1-1-2015

Constrained Motion Particle Swarm Optimization for Non-Linear Time Series Prediction

Nicholas Sapankevych
Unversity of South FLorida, nsapankevych@aol.com

Constrained Motion Particle Swarm Optimization for Non-Linear Time Series Prediction

by

Nicholas I. Sapankevych

A dissertation submitted in partial fulfilment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Ravi Sankar, Ph.D.
Thomas Weller, Ph.D.
A. D. Snider, Ph.D.
Kandethody M. Ramachandran, Ph.D.
Rangachar Kasturi, Ph.D.

Date of Approval:
March 13, 2015

Keywords: Support Vector Regression, Convex Optimization, EUNITE, Competition for Artificial Time Series, Mackey-Glass

DEDICATION

This dissertation is dedicated to my wife Susan, to my children Tyler and Alex, to my sisters Kathryn and Anne Marie, and to my brother Jim, and to my parents William and Karen Sapankevych.

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. Ravi Sankar for his support and patience in guiding me through my Ph. D studies. His continued guidance, advice and generosity have been invaluable throughout my experience at USF. I am also very grateful for the effort spent by my dissertation committee members Prof. Weller, Prof. Snider, Prof. Ramachandran, and Prof. Kasturi in helping me reach my goal and for the time they spent in broadening my engineering experience.

I would also like to thank my employer, Raytheon, for making this opportunity a reality. To all of my fellow colleagues at Raytheon who have supported and encouraged me over the years, thank you.

Many thanks to all of the engineers and schoolmates I have had the privilege to meet at our Interdisciplinary Communication Networking and Signal Processing (iCONS) group at USF. Their commentary and feedback regarding my work is always appreciated.

To my family for their sacrifice and understanding during my time of study, thank you for your prayers and thoughts.

Thanks be to God for giving us the light of knowledge and making all things possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Time series prediction techniques have been used in many real-world applications such as financial market prediction, electric utility load forecasting, weather and environmental state prediction, and reliability forecasting. The underlying system models and time series data generating processes are generally complex for these applications and the models for these systems are usually not known a priori. Accurate and unbiased estimation of time series data produced by these systems cannot always be achieved using well known linear techniques, and thus the estimation process requires more advanced time series prediction algorithms.

One type of time series interpolation and prediction algorithm that has been proven to be effective for these various types of applications is Support Vector Regression (SVR) [1], which is based on the Support Vector Machine (SVM) developed by Vapnik *et al.* [2, 3]. The underlying motivation for using SVMs is the ability of this methodology to accurately forecast time series data when the underlying system processes are typically nonlinear, non-stationary and not defined a-priori. SVMs have also been proven to outperform other non-linear techniques including neural-network based non-linear prediction techniques such as multi-layer perceptrons.

As with most time series prediction algorithms, there are typically challenges associated in applying a given heuristic to any general problem. One difficult challenge in using SVR to solve these types of problems is the selection of free parameters associated with the SVR algorithm. There is no given heuristic to select SVR free parameters and the user is left to adjust these parameters in an ad hoc manner.

The focus of this dissertation is to present an alternative to the typical ad hoc approach of tuning SVR for time series prediction problems by using Particle Swarm Optimization (PSO) to assist in the SVR free parameter selection process. Developed by Kennedy and Eberhart [4-8], PSO is a technique that emulates the process living creatures (such as birds or insects) use to discover food resources at a given geographic location. PSO has been proven to be an effective technique for many different kinds of optimization problems [9-11].

In this dissertation, the general problems associated with time series prediction and their associated algorithms are presented first. Second, both the SVR and PSO algorithms are discussed in detail which leads to the third part, formulating the fusion of both SVR and PSO algorithms, called Constrained Motion Particle Swarm Optimization (CMPSO). CMPSO not only provides a method for SVR free parameter selection, but also provides computational efficiency benefits. Finally the CMPSO algorithm is benchmarked using both artificially generated and real world data and is compared to other published non-linear time series prediction algorithms against the same benchmark data. Specifically, CMPSO performance is analyzed against artificial, arbitrary functions as well as real world financial market index data (S&P 500). CMPSO is also benchmarked against Mackey-Glass non-linear data, real power company electrical load data and another artificial time series used for a competition. CMPSO performed well and also would have been placed first in at least one of the time series prediction competitions.

CHAPTER 1: INTRODUCTION

1.1 Time Series Prediction

Fundamentally, the goal of time series prediction is to estimate some future time series value based on current and past data samples. Mathematically stated in Equation (1.1):

$$\hat{x}(t+\Delta_t) = f(x(t-a), x(t-b), x(t-c), \ldots) \tag{1.1}$$

where, in this specific example, $\hat{x}$ is the predicted value of a (one dimensional) discrete time series x. The variables a, b, c, etc., are (positive) time offsets and Δ_t is a positive number greater or equal to zero representing a future time for an estimated sample.

The objective of time series prediction is to find a function f(x) such that $\hat{x}$, the predicted value of the time series at a future point in time is unbiased and consistent. An unbiased estimator [12] is one where the expected value of the estimate of x should approach the actual value of x. A consistent estimator [12] is one where the variance of the predicted value will approach zero as the number of samples increases and is shown in Equation 1.2:

$$\lim_{N \to \infty} E[(\hat{x}_N - x)^2] = 0 \tag{1.2}$$

where x is the time series function to be estimated, $\hat{x}$ is the estimated function, N is the number of time series samples, and E[.] is the expected value operator.

An unbiased and consistent estimator will tend to have an expected value of the difference in the predicted and actual functions as well as the variance of this difference both approach zero simultaneously. Specific time series prediction metrics related to bias and consistency will be discussed in Section 5.1.

Estimators generally fall into two categories: linear and non-linear. Over the past several decades, a vast amount of technical literature has been written about linear prediction: the estimation of a future value based on the linear combination of past and present values. Real world time series prediction applications generally do not fall into the category of linear prediction. Instead, they are typically characterized by non-linear models.

1.2 Challenges in Time Series Prediction

There are many different kinds of time series prediction algorithms, each one performing differently depending on the type of data being analyzed. Table 1.1 highlights the benefits and challenges for many different kinds of prediction algorithms, including SVR.

Although the advantages of SVR have been shown to be significant as compared to other algorithms for time series prediction, the challenge associated with selecting SVR free parameters is considerable.

1.3 Research Motivation

Non-linear time series regression and prediction applications range from financial market prediction, electrical load forecasting, dynamic control system design, to a vast array of other real world problems. As stated earlier, there are many methods to solve such problems including Auto Regressive Moving Average (ARMA) algorithms (in many different forms) [12, 13], Kalman Filtering (also in many different forms) [12-17], Artificial Neural Networks (ANNs) [18, 19], Support Vector Machines (SVMs) and SVR [20-45], as well as several others.

Any of the above algorithms can be applied to real world problems, some with greater success than others. In many cases the success of the algorithm for a given application depends heavily on algorithm "tuning": the process of optimizing the algorithm for the specific problem space. Some examples of "tuning" include model selection (as with Kalman Filtering) and free variable constant selection (as with SVR). The employment of some algorithms such as SVR further requires the use of a Quadratic Program (QP) to solve for the given algorithmic parameters, thus increasing the computational complexity of these kinds of approaches [21, 46-50].

Although SVMs/SVR algorithms are generally considered computationally complex, it has been well documented that they are effective for time series prediction applications [1] as well as regression (interpolation) applications. The challenge remains to optimize the SVR free parameters effectively while accurately estimating the time series. Given the advantages of using SVR for time series prediction, the motivation of this research is to identify an optimization algorithm that can address the SVR parameter tuning challenge while simultaneously adapt to a wide variety of applications.

1.4 Contributions and Organization

This dissertation proposes an SVR based approach for both time series prediction and regression while simultaneously using Particle Swarm Optimization (PSO) [51-60] to optimize SVR free parameters. Although PSO based SVR optimization techniques have been developed into one approach, the methodology presented in this research is unique in that the process necessary to compute an SVR estimate of a time series is integrated with the PSO optimization formulation, making the overall computation more efficient. By constraining the motion of the particles, there is a reduction in the solution space required to examine to find an optimal

solution. In addition, this PSO and SVR framework is adaptable to a wide variety of applications without the need of the user to tune any parameters. These specific algorithmic features detailed in this research are unique and have not been presented before in literature.

As with any time series estimation method, this approach will be benchmarked against other well-known time series sets for comparison to other published algorithms. It will be shown that the exact same framework, without any modification, will produce a time series estimate for many different kinds of problems. The applications range from common mathematical time series functions, to more complicated real world applications such as stock market index prediction, power load forecasting, and other synthetic data interpolation and extrapolation applications.

In summary, the main contributions and results of this research include:

- A general heuristic for selecting required SVR user defined parameters.
- A computationally more efficient method for PSO based SVR user defined parameter optimization.
- A generic framework adaptable to many different one dimensional time series regression and prediction problems that does not require user adjustments.
- A time series interpolation and extrapolation method that produced significant results as compared to other similar approaches. The results are based against worldwide time series regression competitions, in which CMPSO provided a goodness of fit that met or in some cases exceeded all other competitive algorithms.

The organization of the dissertation is as follows:

In Chapter 2, we discuss the wide variety of real world time series prediction applications associated with SVM and SVR, along with PSO based optimization of SVM/SVR algorithms. We also introduce time series prediction benchmarks and summarize their use as evaluation tools for any type of time series prediction method.

In Chapter 3, we review the formulation of SVR and identify parts of the formulation that are adapted for optimization using PSO. We will discuss the concept of primal and dual objective functions as well as the use of kernel functions to cast time series data into another form.

In Chapter 4, we review the formulation of PSO. Specifically, we will review the terminology and definitions associated with PSO, the PSO framework and process, and some specific parameters that will be linked to SVR.

In Chapter 5, we present the CMPSO framework, which includes the fusion of both SVR and PSO algorithms for time series prediction and interpolation. Additionally, we discuss the advantages of this approach including the flexibility in adapting this technique to a wide variety of different time series applications.

In Chapter 6, we test the CMPSO algorithm against both artificial and real world time series data and compare its performance to other published algorithms. First, we define time series prediction metrics by which time series prediction algorithms are measured. We briefly discuss computational complexity in the context of the integrated SVR/PSO CMPSO approach vs. other typical approaches where SVR optimization is executed outside the PSO optimization routine. Next we illustrate CMPSO performance with an arbitrary function, an arbitrary function missing data and a real world stock market index example. Then we test CMPSO's performance against other published algorithms. The first example illustrates CMPSO performance using a

common, artificial time series benchmark called Mackey-Glass data [61] and is compared to other comparable time series prediction algorithms. The second example shows CMPSO performance using the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE) competition data [62, 63]. The goal of EUNITE is to test several algorithms by predicting maximum electrical power loads for the East-Slovakia Power Distribution Company for one future month given two prior years' worth of daily electrical load data. The last example illustrates CMPSO performance via Competition on Artificial Time Series (CATS) data where several algorithms were to predict (interpolate) four sets of 20 missing data points out of a total of 4980 time series samples [64]. In addition, the same algorithms were to predict (extrapolate) the last 20 time series data points of the CATS data.

In Chapter 7, we summarize our findings and propose recommended further research areas.

Table 1.1: Summary of Advantages and Challenges of Classical, ANN Based, and SVR Time Series Prediction Methods

Time Series Prediction Method	**Advantages**	**Challenges**
Autoregressive Filter [12,13]	- Can be computationally efficient for low order models - Convergence guaranteed - Minimizes mean square error by design	- Assumes linear, stationary processes - Can be computationally expensive for higher order models
Kalman Filter [12-17]	- Computationally efficient by design - Convergence guaranteed - Minimizes mean square error by design	- Assumes linear, stationary processes - Assumes process model is known
Multi-layer Perceptron [18,19]	- Not model dependent - Not dependent on linear, stationary processes - Can be computationally efficient (feed forward process)	- Number of free parameters large - Selection of free parameters usually calculated empirically - Not guaranteed to converge to optimal solution - Can be computationally expensive (training process)
SVR [20-45]	- Not model dependent - Not dependent on linear, stationary processes - Guaranteed to converge to optimal solution - Small number of free parameters - Can be computationally efficient (estimation process)	- Selection of free parameters usually calculated empirically - Can be computationally expensive (training process)

CHAPTER 2: SVM BASED TIME SERIES PREDICTION AND BENCHMARKS

2.1 Time Series Prediction and Support Vector Machines

Time series prediction techniques have been used in many real-world applications such as financial market prediction, electric utility load forecasting, weather and environmental state prediction, and reliability forecasting. The underlying system models and time series data generating processes are generally complex for these applications and the models for these systems are usually not known *a priori*. Accurate and unbiased estimation of the time series data produced by these systems cannot always be achieved using well known linear techniques, and thus the estimation process requires more advanced time series prediction algorithms.

The underlying motivation for using SVMs is the ability of this methodology to accurately forecast time series data when the underlying system processes are typically nonlinear, non-stationary and not defined a-priori. SVMs have also been proven to outperform other non-linear techniques including neural networks. Traditionally Support Vector Machines (SVMs), as well as other machine learning algorithms are used for classification in pattern recognition applications. These learning algorithms have also been applied to general regression analysis: the estimation of a function by fitting a curve to a set of data points. The application of SVMs to general regression analysis case is called Support Vector Regression (SVR). A detailed general survey of SVM applications for time series can be found in [1].

2.1.1 Support Vector Machine Applications

Of all the practical applications using SVR for time series prediction, financial data time series prediction appears to be one research area that has a considerable amount of research

interest. The inherent noisy, non-stationary and chaotic nature of this type of time series data appears to lend itself to the use of non-traditional time series prediction algorithms such as SVR. As seen in references [65] through [85] alone, there are many different SVM based approaches to many different kinds of financial data. All of the research noted here has at least one thing in common (other than an SVM based solution): there is no given heuristic to tune the required set of SVR user defined parameters. The cited research uses many alternative optimization algorithms, such as genetic algorithms, to assist the SVR process in finding an optimal curve fit. The methods presented in this research tend to include an ε insensitive loss function as well as a radial basis kernel function, both of which will be discussed in more detail in Chapter 3. Another common thread in the research is the reliance of the stated methods to use specific Quadratic Programming (QP) methods to solve for the SVR estimation.

A non-linear prediction problem found in power systems research is the forecasting of electrical power consumption demands by consumers. There are many beneficial aspects to the accurate prediction of electrical utility load forecasting including proper maintenance of electrical energy supply, the efficient utilization of electrical power resources, and the proper administration and dissemination of these resources as related to the cost of these resources to the consumer. Power load forecasting is yet another widely researched application for SVR. As with the financial market forecasting citations, the SVR based power load prediction applications also tend to use an e insensitive loss function, a radial basis kernel function, and publically available QP programs [46-50].

Although the two research areas mentioned above appear to be the most prevalent applications for SVR, there are many other real world applications such as control systems and

signal processing, machine reliability forecasting, meteorological data forecasting, and many others [1].

2.1.2 Support Vector Machines and Particle Swarm Optimization

As mentioned in the previous section, there are a wide variety of SVM/SVR based real world applications. One of the common threads found in published SVR based solutions is that there is no one heuristic to tune SVR for any given application. However, there have been many applications using Swarm Optimization (SO) based techniques, including Particle Swarm Optimization (PSO), to assist in finding an optimal SVM/SVR based time series estimate. This dissertation explores a unique implementation of PSO and SVR that will represent a general framework for solving these kinds of problems.

Wang *et al.* [51] uses PSO to solve for the three user defined SVR parameters, error sensitivity control parameter, a regularization constant, and a user defined parameter associated with the SVR kernel function (often referred to as ε, C, and σ, respectively; these parameters will be discussed in detail in Chapter 3) for a real world application (coal working face gas concentration forecasting). This is a similar approach to the presented research in this dissertation, although the SVR optimization in this citation still requires a separate QP algorithm. Although Wang presents a feasible approach, there is still the computational overhead associated with using SVR and PSO in a non-integrated methodology.

There are many other recent and similar SVR and PSO combined approaches that have been published recently that use PSO to optimize SVR free parameters [52 through 60]. The applications in these publications range from power load forecasting, traffic flow optimization, to benchmark data estimation, and many others. There are also other general modifications to the algorithms in the citations, but the core processing algorithms are based on both SVM/SVR and

PSO. All of these publications show that a PSO based approach to SVR parameter tuning is viable and effective for many different problems. However, none of the researched publications address the computational overhead involved with using a QP solver to find an appropriate SVR solution. Also, there is no explicit discussion of the necessary data preprocessing required to adapt any given approach to any given problem. The research in this dissertation addresses both of these challenges.

