & \phi_2(\mathbf{x}_2) & \cdots & \phi_N(\mathbf{x}_2) \\ \vdots & & \ddots & \vdots \\ \phi_1(\mathbf{x}_p) & \phi_2(\mathbf{x}_p) & \cdots & \phi_N(\mathbf{x}_p) \end{bmatrix} \tag{7.27}$$

and, $\mathbf{x}_i$ denotes the spatial coordinates for the i^{th} control variable u_i.

Further, substituting for $\mathbf{u}(t)$ from Eq. (7.26) in Eqs. (7.21)-(7.23), we get following optimization problem for the amplitude vector $\mathbf{a}$

$$\min : \|\boldsymbol{\Phi}(\mathbf{x})\mathbf{a}\|_p \tag{7.28}$$

subject to

$$\mathbf{B}\boldsymbol{\Phi}(\mathbf{x})\mathbf{a} = \mathbf{H}(\mathbf{x})\mathbf{a} = \mathbf{v} \tag{7.29}$$

$$\mathbf{u}_l \leq \mathbf{u} \leq \mathbf{u}_u \tag{7.30}$$

It should be noticed that in this case one needs to solve for N-dimensional amplitude vector $\mathbf{a}(t)$ as compared to the p-dimensional true control variable vector $\mathbf{u}(t)$ in case of the optimization problem described by Eqs. (7.21)-(7.23). So the reduction in the

dimensionality of the problem depends upon the relative values of p and N. A key question regarding the selection of distribution functions $\phi_i(.)$ is "How irregular is the feasible solution set $\Omega(t)$?". A globally valid set of distribution functions should be sufficient if $\Omega(t)$ is well connected set and all feasible solutions are globally smooth. However, if $\Omega(t)$ is a complicated set or in the presence of high frequency local features in the feasible solution set, a more judicious selection of distribution functions will be required. While the brute force approach of using infinitely many basis functions is a theoretical possibility, it is intractable in a practical application because such an optimizer will have far too many parameters to determine and will not give any advantage in terms of dimensionality reduction. As discussed in Chapters II-IV, we consider Radial Basis Functions (RBF) and global/local orthogonal polynomial basis functions as possible candidates for control distribution functions.

1. Radial Basis Functions

As discussed in Chapter II, RBF are the basis functions whose response decreases monotonically with the increase in radial distance from their center location and can be confined to a local region around their center location $\boldsymbol{\mu}$. Among many choices for the radial basis functions, the Gaussian function is the most widely used because, among other reasons, the different parameters appearing in its description live in the space of inputs and have physical and heuristic interpretations that allow good starting estimates to be locally approximated. The most general Gaussian function can be written as:

$$\phi_i(\|\mathbf{x}-\boldsymbol{\mu}_i\|,\boldsymbol{\sigma}_i,\mathbf{q}_i)=\exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\mathbf{R}^{-1}(\boldsymbol{\sigma}_i,\mathbf{q}_i)(\mathbf{x}-\boldsymbol{\mu}_i)\} \tag{7.31}$$

where, $\boldsymbol{\mu}_i \in \mathcal{R}^M$ represents the center location of the Gaussian basis function ϕ_i whereas $\mathbf{R} \in \mathcal{R}^{M\times M}$ is a positive definite symmetric matrix which describes shape

and size of a Gaussian basis function. Now, substituting for $\phi_i(.)$ from Eq. (7.31) in Eq. (7.26), we get:

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^{N} \phi_i(\|\mathbf{x} - \boldsymbol{\mu}_i\|, \boldsymbol{\sigma}_i, \mathbf{q}_i) a_i \tag{7.32}$$

Therefore, we need to solve for following parameters to use Gaussian basis functions as control distribution functions:

1. M parameters for the centers of the Gaussian function i.e. $\boldsymbol{\mu}_i$.
2. M parameters for the spread (shape) of the Gaussian function i.e. $\boldsymbol{\sigma}_i$.
3. $\frac{n(n+1)}{2}$ parameters for rotation of the principal axis of the Gaussian function i.e. $\mathbf{q}_i$.
4. Amplitude a_i of the Gaussian function ϕ_i.

The main problem with the use of the Gaussian functions as control distribution functions is that, except the amplitude vector, the various other parameters appear nonlinearly in Eq. (7.32) and necessitate the use of a nonlinear optimization algorithm to solve for their optimal value. The use of nonlinear optimization algorithm may not be desirable for many practical reasons. To simplify the problem, one can pre-define "good choice" of the various parameters except the amplitude vector $\mathbf{a}$ whose optimal value can be found by solving the simpler algebraic optimization problem defined by Eqs. (7.28)-(7.30). The centers $\boldsymbol{\mu}_i$ for various Gaussian basis functions can either be distributed uniformly over the input space or they can be selected to make use of some a-priori information about the grouping of actuators. Further, the spread parameter vector $\boldsymbol{\sigma}_i$ can be chosen proportional to the shortest distance between $\boldsymbol{\mu}_i$ and the existing centers

$$\boldsymbol{\sigma}_i = \kappa \|\boldsymbol{\mu}_i - \boldsymbol{\mu}_{nearest}\| \tag{7.33}$$

where κ is a user-defined parameter which accounts for the amount of overlap between different Gaussian functions. The rotation parameter $\mathbf{q}_i$ can be assumed to zero until some information is available on the control distribution surface. In the limiting case when $\kappa \rightarrow 0$ the Gaussian basis function approaches a dirac-delta function and in this particular case, one would like to choose as many basis functions as the number of control variables i.e. $p = N$. As a consequence of this, we get back the original optimization problem defined by Eqs. (7.28)-(7.30). Experience indicated that one would like to choose the parameter κ such that two neighboring basis functions overlap by at least 50%. Finally, one can iterate on the number of basis functions N depending upon whether a feasible solution exist or not. This provides a good compromise between "local dominance" and "trend sensing" of the RBF model. Initially, one can choose small number of distribution functions distributed uniformly in spatial coordinates $\mathbf{x}_i$ and if a feasible solution is not found then one can keep on increasing the number of basis functions until a feasible solution is obtained. The outline of the control distribution algorithm using RBF function is shown in Fig. 63

2. Global/Local Orthogonal Basis Functions

In Chapter III, we introduced the idea of the Global/Local Orthogonal MAPping (GLO-MAP) algorithm to approximate irregular surfaces. The same idea can be to interpolate an irregular control distribution surface $\mathbf{u}(\mathbf{x}, t)$ with all the global and local advantages of the GLO-MAP approach to approximation. Introducing a set of grid points $\{\bar{\mathbf{x}}_i\}_{i=1}^{Q}$ as approximation vertices having associated with weight function w_i and local approximations $\boldsymbol{\psi}_i(\mathbf{x}, \bar{\mathbf{x}}_i)$, we can approximate unknown control distribution surface $\mathbf{u}(\mathbf{x}, t)$ as follows:

$$\mathbf{u}(\mathbf{x}, t) = \sum_{i=1}^{Q} w_i(\mathbf{x}, \bar{\mathbf{x}}_i)\boldsymbol{\psi}_i(\mathbf{x}, \bar{\mathbf{x}}_i) \tag{7.34}$$

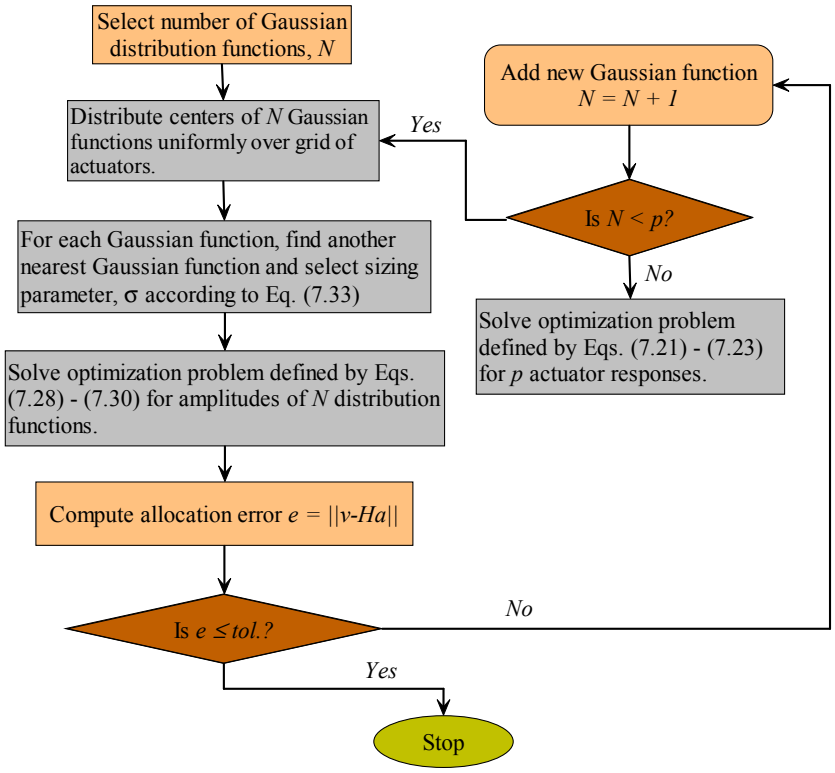


Fig. 63. Flow chart for the control distribution algorithm using RBF distribution functions.

where, $w_i(.)$ represents specially designed GLO-MAP weight function and local approximation $\boldsymbol{\psi}_i(.)$ can be written as a linear combination of N polynomial basis functions ϕ_{l_j}:

$$\boldsymbol{\psi}_i(\mathbf{x}, \bar{\mathbf{x}}) = \sum_{j=1}^{N} a_{ij}(t)\phi_{l_j}(\mathbf{x}, \bar{\mathbf{x}}_i) = \boldsymbol{\phi}_l^T(\mathbf{x}, \bar{\mathbf{x}}_i)\mathbf{a}_i \tag{7.35}$$

where, $\phi_{l_j}(.)$ can be chosen as the orthogonal polynomials of the GLO-MAP process. Now, substitution of Eq. (7.35) in Eq. (7.34) leads to the following equation for the feasible control distribution surface $\mathbf{u}(\mathbf{x}, t)$:

$$\mathbf{u}(\mathbf{x}, t) = \boldsymbol{\phi}(\mathbf{x}, \bar{\mathbf{x}}_i)\mathbf{a}(t) \tag{7.36}$$

where $\boldsymbol{\phi}(.)$ is a vector of various distribution functions and $\mathbf{a}$ is a vector of corresponding amplitudes:

$$\boldsymbol{\phi}(\mathbf{x}, \bar{\mathbf{x}}_i) = \left\{ \begin{array}{cccc} w_1(\mathbf{x}, \bar{\mathbf{x}}_1)\boldsymbol{\phi}_l(\mathbf{x}, \bar{\mathbf{x}}_1) & \cdots & w_Q(\mathbf{x}, \bar{\mathbf{x}}_Q)\boldsymbol{\phi}_l(\mathbf{x}, \bar{\mathbf{x}}_Q) \end{array} \right\} \tag{7.37}$$

$$\mathbf{a}(t) = \left\{ \begin{array}{cccc} \mathbf{a}_1(t) & \cdots & \mathbf{a}_Q(t) \end{array} \right\} \tag{7.38}$$