2.2 Time Series Prediction Performance Benchmarks

As with any algorithm, there are usually defined methods by which an algorithm can be evaluated for performance. Data sets referred to as benchmarks are developed and used worldwide in order to facilitate the evaluation and comparison of any given algorithm. For time series prediction and regression based applications, there are many benchmark data sets available for use. This dissertation focuses on three common benchmarks that are found in many algorithmic performance evaluations: Mackey-Glass data [61], EUNITE Competition data [62, 63] and CATS competition data [64]. A brief description is given below with detailed numeric results and comparisons to other published approaches are given in Chapter 6.

2.2.1 Mackey-Glass Data

Mackey-Glass data is highly non-linear chaotic time series data that is used to evaluate time series prediction and regression algorithms. It should be noted that it is synthetic data and can be generated by solving a specific, time delayed differential equation [61].

Many SVR based methods researched use Mackey-Glass data as the reference data set to estimate performance [25, 28, 34, 86 through 91]. The PSO/SVR methodology described in this dissertation is compared to eight different algorithms that use Mackey-Glass benchmark data to show a comparison of performance. Several of the methodologies used for comparison use some

form of SVM/SVR as part of the approach. Section 6.5.1 details the performance comparisons between this research and other published approaches.

2.2.2 EUNITE Competition

The Mackey-Glass benchmark data set is a common data set used to evaluate time series prediction problems and it is synthetically generated from the solution of a differential equation. Another common benchmark data set is from the European Network on Intelligent Technologies for Smart Adaptive Systems (EUNITE; see references [62, 63]) competition which is seen in many publications. Unlike Mackey-Glass data, this data is from real world power utility company loading data. The goal of the competition is to predict one month of maximum power loads based on the prior two years of both daily power load data as well as daily temperature data.

Many of the approaches used are SVR and Artificial Neural Network (ANN) based approaches. The PSO/SVR approach shown in this research is compared to the top ten results of the competition [92-101]. In addition, many recent publications have used this data as a benchmark for their algorithm design and performance [102-106]. Again, the PSO/SVR approach performance is compared to these recently published results for comparison. More details are provided in Section 6.5.2.

2.2.3 CATS Data

The Competition for Artificial Time Series (CATS) [64] is the last benchmark data this research is tested against. As with the EUNITE competition, CMPSO performance is compared to the top ten contestants [107-116] as well as recent publications using this dataset for evaluation [117-121].

The CATS competition included the estimation of four 20 point sets of missing data gaps in a time series of 4980 points. In addition, the last 20 data points are extrapolated and compared to the original data for scoring. The PSO/SVR approach is compared to the top ten algorithms submitted to the competition as well as several recent publications and algorithms that use the CATS benchmark data for reference.

In conclusion, there is significant research interest in advanced learning methods applied to time series prediction and estimation problems. It has been well documented that PSO can be applied to aid in the optimization of an SVR based approach. It is also well documented that SVR approaches, including the one presented in this research, perform well against many different data sets, including standard benchmark data sets described in this chapter. As will be seen in this dissertation, the core of the time series regression formulation is Support Vector Regression.

CHAPTER 3: SUPPORT VECTOR REGRESSION

3.1 Introduction to Support Vector Machines

SVM is a pattern recognition algorithm based on statistical learning theory developed by Vapnik and Chervonenkis [2, 3]. SVMs are a form of neural network and employ a supervised learning training paradigm. Neural network based algorithms have many significant benefits and properties [18]:

- Nonlinearity: an important property that can adapt to non-linear inputs.
- Input-Output Mapping: Inputs to SVM are weighted as training samples are presented to the SVM. As more training samples are presented, the SVM will adapt the weights based on an expected response. This is essentially the supervised learning paradigm.
- Adaptive: SVMs can change, or adapt, to different training inputs dynamically.
- Application Independence: Regardless of the underlying processes used to generate the input and desired output, the same fundamental architecture can be used.
- Sparse Data Representation: Typically the number of support vectors required to define the SVM or SVR function is significantly less than the total number of training points required.

Many books, journal publications, and electronic references currently exist regarding the formulation of SVM as well as SVR. The following formulations of SVM and SVR found in

this chapter are based on Vapnik's reference books [2, 3], a reference book by Smola and Scholköpf [20], an SVM/SVR overview with references to Quadratic Programing (QP) specific to SVM by Cristianini and Shawe-Taylor [21], a rigorous tutorial by Smola and Scholköpf [22], and several other relevant SVM/SVR references found in [23-37]. There are also many web based references available in references [38-45].

The fundamental purpose of an SVM is to mathematically define a surface that separates two different classes of data. This surface is typically called a hypersurface. As an example, data that lies on one side of the hypersurface will be numerically classified as +1 and data that lies on the opposite side will be represented by a -1. Figure 3.1 illustrates a simple two dimensional classification problem with two classes of data.

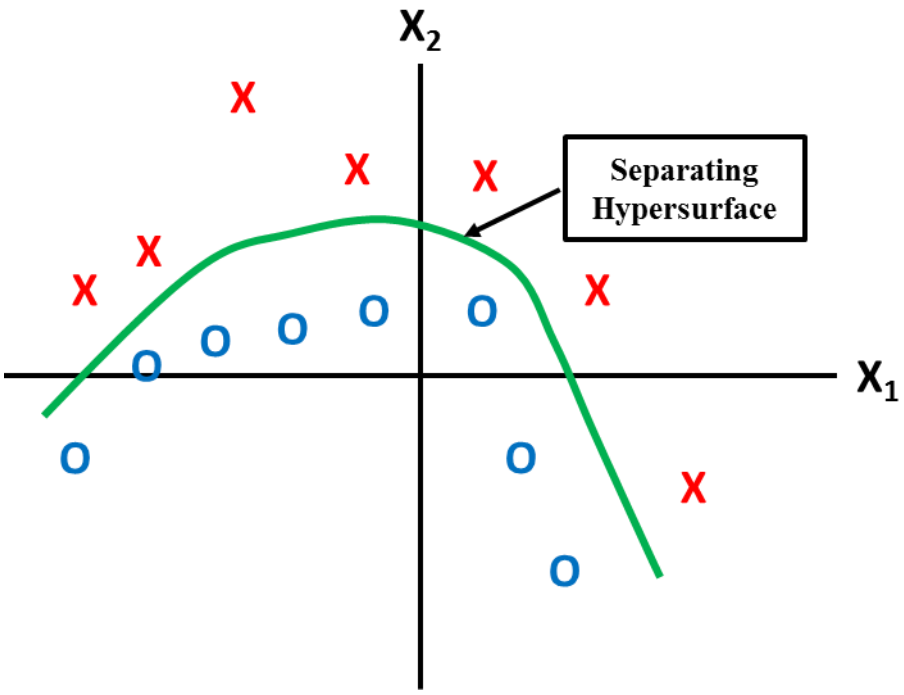

Figure 3.1: Pattern Recognition Example with Two Class Data in Two Dimensions

In Figure 3.1, the red X's represent one class of data and the blue O's represent the other. These two sets of two dimensional data as defined by the variables X_1 and X_2, are separated by a

non-linear curve as illustrated by the green line between the individual data points. The green line separating the two data sets is the hypersurface and is computed by the SVM. Each data point in this example would be one sample of a training data set that would be presented to the SVM in order to generate the separating hypersurface. Equation 3.1 defines the training data set as:

$$Training\ Samples: \{(\boldsymbol{x}_i, d_i)\}|_{i=1}^{N} \tag{3.1}$$

where $\mathbf{x}_i$ is any given point location illustrated in Figure 3.1, d_i are the classifications for each point (+/- 1 as an example), and N are the total number of points in the training set.

The SVM formulation starts with an assumption of linearly separable data as represented by the equation of a linear hypersurface shown in Equation 3.2:

$$f(\boldsymbol{x}) = \sum_{i=0}^{N} w_i \boldsymbol{x}_i + b \tag{3.2}$$

where $\mathbf{x}_i$ is the multi-dimensional variable, w_i are real valued weights and b is a bias term offset. For the case of a two class pattern recognition problem, Equations 3.3 and 3.4 represent the decision making process for any given data point:

$$\sum_{i=0}^{N} w_i \boldsymbol{x}_i + b \geq 0 \; for \; d_i = +1 \tag{3.3}$$

$$\sum_{i=0}^{N} w_i \boldsymbol{x}_i + b < 0 \; for \; d_i = -1 \tag{3.4}$$

Figure 3.2 illustrates an example of linearly separable data. In this example, the data can be separated by the function defined in Equation 3.2 for some optimal set of *w* and *b*. Also illustrated is a subset of the training points located on the green dashed lines which lay some

distance ε from the optimal hypersurface. Data points that are found to lie on these dashed lines are known as support vectors.

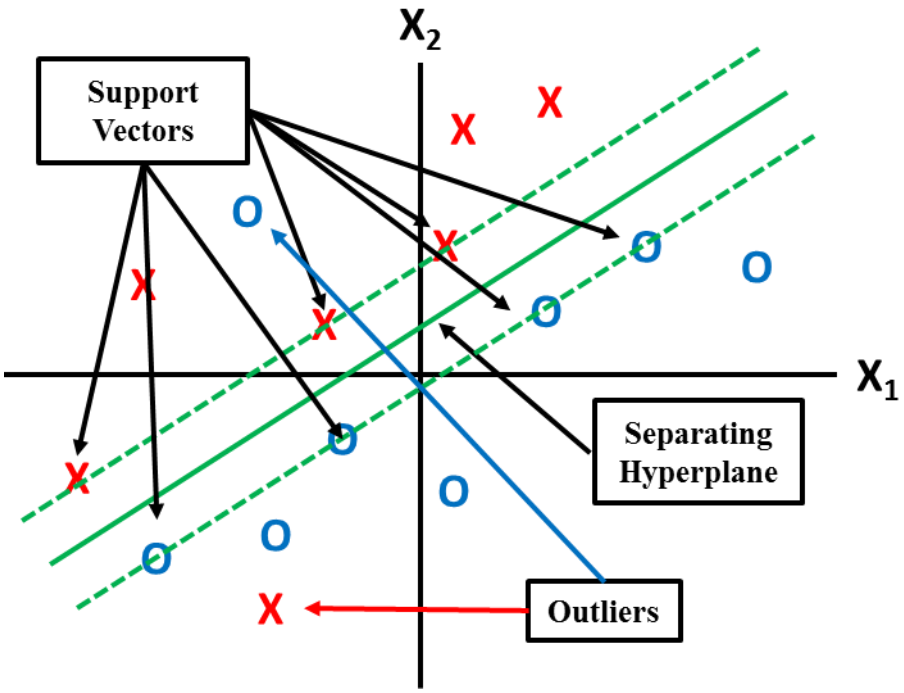

Figure 3.2: Linear Pattern Recognition Example with Two Class Data in Two Dimensions Illustrating Support Vectors and Outliers

It will be shown that the support vectors and their associated weights (plus a bias term) are all that is required to define the separating hypersurface. Also shown in Figure 3.2 are outliers. These are data points that fall on the opposing classification side of the separating hypersurface and would normally be classified incorrectly. The data points are said to be inseparable. The SVM (and SVR) formulation allows for such errors and the handling of these errors will be addressed later in this chapter.

Equation 3.5 below defines the equation for the optimal hypersurface $f_0(\mathbf{x})$, rewritten as the dot product between a set of optimal weights and the independent variable (vector) **x**:

$$f_0(\boldsymbol{x}) = \boldsymbol{w}_0 \cdot \boldsymbol{x} + b_0 \tag{3.5}$$

The goal is to find the optimal hypersurface such that the distance between any given point and the hypersurface is maximized. From [18], Equation 3.6 shows that the distance *r* from any given point **x** to the hypersurface $f_0(\mathbf{x})$ is defined as:

$$r = \frac{f_0(\boldsymbol{x})}{\|\boldsymbol{w}_0\|} \tag{3.6}$$

where $\|\mathbf{w}_0\|$ represents the Euclidean norm of the optimal weight vector. It turns out support vectors lay on the boundary of the equality constraint in Equation 3.3 and a similar equality Equation in 3.4 for both cases where *d* is +/- 1 [18]. Given this, the optimal distance for support vectors can be defined in Equation 3.7:

$$r = \frac{f_0(\boldsymbol{x}^S)}{\|\boldsymbol{w}_0\|} = \begin{cases} \frac{1}{\|\boldsymbol{w}_0\|} & if\ d^S = +1 \\ -\frac{1}{\|\boldsymbol{w}_0\|} & if\ d^S = -1 \end{cases} \tag{3.7}$$

where the superscript S denotes a support vector.

Given the distance *r* from a support vector of each class to the optimal hypersurface, the separation margin can be defined as twice the distance *r* from one class to the other class. In summary, optimizing SVM for pattern recognition requires maximizing the separation between classes of data which is determined by minimizing the Euclidian norm of the weight vector **w**.

3.2 Time Series Regression

The SVM framework outlined in Section 3.1 can now be applied to time series regression problems. First, we define time series regression as "a method for fitting a curve (not necessarily a straight line) through a set of points using some goodness of fit criterion" [122]. The goal is to use a similar framework as defined for SVM for time series prediction and estimation applications. This framework will lend itself to the framework of Support Vector Regression, or SVR.

The SVM framework described in Equation 3.1 is manipulated for regression applications by first changing the target data representation. Instead of a discrete classification of +/- 1 represented by the variable *d* in Equation 3.1, we now introduce a real valued function f(x) as shown in Equation 3.8:

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \tag{3.8}$$

where **w**, **x**, are real valued vectors, f(**x**) and *b* are real valued numbers and the $\langle \cdot \rangle$ notation represents the dot product of the two vectors. For the purposes of this dissertation, we are only going to examine data sets that are two dimensional functions, defined by the training set in Equation 3.9:

$$Training\ Samples: \{(x_i, y_i)\}|_{i=1}^{N} \tag{3.9}$$

We will use bold face letters such as **x** and **w** to represent a one dimensional vector and subscripted variables such as x_i and y_i to represent individual data points.

As with the pattern recognition problem, the goal once again is to find optimal values of **w** and *b* such that the Euclidian norm of **w** is minimized. The process of minimizing **w** is often referred to as "flattening". As seen in Equation 3.9, the real vector **x** has a one to one mapping to a single real value **y** for all N training samples. Clearly the formulation in Equation 3.7 is essentially the same as defined for pattern recognition in Equation 3.2, with the exception of the real valued output.

The goal is to find the regression function f(x) such that it lies within an error bound ε as shown in Figure 3.3. In this example, the red points represent N training data points, the solid green line represents the estimated function f(x) and the two dashed green lines represent the error bound ε around the estimated function f(x).

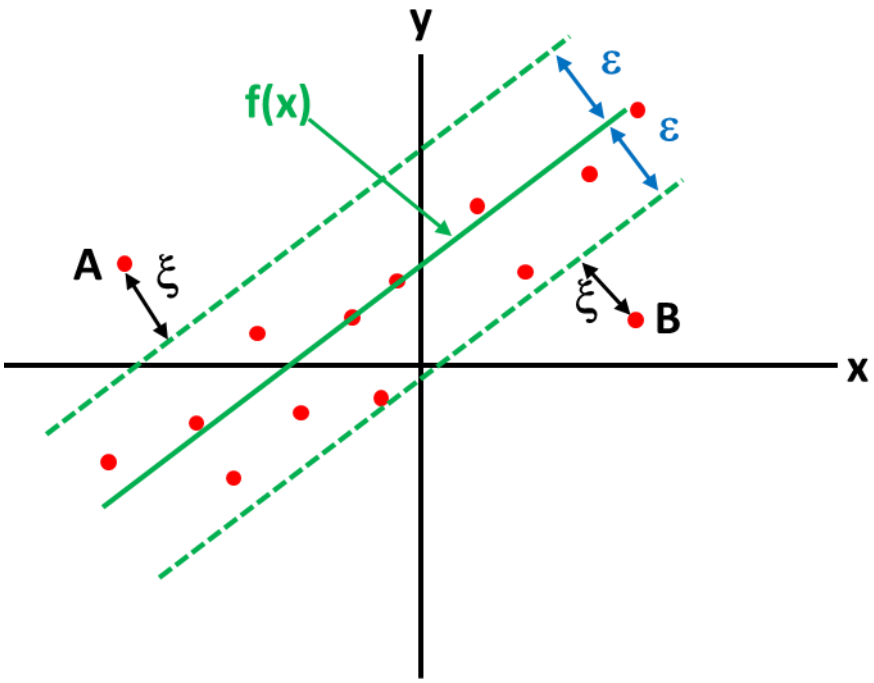

Figure 3.3: Support Vector Regression Linear Approximation Example

As seen in Figure 3.3, this example shows a linear regression type problem where the fitted curve f(x) is a straight line as defined in Equation 3.8. Again, the goal is to find f(x) such that all of the training points fall within +/- ε of f(x). As can be seen, there are two outlier points in Figure 3.3 marked as points A and B. These points fall outside the ε error bound and must be somehow be tolerated in the SVR formulation. These cases will be explained in the primal objective function discussion in Section 3.4.

It is clear that not every time series regression problem will have a linear form of solution as defined in Equation 3.8. Figure 3.4 is a more typical example of a real world time series plot. Again, the red points are the training data and f(x), shown as the green solid line, is the approximation to those points. The dashed green lines illustrate the e error bound. Also shown in this example are two outlier points marked A and B. Again, these conditions will be handled in Section 3.4.

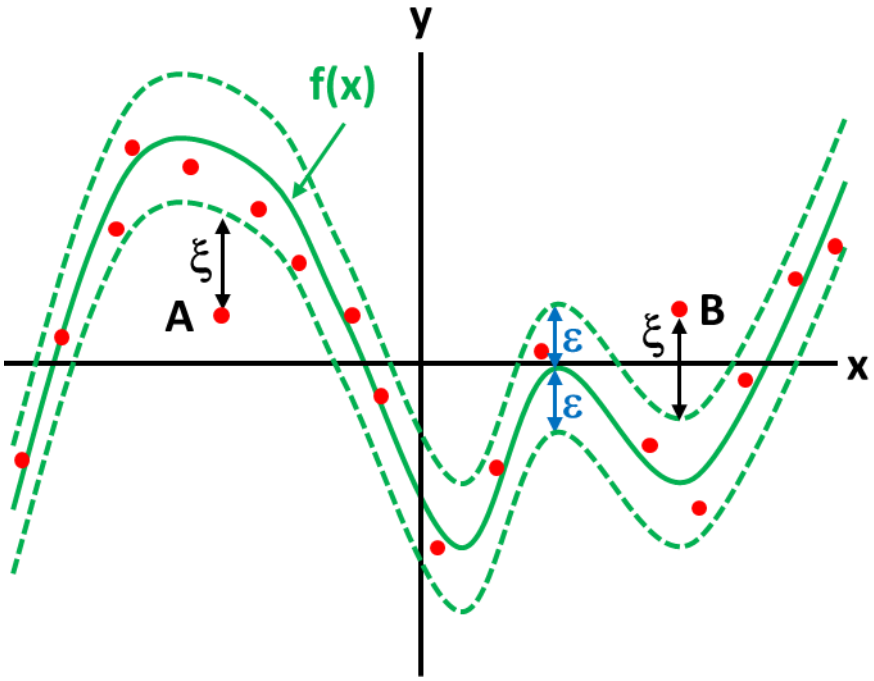

Figure 3.4: Support Vector Regression Non Linear Approximation Example

It is now clear that the approximating function defined in Equation 3.8 will no longer work for this kind of problem as a linear fit is not appropriate unless the error bounds are set to a very high value. We would still like to use a linear formulation as stated in Equation 3.8, but a data transformation will have to be used.

3.3 Kernel Functions

Figure 3.4 illustrates a time series regression problem where a linear curve fit will not be suitable as an approximation. The goal is to try and maintain the SVM/time series regression framework defined in Sections 3.1 and 3.2 where a linear approximation is used. One way of manipulating the data is to use a transformation function that will map the training set points (x_i,y_i) into what is commonly referred to as feature space. This transformation process is done to attempt to cast the variables into another space as shown in Equation 3.10.

$$\boldsymbol{x} = (x_1, \dots, x_n) \rightarrow \boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \dots, \phi_M(\boldsymbol{x})) \tag{3.10}$$

where n are the number of sample data points and M is the number of dimensions in feature space. The linear regression in Equation 3.8 can now be restated in feature space in Equation 3.11.

$$f(x) = \sum_{i=1}^{M} w_i \phi_i(x) + b \tag{3.11}$$

We now have the same linear combination of weights and bias term representing the time series to be estimated. Figure 3.5 illustrates the transformation of variable space.

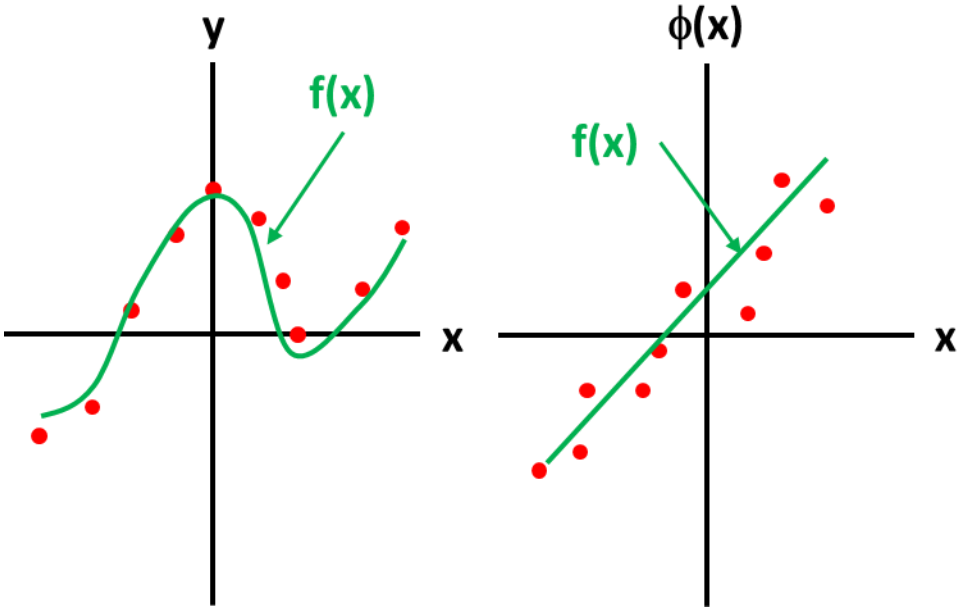

Figure 3.5: Time Series Example Mapped in to Feature Space

Figure 3.5 illustrates how the transformation of time series data points from their original space to an alternative feature space enables the use of the linear approximation shown in Equation 3.8 to be used for estimation. The goal of SVR is to use an appropriate transformation function that will allow any arbitrary time function to be translated to a space where a linear fit would yield an accurate estimation.

The SVR primal objective function formulation (described in the following section) requires the product of a given data point cast into feature space and the training samples also cast in to feature space to be computed. This computation, as shown in Equation 3.12, defines the kernel function, or sometimes referred to as the inner-product kernel, which is essential to the formulation of the SVR optimization problem. Equation 3.13 is the more general inner product representation.

$$K(x, x_i) = \sum_{j=1}^{M} \phi_j(x)\phi_j(x_i) \tag{3.12}$$

$$K(x, z) = \langle \phi(x), \phi(z) \rangle \tag{3.13}$$

where M is the number of dimensions in feature space and *i* is an arbitrary index to a time series data point.

There are several requirements and features associated with kernel functions, with some of them listed below:

- Kernel functions must be semi-positive definite (this requirement is associated with Mercer's Theorem as related to kernel functions).
- Kernel functions are symmetric.
- A linear combination of kernel functions is also a kernel function.

A more exhaustive list of kernel function requirements can be found in references [22, 23].

One can now formulate an arbitrary kernel function as necessary following the requirements stated above. For the purpose of this research, three candidate kernel functions were considered for CMPSO. The first is typically called a polynomial kernel or polynomial learning machine as is defined in Equation 3.14.

$$K(\boldsymbol{x}, \boldsymbol{x_i}) = \left(\boldsymbol{x^T x_i} + 1\right)^p \tag{3.14}$$

where *i* is an arbitrary index to a time series data point and *p* is an integer that is typically defined a priori by the user.

The second type of kernel is one that uses the hyperbolic tangent function, sometimes referred to as a two-layer perceptron and is shown in Equation 3.15.

$$K(\boldsymbol{x}, \boldsymbol{x_i}) = tanh\left(b_0 \boldsymbol{x^T x_i} + b_1\right) \tag{3.15}$$

where b_0 and b_1 are selected parameters (in some cases the selection of b_0 and b_1 might violate the kernel restriction requirements).

The last kernel function considered is the radial basis function or exponential function as shown in Equation 3.16.

$$K(\boldsymbol{x}, \boldsymbol{x_i}) = e^{-\frac{1}{2\sigma^2}\|x - x_i\|^2} \tag{3.16}$$

It is observed that for the vast majority of the literature reviewed for this research as documented in [1], the radial basis function is the most widely used in all of the non-linear time series prediction applications. It should also be noted that there is no specific mathematical or computational complexity constraint in using this function vs. using any of the other listed functions. A study of specific kernel function performance for time series estimation should be considered for future research.

3.4 Primal Objective Function Formulation

Beginning with the SVM problem formulation in Section 3.1, we have defined a linear representation of an optimal separating hypersurface between two representative classes of data. This formulation showed that finding the optimal hypersurface meant that one had to find the maximum separation distance between the two classes. This was equivalent to minimizing the

Euclidian norm of the weights in the linear representation shown in Equation 3.2. This concept was extended to time series regression problems where the same linear approximation was illustrated in Equation 3.8. As it turns out, the same minimal flatness criterion for the weights is required for the regression problem as well as the pattern recognition problem. The formulation in Equation 3.8 is useful for applications where a linear fit to the data is sufficient; however almost all real world applications require some sort of nonlinear curve fitting. The same framework can be applied to nonlinear applications if the data is transformed into feature space via the use of kernel functions as defined in Section 3.3.

Given the above nonlinear curve fitting problem and associated support vector and kernel function techniques described previously, we can now develop what is referred to as the primal objective function definition of SVR. The primal objective function is formulated to be a convex optimization problem that ensures the flatness criterion is met. A convex optimization problem is defined as finding an optimal point of some function that satisfies Equation 3.17 [123].

$$f(x^*) \leq f(x) \text{ for all } x \in X \tag{3.17}$$

where f(x) is a real valued function and x* is the point at which the function is the smallest.

For SVR, we formulate a convex optimization problem with two linear constraints as shown in Equations 3.18 and 3.19.

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3.18}$$

$$\text{subject to} \begin{cases} y_i - f(x_i) \leq \varepsilon \\ f(x_i) - y_i \leq \varepsilon \end{cases} \tag{3.19}$$

where y is a sample of the training set and f(x) is the estimation of the same sample in the training set. The 0.5 scaling constant is presented for convenience relative to the dual objective function formulation presented in the next section. The error bound ε was shown in Figure 3.4.

This type of optimization problem is also referred to as a quadratic programming problem as the objective function, in this case the minimization of the Euclidean norm of the weights squared as seen in Equation 3.18, is quadratic. Also associated with quadratic programs is set of linear constraints as formulated for SVR in Equation 3.19. As stated in Section 3.2, we would like to minimize the error associated with the difference between the actual data represented as y and the estimated data set represented by f(x). The constraints shown in Equation 3.19 do not allow for errors outside the ε error bound. Any errors found outside this bound would make this optimization problem infeasible to solve.

In reality, one would like to allow for some amount of error in the formulation shown in Equation 3.18 and make the overall quadratic program have a feasible solution. This can be seen by the errors shown in Figure 3.4 which are represented by points A and B. The error, denoted as ξ, is clearly outside the ε error bound, violating the linear constraints in Equation 3.19. The variable ξ as shown in Figure 3.4 is generally referred to as a slack variable, with the obvious connotation for allowing errors outside the ε bound. A reformulation of the quadratic function in Equation 3.18 and the linear constraint functions in Equation 3.19 are now presented in Equations 3.20 and 3.21.

$$\text{minimize} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{3.20}$$

$$\text{subject to} \begin{cases} y_i - f(x_i) \leq \varepsilon + \xi_i \\ f(x_i) - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \\ C > 0 \end{cases} \tag{3.21}$$

The reformulation of the objective function in Equation 3.20 now allows for error beyond the ε bound, but at a cost. This is also reflected in the linear constraint equations shown in

Equation 3.21. It should be noted that the two slack variables represented in this formulation only represent the amount of error beyond the ε error bound. Only one of the two slack variables will be non-zero, depending if the estimated function f(x) is above or below the training value y. The constant C is typically referred to as a regularization or capacity constant and controls the tradeoff between the amount of slack (error) tolerated in the formulation vs. the flatness (original objective) in the final estimation.

The quadratic programming problem stated in Equation 3.20 is related to the "soft margin" loss function as stated in [23, 37]. The actual loss function itself, otherwise referred to as the ε-insensitive loss function, is defined in Equation 3.22 and illustrated in Figure 3.6.

$$|\xi| = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases} \tag{3.22}$$

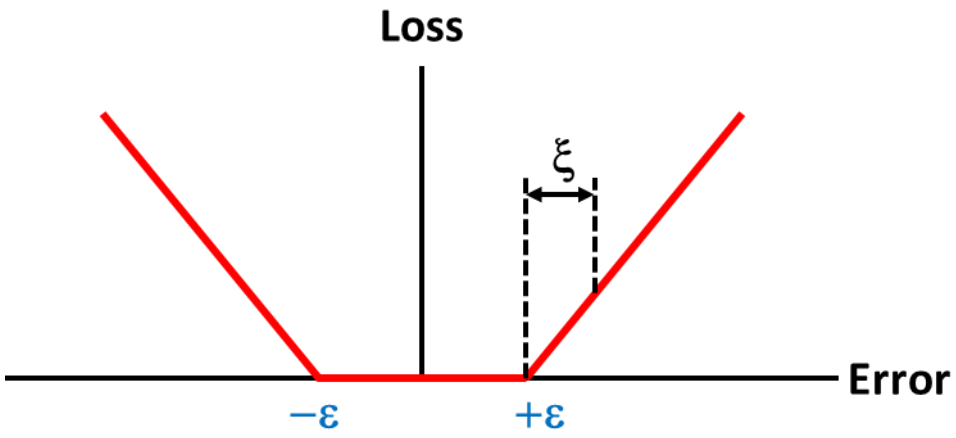

Figure 3.6: ε Insensitive Loss Function

From Figure 3.6, it is clearly seen that there is no loss, or penalty, for estimates that lie within the ε error bound and the error outside the e error bound grows linearly. There are many different kinds of loss functions that can be used to manage the error that may occur beyond the ε

error bound. This research focuses on the ε insensitive loss function as described above. It is also the most prevalent loss function observed in the literature search for this research.

We have now formulated what is referred to as the primal objective function and its associated linear constraints. In general, solving this type of optimization problem in its present (primal) form can be challenging. An alternative form of the optimization problem must be formulated.

3.5 Dual Objective Function Formulation

Solving the quadratic programming problem shown in Section 3.4 requires the use of Lagrange multipliers. The Lagrange function L defined in Equation 3.23 is the sum of the primal objective function in Equation 3.20 and the negative sum of all the products between the constraints and corresponding Lagrange multipliers.

$$L = g_1 + g_2 + g_3 + g_4 + g_5 \tag{3.23}$$

$$g_1 = \frac{1}{2}\|w\|^2 \tag{3.24}$$

$$g_2 = C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{3.25}$$

$$g_3 = -\sum_{i=1}^{N}\beta_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) \tag{3.26}$$

$$g_4 = -\sum_{i=1}^{N}\beta_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b) \tag{3.27}$$

$$g_5 = -\sum_{i=1}^{N}(\eta_i\xi_i + \eta_i^*\xi_i^*) \tag{3.28}$$

$$\text{subject to: } \beta_i, \beta_i^*, \eta_i, \eta_i^* \geq 0 \tag{3.29}$$

where L is the Lagrange function, N is the total number of data samples, $\beta^{(*)}$ and $\eta^{(*)}$ are the set of Lagrange multipliers, and w, x, y, b, ε, ξ, and C are all defined in the previous section. Note that w, b, ξ, and ξ* are all considered primal variables.

In order for an optimal solution to be found, the partial derivative of L with respect to each of the primal variables has to equal zero as shown in Equations 3.30, 3.31, and 3.32.