Now, making use of Eq. (7.37), the optimization problem to solve for total NQ amplitude variables associated with various distribution functions can be defined as follows:

$$\min : \|\boldsymbol{\Phi}(\mathbf{x}, \bar{\mathbf{x}}_i)\mathbf{a}\|_p \tag{7.39}$$

subject to

$$\mathbf{H}\mathbf{a} = \mathbf{v} \tag{7.40}$$

$$\mathbf{u}_l \leq \boldsymbol{\Phi}(\mathbf{x}, \bar{\mathbf{x}}_i)\mathbf{a} \leq \mathbf{u}_u \tag{7.41}$$

where, $\mathbf{H} = \mathbf{B}\mathbf{\Phi}(.)$ and $\mathbf{\Phi}(.) \in \mathcal{R}^{p \times NQ}$ is given by the following equation:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_1(\mathbf{x}_1, \bar{\mathbf{x}}) & \phi_2(\mathbf{x}_1, \bar{\mathbf{x}}) & \cdots & \phi_{NQ}(\mathbf{x}_1, \bar{\mathbf{x}}) \\ \phi_1(\mathbf{x}_2, \bar{\mathbf{x}}) & \phi_2(\mathbf{x}_2, \bar{\mathbf{x}}) & \cdots & \phi_{NQ}(\mathbf{x}_2, \bar{\mathbf{x}}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\mathbf{x}_p, \bar{\mathbf{x}}) & \phi_2(\mathbf{x}_p, \bar{\mathbf{x}}) & \cdots & \phi_{NQ}(\mathbf{x}_p, \bar{\mathbf{x}}) \end{bmatrix} \tag{7.42}$$

Depending upon the norm selected, the above algebraic optimization problem can be easily solved for the finite dimensioned amplitude vector $\mathbf{a}$ and thereby affect the control distribution. So, the problem of finding p control variables has reduced to finding the amplitudes of NQ distribution functions. Note that $Q = 1$ corresponds to the problem of finding a global distribution surface while as Q increases the distribution functions becomes more capable of approximating local features of the underlying distribution surface $\mathbf{u}(\mathbf{x}, t)$. Ideally, one would like to choose N and Q such that a substantial dimensionality reduction results ($NQ \ll p$). Initially, one can start with a global distribution surface described by N distribution functions and if feasible solution is not found to the optimization problem described by Eqs. (7.39)-(7.41) then more local approximations can be introduced by increasing N until a feasible solution is found. It should be mentioned that the GLO-MAP process provides a zeroth level hierarchy in the control distribution and allows us to make distribution decisions at various scales. The outline of the control distribution algorithm using GLO-MAP process is shown in Fig. 64

Finally, it should be noticed that the success of both the algorithms depends upon how well the basis functions span the feasible solution set $\Omega(t)$ and the performance of both the algorithms is dictated by the total number of distribution functions required to have a reasonable approximation of $\Omega(t)$. Although the iterative nature of both the algorithms seeks a good approximation of the feasible solution set $\Omega(t)$ with

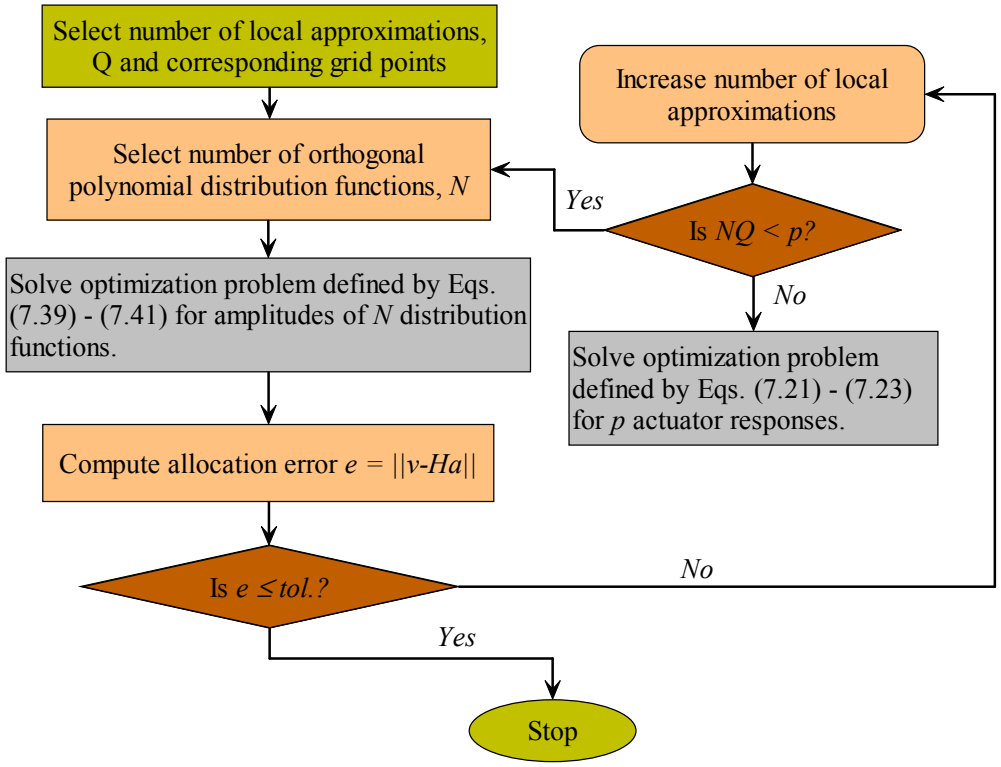


Fig. 64. Flow chart for the control distribution algorithm using the GLO-MAP orthogonal polynomials as distribution functions.

a minimal number of distribution functions, there may remain cases when both the algorithms fail to provide a feasible solution. We mention that the occasional failure of distribution function approach can be attributed to the irregularity of the feasible solution set which is sometime difficult to anticipate in high-dimensioned nonlinear problems. In the next section, we describe a hierarchical control distribution approach that makes use of these algorithms to deal with this occasional failure problem. The hierarchical approach not only allows a multi-resolution approximation of the feasible solution set $\Omega(t)$ but it also provides a mechanism for parallelizing the control distribution algorithms.

D. Hierarchical Control Distribution Algorithm

In the previous section, we introduced the concept of distribution functions to approximate the feasible solution set $\Omega(t)$ and it is shown that how the use of distribution functions improves the performance of the control distribution algorithm while keeping in check the "curse of dimensionality". However, the improvement in computational speed is generally accompanied by the degradation of the optimal solution due to errors in approximating the feasible solution set. In many cases, this degradation may not matter, especially until we get a feasible, sub-optimal solution. But, in some cases these approximation errors can lead to a situation when distribution functions fail to approximate the feasible solution set at all. This kind of failure may arise due to the irregularity of the feasible solution set $\Omega(t)$. As discussed in Chapters III and IV, decreasing the domain of validity of different distribution functions may lead to the improvement in the approximation error and consequently, increase the region of validity of the distribution function approach. In this section, we discuss hierarchical control distribution algorithm to improve the performance of the distribution

function approximation of the feasible solution set $\Omega(t)$ with the goal of computing the feasible solution, if it exists, with minimal computation. We mention that in the worst case scenario the proposed algorithm requires as much resources as any other conventional control distribution algorithm does.

The proposed hierarchical method decomposes the large scale control distribution problem in to many regional control distribution problems to compromise the need for real-time computation against optimality. We mention that the proposed algorithm is inspired by the work of Fromherz et al. [101], Luntz et al. [93] and Jackson et al. [102]. The main steps of the algorithms can be summarized as follows:

1. Group spatially distributed actuators to generate finite number (say, G) of small scale (regional) subsets to take advantage of regionally correlated control input distributions.

2. Combine the effects of the actuators in a particular group to form an "aggregated" actuator which represents all the actuators of that group. The discrete spatial coordinates associated with "aggregated" actuator can be taken as the mean position (centroid) of various actuators it represents.

3. Distribute total control effort among G "aggregated" actuators. In another words, assign responsibility to each "aggregated" actuator to produce total control effort.

4. Solve the control distribution problem recursively with the help of adaptive distribution functions in each subset.

In the previous section, we have already developed the procedure for step 4. However, the main issues with this approach is the implementation of steps 1 and 2. Basically, the problem is how to aggregate different actuators and then combine them to form

an "aggregated" actuator. Like in Refs. [93, 101, 102], a simple hierarchy scheme will involve the grouping of actuators on the basis of their spatial coordinates. If control effectiveness for each actuator is same or is a function of spatial coordinates then according to this approach, the actuators are grouped on the basis of their control effectiveness. For example, if consider the example of SJA actuators distributed spatially over the aircraft wing then their control effectiveness is the function of their frequency and the spatial position of the actuators [91]. Further, this hierarchical approach facilitates the use of distribution functions inside each subset to solve the control distribution problem recursively.

Once the actuators are grouped together then the next step is to come up with an "aggregated" effective actuator which conveys some averaged information about the whole group. To form an "aggregated" actuator by combining various actuators in a group, we need following information about the collective response of the whole group:

1. First, the "aggregated" actuator needs to represent the response of all the actuators in a particular group in some average sense.

2. Secondly, the constraints on the "aggregated" actuator should contain the information about the constraints on each individual actuator.

3. Finally, the weighting factors for each "aggregated" actuator's contribution to the total physical control effort is required. These weights provide an opportunity to account heuristically for the capability of each group during control distribution among each group (e.g., the type and number of actuators in each group).

The first two sets of information are easy to obtain. Usually, the response of the whole group is assumed to be the sum of the response of each individual actuators.

Let $\mathbf{u}^i = \{u_1, u_2, \cdots, u_l\}$ is a vector of the control responses of all the actuator in the i^{th} group then the combined/aggregated contribution of the i^{th} group can be written as:

$$\mathbf{v}^i = \mathbf{B}^i \mathbf{u}^i \tag{7.43}$$

where $\mathbf{B}^i$ is a aggregated control distribution ("influence") matrix for i^{th} actuator group and consists of rows of the original $\mathbf{B}$ matrix.

$$\mathbf{B}^i = \begin{bmatrix} \mathbf{B}_1^T & \mathbf{B}_2^T & \cdots & \mathbf{B}_l^T \end{bmatrix} \tag{7.44}$$

where $\mathbf{B}_i$ is the i^{th} row of control distribution matrix $\mathbf{B}$. Further, the constraints on the aggregated group response can be found by taking the average value of the constraints on each individual actuators. Let $\bar{\mathbf{u}}_i$ and $\underline{\mathbf{u}}_i$ are vectors of upper and lower limit on actuators in i^{th} group. Now, the constraints on aggregated response can be given as follows:

$$\mathbf{v}_l^i \leq \mathbf{v}^i \leq \mathbf{v}_u^i \tag{7.45}$$

where,

$$\mathbf{v}_l^i = \min\left(\mathbf{B}^i \underline{\mathbf{u}}^i, \mathbf{B}^i \bar{\mathbf{u}}^i\right) \tag{7.46}$$

$$\mathbf{v}_u^i = \max\left(\mathbf{B}^i \underline{\mathbf{u}}^i, \mathbf{B}^i \bar{\mathbf{u}}^i\right) \tag{7.47}$$

Finally, third and the most important part is to find the appropriate weighting function to give proper weighage to the aggregated information of each group so that the required physical control effort can be divided among each group. Generally, weighting function is chosen as the inverse of number of actuators in that particular group or depending upon the measure of controllability of the aggregated group. We choose 2-norm of control distribution matrix as a measure of controllability for the aggregated group.