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{N} (\beta_i - \beta_i^*)\, x_i = 0 \tag{3.30}$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{N} (\beta_i^* - \beta_i) = 0 \tag{3.31}$$

$$\frac{\partial L}{\partial \xi^{(*)}} = C - \beta_i^{(*)} - \eta_i^{(*)} = 0 \tag{3.32}$$

The next step is to substitute the findings in Equations 3.30, 3.31, and 3.32 into the Lagrange function L. By combining all the terms in Equations 3.25 through 3.28 (g_2 through g_5) and applying the optimality point condition in Equation 3.32, all terms associated with $\eta^{(*)}$ and $\xi^{(*)}$ disappear in the Lagrange function. The remaining terms are gathered along with Equation 3.24 and restated as the dual objective function in Equation 3.33 along with the remaining constraints shown in Equation 3.34.

$$\text{maximize} \quad \sum_{i=1}^{N} y_i(\beta_i - \beta_i^*) - \varepsilon \sum_{i=1}^{N} (\beta_i + \beta_i^*) - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_i^*)\big(\beta_j - \beta_j^*\big)\, \langle x_i, x_j \rangle \tag{3.33}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{N} (\beta_i - \beta_i^*) = 0 \\ 0 \le \beta_i, \beta_i^* \le C \end{cases} \tag{3.34}$$

There are several important observations regarding the dual objective function formulation:

- There are only one set of Lagrange multipliers remaining ($\beta^{(*)}$).
- Both the bias term b and the weights w disappear in the dual optimization problem.
- The sum of the Lagrange multipliers is equal to zero (Equation 3.34).

In order for the dual objective formulation to be valid, the Karush-Kuhn-Tucker (KKT) constraints (see literature sited in Section 3.1) must hold. Simply stated, the product of the Lagrange multipliers and the primal objective function constraints must equal zero. Equation 3.35 shows the KKT constraint for the time series estimate and error bound and Equation 3.36 shows the KKT constraint for the slack variables (with a substitution for $\eta^{(*)}$ from Equation 3.32).

$$\beta_i^{(*)}\left(\varepsilon + \xi_i^{(*)} \mp y_i \pm (\langle w, x_i\rangle + b)\right) = 0 \tag{3.35}$$

$$\left(C - \beta_i^{(*)}\right)\xi_i^{(*)} = 0 \tag{3.36}$$

There are several additional observations regarding the dual optimization problem with regards to the KKT constraints:

- The product of the Lagrange multipliers β_i and β_i* are zero as you cannot have two non-zero slack variables for the same data point simultaneously.
- The value of the Lagrange multipliers at the optimal solution point is equal to the capacity term C.
- Only data points that lie outside the e error bound have a corresponding Lagrange multiplier equal to C. The remaining data points that lie within that bound have a

zero Lagrange multiplier value. This means the SVR solution is sparse in the sense that not every data point is necessary to compute a solution. The data points that are associated with nonzero Lagrange multipliers are known as support vectors.

The partial derivative shown in Equation 3.30 leads to another important conclusion. Solving for w, we can now substitute Equation 3.30 into our original time series regression estimation Equation 3.8 and formulate the SVR time series estimation equation as shown in Equation 3.37.

$$f(x) = \sum_{i=1}^{N} (\beta_i - \beta_i^*) \langle x, x_i \rangle + b \tag{3.37}$$

We can now represent an approximation to a time series with only the sample (training set) data points and the Lagrange multipliers found from the dual optimization problem (and a bias term b). However, the formulation in Equation 3.37 is for a linear regression fit only. For a non-linear curve fit, we can now use a kernel function as described in Section 3.3 by substituting Equation 3.13 into Equation 3.37 as shown in Equation 3.38.

$$f(x) = \sum_{i=1}^{N} (\beta_i - \beta_i^*) K(x, x_i) + b \tag{3.38}$$

Equation 3.38 is the general SVR formula for estimating any given non-linear time series. In addition, for the CMPSO formulation (to be discussed in Chapter 5), we would like to make a change of variables for the Lagrange multipliers as shown in Equation 3.39.

$$\alpha_i = \beta_i - \beta_i^* \tag{3.39}$$

Substituting the Kernel function found in Equation 3.38 and the change of variables in Equation 3.39 into the dual optimization objective function and constraints in Equations 3.33 and 3.34 as well as the time series estimation Equation in 3.38, we now have the final version of the SVR formulation objective function (Equation 3.40), the associated constraints (Equation 3.41), and the SVR time series estimation function (Equation 3.42).

$$\text{maximize} \quad \sum_{i=1}^{N} \alpha_i y_i - \varepsilon \sum_{i=1}^{N} |\alpha_i| - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \tag{3.40}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{N} \alpha_i = 0 \\ -C \leq \alpha_i \leq C \end{cases} \tag{3.41}$$

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i) + b \tag{3.42}$$

3.6 SVR Optimality, Architecture and Free Variables

Now that we have defined both the primal and dual objective functions for SVR, we must solve the dual objective function by finding a set of Lagrange multipliers such that the objective function is maximized subject to its constraints. The condition of optimality for the SVR problem can be found by finding the difference in the primal and dual objective function outputs, otherwise known as the feasibility gap. For optimality, the feasibility gap is equal to zero as the primal and dual optimization problems converge to the same optimal point. Equation 3.43 shows a typical calculation of this gap (this is sometimes referred to as the duality gap, which will be referred to with the variable γ).

$$\text{Duality Gap} = \frac{\text{Primal Objective - Dual Objective}}{|\text{Primal Objective}| + 1} = \gamma \tag{3.43}$$

The calculation of the primal objective value in Equation 3.43 can be derived by adding the last term in Equation 3.30 to the ξ slack values calculated by finding difference between the actual data (training) point y plus the ε bound and the estimate f(x). Other computational strategies exist, including specific implementation alternatives that can be used to find the optimal point. Regardless of implementation, the optimization problem is solved when the primal and dual objective functions approach the same value.

A suitable duality gap value associated with the calculation in Equation 3.43 is typically 0.001; however in Chapters 5 and 6 we show that the duality gap, in a more practical sense, can depend on the user's requirements and can be somewhat higher than this suggested value. The topic of techniques for finding optimal solutions, also known as quadratic program (QP) solvers, will be discussed later in Chapter 5.

The construct of the SVR is shown in Figure 3.7. A training data set is necessary to pass through the QP process in order to estimate the Lagrange multipliers, also noting that a specific Kernel function must be selected along with its user defined parameters. Once the Lagrange multipliers are found along with the bias term b, the support vectors are identified based on the values of the Lagrange multipliers, passed through the same kernel function used in training, and finally weighted by the Lagrange multipliers to estimate any arbitrary data point.

Table 3.1 summarizes the SVR formulation and identifies the data and processes shown in the SVR Architecture in Figure 3.7.

The challenge with implementing SVR is the determination of the three user defined variables ε, C, and σ. The focus of this research is to find an efficient method to determine these values for any arbitrary non-linear time series application.

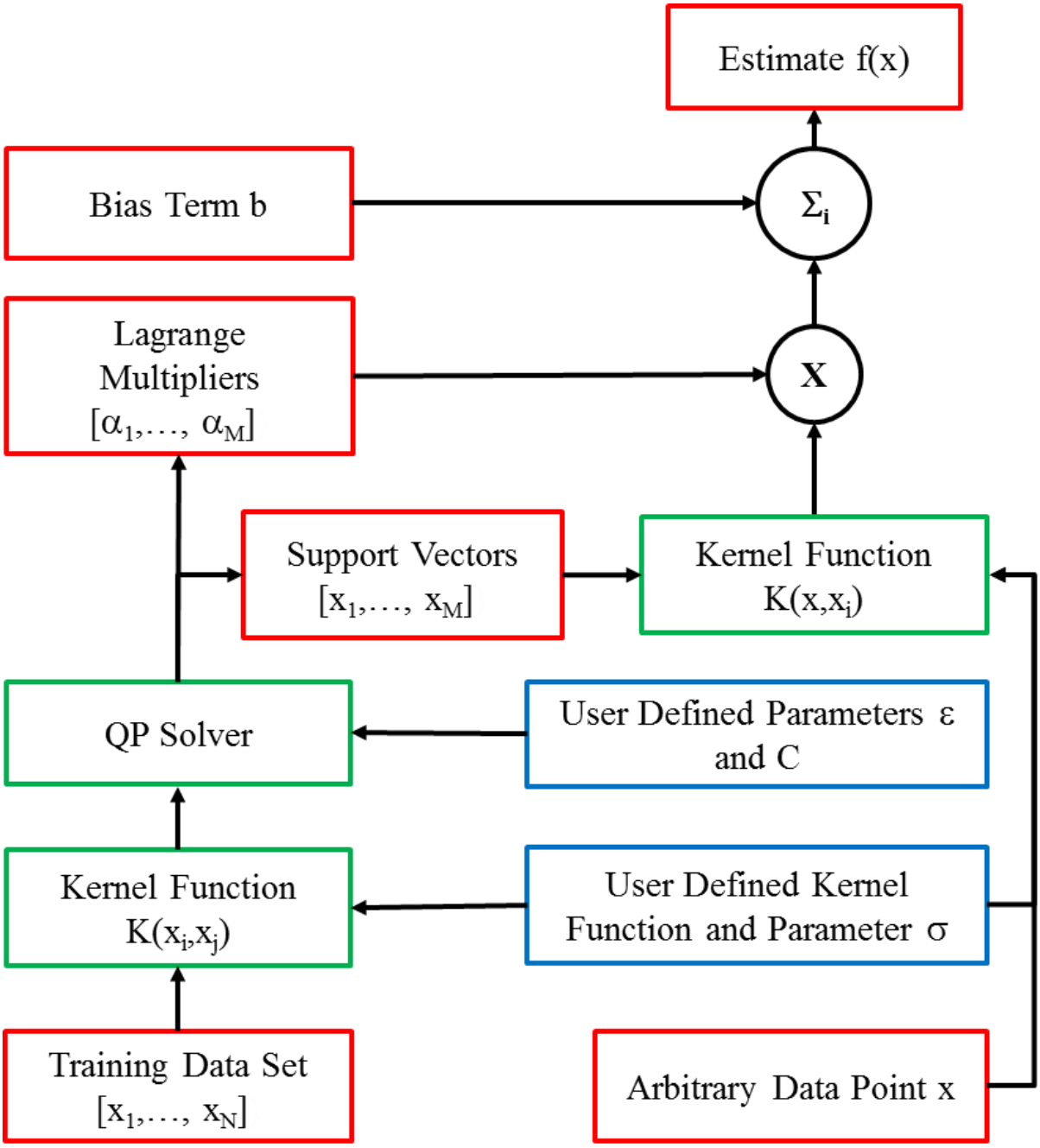

Figure 3.7: SVR Architecture

Table 3.1: A Summary of SVR Parameters, Processes and Functionality

Data or Process	Functionality	Notes
Training Set	Used to train the SVR	Defined as N real valued pairs (x,y).
Kernel Function	Project the data into feature space	User must define function type and associated parameters. For this research, the radial basis function (exponential function) is used.
QP Solver	A process that will solve the SVR dual objective function	User must define the capacity term C and the error bound ε. Many different types of QP solvers exist.
Lagrange Multipliers	Used to weight the output of the kernel function to estimate a real valued function.	Lagrange multipliers are found by the QP process and there are typically M non zero values where M < N. There is one nonzero Lagrange multiplier for each support vector.
Bias Term	An offset applied to the final time series estimate.	Can be computed in different ways; usually estimated to minimize the error between estimate and truth.
Support Vectors	A subset of the training data that have associated nonzero valued Lagrange multipliers.	Used to compute the estimate of f(x) for any arbitrary real valued point x.
User Defined Parameters ε, C, and σ	Free variables required to be selected by the user for any given application	Necessary to "tune" the SVR algorithm and its associated kernel function.

CHAPTER 4: PARTICLE SWARM OPTIMIZATION

4.1 SVR Parameter Optimization

As shown in Chapter 3, SVR is a viable time series prediction and regression algorithm that has many significant attributes. As with most time series prediction algorithms, there are usually challenges associated with tuning an algorithm to make it effective for any given application. SVR also has challenges in how it is tuned. As stated at the end of Chapter 3, SVR relies on the user to define a set of parameters, sometimes referred to as free variables, in order to work. For the purposes of this research, we are focusing specifically on the error bound ε, the capacity term C, and the radial basis function parameter σ. Note that we have selected a specific kernel function, the radial basis function, for this research. The optimization problem can get more difficult if another kernel function is used as more parameters may be necessary to tune. The problem of finding an optimal set of these parameters has been a subject of much research and it should be noted that no one heuristic has been identified. It should also be noted that the QP solver used to find the Lagrange multipliers (shown in Figure 3.7) is essentially an optimization problem as well. The QP program solves for the dual objective function subject to its constraints.

Finding the optimal set of SVR user selectable parameters (free variables) is a mathematical optimization problem. The general form of this optimization is given in Equation 4.1 [124].

$$\text{maximize (or minimize)} \quad f(\boldsymbol{x})$$
$$\text{subject to:} \begin{cases} \boldsymbol{a}^T\boldsymbol{x} \geq b_i, & i \in M_1 \\ \boldsymbol{a}^T\boldsymbol{x} \leq b_i, & i \in M_2 \\ \boldsymbol{a}^T\boldsymbol{x} = b_i, & i \in M_3 \\ x_j \geq 0, & i \in M_4 \\ x_j \leq 0, & i \in M_5 \end{cases} \tag{4.1}$$

Equation 4.1 is an example of an optimization problem with linear constraints. The variables [x_1, …, x_M] are called decision variables. There are M total variables, each divided in to five subsets M_1 through M_5. The vector **a** contains linear multipliers for the decision variables. The function f(**x**) is said to be a cost or objective function. A vector **x*** that satisfies all of the constraints and maximizes (or minimizes) the objective function is said to be a feasible solution. It should be noted that the primal and dual examples in the previous chapter are said to be convex as they are quadratic in nature.

Given the definition above, we would now like to formulate an optimization problem for the SVR free parameters. We have already identified the three variables we would like to optimize: ε, C and σ. Note this formulation assumes a radial basis kernel function as there is only one parameter (σ) associated with that function. In addition, we also need to solve for the Lagrange multipliers as well as the bias term b to complete the SVR solution. Given a training set **y** of real valued numbers, we would like to find an estimate f(x) for any real valued number x using the SVR formulation. The SVR formulation is found by minimizing the duality gap γ defined in Equation 3.43. Equation 4.2 is the formulation of the optimization problem we are trying to solve along with it constraints in Equation 4.3.

$$\text{minimize } h(f(\boldsymbol{x}), \boldsymbol{y}) \tag{4.2}$$

$$\text{subject to: } \begin{cases} \gamma \leq 0.001 \\ \sum_{i=1}^{N} \alpha_i = 0\,, \quad N > 0 \\ -C \leq \alpha_i \leq C, \quad i \in N \\ \varepsilon > 0 \\ C > 0 \\ \sigma > 0 \end{cases} \tag{4.3}$$

The variable γ is the duality gap defined in Equation 4.42. The Lagrange multipliers defined in Chapter 3 are denoted as the vector $\boldsymbol{\alpha}$ (one for each training set sample). The vector **x** represents the N independent variables (time in this case) and the **y** vector represents the corresponding N training (dependent) values respectively. The objective function h(.) is the optimization objective function that needs to be defined. Its purpose is to evaluate the estimate f(**x**) generated by the SVR process against the actual training values **y**. The formulation of h(.) will be discussed further in Chapter 5.

4.2 Introduction to Particle Swarm Optimization

Equations 4.2 and 4.3 are the formulation of the objective function and its associated constraints for finding the SVR user defined parameters for any arbitrary time series. The optimization process selected for this research is Particle Swarm Optimization (PSO).

First developed by Kennedy and Eberhart ([4] through [8]) in 1995, PSO mimics the real life process of a swarm of animals or insects searching for food. The goal for each individual of the swarm, where each individual is termed a "particle", is to scavenge in a pre-defined search region for the place that has the most food. Each particle has no prior knowledge of the amount of food in any given location in the search region. As a particle travels through the search region, each will remember the location in their search region where they have found the highest density of food and the entire swarm will remember collectively where the highest density of food was found in the areas that have been searched. As time passes, the particles move through

the search region remembering where they have found the most food and where the swarm has found the most food collectively. Each particle is driven by three motivational forces: 1) the current direction they are traveling, 2) the location where the individual particle has found the most food, and 3) the location where the group has collectively found the most food. These three motivational forces are also modified by a "wandering" factor. This factor mimics the real world behavior of animals or insects, meaning that particles do not exactly vector towards the exact locations of their individual or collective findings. There is some randomness to their motion in time. Eventually, the collection of particles, or "swarm", will all eventually gravitate to some location where the most food has been found.