Further, let $\mathbf{v}_G$ is a vector of aggregated responses of each group i.e. $\mathbf{v}_G = \{\mathbf{v}^1, \mathbf{v}^2, \cdots, \mathbf{v}^G\}$. Now, the aggregated response vector $\mathbf{v}_G$ can be solved by posing following optimization problem:

$$\min : \|\mathbf{W}\mathbf{v}_G\|_p \tag{7.48}$$

subject to

$$\underline{\mathbf{v}}_G \leq \mathbf{v}_G \leq \bar{\mathbf{v}}_G \tag{7.49}$$

$$\mathbf{v} = \sum_{i=1}^{G} \mathbf{v}^i \tag{7.50}$$

where,

$$\mathbf{W}_{ij} = \|\mathbf{B}_i\|\delta_{ij} \tag{7.51}$$

$$\underline{\mathbf{v}}_G = \left\{ \begin{array}{cccc} \mathbf{v}_l^1 & \mathbf{v}_l^2 & \cdots & \mathbf{v}_l^G \end{array} \right\} \tag{7.52}$$

$$\bar{\mathbf{v}}_G = \left\{ \begin{array}{cccc} \mathbf{v}_u^1 & \mathbf{v}_u^2 & \cdots & \mathbf{v}_u^G \end{array} \right\} \tag{7.53}$$

It should be noticed that a large weighting factor $\mathbf{W}_{ii}$ causes the i^{th} group to have a greater role in meeting the total virtual force vector $\mathbf{v}$. Further, depending upon the definition of norm in Eq. (7.48), various numerical methods as discussed in the last section can be used to find aggregated group contribution vector $\mathbf{v}_G$.

Once the total control effort is distributed among various groups then the following optimization problem is solved using the distribution function method as discussed in the previous section.

$$\min : \|\mathbf{u}^i\|_p \tag{7.54}$$

subject to

$$\underline{\mathbf{u}}^i \leq \mathbf{u}^i \leq \bar{\mathbf{u}}^i \tag{7.55}$$

$$\mathbf{B}^i \mathbf{u}^i = \mathbf{v}^i \tag{7.56}$$

Now, we present a generic algorithm for hierarchical control allocation. The first major step of this algorithm is the task of grouping of different actuators. The grouping of actuators may be pre-determined or done in real time while solving the control distribution problem. We use a hierarchical approach for the grouping of different actuators depending upon their spatial coordinates. According to the hierarchical grouping algorithm, first, there is only one group consisting of all the actuators and the control distribution problem is solved by using the RBF or the global/local orthogonal polynomial distribution functions as discussed in section C. If a feasible solution is not found for the control distribution problem, then the actuators are divided into two groups. First, the required control effort is distributed among these two groups by solving the optimization problem of Eqs. (7.48)-(7.50). If a feasible solution to this optimization problem does not exist, then each group is further divided into two sub-groups else within each group another optimization problem of Eqs. (7.54)-(7.56) is solved to compute the response of each actuator. Further, if we fail to compute a feasible solution in a particular group then that particular group is again divided into two sub-groups. Also, if in a particular group the number of distribution functions N exceeds the number of actuators contained in that group then the control distribution problem for that particular group is solved for control variables instead of amplitudes of distribution functions. This whole process is repeated recursively until a feasible solution is found or all control variables are solved for simultaneously. The flowchart for this hierarchical control distribution algorithm is shown in Fig. 65.

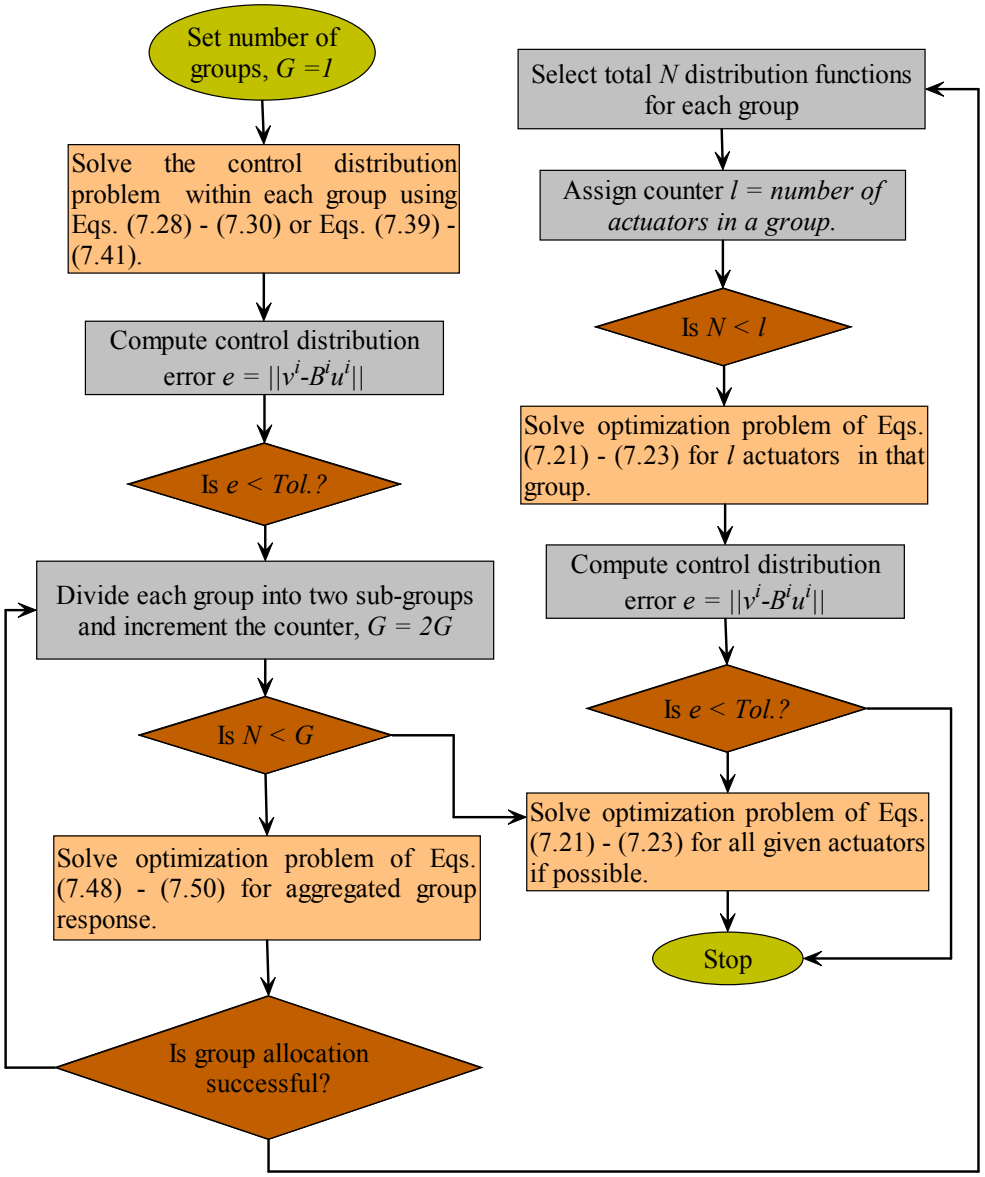


Fig. 65. Flow chart for the hierarchical control distribution algorithm.

We mention that for real-time implementation, one might need to compromise between computational time and allocation errors. Also, based upon some previous experience or knowledge of the system, the number of groups can be pre-determined and "frozen" to save some computational time. The main advantage of this hierarchical approach is that the control distribution in each group is decoupled from the distribution problem in other groups and thus this algorithm can be highly parallelized to reduce the elapsed computation time required. Beside this, there are many other advantages of this kind of hierarchical approach. First, the distributed nature of the actuators can be fully exploited without having the dimensionality of the optimization problem approach infinity. Secondly, in case of actuator failures in one particular group, the redistribution can be adaptively performed without affecting the distribution in all groups. Finally, actuator utilization can be optimized independently for specific applications and, in principle, changed adaptively, on-the-fly e.g.: Leading edge actuation may be preferable at high Angle Of Attack (AOA) while trailing edge actuation may be best for low AOA; with sufficient intelligence or rule-based logic, adaptive algorithms may be able to automatically shift emphasis of the control allocation in real time.

E. Numerical Results

The proposed control distribution algorithm is tested on a simulated control allocation problem for a morphing wing embedded with millions of hypothetical actuators. In this section, some results from these studies are presented.

1. Control Allocation For a Morphing Wing

There is a significant thrust in aerospace industry to develop advanced technologies that would enable adaptive, intelligent, shape controllable micro and macro structures, for advanced aircraft and space systems. These designs involve precise control of the shape of the structures with micro and macro level manipulations (actuation). In pursuit of these objectives, a novel morphing wing is being designed and built that can achieve an infinity of different configurations upon command. The morphing wing represents an alternative technology that adaptively shapes the flow and pressure fields over the wing by changing the curvature of the wing. This morphing technology could lead to replacement of hinged control surfaces thereby achieving hingeless control. This morphing of the wing can be achieved by embedding actuators at micro scales of an aerodynamic structure. The desired force and moment profile are achieved by generating moments using these actuators to deform the geometry and thereby creating a desired flow and pressure distribution over the surface.

As part of initial effort, a first prototype of the morphing wing is built and installed in the $3' \times 4'$ wind tunnel of the Texas A&M Aerospace Engineering Department (Fig. 66). The wing is built by using ABS plastic structure material supported by telescopic tubes and the skin over the wing is made of silicone rubber elastomer mixed with a small quantity of Carbon Nano-Tubes (CNT) to tailor the structure for the desired stiffness in bending and torsion. While micro sensors are being developed, we test the idea of morphing by twisting the wing at three cross-sections using electric motors. Various experiments were performed to get an idea of torque required to morph the wing. Fig. 67 shows the plot of total three moments required to twist the wing in one of these experiments. The future will bring some significant embedded actuation capability, our goal here is to establish methods for distributing control for

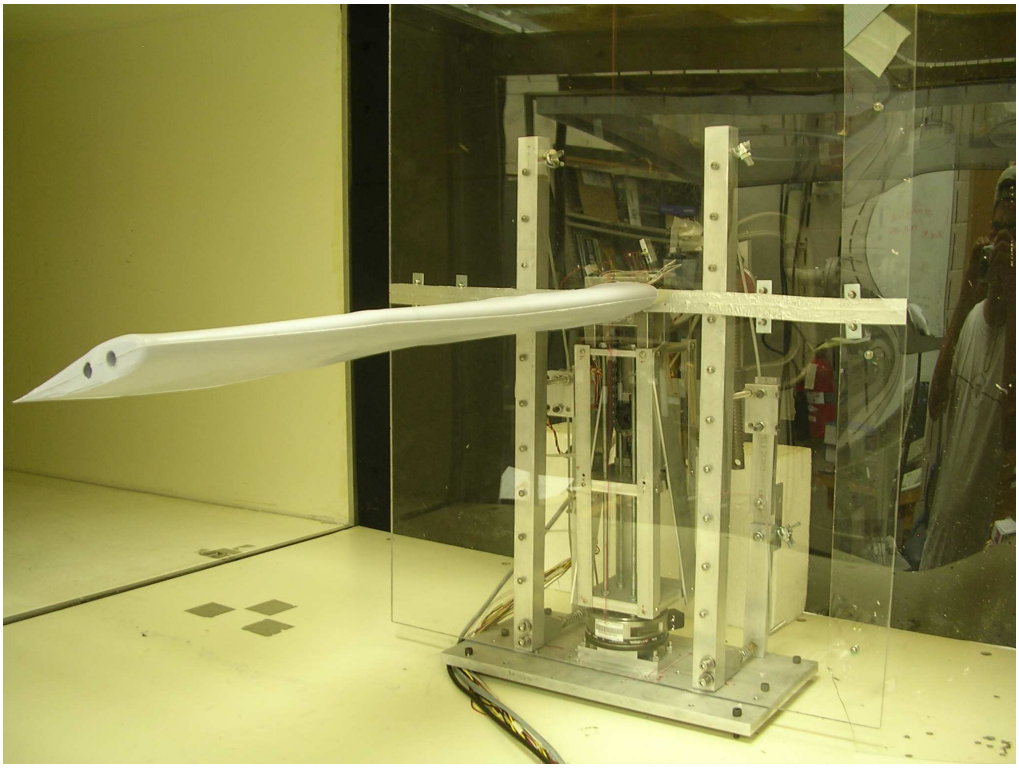


Fig. 66. Morphing wing experimental set-up.

such high-dimensioned problems.