This optimization process has been shown to outperform other optimization methods such as genetic algorithms [10]. In addition, this technology in general has been shown to be more efficient to implement as compared to genetic or evolutionary optimizers [10, 11]. The ease of implementation is one of the highlighted attributes in using this specific algorithm. The following PSO formulation is based on the previously stated references in this section along with the work found in [9, 11, and 125].

4.3 Particle Swarm Optimization Terminology

PSO has a specific set of terminology and definitions that are somewhat unique to this optimization process. This section outlines the terminology and definitions that will be used throughout the dissertation.

The first definition is the particle. It represents the individual representation of a point in the search space. It is defined by its location and direction of motion, both in position and velocity. It also has a memory function, as each particle remembers the location where it found

the most food and it also remembers the location where the entire swarm has found the most food. The collection of the entire set of particles is called the swarm.

Position is defined as a particle's location in space. Unlike a bird or insect real world representation of position in a three dimensional space, a particle's position can be N dimensional. The particle's velocity is also defined in the same N dimensional space and can be limited by a user defined bound.

As mentioned with the particle definition is the memory function which is used to store the location of the most food found by the individual particle and the swarm. The term used for the location of the place where the particle found the most food is referred to as the particle's "personal best", "particle best", or more commonly pbest for short. The term used for the location of the place where the entire swarm has found the most food is referred to as the "global best", or gbest for short.

As in the real world, animals and insects are usually limited to a certain region where they can search for food. PSO has similar constraints for the particles. As the particles move through space, eventually they will reach a boundary of their search region. The PSO algorithm has different associated methods by which to manage these situations and is discussed further in Section 4.5.

4.4 Particle Swarm Optimization Algorithm

Given the above description of PSO and its associated functions and processes, the goal in this section is to define the PSO algorithm and tailor it to the SVR time series regression problem and SVR free variable selection.

The first step in the PSO algorithm definition process is to define the solution space. This is analogous to defining a physical region for a swarm of birds or insects to hunt for food.

The individual particle parameter definitions and boundary limits that define the solution space for this problem are given in Table 4.1.

Table 4.1: PSO Particle Definitions and Solution Space Boundaries

Particle Parameter	Definition	Parameter Bounds
Error Bound ε	The amount of error tolerated by the SVR time series estimation process.	$\varepsilon \in \mathbb{R}$ $0 < \varepsilon \leq \varepsilon_{max}$
Capacity Term C	A user defined variable that balances the flatness of the estimating function and the amount of error beyond the error bound to tolerate.	$C \in \mathbb{R}$ $0 < C \leq C_{max}$
Radial Basis Kernel Function Parameter σ	A user defined variable that will adjust the projection of the sample time series into feature space via the kernel function.	$\sigma \in \mathbb{R}$ $0 < \sigma \leq \sigma_{max}$
Lagrange Multipliers $[\alpha_1, \ldots \alpha_N]$	The Lagrange multipliers are used to weight the support vectors to produce an estimate of a time series function. There is one Lagrange multiplier for each data point in the training set (total of N points).	$\alpha_i \in \mathbb{R}$ $-C \leq \alpha_i \leq C$
Bias Term *b*	The amount of offset between the estimated function and the training set values.	$b \in \mathbb{R}$ $-b_{max} \leq b \leq b_{max}$

As can be seen, the solution space boundaries are given in Table 4.1 in the "Parameter Bound" column. There are a total of N+4 parameters that define a particle's solution space, where N is the number of training samples for a given time series data set. The actual bound limits that will be used in implementation will be discussed in Chapter 5.

The next step in the PSO algorithm development process is the definition of the PSO fitness function. This was defined in Equation 4.2 as h(f(**x**),**y**), with the underlying assumption that this function would produce a single real value that would give some measure of "goodness"

between the estimated value f(**x**) and the actual training set **y**. Also recall the necessary process to complete the SVR optimization shown in Chapter 3 that an optimization had to be completed and a duality limit γ needed to be achieved as seen in Equation 3.43. It now appears that there are more than one set of objectives to be met for this particular problem. These types of problems with multiple objectives using PSO are often referred to as Multiple Objective Particle Swarm Optimization (MOPSO).

MOPSO is a candidate technique that can be considered for this application since more than one "fitness" criteria is required. A survey of MOPSOs can be found in [126] along with their implementations and applications. Reference [127] also details the use of MOPSOs and qualitative performance results associated with applications with more than one fitness criteria. Many of the MOPSO techniques reviewed and referenced above were considered for solving this specific PSO/SVR problem. However, the solution of the SVR optimization problem is dependent on ε, C, and σ, making the MOPSO techniques cited difficult to implement for this type of problem. A further discussion of the PSO fitness function formulation will be presented in Chapter 4 specifically tailored for the SVR optimization problem.

The next step in defining the PSO algorithm is to select the total number of particles to use and initialize the position and velocity of the particles in solution space. Typically the total number of particles is a function of the performance of the computing hardware available for solving the specific problem. The initial position of any given particle for each dimension (N+4 in this case) is done by selecting a random point for each of the parameters (dimensions) detailed in Table 4.1. The value of the initial positions parameters is a uniformly distributed random number scaled by the boundary limit of each parameter. Equations 4.4 and 4.5 are the representations of the particle's position and velocity respectively as this notation will be used

throughout the dissertation. Equations 4.6 through 4.10 show the initial position calculations for each particle parameter, with each particle denoted by the index k.

$$\text{particle position: } p_{k(\varepsilon,C,\sigma,\alpha_1 \ldots \alpha_N,b)} \tag{4.4}$$

$$\text{particle velocity: } v_{k(\varepsilon,C,\sigma,\alpha_1 \ldots \alpha_N,b)} \tag{4.5}$$

$$p_{k,0(\varepsilon)} = \varepsilon_{max} * rand \tag{4.6}$$

$$p_{k,0(C)} = 2 * C_{max} * rand - 1 \tag{4.7}$$

$$p_{k,0(\sigma)} = \sigma_{max} * rand \tag{4.8}$$

$$p_{k,0(\alpha_i)} = 2 * C_{max} * rand - 1 \tag{4.9}$$

$$p_{k,0(b)} = 2 * b_{max} * rand - 1 \tag{4.10}$$

The function "rand" is a uniform random number generator that produces real numbers between zero and one inclusive. Each instance of "rand" produces an independent and identically distributed (iid) random number. For convenience, we have shortened the particle (and velocity) notation to identify only the dimension of the particle of interest. Also, we introduced the zero subscript to indicate the number is the initial value. As seen in Equations 3.6 through 3.10, the initial particle location is within the search boundaries.

Using the same notation in Equations 4.4 through 4.10, a similar process is used to initiate the particle velocities as shown in Equation 4.11.

$$v_{k,0(z)} = \delta_z f_z(p_{k,0(z)}) \tag{4.11}$$

The initial velocity calculations for each dimension uses the exact same random value generation process shown in Equations 4.6 through 4.10, represented by the $f_z(.)$ function in 4.11. The variable z represents any of the PSO parameters. The factor δ_z is used to scale the value of the random number generated for each parameter and will generally be less than one. More

details regarding the selection of the bounds and velocity scale factor will be discussed in Chapter 5.

At this point every particle has been given and initial position and velocity. Before the particle can move through solution space, the pbest values for each particle need to be calculated. This is done by evaluating the fitness function h(f(**x**),**y**) defined in Equation 4.2. The pbest value scores are stored for each particle as well as the position where pbest was found. The highest value of all the pbest values is then selected to be the gbest value as well as the location where gbest was found. Again, location is defined as the value of the PSO parameters where the pbest and gbest values were identified.

For notation, we would like to define N_s as the total number of time series samples in any given time series training set and N_p as the total number of particles in the PSO process. We have now evaluated the fitness function for each particle and have selected the best result (gbest). Equation 4.2 defined the fitness function objective as a minimization. In practice, there must be some practical limit to be obtained in order to find a stopping point for the PSO process. If the gbest value is found to less than some optimal number, the PSO process will end. If a limit is not reached, then the particles must move in order to find a better fitness function value.

The motion of the particles through solution space is analogous to the physical motion of birds or insects through physical space. The general equation of particle motion is defined in Equation 4.12.

$$p_{k,j+1} = p_{k,j} + \Delta_t v_{k,j} \tag{4.12}$$

For convenience we have dropped the (z) notation with the understanding that each particle's (indexed by k) new position at step j+1 is dependent on the same particle parameter at the previous step j plus the velocity term of the same parameter multiplied by a time factor Δ_t.

This is the basic physical world representation of a constant velocity equation of motion. The time factor Δ_t should not be confused with the time series **x** independent variable as stated in the regression formulation in Chapter 3. PSO defines the factor Δ_t to be unity. It should be noted that the notation j refers to each step in an iterative process, where each step is referred to as an epoch.

Equation 4.12 is the equation of motion that is used to update the position of every particle in the swarm. The particles will move through solution space until some minimum value of the fitness function is found. For every epoch, the fitness function will be evaluated for every particle and if a better value for the fitness function is found for any individual particle, its pbest will be updated and stored. Also at every epoch, the best pbest score of all the particles will be compared to the stored gbest value. If a better pbest value is found, the current gbest location and value will be updated and stored.

After the evaluation of pbest and gbest at every epoch, the velocity of each individual particle will need to be updated in order to continue moving through solution space. Equations 4.13 through 4.16 are the update functions for the particle velocity.

$$v_{k,j+1} = m_1 g_1 + m_2 g_2 + m_3 g_3 \tag{4.13}$$

$$g_1 = v_{k,j} \tag{4.14}$$

$$g_2 = rand * \left(pbest_{k,j} - p_{k,j}\right) \tag{4.15}$$

$$g_3 = rand * \left(gbest_j - p_{k,j}\right) \tag{4.16}$$

$$m_1, m_2, m_3 \in \mathbb{R}, m_1, m_2, m_3 > 0 \tag{4.17}$$

A particles' velocity defined in Equation 4.13 for any given epoch is the weighted sum of three "forces" acting on the particle. The first term, expressed in Equation 4.14, is referred to as the inertial term with its associated inertial weight m_1. This term forces the particle to go in the

same direction it was travelling in the previous epoch, as related to inertia in the physical sense. The second term, expressed in Equation 4.15, is referred to as the cognitive term with its associated cognitive weight m_2. This term forces the particle in the direction where the particle's pbest location was found, as related to a cognitive decision by an individual bird or insect. The last term, expressed in Equation 4.16, is referred to as the social term with its associated social weight m_3. This term forces the particle in the direction where the swarm's gbest location was found, as related to the social swarm locating the best location in the searched area.

The three weights m_1, m_2, and m_3 are identified in Equation 4.17 and are generally user defined. It should be noted that there is no single heuristic to determine these three values, although the referenced literature makes some suggestions regarding recommended values. It is also clear that the ratio of these weights relative to each other will change the behavior of the particle as it moves through space.

One real world analogy associated with setting the three weights could be expressed as a definition of a particles "personality". A high inertial weight along with lower cognitive and social weights would tend to make the particle "stubborn" and not tend to be influenced by its own experience or the experience of the swarm. Another example would be a high cognitive weight with lower inertial and social weights, making the particle appear to be "selfish" or "independent" in the sense it is only worried about the maximum fitness function location for itself. The last analogy would be one where the social weight is higher than both the inertial and cognitive weights, making the particle tend to be more "social" or "dependent" as it will tend to be more influenced by all the other particles and tend to the location where the swarm has found the best fitness function. The study of determining an optimal set of weights along with their classifications is a subject for further research.

Figure 4.1 is a general view of one particle moving through a two dimensional space.

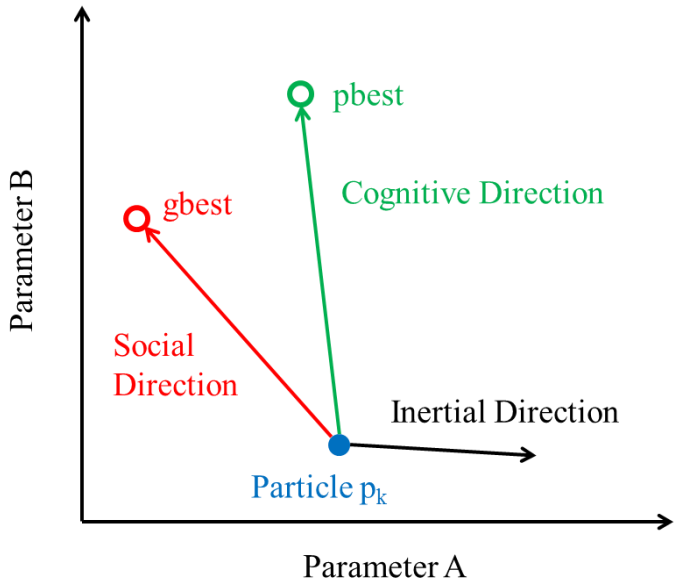

Figure 4.1: Particle Motion Through Solution Space

As seen in the two dimensional example in Figure 4.1, the particle (shown as the blue point) is being moved by three forces: 1) the inertial force (black vector), 2) cognitive force (green vector), and 3) the social force (red vector). The particle's position at the next epoch will result from the vector sum of those three forces for every parameter (N_S+4 dimensions in this PSO formulation).

Another important aspect of the PSO algorithm formulation is the use of the "rand" functions in the cognitive and social "force" Equations in 4.15 and 4.16. Again, the rand function is a uniformly distributed iid function that produces a random value [0, 1]. The rand function produces two separate values for each term and two new random numbers for every epoch. The purpose of this factor is to mimic the real world "wandering" behavior of birds and insects while they are attempting to find the location with the best fitness function. This adds an additional weighting factor to those terms and forces the particle to move in a random fashion towards its pbest and the swarm's gbest value.

Figure 4.2 illustrates the general PSO algorithm process.

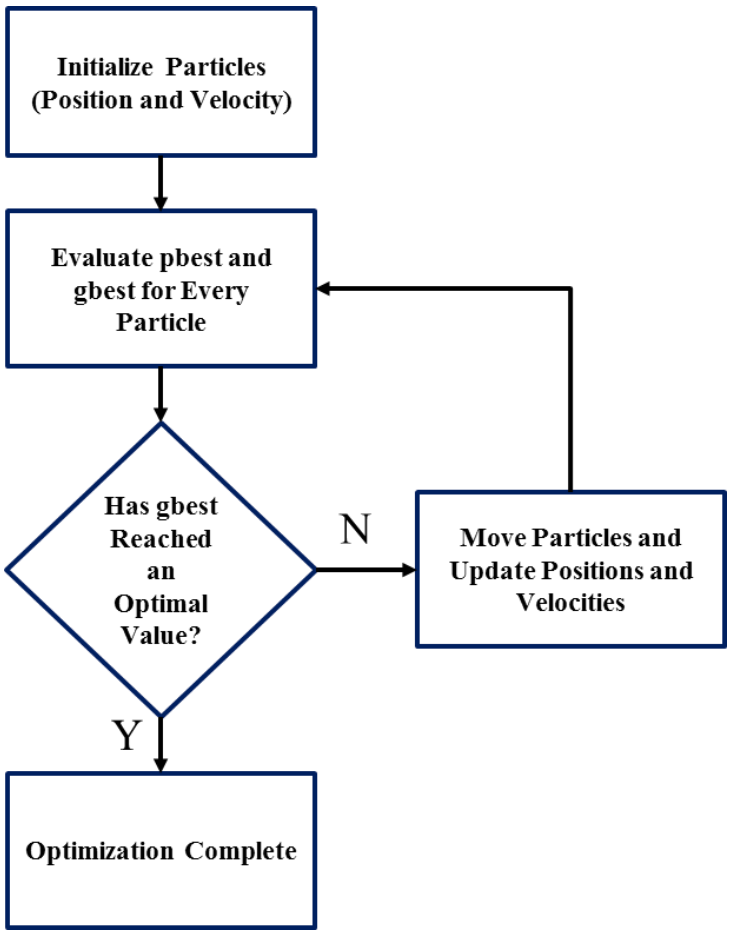

Figure 4.2: General PSO Optimization Process

The last part of the optimization described in the next section deals with the case when a particle travels outside the solution space boundary for any given PSO parameter.

4.5 PSO Boundary Conditions

There are times when a particle's trajectory can force the particle outside any given boundary of the N_s+4 PSO parameters (the solution space). There are several different methods available to handle these conditions for PSO.