To drive our simulation study, we assume that morphing wing has embedded $22,500$ micro torsional actuators to impose these three moments. We mention that this is a hypothetical situation but provides us a good simulation platform to test the control distribution algorithm developed in this chapter.

For simulation purposes, the control effectiveness of each actuator is defined by scaling an electric motor model (currently used to twist morphing wing profile) by a factor of 10^{-3}.

$$\mathbf{B}_i = \begin{bmatrix} -3.7239 & 19.8465 & 15.6663 \end{bmatrix} 10^{-3} \tag{7.57}$$

Further, each actuator is constrained to produce at max moment of 0.1 N-m in either

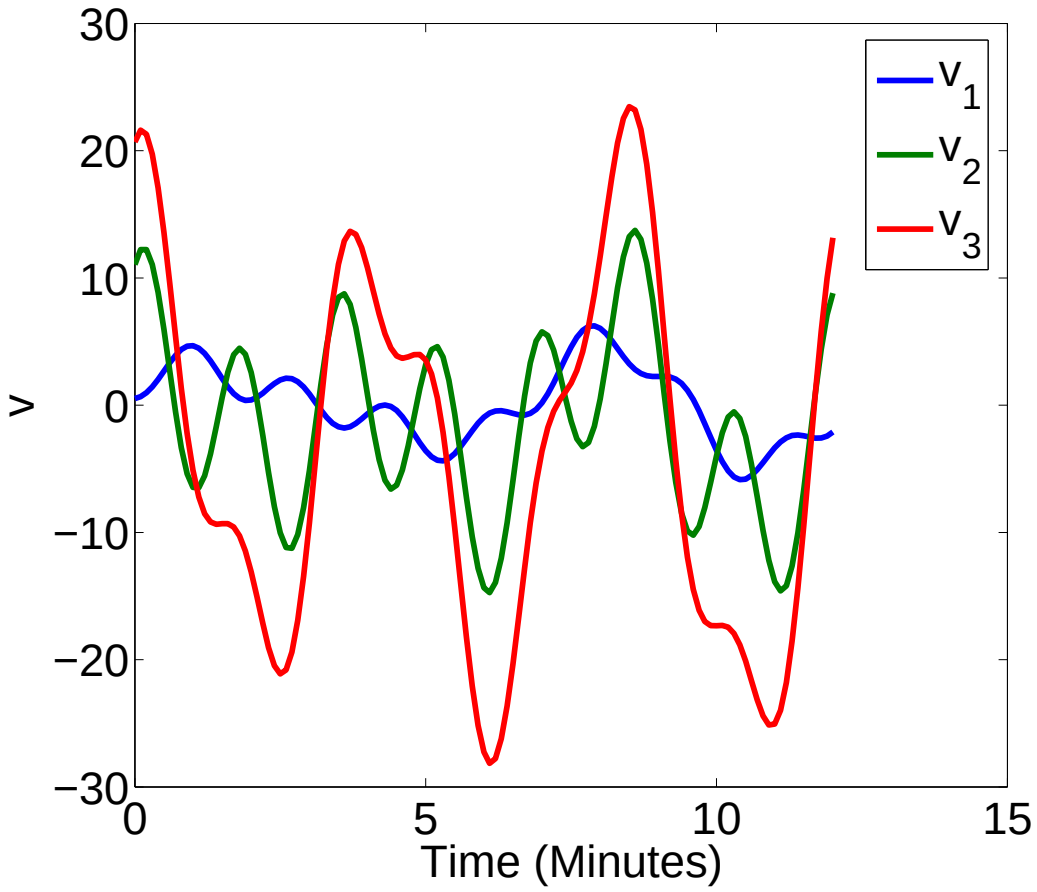
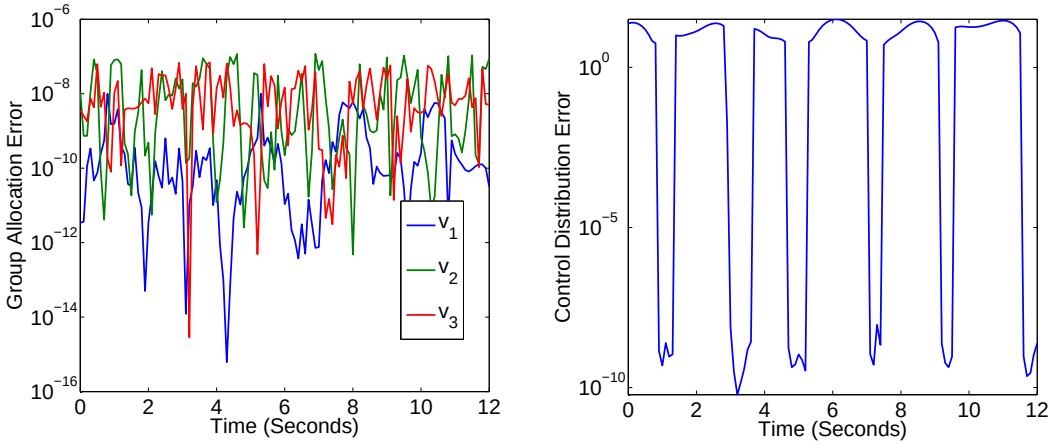


Fig. 67. Physical/virtual control effort.



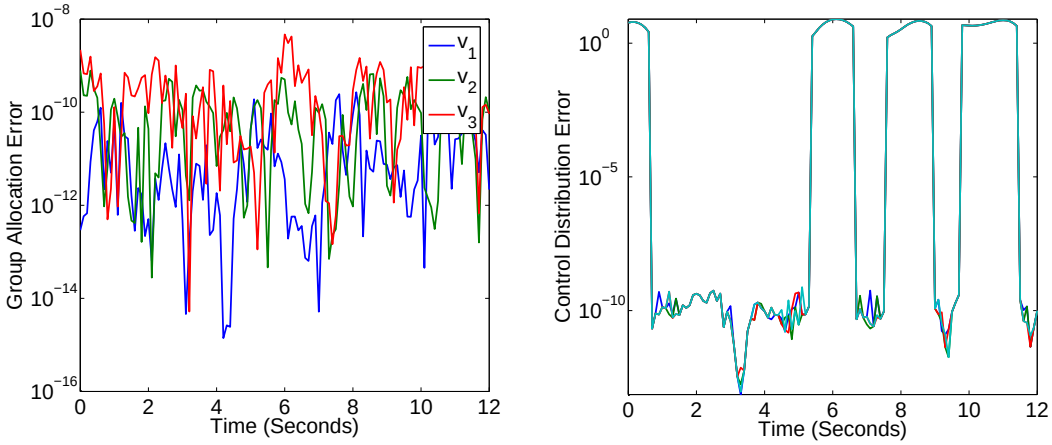
(a) Control allocation error among different groups

(b) Control distribution error within each group

Fig. 68. Control distribution results by dividing the actuators in 1 group.

direction.

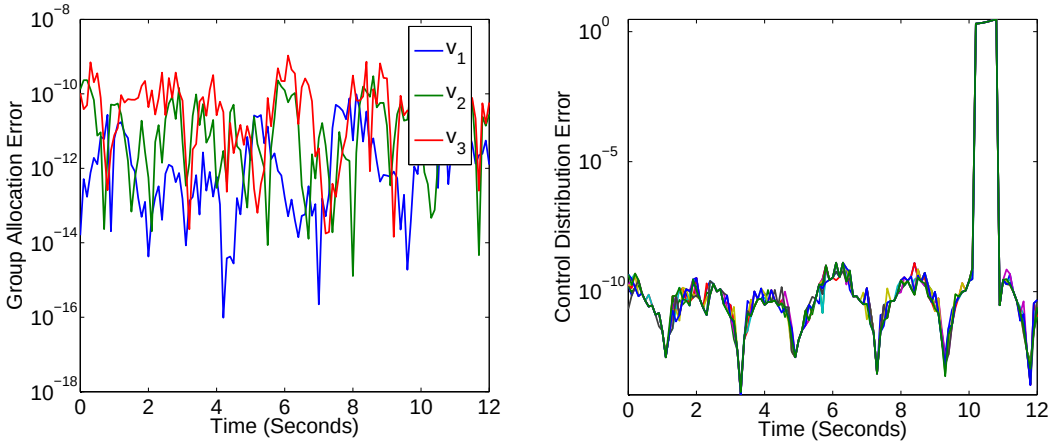
To allocate required control moments among $22,500$ embedded actuators the hierarchical algorithm (shown in Fig. 65) is used. First, the whole control effort is assigned to all the actuators and response of each actuators in that regional group is approximated by orthogonal polynomials of the GLO-MAP algorithm as discussed in section C. Figs. 68(a) and 68(b) show the plots of a group allocation error and net control distribution error, respectively. From, these plots it is clear that with just one group of actuators control distribution function approach fails to compute the feasible solution. According to the hierarchical control distribution algorithm, we divide all actuators in 4 groups. Fig. 69(a) shows the plot of allocation error among the four groups. From, this plot it is clear that the optimization problem of Eqs. (7.48)-(7.50) is solved successfully. Now, within each group, we use the orthogonal polynomial functions to distribute (interpolate) the control error among the actuators contained in that particular group. Fig. 69(b) shows the plot of control distribution error within each group. From this figure, it is clear that although results have



(a) Control allocation error among different groups

(b) Control distribution error within each group

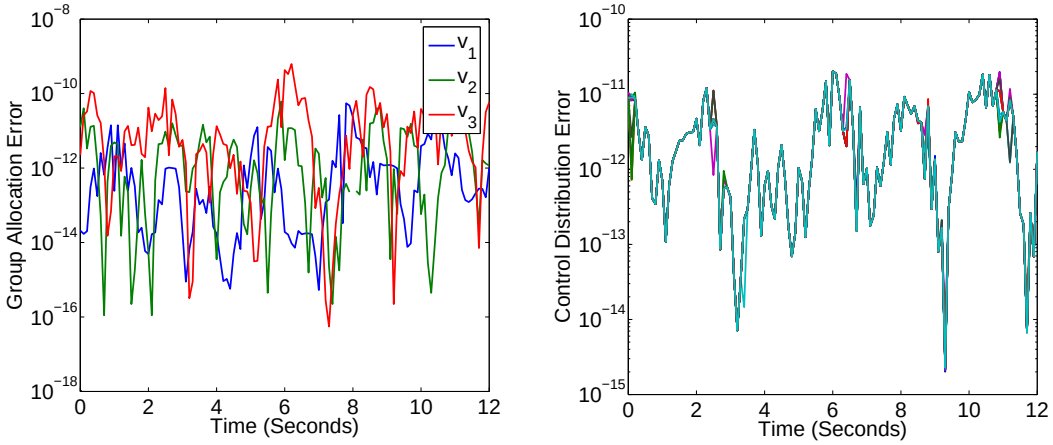
Fig. 69. Control distribution results by dividing the actuators in 4 groups.