The first method for handling a boundary condition violation is called an "absorbing wall". In this case when any given dimension of the particle exceeds the upper or lower bound,

the velocity related to that dimension is set to zero. This technique mimics the particle traveling along the bound of that parameter, or being "absorbed" by the wall. In this case, the only forces acting on future movements of the particle are due only to the cognitive and social forces, which should eventually pull the particle back to the defined search space. There may be an issue, in general, in trying to evaluate the fitness function for parameters outside the solution space. Using this technique will depend on the application and more specifically the fitness function calculation's tolerance to parameter values outside their defined bounds. If necessary, the parameter that exceeds the boundary value can be set to the boundary value, making for a more representative "absorption" of the boundary wall. It should be noted that this technique modification was not explicitly mentioned in the researched literature.

The next technique considered for PSO boundary violation is called "reflecting walls". When a particle exceeds its upper or lower boundary, the sign of the velocity for that parameter is reversed. This mimics the physical action of a ball bouncing off a wall. In this case, the particle's parameter that was out of bounds should return inside the boundary limit.

The last technique considered is called "invisible" walls. In this case when a particle's parameter exceeds the boundary limit, the velocity (of any parameter) is unchanged and the particle is allowed to pass through the boundary. The key aspect of this technique is the fact that the fitness function is not evaluated when any of the PSO parameters exceed their defined bounds. This will save on computation resources. Since there is no fitness evaluation, pbest will not be updated for the particle. Similar to the absorbing walls technique, the particle is expected to be pulled back into the solution space by the cognitive and social forces.

Table 4.2 illustrates the advantages and challenges associated with each of the three boundary condition techniques and Figure 3.3 illustrates the particle behavior when the parameter boundaries are exceeded.

Table 4.2: PSO Boundary Condition Advantages and Challenges

Boundary Condition Technique	**Advantages**	**Challenges**
Absorbing Wall	• Particle will not (necessarily) leave the solution space, making each evaluated pbest, gbest feasible. • Condition evaluation for changing velocity and technique implementation are simple.	• Evaluation of fitness function while particle parameter is outside the boundary may not be possible. • Time (number of epochs) required for particle to reenter the solution space may be considerable depending on the weighting factors $m_{1,2,3}$. Also need to consider the case when (if) gbest is selected outside the solution space.
Reflecting Wall	• Particle will (should) only leave the solution space for one epoch • Condition evaluation and change of velocity sign simple to implement and evaluate.	• Evaluation of fitness function while particle parameter is outside the boundary for at least one epoch may not be possible. • When the particle is outside the boundary, the total number of swarm particles is decreased and the number of potential contributors to the gbest estimation is reduced (efficiency penalty). • Time (number of epochs) required for particle to reenter the solution space may be considerable depending on the weighting factors $m_{1,2,3}$ and if a true "absorption" of that parameter to the boundary is not implemented. Also need to consider the case when (if) gbest is selected outside the solution space.
Invisible Wall	• There is only one test to evaluate if the particle is outside the boundary; no necessary updates (calculations) to velocity or position. • Increase in computational efficiency as the fitness function will not be evaluated unless all the particle's parameters are within their bounds.	• Time (number of epochs) required for particle to reenter the solution space may be considerable depending on the weighting factors $m_{1,2,3}$. • When the particle is outside the boundary, the total number of swarm particles is decreased and the number of potential contributors to the gbest estimation is reduced (efficiency penalty).

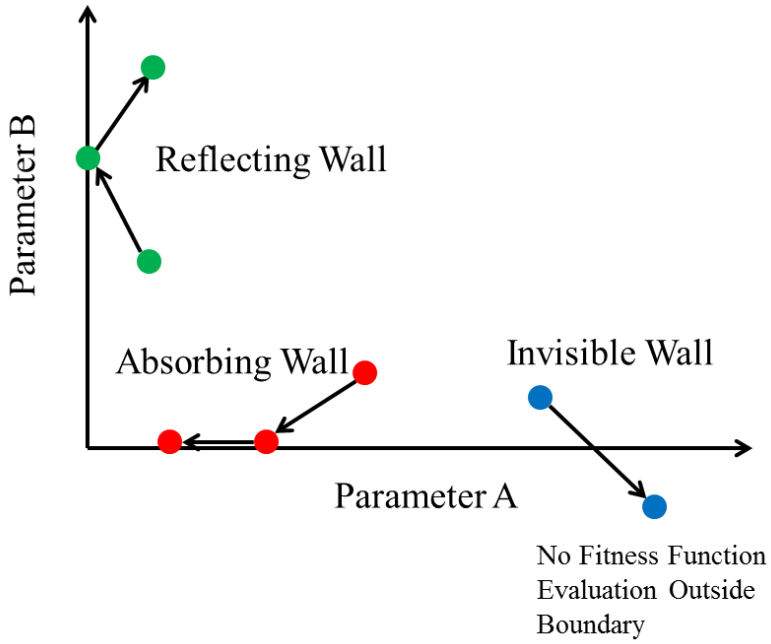

Figure 4.3: PSO Boundary Condition Illustration

As can be seen in the two dimensional example in Figure 4.3, the red points illustrate the absorbing wall technique as the motion of the particle is being absorbed into the lower bound of parameter B. The green points illustrate the reflecting wall technique as the boundary of parameter A is reached the velocity of the particle in that dimension is reversed. Finally the blue points illustrate the invisible wall technique where the particle is allowed to violate the lower bound of parameter B but once outside this boundary, the fitness function is not update and consequently the pbest (and possibly gbest) values are not updated.

4.6 Parameter Selection

The PSO algorithm described in the previous section requires the definition of three specific parameters: 1) inertial weight, m_1, 2) cognitive weight, m_2, and 3) social weight, m_3. These are user defined parameters and may be application dependent as noted in researched literature.

For the purposes of this study, the initial m_1 weight is set to 0.75 (approximate value of recommended settings in literature). However, in adapting the other two weights m_2 and m_3, it

was discovered for the CMPSO implementation that the recommended values above 1.0 resulted in the PSO algorithm not converging for the examples studied. The values for m_2 and m_3 are set to 0.75 and 0.25 respectively for the CMPSO implementation described in Chapter 5 based on similar convergence issues and observations.

The process of selecting the three weights is further complicated by the setting of the initial particle velocity. If the weights are too large, the second epoch of the PSO algorithm iteration (the first movement of the particles) could have the particles fly outside many if not all of the PSO parameter boundaries. This may require more epochs for the velocities to "settle" to reasonable values that maintain the particles' position in the solution space. Another factor in setting the weights is the computational power and efficiency of the computing platform. If a suitable computing environment is available, the sensitivity of these settings becomes less important as long as at some point the PSO algorithm converges. Again, it is user and application specific.

In summary, an "optimal" set of PSO parameters would require a completely different optimization formulation and could potentially be biased towards any one application. It is clear that this is a further research topic.

CHAPTER 5: CONSTRAINED MOTION PARTICLE SWARM OPTIMIZATION

5.1 Motivation and Objectives

We have now formulated the SVR solution for time series estimation and regression outlined in Chapter 3, noting that there are N_s+4 SVR specific parameters to optimize and three of those parameters are user defined with no given solution heuristic. In Chapter 4 we have suggested an optimization algorithm, PSO, to be used to find all these parameters.

There are many challenges associated with using PSO to optimize SVR parameters. First is defining the PSO fitness function in Equation 4.2 along with the requirement to minimize the duality gap γ defined in Equation 3.43 for SVR optimization. This would suggest a MOPSO based approached suggested in Section 4.4.

Alternative PSO approaches not based on multiple objectives have been proposed for SVR free parameter (ε, C, and σ) estimation. Hong [128] proposed the use of Chaotic PSO in use with SVR for electrical load forecasting. The technique uses a parallel approach in applying PSO to find the SVR free parameters, but the technique still requires an SVR solution algorithm for the QP problem. Guo *et al.* [129] uses PSO for finding the free parameters for a Least Squares Support Vector Machine (LS-SVM) application using medical related data for benchmarking (note that this application was specifically for hyper-parameter selection for SVM classification, not SVR based regression).

Other PSO related applications involving SVR and linear constraint problems have been studied. Yuan *et al.* [130] introduced a modified PSO algorithm for SVM training based on

linearly-constrained optimization using PSO and techniques proposed by Paquest and Engelbrecht [125, 131]. The method presented by Yuan, the "Modified Linear PSO - MLPSO", initializes the Lagrange multipliers randomly, but relies on re-initialization of the Lagrange multipliers should they go beyond the capacity value boundary C. The focus of the research in these citations is a PSO applied process for solving linearly constrained optimization problems.

None of the cited research offers a solution that will find a candidate solution for the PSO fitness function while optimizing the SVR objective function simultaneously. Note the SVR objective function solution also requires certain constraints to be met as shown in Equation 3.41. In addition, the cited references are generally tuned to a given application, which in general can be a limitation as different applications may require different implementations and setups of any given candidate solution.

We would now like to formulate a general solution for any given time series prediction and regression problem that will meet the functional objectives outlined in Table 5.1. Meeting all of the stated objectives in Table 5.1 would be advantageous as a general time series regression and estimation framework could be employed for any application. However, the development of such an optimizer will require some modifications to the given SVR and PSO formulations.

5.2 Support Vector Regression Dual Objective Function Reformulation

The first step in meeting the objectives stated in Table 5.1 is to reformulate the SVR dual optimization problem presented in Chapter 3, Equations 3.40 and 3.41 which are restated in Equations 5.1 and 5.2, respectively. Additionally, the associated Lagrange multiplier initial limits shown in Equation 4.9 are restated in Equation 5.3.

$$\text{maximize} \quad \sum_{i=1}^{N_S} \alpha_i y_i - \varepsilon \sum_{i=1}^{N_S} |\alpha_i| - \frac{1}{2} \sum_{i=1}^{N_S} \sum_{j=1}^{N_S} \alpha_i \alpha_j K(x_i, x_j) \tag{5.1}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{N_S} \alpha_i = 0 \\ -C \leq \alpha_i \leq C \end{cases} \tag{5.2}$$

$$p_{k,0(\alpha_i)} = 2 * C_{max} * rand - 1 \tag{5.3}$$

Table 5.1: General Time Series Regression and Estimation Functional Objectives

Objectives	Details	Notes
SVR Based Regression and Estimation	The basis of the general formulation will include SVR as the regression and estimation engine.	SVR requires, at a minimum, an optimization routine that will solve for the Lagrange multipliers and bias value. In addition, SVR free variables will have to be selected.
PSO Based Optimization	PSO will be used to, at a minimum, find an optimal set of SVR free variables.	PSO requires the definition of a fitness function as well as a method to handle parameter boundary violations.
Limited QP Use	Some form of QP solver will be necessary to find the Lagrange multipliers and bias term b for the SVR formulation.	A modified version of SMO can be used to solve the SVR problem. However, the use of this solver will be computationally expensive if every Lagrange multiplier and bias term needs to be calculated for every particle over every epoch.
Radial Basis Kernel Function Use	Selected based on general use in many applications.	Radial Basis Kernel functions only require one parameter to be tuned (typically user defined).
Adaptability	Any given two dimensional time series can be estimated regardless of scaling.	This will require a scaling of the sample data and a general solver to operate in the scaled space.

We now have a clear problem in bounding the Lagrange multiplier limits as they are dependent on the regularization parameter C. This means the candidate Lagrange multipliers solution space could contract or expand as a function of C for every particle over every epoch, which would be infeasible to implement.

The PSO framework described in Chapter 4 further requires that each PSO parameter have fixed boundary values. This can be achieved by scaling the dual objective function by the constant C (noting that C > 0) as well as scaling the constraint equations. The reformulation of the SVR dual objective function, its associated constraints, and particle initialization for the parameter are given in Equations 5.4 through 5.6, respectively.

$$\text{maximize} \quad C\left(\sum_{i=1}^{N_S}\alpha_i y_i - \varepsilon\sum_{i=1}^{N_S}|\alpha_i| - \frac{C}{2}\sum_{i=1}^{N_S}\sum_{j=1}^{N_S}\alpha_i\alpha_j K(x_i, x_j)\right) \tag{5.4}$$

$$\text{subject to} \begin{cases} \sum_{i=1}^{N_S}\alpha_i = 0 \\ -1 \le \alpha_i \le 1 \end{cases} \tag{5.5}$$

$$p_{k,0(\alpha_i)} = 2 * rand - 1 \tag{5.6}$$

Although the reformulated dual objective function in Equation 5.4 has an outer scale factor C that would not affect a maximum value solution, it will be necessary to use in finding the duality gap γ defined in Equation 3.43, thus it is kept in Equation 5.4 for completeness. Also noted is the second constraint in Equation 5.5. The Lagrange multipliers are now bounded by fixed values of +/- 1. Subsequently, the initialization of the Lagrange multipliers shown in Equation 5.6 is no longer dependent on any other PSO parameter. All of the PSO parameters are now independent based on the SVR dual objective function reformulation.

Based on this reformulation, the SVR estimation function in Equation 3.42 is now restated in Equation 5.7 with the appropriate scale factor.

$$f(x) = C\sum_{i=1}^{N}\alpha_i K(x, x_i) + b \tag{5.7}$$

This formulation allows the use of PSO to solve for the required parameters.

5.3 Particle Initialization and Constrained Motion

We now have a way to set up the PSO parameter bounds to fixed values due to the reformulation of the dual objective function as shown in Equation 5.4. The initialization equations given in Equations 4.6 through 4.10, with the substitution of Equation 5.6 for Equation 4.9, now define the initial positions and velocities for each of the particles. However, there is another constraint that must be held in Equation 5.5 which states the sum of the Lagrange multipliers must equal zero in order to have a feasible solution for the SVR dual objective function. The Lagrange multiplier initialization given in Equation 5.6 does not account for this constraint.

One option is to leave the Lagrange multiplier initialization as is and define a bound for the sum of the Lagrange multipliers as shown in Equation 5.8.

$$\left| \sum_{i=1}^{N_S} \alpha_i \right| \leq r \tag{5.8}$$

The problem with this formulation is that there is the introduction of yet another optimization parameter r and the actual estimation of the time series may not be optimal depending on this (possibly user defined) value. One would like to minimize the number of parameters to optimize as well as maintain the PSO formulation stated in Chapter 4 without adding additional computational complexity.

One way to solve this issue is to initialize the particles' position and velocity for the Lagrange multiplier parameters such that the Lagrange multiplier zero summation constraint is satisfied. This can be done by initializing N_S-1 particles' Lagrange multiplier positions and velocities as stated in Equation 5.6. The identification of the N_S-1 Lagrange multipliers to be set is randomly selected as well. The remaining Lagrange multiplier is set to the negative of the

summation of the N_S-1 Lagrange multipliers. An iterative process can be set up at initialization such that if the sum of the last Lagrange multiplier is greater than the unity bound (the second bound constraint in Equation 5.5), another set of random candidate Lagrange multipliers can be selected until the bound is met.

Figure 5.1 illustrates the PSO particle Lagrange multiplier position and velocity initialization process.