(a) Control allocation error among different groups

(b) Control distribution error within each group

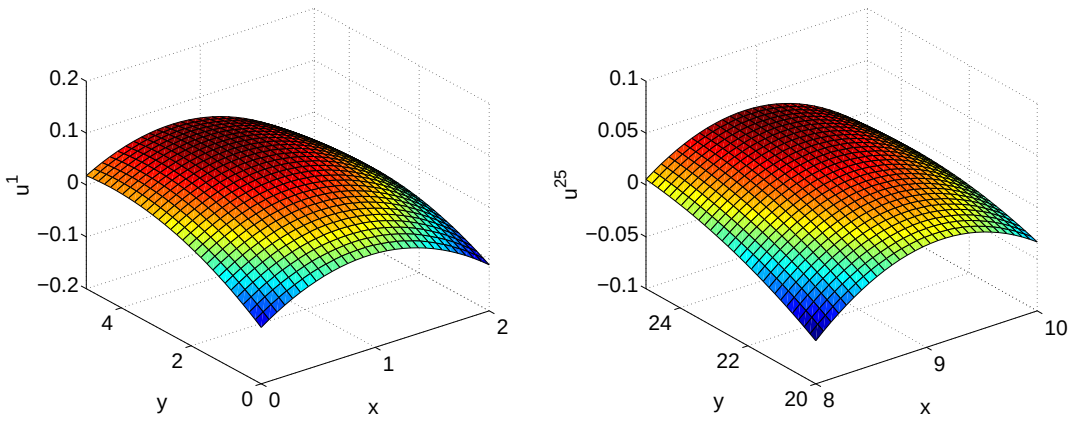
Fig. 70. Control distribution results by dividing the actuators in 8 groups.



(a) Control allocation error among different groups

(b) Control distribution error within each group

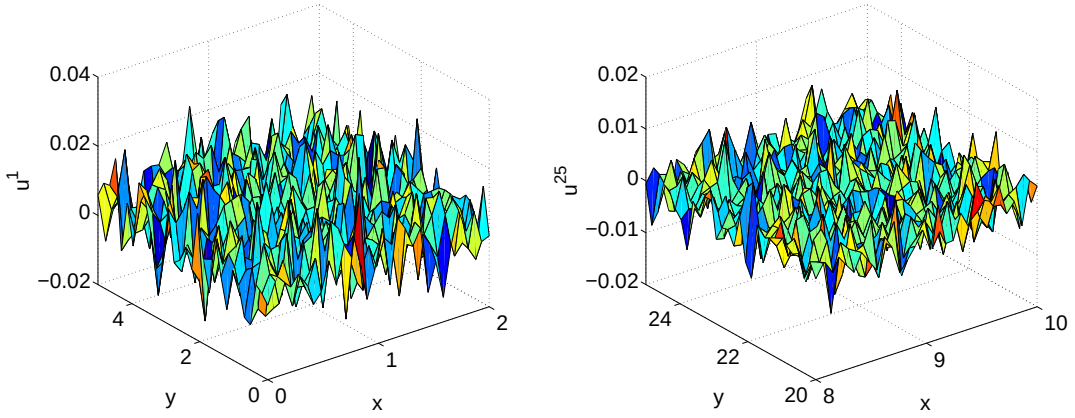
Fig. 71. Control distribution results by dividing the actuators in 25 groups.



(a) Control distribution surface for group 1

(b) Control distribution surface for group 25

Fig. 72. Control distribution surface by using the distribution function approach.



(a) Control distribution surface for group 1

(b) Control distribution surface for group 25

Fig. 73. Control distribution surface without using the distribution function approach.

improved from previous step, there are some instances when distribution function approach fails to provide a solution. Similarly, Figs. 70(a) and 70(b) show the plot of control distribution among 8 different groups of actuators and control distribution error within each group. Once again, there is an improvement from previous step but still there are some instances when control distribution algorithm fails to provide a feasible solution. We repeat this process of dividing the actuators in groups recursively and finally settle down to a total of 25 groups of actuators. Fig. 71(a) shows the plot of control distribution error among 25 groups of actuators whereas Fig. 71(b) shows the plot of control distribution error within each group. From these plots, we can conclude that the hierarchical approach performed very well in allocating the total control effort among all $22,500$ actuators. We mention that 25 groups of actuators are used at only those time instances when we are not able to find the solution using less number of groups. We also mention that the failure of the control distribution algorithm within each group at some instant with less number of actuator groups can be attributed to the irregular nature of the feasible solution set. To make this point clear, we

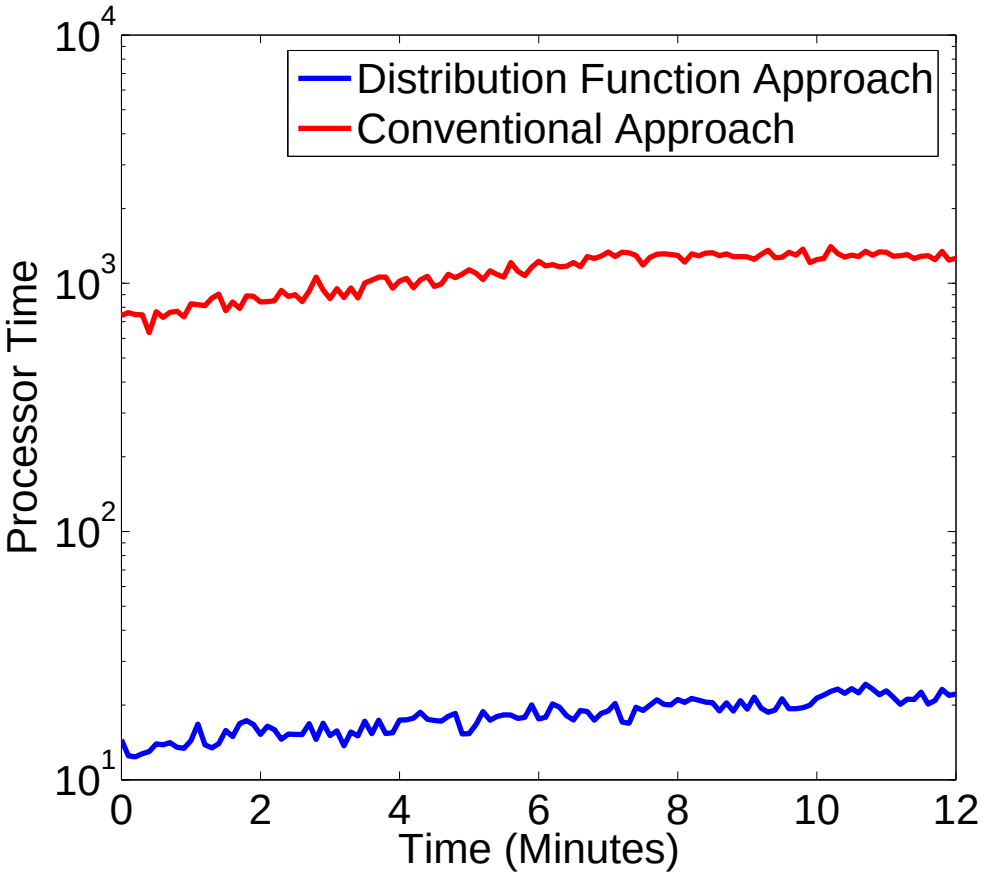


Fig. 74. Processor time to solve the control distribution problem.

solved the control distribution problem without using distribution functions in all 25 groups. Figs. 72 and 73 show the plot of control distribution surface for group 1 and 25 at a particular time instant with and without making use of distribution functions, respectively. Note that the surfaces in Fig. 73 are highly irregular whereas surface in Fig. 72 are very smooth. From these figures, it is clear that the solution obtained by using distribution functions is not necessarily the optimal. However, it is the optimal smooth solution using those particular distribution functions. Further, we mention that we used only 6 distribution functions in all the groups and at all the levels. That means, within each group we are solving for only 6 amplitudes of these distribution functions. However, we need to solve for $22,500/25 = 900$ control variables if we do not use distribution functions to approximate the feasible solutions set. Fig. 74 shows the plot of processor time required to solve the control distribution problem with and without using distribution functions. In both methods, we use 25 groups of actuators to divide the problem into many small scale problems. As expected, the processor time decreases significantly (2 order of magnitude) if distribution functions are used to approximate the feasible solutions set. Also, further decrease in processor time is possible by parallelized implementation of the hierarchical approach. It is important that this example is a basis for optimism, but obviously does not prove a general trend.

Finally, we mention that all numerical simulations were performed using MATLAB [57] environment on $1.5GHz$ Sony Vaio Notebook equipped with $768MB$ of RAM and window XP operating system. Also, we would like to mention that all optimization problems are solved using the SeDuMi [103, 104] optimization package interfaced with by YALMIP [105]. SeDuMi stands for Self Dual Minimization and has been proved to solve large scale optimization problems in an efficient manner and YALMIP is a MATLAB toolbox for rapid prototyping of optimization problems.

F. Concluding Remarks

A general hierarchical methodology for control distribution in highly redundant system is presented in this chapter. The new method makes use of distribution functions to approximate the feasible solution set and to keep in check the "curse of dimensionality". Due to the irregular nature of the feasible solutions set, it may be difficult to approximate the feasible solution set with a chosen set of smooth distribution functions. However, the approximation errors can be improved by using compactly supported distribution functions. To improve the performance of the distribution function approach a hierarchical approach is proposed which guarantees the computation of the feasible solution, if it exists. The proposed hierarchical method decomposes a large scale control distribution problem in to many small scale control distribution problems to compromise the need for real-time computation against optimality. The main advantage of the proposed hierarchical approach is the de-coupling of many small scale problems from each other. As a consequence, the algorithm can be highly parallelized to reduce the computation burden involved. The convergence and accuracy of the proposed method are demonstrated by numerical studies. The broad generality of the method, together with simulation results provides a strong basis for optimism for the importance and utility of these ideas. However, more testing is required to reach stronger conclusions about the utility of this algorithm.

CHAPTER VIII

CONCLUSIONS

In this dissertation, novel modeling and control methodologies are developed to address various problems associated with the design of large scale dynamical systems. This dissertation is addressed to solve challenging modeling and control problems, motivated by advanced aerospace systems. The main contribution of this dissertation is the development of adaptable, robust and computationally efficient, multiresolution approximation algorithms based on the Radial Basis Function (RBF) network and the Global-Local Orthogonal MAPping (GLO-MAP) approaches. The main feature of our RBF network approach is the unique direction dependent scaling and rotation of RBF via a novel Directed Connectivity Graph approach. Our contributions have led to a broadly useful approximation approach that leads to global approximations capable of good local approximation for many moderate dimensioned applications. However, many applications with many high frequency local input/output variations and a high dimensional input space remain a challenge and motivate us to investigate entirely new approach. The innovation for the GLO-MAP method is the development of a novel averaging process to determine a piecewise continuous global family of local least-squares approximations while retaining the freedom to vary in a general way the resolution (e.g., degrees of freedom) of the local approximations. These approximation methodologies are compatible with a wide variety of disciplines such as continuous function approximation, dynamic system modeling and system identification, nonlinear signal processing and time series prediction.

Another contribution of this dissertation is the development of the GLO-MAP based methods for the modeling of dynamical systems nominally described by nonlinear differential equations and to solve for static and dynamic response of Distributed

Parameter Systems (DPS) in an efficient manner. The main focus is on understanding the process of producing dynamical models from the experimental data. We accept that there might not exist simple formulas to accurately describe the experimental data. Therefore, we adopt non-traditional modeling approaches, and accept that we are looking for approximation of the experimental data. The new nonlinear system approximation algorithms not only has the approximation ability of ANN but also has model reduction ability of algorithms like POD/PCA. The main advantage of the GLO-MAP based mesh-less FEM algorithms is that for dynamic calculations, the GLO-MAP approximations can in principle be added or subtracted individually as their corresponding weight functions can grow or shrink without disturbing the basis functions. The main advantage of the GLO-MAP approximation method is that any kind of prior knowledge (qualitative or geometrical) about the system can be incorporated in our approximation. For example, one can always prescribe the modes of interest while doing system identification in Chapter V.

The generalized GLO-MAP approach can, in principle, handle any high dimensioned systems. GLO-MAP is the first piecewise continuous approximation to allow the full utilization of locally supported orthogonal function approximation. As things currently stand, the demonstrated promise is there, the general code is not. In the near future, the GLO-MAP based FEM code will be extended for multi-scale modeling in higher dimensions and to solve several classical problems including the Fokker Plank Equation to generate response PDF and the Hamilton-Jacobi-Bellman equation to compute optimal cost-to-go for a given dynamical system.

In addition, a hierarchical control allocation algorithm is presented which makes use of the concept of distribution functions to keep in check the "curse of dimensionality" while solving the control allocation problem for highly over-actuated systems that might arise with the development of embedded systems. The main advantage of

the proposed hierarchical approach is the de-coupling of many small scale problems from each other. As a consequence of that the algorithm can be highly parallelized to reduce the computation burden involved.

Whereas the theoretical framework of this dissertation lies in fundamental research in approximation theory, the motivation and applications of the methodology have already been demonstrated for diverse problems, drawn from autonomous and intelligent systems, flow control, spacecraft maneuvers, active materials/structures; and thus we have already shown that the resulting methodology has broad applications. The studies in this dissertation focus on demonstrating, through analysis, simulation, and design, the applicability and feasibility of several novel approximation ideas to a variety of examples. The reliability and limitations of the newly established approximation methods are assessed by considering various academic and engineering problems where traditional methods either fail or perform very poorly. The results from these studies are of direct utility in addressing the "curse of dimensionality" and frequent redundancy of neural network approximation. Building upon the results of this work, there are many exciting directions to pursue. I plan to concentrate on the design of nonparameteric/semiparametric hierarchical functional methods for generalized input-output models derived from measurements. I further plan to focus on solving both low and high dimensioned dynamics that arise due to multi-scale discretization/aggregation problems in representing large numbers of sensor and actuators in embedded systems i.e. modeling of complete system of autonomous systems (so-called systems of systems).

REFERENCES

[1] A. N. Kolmogorov, "On the representation of continous function of several variables by superposition of continuous functions of one variable and addition," *Doklady Akademiia Nauk SSSR*, vol. 114, no. 5, pp. 953–956, 1957.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification.* New York, NY: John Wiley & Sons, Inc., 2001.

[3] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Upper Saddle River, NJ: Prentice Hall, 1998.

[4] K. Tao, "A closer look at the radial basis function networks," in *Conference record of 27th Asilomar Conference on Signals, System and Computers.* Pacific Grove, CA, May 1993, pp. 401–405.

[5] J. Platt, "A resource allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, Nov. 1991.

[6] J. Moody and C. Darken, "Fast learning in network of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, Mar. 1989.

[7] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural network," *Neural Computation*, vol. 5, no. 6, pp. 954–975, Nov. 1993.

[8] J. L. Junkins and R. S. Engels, "The finite element approach in gravity modeling," *Manuscripta Geodaetica*, vol. 4, pp. 185–206, Feb. 1979.

[9] W. K. Liu, S. Li, and T. Belytschko, "Moving least square reproducing kernel methods (I) methodology and convergence," *Computer Methods in Applied Mechanics and Engineering*, vol. 143, no. 1-2, pp. 143–157, Apr. 1997.

[10] D. Levin, "The approximation power of moving least squares," *Mathematics of Computations*, vol. 67, no. 224, pp. 1517–1531, Oct. 1998.

[11] H. Wendland, "Local polynomial reproduction and moving least squares approximation," *IMA Journal of Numerical Analysis*, vol. 21, no. 1, pp. 285–300, Jan. 2001.

[12] N. Sundararajan, P. Saratchandran, L. Y. Wei, Y. W. Lu, and Y. Lu, *Radial Basis Function Neural Networks With Sequential Learning: MRAN and Its Applications*, ser. Progress in Neural Processing. River Edge, NJ: World Scientific Pub. Co., Dec., 1999, vol. 11.

[13] M. Musavi, W. Ahmed, K. Chan, K. Faris, and D. Hummels, "On training of radial basis function classifiers," *Neural Networks*, vol. 5, no. 4, pp. 595–603, Jul. 1992.

[14] M. Powell, "Algorithms for approximation," in *Radial Basis Function for Multivariate Interpolation: A Review*, J. C. Mason and M. G. Cox, Eds. Oxford: Clarendon Press, 1987, pp. 143–168.

[15] J. J. Benedetto, *Harmonic Analysis and Applications*. Boca Raton, FL: CRC Press Inc., 1997.

[16] F. J. Narcowich, "Recent developments in error estimates for scattered-data interpolation via radial basis functions," *Numerical Algorithms*, vol. 39, no. 1-3, pp. 307–315, Jul. 2005.

[17] F. J. Narcowich, J. D. Ward, and H. Wendland, "Sololev bounds on functions with scattered zeros, with applications to radial basis function surface fitting," *Mathematics of Computation*, vol. 74, no. 250, pp. 743–763, Jul. 2005.

[18] ——, "Refined error estimates for radial basis function interpolation," *Constructive Approximation*, vol. 19, no. 4, pp. 541–564, Aug. 2003.

[19] J. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, Mar. 1991.

[20] S. Chen, C. F. N. Cowman, and P. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transaction on Neural Networks*, vol. 2, no. 2, pp. 302–309, Mar. 1991.

[21] T. F. Junge and H. Unbehauen, "On-line identification of nonlinear time variant systems using structurally adaptive radial basis function networks," in *Proceedings of the American Control Conference*, Albuquerque, NM, June 1997, pp. 1037–1041.

[22] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.

[23] S. Lee and R. M. Kil, "A Gaussian potential function network with hierarchically self-organizing learning," *Neural Networks*, vol. 4, no. 2, pp. 207–224, Mar. 1991.

[24] J. L. Junkins and Y. Kim, *Introduction to Dynamics and Control of Flexible Structures*, ser. AIAA Education Series. Washington, DC: American Institute of Aeronautics and Astronautics, 1993.

[25] P. Singla, K. Subbarao, and J. L. Junkins, "Direction dependent learning for radial basis function networks," *IEEE Transaction on Neural Networks*, Jan. 2006, accepted for Publication.

[26] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960, series D.

[27] K. Levenberg, "A method for the solution of certain nonlinear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[28] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[29] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamical Systems*, ser. Applied Mathematics and Nonlinear Science. Boca Raton, FL: CRC Press Inc., Mar. 2004, vol. 2.

[30] P. Singla, "A new attitude determination approach using split field of view star camera," M.S. Thesis, Aerospace Engineering Department, Texas A&M University, College Station, TX, Aug. 2002.

[31] N. Sundararajan, P. Saratchandran, and Y. Li, *Fully Tuned Radial Basis Function Neural Networks for Flight Control*, ser. The International Series on Asian Studies in Computer and Information Science. Norwell, MA: Kluwer Academic, 2002, vol. 12.

[32] I. N. N. C. S. Committee, *Benchmark Group on Data Modeling.* [Online] Available: http://neural.cs.nthu.edu.tw/jang/benchmark, Jan. 2005.

[33] J. L. Junkins and J. R. Jancaitis, "Smooth irregular curves," *Photogrammetric Engineering*, vol. 38, no. 6, pp. 565–573, June 1972.

[34] J. O. Moody and P. J. Antsaklis, "The dependence identification neural network construction algorithm," *IEEE Transaction on Neural Networks*, vol. 7, no. 1, pp. 3–15, Jan. 1996.

[35] S. Tan, J. Hao, and J. Vandewalle, "Identification of nonlinear discrete-time multivariable dynamical system by RBF neural networks," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 5, Orlando, FL, 1994, pp. 3250–3255.

[36] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[37] B. A. Whitehead and T. D. Choate, "Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction," *IEEE Transactions on Neural Networks*, vol. 7, no. 4, pp. 869–880, Jul. 1996.

[38] J. Gonzlez, I. Rojas, J. Ortega, H. Pomares, F. J. Fernndez, and A. F. Daz, "Multiobjective evolutionary optimization of the size, shape, and position parameters of radial basis function networks for function approximation," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1478–1495, Nov. 2003.

[39] I. Rojas, H. Pomares, J. Gonzlez, E. Ros, M. Sallmeron, J. Ortega, and A. Prieto, "A new radial basis function networks structure: Application to time series prediction," in *Proceedings of the IEEE-INNS-ENNS*, vol. IV, Jul. 2000, pp. 449–454.

[40] R. C. Engels and J. L. Junkins, "Local representation of the geopotential by

weighting orthonormal polynomials," *AIAA Journal of Guidance and Control*, vol. 3, no. 1, pp. 55–61, Jan. 1980.

[41] J. L. Junkins, P. Singla, T. D. Griffith, and T. Henderson, "Orthogonal global/local approximation in n-dimensions: Applications to input/output approximation," in 6^{th} *International Conference on Dynamics and Control of Systems and Structures in Space*, Cinque-Terre, Italy., 2004.

[42] J. L. Junkins, *An Introduction to Optimal Estimation of Dynamical Systems*, L. Meirovitch, Ed. The Netherlands: Sijthoff & Noordhoff International Publishers, 1978.

[43] J. L. Junkins, G. W. Miller, and J. R. Jancaitis, "A weighting function approach to modeling of geodetic surfaces," *Journal of Geophysical Research*, vol. 78, no. 11, pp. 1794–1803, Apr. 1973.

[44] J. R. Jancaitis and J. L. Junkins, "Modeling in n dimensions using a weighting function approach," *Journal of Geophysical Research*, vol. 79, no. 23, pp. 3361–3366, Aug. 1974.

[45] J. L. Junkins, "An investigation of finite element representations of the geopotential," *AIAA Journal*, vol. 14, no. 6, pp. 803–808, June 1976.

[46] P. Singla, T. D. Griffith, K. Subbarao, and J. L. Junkins, "Autonomous focal plane calibration by an intelligent radial basis network," in *Proceedings of 2004 AAS/AIAA Spaceflight Mechanics Meeting*, ser. Advances in Astronautical Sciences, vol. 119, no. I, Maui, HI, 2004, pp. 275–300.

[47] P. Singla, K. Subbarao, O. Rediniotis, and J. L. Junkins, "Intelligent multiresolution modeling: Application to synthetic jet actuation and flow control,"

in *42nd AIAA Aerospace Sciences Meeting and Exhibit*, no. AIAA-2004-0774, Reno, NV, Jan. 5-8 2004.

[48] T. D. Griffith, P. Singla, and J. L. Junkins, "Autonomous on-orbit calibration approaches for star tracker cameras," in *Proceedings of 2002 AAS/AIAA Spaceflight Mechanics Meeting*, ser. Advances in Astronautical Sciences, vol. 112, no. I, San Antonio, TX, 2002, pp. 39–51.

[49] K. Weierstrass, "Über die analytische darstellbarkeit sogenannter willkürlicher functionen einer reellen veränderlichen," *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, pp. 633–639, 1885, erste Mitteilung, part I.