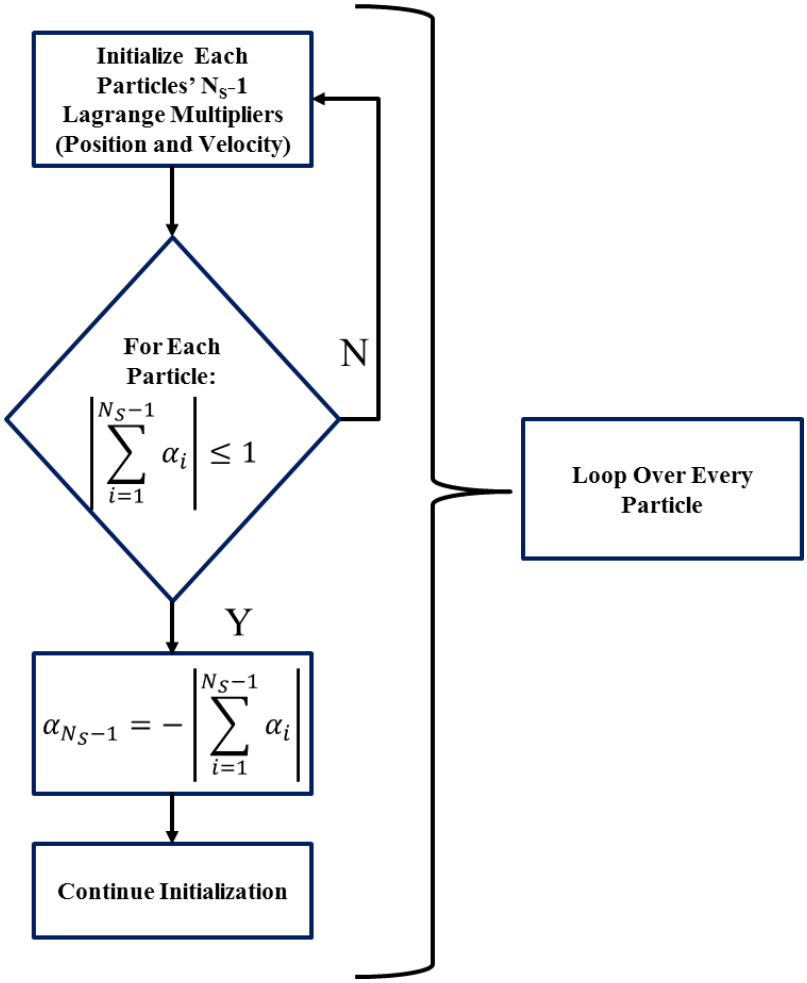

Figure 5.1: Particle Lagrange Multiplier Initialization Process

It should be noted that the notation in Figure 5.1 does not differentiate particle position or velocity as the initialization process is generalized for both. Also note that each randomly developed Lagrange multiplier is relocated to a random index value such that the last index is not

necessarily the location for the negative sum of the N_s-1 Lagrange multipliers. The setting of the initial Lagrange multipliers for each particle Lagrange multiplier position and velocity now satisfies the summation constraint in Equation 5.5. By definition, each particle is now a feasible solution to the SVR dual objective function defined in Equation 5.4 with the associated constraints shown in Equation 5.5.

As the PSO process iterates, each particle moves through solution space. We now restate the particles' equation of motion in Equation 4.12 in Equation 5.9 for any given dimension as well as the particle velocity update in Equation 5.10.

$$p_{k,j+1} = p_{k,j} + \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}^T \begin{bmatrix} v_{k,j} \\ rand * \left(pbest_{k,j} - p_{k,j}\right) \\ rand * \left(gbest_j - p_{k,j}\right) \end{bmatrix} \tag{5.9}$$

$$v_{k,j} = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}^T \begin{bmatrix} v_{k,j-1} \\ rand * \left(pbest_{k,j-1} - p_{k,j-1}\right) \\ rand * \left(gbest_{j-1} - p_{k,j-1}\right) \end{bmatrix} \tag{5.10}$$

Equation 5.9 states that for any particle k, its position in any dimension at epoch j+1 is computed as the sum of where it was (in any dimension) at epoch j plus the weighted sum of its velocity terms at epoch j. The cognitive and social velocity terms, shown as the difference between and the particles' pbest location and swarm's gbest location at epoch j and where the particle was at epoch j, are modified by a uniformly distributed iid random scale factor. Note the **m** vector is defined as $\mathbf{m} = [0.75\ 0.75\ 0.25]^T$ as discussed in Chapter 4.

Based on the initialization process shown in Figure 5.1 and the particle position and velocity update Equations 5.9 and 5.10, we now conclude the sum of the Lagrange multipliers in both position and velocity for any particle are zero for any epoch. This is achieved by initializing both the Lagrange multiplier position and velocity to zero simultaneously. Every

update to position is now a weighted sum of position and velocity terms that sum to zero, forcing the sum of the Lagrange multipliers to be zero at every epoch.

This concept constrains the motion of the particles' Lagrange multipliers to always sum to zero which means the zero summation constraint in Equation 5.5 is always met. Now every particle at every epoch is a feasible, but not necessarily optimal, candidate solution for the SVR optimization problem because both constraints in Equation 5.5 are met simultaneously. The constraining of the motion of the particles effectively reduces the solution space. This is the essential part of the PSO formulation and this formulation is called Constrained Motion Particle Swarm Optimization or CMPSO for short.

5.4 PSO Boundary Condition Selection

As described in Section 4.5, the PSO algorithm needs to handle situations where the particle might fly outside its boundaries for any given dimension. The selection of the boundary method for any of the parameters other than the Lagrange multipliers should not be a factor, but to ensure a feasible SVR solution for every particle at every epoch, the constraint conditions must hold, even during boundary condition violations.

Since the absorbing walls technique formulation will nullify the velocity for any one Lagrange multiplier, and in some cases may force a given Lagrange multiplier to be set to the boundary value, the zero summation constraint in Equation 5.5 cannot be guaranteed to be met. It is clear that this method will not work for the CMPSO framework as the goal is to have every particle at every epoch be a feasible SVR candidate.

The reflecting wall technique manipulates the velocity for any given Lagrange multiplier that falls outside the solution region by reversing the sign of the velocity for the parameter that has violated a bound. The use of this technique will also violate the summation constraint in

Equation 5.5 and will not be suitable as a PSO boundary condition violation remedy for the same reasons as stated for the absorbing walls technique.

The remaining technique, invisible walls, has unique features which adapt well to the requirement of having every particle at every epoch represent a feasibility solution. Recall for this technique that the particle is allowed to travel outside any given boundary without position or velocity adjustment. This means that the summation constraint in Equation 5.5 is always met regardless if the particle is in the Lagrange solution space boundaries or not. However, if any of the Lagrange multipliers violate the +/- 1 boundary, the second constraint in Equation 5.5 will be violated. It turns out this situation is manageable because pbest for a particle that has any of the PSO parameters falling outside their respective boundaries will not be calculated, and subsequently gbest will not be affected. An infeasible solution when a Lagrange multiplier is outside its bound for any given particle will never be considered as a candidate solution to the problem. The penalty paid for using this technique is the particle will not be able to contribute to a solution when its Lagrange multipliers (or other parameters) fall outside a boundary. However, the loss of computational efficiency can be offset by the fact that the fitness function will not need to be evaluated for a particle outside a boundary.

5.5 PSO Fitness Function, Iteration Bounds and Stagnation

As stated in Chapter 4, defining the fitness function for any given optimization problem can be the most difficult task in the formulation. A notional fitness function was given in Equation 4.2, which suggested that the "goodness" of the CMPSO process would be dependent on some type of comparison between a candidate training set and the CMPSO estimate. Different techniques were considered, including MOPSO, but as it turns out, the SVR formulation itself is sufficient as the CMPSO fitness function.

The SVR time series regression and estimation process is the underlying regression engine in CMPSO, along with the optimization of the SVR user defined parameters. There is no further need to develop a more complicated fitness function to evaluate the "goodness" of a fit to the training data as it is embedded in the SVR formulation itself. Therefore the duality gap value formulation in Equation 3.43 is used as the fitness function for the PSO optimization problem. Equation 4.2 is now restated below as Equation 5.11 with its associated parameter constraints in Equation 5.12.

$$\text{CMPSO Fitness Function:} \quad \text{Minimize } \gamma \tag{5.11}$$

$$\text{Subject to: } \begin{cases} 0 < \varepsilon \leq \varepsilon_{max} \\ 0 < C \leq C_{max} \\ 0 < \sigma \leq \sigma_{max} \\ -1 \leq \alpha_i \leq 1 \end{cases} \tag{5.12}$$

The formulation above assumes the SVR estimation function found in Equation 5.7, which includes the scale factor C with the Lagrange multiplier limits set to +/- 1. Note i is defined as the index to the N_S training samples $[1,\ldots,N_S]$.

The use of the duality gap for the PSO fitness function is an important feature of the CMPSO formulation. The gap value verifies if the SVR process is complete, which will essentially eliminate the use of a QP algorithm that normally would be required to complete an SVR solution for every particle over every epoch. This approach ensures a significant savings in computational time as will be shown in Chapter 6.

It is interesting to note that the bias parameter *b* in Equation 5.7 no longer appears in the formulation or in the constraints Equation 5.12 as it is not a necessary part of the dual objective function calculation, but will be necessary for the primal objective function calculation, specifically in the estimation of the slack variables. As it turns out, *b* can be calculated directly

and will not be necessary to include in the optimization parameter list. The calculation of b is discussed in the next section.

The PSO algorithm is an iterative process, as it tries to find a set of parameters that will satisfy the fitness function in Equation 5.11. In the cases observed in this research, it was found that a two stage approach for achieving a minimal fitness value for γ in Equation 5.11 can be obtained in a timely manner with limited computing resources. The first stage of CMPSO moves the particles through solution space as described above, but ends at a slightly higher γ value of 0.025. The diverse empirical data results shown in Chapter 5 indicates this is a fitness level bound sufficient enough to hold the ε, C, and σ PSO parameters at their current value. The second phase simply uses a QP algorithm to complete the Lagrange multiplier calculation to a fitness function value of 0.001 as required in the formulation. The QP algorithm used in CMPSO is a modified version of the SMO algorithm detailed in [21, 50]. In the (unlikely) event that the CMPSO iterative process never reaches the first stage fitness function threshold, the CMPSO processing will halt after a maximum number of iterations has passed.

As the PSO algorithm progresses, the particles will eventually tend towards the gbest value. It is possible that all of the particles may converge to a location that might not meet the fitness function criteria. It can be seen that the velocity update will tend to approach zero as the difference in its location relative to pbest and gbest will approach zero. When this situation happens, the swarm is said to "stagnate". This is the case where the swarm has settled on a "local minimum", which is defined as a place where the swarm thinks it has found a gbest location which approaches the fitness function criteria, but actually has not met the threshold. To handle these situations, it is necessary to reinitialize a subset of the particles. CMPSO will

reinitialize almost all of the particles (98%) every 50 epochs to ensure a stagnation condition does not persist.

It is important to note that the use of a two stage process including a QP algorithm and the occurrence of stagnation conditions are dependent on the computing platform's computational performance limits, both in memory and speed. If enough computing power is available, many more particles could be used in the swarm to find the optimal solution, decreasing the likelihood that stagnation will occur. Results shown in Chapter 6 indicate that these method work for the given applications and the available computational resources.

5.6 Time Series Data Scaling

As noted in Table 5.1, one of CMPSO's required attributes is its ability to adapt to different time series applications. This can be difficult as the dimensions of any given time series data, both in the independent and dependent variable, can vary application to application by a large amount. To manage this requirement, a data scaling factor is introduced to the CMPSO process.

Prior to the PSO optimization process, the input training time series data is scaled to fit a standard (x, y) frame depending on the application. If the goal is to use CMPSO to interpolate a function, the independent data values will be scaled to fit within a [0.0, 1.0] window. If the objective is to use CMPSO to extrapolate data, the independent data values will be scaled to fit within a [0.0, x_{max}] window. The value x_{max} is at least 0.9 in the cases presented in this research. In other cases where the required extrapolated value is very far from the last training point, the x_{max} value may need to be rescaled.

The dependent variable, regardless of an interpolation or extrapolation application, is scaled to [-1.0, 1.0]. The scaled data is then input to CMPSO and processed until the PSO fitness function is reached as described above. The final result is then scaled back to the original

dimensions of the problem at which time the time series estimate can be compared to the original data set for performance evaluation.

The other method that could be used is to set the PSO parameter limits to very large numbers that would encompass many possible applications. This would be inefficient as the solution space would be too large and the likelihood that a particle will converge to an optimal solution in a reasonable amount of time would be reduced. This scaling function is the means by which a general time series regression or estimation process can be made regardless of application for CMPSO for the available computational resources.

5.7 CMPSO Framework Summary and Parameter Settings

There are many CMPSO formulation aspects detailed in this chapter that make the combination of SVR and PSO work for a wide range of time series prediction and regression problems. This section summarizes the CMPSO framework and illustrates the details of the CMPSO algorithm and its specific parameter settings.

Figure 5.2 shows the general CMPSO process framework. Starting with Figure 5.3, each of the functional areas of the CMPSO framework is defined in detail. First is the input time series scaling. Figure 5.4 illustrates particle initialization process. Figure 5.5 illustrates the fitness function evaluation. Figure 5.6 shows the particle motion update function and Figure 5.7 shows the particle parameter re-initialization function. Finally Figure 5.8 shows the last optimization steps and evaluation process.