[50] M. H. Stone, "Applications of the theory of boolean rings to general topology," *Transactions of the American Mathematical Society*, vol. 41, no. 3, pp. 375–481, 1937.

[51] ——, "The generalized Weierstrass approximation theorem," *Mathematics Magazine*, vol. 21, no. 4, pp. 167–184, 1948.

[52] ——, "The generalized Weierstrass approximation theorem," *Mathematics Magazine*, vol. 21, no. 5, pp. 237–254, 1948.

[53] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C : The Art of Scientific Computing*, 2nd ed. New York, NY: Cambridge University Press, 1992.

[54] A. F. Nikiforov, S. K. Suslov, and V. B. Uvarov, *Classical Orthogonal Polynomials of a Discrete Variable*. New York, NY: Springer, 1991.

[55] L. Traub, A. Miller, P. Signla, M. Tandale, J. L. Junkins, and O. Rediniotis, "Distributed hingeless flow control and rotary synthetic jet actuation," in *42nd AIAA Aerospace Sciences Meeting and Exhibit*, no. AIAA-2004-0224, Reno, NV, Jan. 5-8 2004.

[56] NASTRAN, MSC *Software*. [Online] Available: http://www.mscsoftware.com, Jan. 2006.

[57] MATLAB, *The* M*ath*W*orks*-MATLAB *and Simulink for Technical Computing*. [Online] Available: http://www.mathworks.com, Jan. 2006.

[58] P. Singla, T. Henderson, J. L. Junkins, and J. Hurtado, "A robust nonlinear system identification algorithm using orthogonal polynomial network," in *Proceedings of 2005 AAS/AIAA Spaceflight Mechanics Meeting*, ser. Advances in Astronautical Sciences, vol. 120, no. I, Copper Mountain, CO, 2005, pp. 983–1002.

[59] NEO, *Near EARTH Object Program: NASA Web Resource*. [Online] Available: http://jpl.nasa.gov, Jan. 2006.

[60] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*, ser. AIAA Education Series. Washington, DC: AIAA, 2003.

[61] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics*, ser. AIAA Education Series. Washington, DC: AIAA, 1999.

[62] E. W. Weisstein, "Wavelet explorer documentation: A wolfram web resource," [Online] Available, Jan. 2006.

[63] E. J. Stollnitz, T. D. DeRose, and D. H. Salesin, "Wavelets for computer graphics: A primer part 2," *IEEE Computer Graphics and Applications*, vol. 15, no. 4,

pp. 75–85, Jul. 1995.

[64] I. Daubechies, *Ten Lectures on Wavelets*, 1st ed., ser. CBMS-NSF Regional Conference Series in Applied Mathematics. Philadelphia, PA: Society for Industrial & Applied Math, 1992, no. 61.

[65] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed. San Diego, CA: Academic Press, 1999, no. 61.

[66] W. Cheney, *Analysis for Applied Mathematics*, ser. Graduate Text in Mathematics, S. Axler, F. W. Gehring, and K. A. Ribet, Eds. New York, NY: Springer, 2001.

[67] C. S. Fraser, "Digital camera self-calibration," *Journal of Photogrammetry and Remote Sensing*, vol. 52, no. 4, pp. 149–159, 1997.

[68] K. R. Lenz and Y. R. Tsai, "Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 713–720, Sep. 1988.

[69] M. A. Samaan, D. Mortari, and J. Junkins, "Non-dimensional star identification for un–calibrated star cameras." Ponce, Puerto Rico: AAS Paper 03–131 of the AAS/AIAA Space Flight Mechanics Meeting, February, 9–13 2003.

[70] P. A. Ionnaou and J. Sun, *Robust Adaptive Control*. Upper Saddle River, NJ: Prentice Hall Inc., 1996.

[71] K. Subbarao, M. Steinberg, and J. L. Junkins, "Structured adaptive model inversion applied to tracking aggressive aircarft maneuvers," in *AIAA Guidance,*

Navigation and Control Conference and Exhibit. Montreal, Canada, August 2001.

[72] J. N. Juang and R. S. Pappa, "An eigensystem realization algorithm for modal parameter identification and model reduction," *Journal of Guidance, Control and Dynamics*, vol. 8, no. 5, pp. 620–627, Sep.-Oct. 1985.

[73] A. U. Levin and K. S. Narendra, "Control of nonlinear dynamical systems using neural networks-part *ii*: Observability, identification, and control," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 30–42, Jan. 1996.

[74] H. N. Mhaskar and N. Hahm, "Neural networks for functional approximation and system identification," *Neural Computation*, vol. 9, no. 1, pp. 143–159, Nov. 1997.

[75] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[76] J. N. Juang and R. W. Longman, "Optimized system identification," NASA, Langley Research Center, Hampton, Virginia, Tech. Rep. TM-1999-209711, Oct. 1999.

[77] J. N. Juang, *Applied System Identification.* Englewood Cliffs, NJ: PTR Prentice Hall, 1994.

[78] J. N. Juang, M. Phan, L. G. Horta, and R. W. Longman, "Identification of observer/Kalman filter Markov parameters: Theory and experiments," *Journal of Guidance, Control and Dynamics*, vol. 16, no. 2, pp. 320–329, Mar.-Apr. 1993.

[79] A. C. Antoulas, D. C. Sorensen, and S. Gugercin, "A survey of model reduction methods for large-scale systems," *Contemporary Mathematics*, vol. 280, pp. 193–219, 2001.

[80] A. M. Lyapunov, *The General Problem of the Stability of Motion*, A. T. Fuller, Ed. Washington, DC: Taylor and Francis, 1992, english Translation of A. M. Lyapunov's work by A. T. Fuller (editor).

[81] R. A. Gingold and J. J. Monaghan, "Kernel estimates as a basis for general particle methods in hydrodynamics," *Journal of Computational Physics*, vol. 46, no. 3, pp. 429–453, June 1982.

[82] T. Belytschko, Y. Y. Lu, and J. Gu, "Element free Galerkin method," *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 229–256, 1994.

[83] W. Liu, S. Jun, and Y. Zhang, "Reproducing kernel particle methods," *International Journal for Numerical Methods in Fluids*, vol. 20, no. 8-9, pp. 1081–1106, 1995.

[84] S. N. Atluri and T. Zhu, "A new Meshless Local Petrov-Galerkin (MLPG) approach in computational mechanics," *Computational Mechanics*, vol. 22, no. 2, pp. 117–127, Aug. 1998.

[85] I. Babuška and J. Melenk, "The partition of unity finite element method," *International Journal for Numerical Methods in Engineering*, vol. 40, no. 4, pp. 727–758, Feb. 1997.

[86] A. Duarte and J. T. Oden, "Hp clouds - an h-p meshless method," *Numerical Methods for Partial Differential Equations*, vol. 12, no. 6, pp. 673–705, 1996.

[87] T. Belytschko, Y. Krogauz, D. Organ, M. Fleming, and P. Krysl, "Meshless methods: An overview and recent developments," University of Texas, Austin, TX, Tech. Rep., May 1996.

[88] S. N. Atluri and S. Shen, *The Meshless Local Petrov-Galerkin (*MLPG*) Method*. Encino, CA: Tech Science Press, 2002.

[89] J. N. Reddy, *An Introduction to the Finite Element Method*. New York, NY: McGraw-Hill, Inc., 1993.

[90] M. Kumar, P. Singla, S. Chakravorty, and J. L. Junkins, "A multi-resolution approach for steady state uncertainty determination in nonlinear dynamical systems," in *38th Southeastern Symposium on System Theory*, Cookeville, TN, 5-7 Mar. 2006, accepted for Publication.

[91] A. Miller, L. Traub, O. Redeniotis, P. Singla, M. Tandale, and J. L. Junkins, "Distributed hingeless flow control and rotary synthetic jet actuation," in *42nd AIAA Aerospace Sciences Meeting and Exhibit*, no. AIAA-2004-224, Reno, NV, Jan. 5-8 2004.

[92] O. Härkegård, "Backstepping and control allocation with applications to flight control," Ph.D Dissertation, Linköping University, SE-581 83 Linköping, Sweden, 2003.

[93] J. E. Luntz, W. Messner, and H. Choset, "Distributed manipulation using discrete actuator array," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 553–583, Jul. 2001.

[94] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. New York, NY: John Wiley & Sons, 1987.

[95] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY: Cambridge University Press, Mar. 2004.

[96] G. B.Dantzig, *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press, 1963.

[97] R. H. Bartels and G. A. Golub, "The simplex method of linear programming using LU decomposition," *Communications of the ACM*, vol. 12, no. 5, pp. 266–268, 1969.

[98] S. J. Wright, *Primal-Dual Interior-Point Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1997.

[99] K. A. Bordignon, "Constrained control allocation for systems with redundant control effectors," Ph.D Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1996.

[100] J. M. Buffington and D. F. Enns, "Lyapunov stability analysis of daisy chain control allocation," *Journal of Guidance, Control, and Dynamics*, vol. 19, no. 6, pp. 1226–1230, Nov.-Dec. 1996.

[101] M. P. Fromherz and W. B. Jackson, "Force allocation in a large scale distributed active surface," *IEEE Transaction on Control Systems Technology*, vol. 11, no. 5, pp. 641–655, Sep. 2003.

[102] W. B. Jackson, M. P. J. Fromherz, D. K. Biegelsen, J. Reich, and D. Goldberg, "Constrained optimization based control of real time largescale systems: Airjet object movement system," in *Proceedings of the 40th International Conference on Decision and Control*, vol. 5, Orlando, FL, Dec. 2001, pp. 4717 – 4720.

[103] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 12, pp. 625–653, 1999.

[104] E. Andersen, J. Gondzio, C. Meszaros, and X. Xu, *Interior point methods for mathematical programming.* Norwell, MA: Kluwer Adademic Publishers, 1996, ch. Implementation of interior point methods for large scale linear programming, pp. 189–252.

[105] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in matlab," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[106] E. W. Weisstein, *Beta Function: From MathWorld–A Wolfram Web Resource.* [Online] Available: http://mathworld.wolfram.com/BetaFunction.html, Jan. 2006.

APPENDIX A

SOLUTION OF BOUNDARY VALUE PROBLEM FOR THE WEIGHTING FUNCTION

The boundary value problem for obtaining an expression for the weighting function can be summarized as follows:

1. The first derivative of the weighting function must have a desired m^{th}-order zero at the center (centroid of validity) of its respective local approximation.

$$\begin{aligned} w(0) &= 1 \\ \tfrac{d^k w}{dx^k}|_{x=0} &= 0 \quad k = 0, 1, \cdots, m \end{aligned} \tag{A.1}$$

2. The weighting function must have an $(m+1)^{th}$-order zero at the center of its neighboring local approximation.

$$\begin{aligned} w(1) &= 0 \\ \tfrac{d^k w}{dx^k}|_{x=1} &= 0 \quad k = 0, 1, \cdots, m \end{aligned} \tag{A.2}$$

3. The sum of two neighboring weighting functions must be unity over the entire closed interval between their corresponding adjacent local functional approximations.

$$w({}^I x) + w({}^I x - 1) = 1 \quad \forall\ x,\ -1 \leq x \leq 1 \tag{A.3}$$

One family of solutions of this boundary value problem can be obtained by assuming the following particular form for weighting function,

$$w(x) = 1 - J(x) \tag{A.4}$$

where, $J(x)$ is a polynomial in the independent variable whose first derivative is given by following expression:

$$\frac{dJ(x)}{dx} = Cx^m(1-x)^m \tag{A.5}$$

It should be noted that this particular form for the weighting function is in accordance with the fact that first m partial derivatives of the weighting function vanishes at end points and $w(0) = 1$. Now the remaining boundary conditions on the weighting function, $w(x)$, will completely define the constants in Eq. (A.5).

$$J(1) = C\int_0^1 x^m(1-x)^m dx = 1 \tag{A.6}$$

That means the appropriate value for constant, C, is given by:

$$C = \left[\int_0^1 x^m(1-x)^m dx\right]^{-1} \tag{A.7}$$

Now, using the fact that integral expression on the RHS is a Eulerian integral of the first kind [106], the constant C is given by following expression:

$$C = \frac{(2m+1)!}{(m!)^2} \tag{A.8}$$

The general form for weighting function can now be written as:

$$w(x) = 1 - \frac{(2m+1)!}{(m!)^2}\int_0^x x^m(1-x)^m dx \tag{A.9}$$

Further, the binomial theorem allows us to expand the above integrand:

$$w(x) = 1 - \frac{(2m+1)!}{(m!)^2}\int_0^x \sum_{n=0}^{m} x^m x^{m-n}(-1)^n dx \tag{A.10}$$

Now, integrating the above expression term by term yield the following expression

for weighting function, $w(x)$:

$$w(x) = 1 - K \sum_{n=0}^{m} A_n x^{2m-n+1} \tag{A.11}$$

where, K and A_n are given by following expressions:

$$K = \frac{(2m+1)!(-1)^m}{(m!)^2} \quad A_n = \frac{(-1)^n \ {}^mC_n}{2m-n+1} \tag{A.12}$$

Finally, to obtain the expression for weighting function in interval $[-1, 1]$ instead of $[0, 1]$ the absolute value of x is used as independent variable than x. The generalized weight functions that guarantee arbitrary order continuity are given in Table XI.

APPENDIX B

GRAM-SCHMIDT PROCEDURE OF ORTHOGONALIZATION

Let $\mathcal{V}$ be a finite dimensional inner product space spanned by basis vector functions $\{w_1, w_2, \cdots, w_n\}$. According to the *Gram-Schmidt Process* an orthogonal set of basis functions $\{\phi_1, \phi_2, \cdots, \phi_n\}$ can be constructed from any basis functions $\{w_1, w_2, \cdots, w_n\}$ by following three steps:

1. Initially there is no constraining condition on the first basis element ϕ_1 therefore we can choose $\phi_1 = w_1$.

2. The second basis vector, orthogonal to the first one, can be constructed by satisfying the following condition:

$$\langle \phi_2, \phi_1 \rangle = 0 \tag{B.1}$$

   Further, if we write:

$$\phi_2 = w_2 - c\phi_1 \tag{B.2}$$

   then we can determine the following value of unknown scalar constant c by substituting this expression for ϕ_2 in orthogonality condition, given by equation (B.1):

$$c = \frac{\langle w_2, \phi_1 \rangle}{\langle \phi_1, \phi_1 \rangle} \tag{B.3}$$

3. Continuing the procedure listed in step 2, we can write ϕ_k as:

$$\phi_k = w_k - c_1\phi_1 - c_2\phi_2 - \cdots - c_{k-1}\phi_{k-1} \tag{B.4}$$

   where, the unknown constants $c_1, c_2, \cdots, c_{k-1}$ can be determined by satisfying

following orthogonality conditions:

$$\langle \phi_k, \phi_j \rangle = 0 \quad \text{For } j = 1, 2, \cdots, k-1 \tag{B.5}$$

Since, $\phi_1, \phi_2, \cdots, \phi_{k-1}$ are already orthogonal to each other therefore the scalar constant c_j can be written as:

$$c_j = \frac{\langle w_k, \phi_j \rangle}{\langle \phi_j, \phi_j \rangle} \tag{B.6}$$

Therefore, finally we have following general *Gram-Schmidt formula* for constructing the orthogonal basis vectors $\phi_1, \phi_2, \cdots, \phi_n$:

$$\phi_k = w_k - \sum_{j=1}^{k-1} \frac{\langle w_k, \phi_j \rangle}{\langle \phi_j, \phi_j \rangle} \phi_j, \quad \text{For } k = 1, 2, \cdots, n \tag{B.7}$$

To construct the orthogonal polynomials of degree $\leq n$ with respect to weight function, $1 - x^2(3 - 2|x|)$ on closed interval $[-1, 1]$, we need to apply the Gram-Schmidt procedure to non-orthogonal monomial basis $1, x, x^2, \cdots, x^n$. First of all, we compute the general expression for $\langle x^k, x^l \rangle$:

$$\langle x^k, x^l \rangle = \int_{-1}^{1} x^{k+l}(1 - x^2(3 - 2|x|))dx = \begin{cases} \frac{2}{k+l+1} - \frac{6}{k+l+3} + \frac{4}{k+l+2} & k+l \text{ is even} \\ 0 & k+l \text{ is odd} \end{cases} \tag{B.8}$$

According to this formula, monomials of odd degree are orthogonal to monomials of even degree. Now, if $p_0(x), p_1(x), \cdots$ denote the resulting orthogonal polynomials then we can begin the process of Gram-Schmidt orthogonalization by letting:

$$\phi_0(x) = 1 \tag{B.9}$$

According to equation (B.7), the next orthogonal polynomial is

$$\phi_1(x) = x - \frac{\langle x, p_0 \rangle}{\langle p_0, p_0 \rangle} p_0(x) = x \tag{B.10}$$

Further, recursively using the Gram-Schmidt formula given by equation (B.7), we can generate the orthogonal polynomials given in Table XII, including the recursive form given for $\phi_n(x)$. In appendix C, we describe an alternative recurrence relation to generate these orthogonal polynomials.

APPENDIX C

THREE TERM RECURRENCE RELATION TO GENERATE ORTHOGONAL POLYNOMIALS

Let $\mathcal{V}_n$ be a finite dimensional inner product space spanned by orthogonal basis vector functions $\{\phi_1, \phi_2, \cdots, \phi_n\}$, where ϕ_n represent a polynomial of degree n. Next, since $x\phi(x) \in \mathcal{V}_{n+1}$, therefore, there exist numbers $c_0, c_1, \cdots, c_{n+1}$ such that following is true:

$$x\phi_n(x) = \sum_{i=0}^{n+1} c_{i,n}\phi_i(x) \tag{C.1}$$

Since, $\phi_0, \phi_1, \cdots, \phi_n$ are orthogonal to each other with respect to weight function $w(x)$, we see that

$$c_{k,n} = \frac{1}{\mu_k^2}\int x\phi_n(x)\phi_k(x)w(x)dx = \frac{1}{\mu_k^2}\langle xv_n, \phi_k\rangle,\ k = 0, 1, \cdots, n+1 \tag{C.2}$$

Where, $\langle ., . \rangle$ denotes the inner product defined by weight function $w(x)$ and $\mu_k = \langle \phi_k, \phi_k \rangle$. Further, notice that for $k \leq n-2$, $xv_k(x) \in \mathcal{V}_{n-1}$ and hence, $c_{k,n} = 0,\ \forall 0 \leq k \leq n-2$ and Eq. (C.1) reduces to:

$$x\phi_n(x) = c_{n-1,n}\phi_{n-1}(x) + c_{n,n}\phi_n(x) + c_{n+1,n}\phi_{n+1}(x) \tag{C.3}$$

Now, let us assume that a_n and b_n are leading coefficients of basis function $\phi_n(x)$. Hence, from Eq. (C.1), we get:

$$a_n = c_{n+1,n}a_{n+1},\ b_n = c_{n,n}a_n + c_{n+1,n}b_{n+1} \tag{C.4}$$

Also, substituting for $k = n-1$ in Eq. (C.2), we get

$$c_{n-1,n} = \frac{1}{\mu_{n-1}^2}\langle x\phi_n, \phi_{n-1}\rangle = \frac{\mu_n^2}{\mu_{n-1}^2}c_{n,n-1} \tag{C.5}$$

Now, from Eqs. (C.4) and (C.5), we get:

$$c_{n+1,n} = \frac{a_n}{a_{n+1}},\ c_{n,n} = \frac{b_n}{a_n} - \frac{b_{n+1}}{a_{n+1}},\ c_{n-1,n} = \frac{\mu_n^2}{\mu_{n-1}^2}\frac{a_{n-1}}{a_n} \tag{C.6}$$

Now, substituting for various $c_i's$ from Eq. (C.6) in Eq. (C.3), we get following three term recurrence relation:

$$x\phi_n(x) = \frac{a_n}{a_{n+1}}\phi_{n+1}(x) + \left(\frac{b_n}{a_n} - \frac{b_{n+1}}{a_{n+1}}\right)\phi_n(x) + \frac{\mu_n^2}{\mu_{n-1}^2}\frac{a_{n-1}}{a_n}\phi_{n-1}(x) \tag{C.7}$$

Finally, From Eq. (C.7), it is clear that given a sequence of numbers $\{a_n\}$ and $\{b_n\}$, one can construct orthogonal polynomials to given weight function $w(x)$. That means, the orthogonal polynomial $\phi_n(x)$ is unique up to a normalizing factor. In appendix D, we give a more detailed proof of this statement.

APPENDIX D

UNIQUENESS OF ORTHOGONAL POLYNOMIALS

In this appendix, we prove that orthogonal polynomials which satisfy the *orthogonality condition* of Eq. (3.28) are unique up to a normalizing factor.

Let $\{\phi_i(x)\}$ and $\{\bar{\phi}_i(x)\}$ are two sets of polynomials which satisfies the following orthogonality condition:

$$\langle \phi_i(x), \phi_j(x) \rangle \equiv \int_{-1}^{1} w(x)\phi_i(x)\phi_j(x)dx = k_i\delta_{ij} \tag{D.1}$$

$$\langle \bar{\phi}_i(x), \bar{\phi}_j(x) \rangle \equiv \int_{-1}^{1} w(x)\bar{\phi}_i(x)\bar{\phi}_j(x)dx = \bar{k}_i\delta_{ij} \tag{D.2}$$

Since, $\bar{\phi}_n(x)$ is a polynomial of degree n, therefore, we can write it as a linear combination of polynomials $\{\phi_0, \phi_1, \cdots, \phi_n\}$ as:

$$\bar{\phi}_n(x) = \sum_{i=1}^{n} c_{i,n}\phi_i(x) \tag{D.3}$$

Note, by Eq. (D.1) $c_{i,n} = 0$ for $k < n$ and therefore, $\phi(x)$ and $\bar{\phi}(x)$ should be proportional to each other. However, if the leading coefficient of the polynomial $\phi_n(x)$ is constrained to be one then it is apparent that $\phi_n(x) = \bar{\phi}_n(x)$.

VITA

Puneet Singla was born in Punjab, India in 1978. He received his baccalaureate (B.Tech.) degree in aerospace engineering from Indian Institute of Technology, Kanpur, India in May, 2000. In August, 2000, he joined the aerospace engineering department of Texas A&M University for his graduate studies and received his master's degree in Aerospace Engineering in August 2002. His master's thesis was focused on the attitude estimation problem of a star tracker mission and his Ph.D dissertation is geared towards solving highly challenging modeling and control problems, motivated by advanced aerospace systems. All graduate level work was performed under the supervision of Dr. John L. Junkins.

He can be reached at p.singla@gmail.com or by contacting Dr. John L. Junkins, Department of Aerospace Engineering, Texas A&M University, College Station, TX-77843.