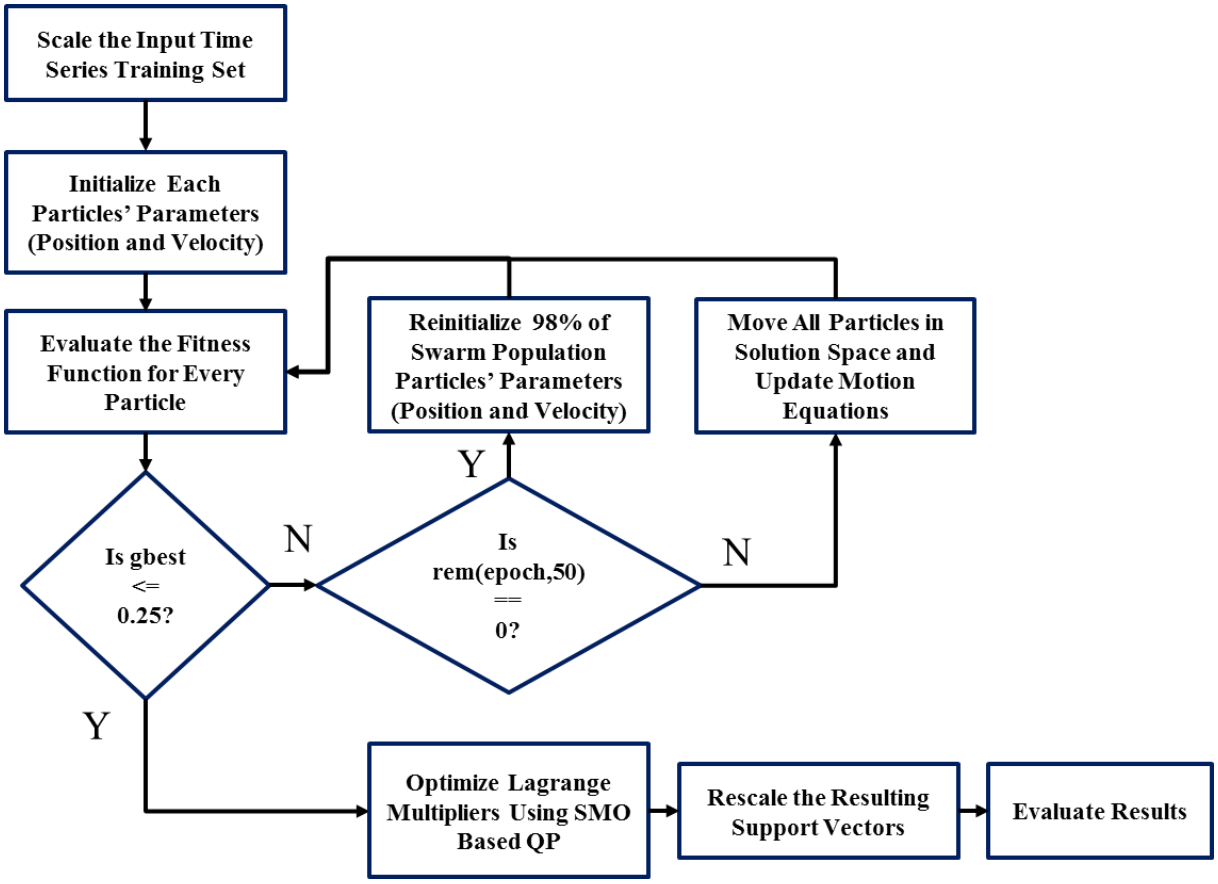

Figure 5.2: CMPSO Process Framework

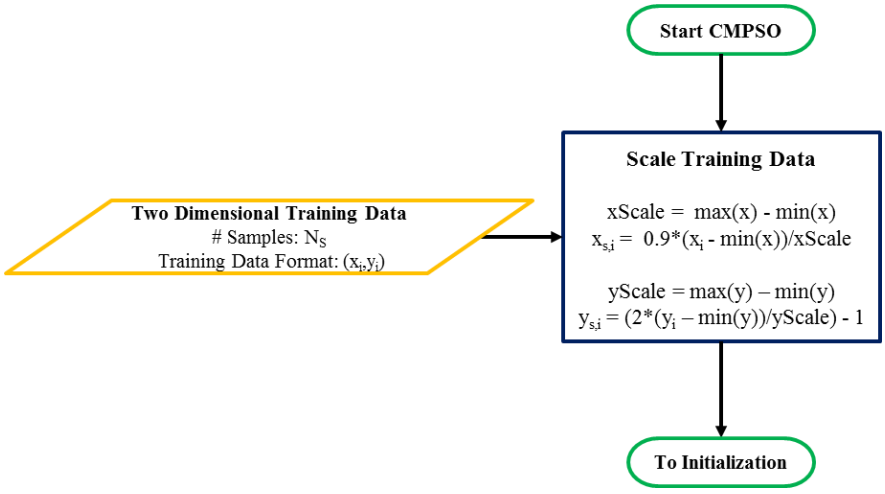

Figure 5.3: CMPSO Input Data Scaling

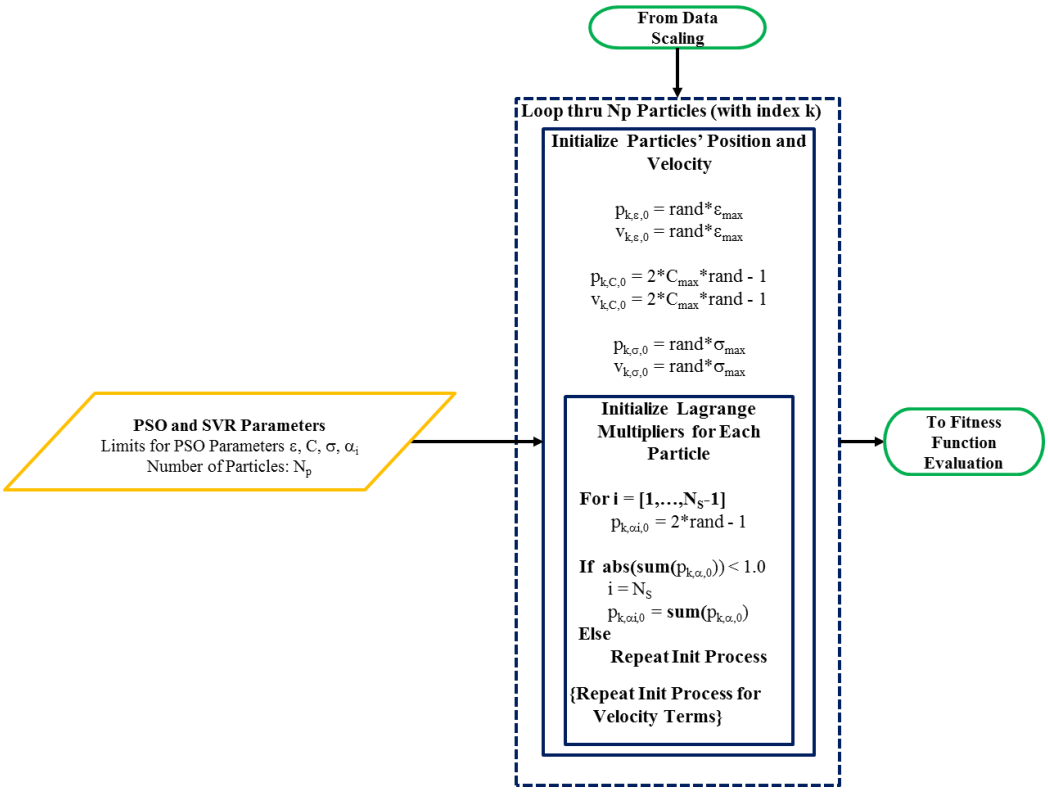

Figure 5.4: CMPSO Particle Parameter Initialization

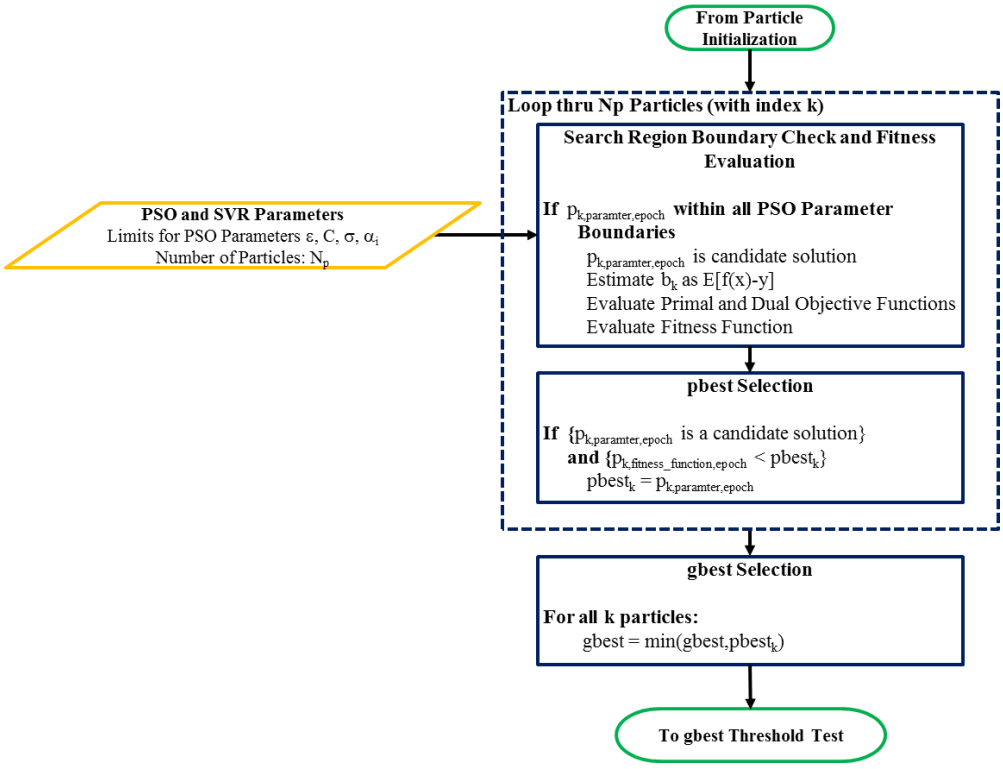

Figure 5.5: CMPSO Particle Fitness Evaluation

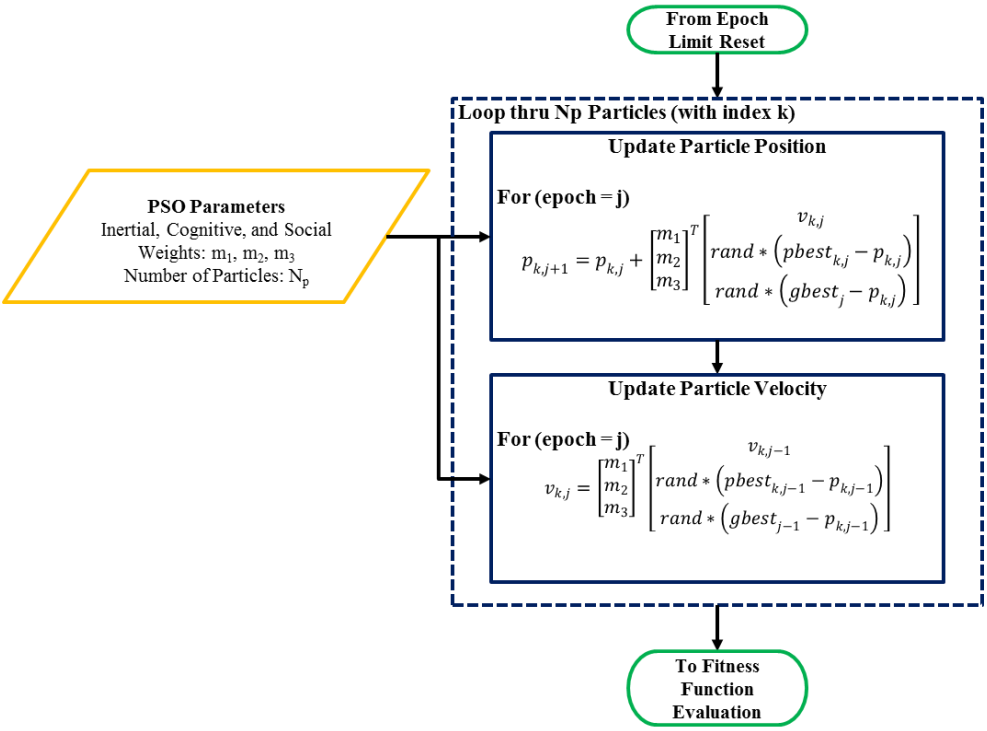

Figure 5.6: CMPSO Particle Motion Update

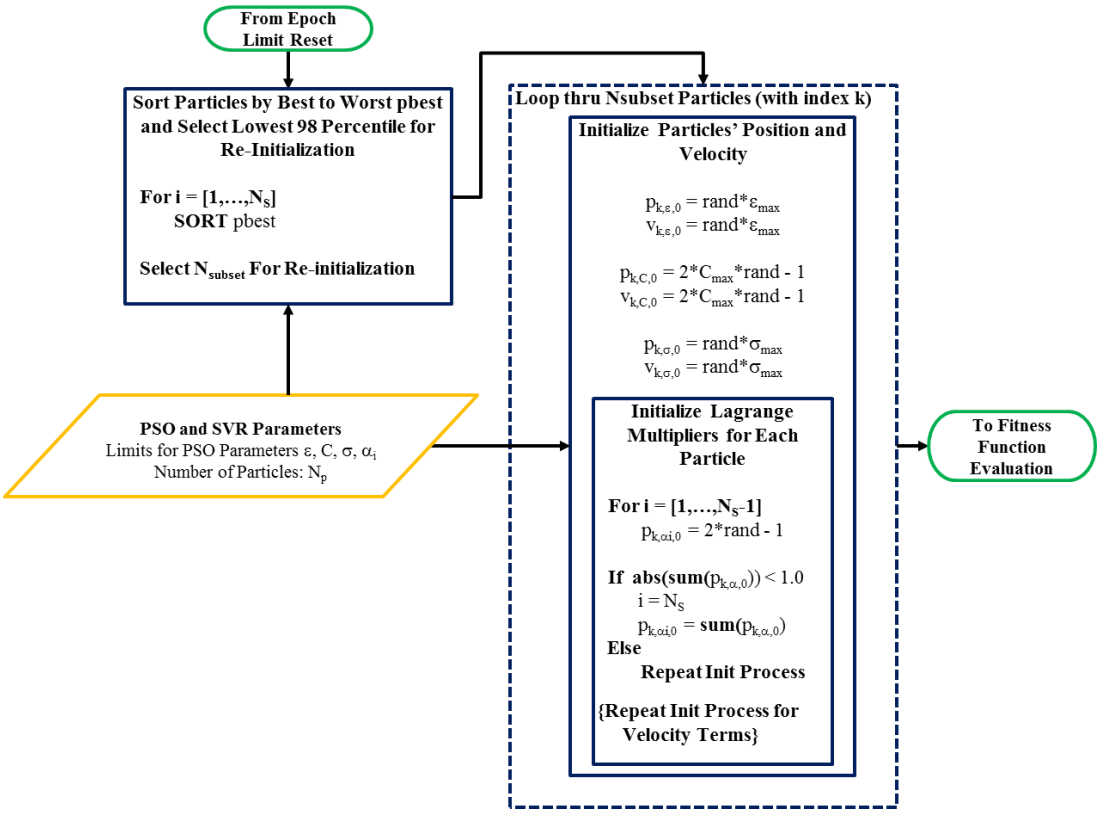

Figure 5.7: CMPSO Particle Re-Initialization

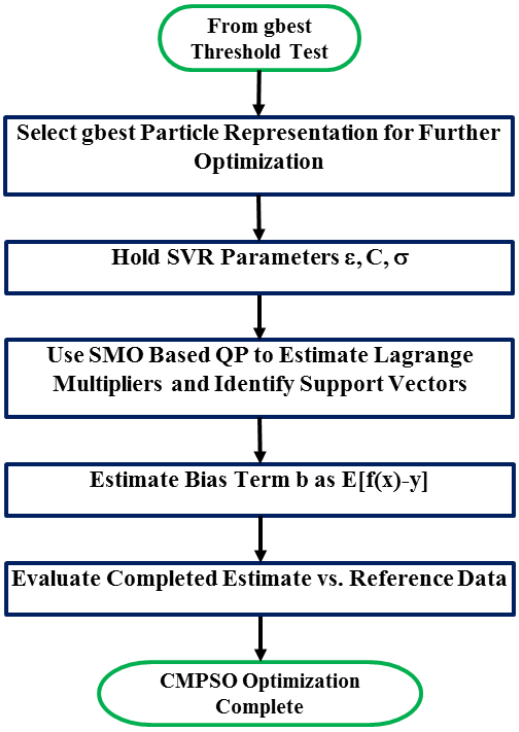

Figure 5.8: CMPSO Final Processing

Figures 5.3 through 5.8 represent the entire CMPSO algorithm. There are many parameters and limits associated with the actual implementation of CMPSO. Table 5.2 details these values.

We have now completely defined the CMPSO algorithm, including functional processes as well as all the relevant parameters associated with the algorithm. The following chapter will illustrate CMPSO's ability to estimate arbitrary time series functions.

Table 5.2: CMPSO Parameter List

CMPSO Parameter	Details	Notes
Number of Particles	CMPSO uses 500 particles regardless of application.	Selected based on computational hardware used for this research (see Chapter 5).
Particle Velocity Limit	During particle velocity and update cycles, the velocity is limited to 10% of the range of bounds for any given parameter.	This limit essentially slows the particles motion through the solution space and helps keep a significant number of particles in the solution space boundaries.
ε Max and Min Values	0.10 and 0.02, respectively.	Determined empirically based on data examples shown in Chapter 5.
C Max and Min Values	2.0 and 0.001, respectively.	Determined empirically based on data examples shown in Chapter 5.
σ Max and Min Values	1.0e-3 and 1.0e-5, respectively.	Determined empirically based on data examples shown in Chapter 5.
Particle Reset Limits	98% of the lowest scoring pbest particles are reset every 50 epochs.	This prevents swarm stagnation.
Fitness Function Thresholds	0.25 and 0.001.	The higher value is used to halt the PSO search process for ε, C, and σ. The second value is used to complete one QP pass on the gbest parameters to complete the CMPSO process.
Epoch Limit	Maximum number of epochs to execute is 20000.	In case the PSO fitness function does not reach a feasible solution, the CMPSO process will halt after the epoch limit is reached.

CHAPTER 6: PERFORMANCE METRICS AND RESULTS

6.1 Statistical Performance Metrics

We have now defined CMPSO and have shown the implementation details for using SVR as the time series regression and estimation engine along with PSO to assist in optimizing user defined variables as well as the standard SVR variables. To illustrate CMPSO performance against any given time series regression or estimation problem, a set of time series statistical comparison metrics needs to be defined.

Recall from Section 1.1 the definition of "goodness" for a time series estimate. First, it must be consistent, meaning the expected value of the estimate as compared to a reference time series must approach zero. Equation 6.1 illustrates the bias estimation equation.

$$\text{Error Mean: } \frac{1}{N_S}\sum_{i=1}^{N_S}(y_i - f(x_i)) \tag{6.1}$$

where N_S is our standard definition of the number of time series samples, y_i is a sample of the time series we are trying to estimate, sometimes referred to as reference or "truth" data, and $f(x_i)$ is the generated estimate.

Equation 6.1 is referred to as the error mean. Related to this metric is the mean absolute error defined in Equation 6.2.

$$\text{Mean Absolute Error: } \frac{1}{N_S}\sum_{i=1}^{N_S}|y_i - f(x_i)| \tag{6.2}$$

Both Equations 6.1 and 6.2 are standard bias measures that are used to evaluate a goodness of fit.

The second measure of estimation performance is the consistency of the estimate. Consistency is defined as the variance of the estimate relative to the closeness of fit to the example data and is defined in Equation 6.3.

$$\text{Estimation Consistency: } \lim_{N_S \to \infty} E[(\boldsymbol{y} - f(\boldsymbol{x}))^2] = 0 \tag{6.3}$$

where N_S is the number of samples in the data set and the E[.] is the expected value operator represented in Equation 6.6.

The most common measure of consistency is the Normalized Mean Square Error (NMSE) as defined in Equations 6.4, 6.5 and 6.6.

$$\text{NMSE: } \frac{1}{\sigma^2 N_S} \sum_{i=1}^{N_S} (y_i - f(x_i))^2 \tag{6.4}$$

$$\sigma^2 = \frac{1}{N_S - 1} \sum_{i=1}^{N_S} (y_i - E[\boldsymbol{y}])^2 \tag{6.5}$$

$$E[\boldsymbol{y}] = \frac{1}{N_S} \sum_{i=1}^{N_S} y_i \tag{6.6}$$

Equation 6.5 is often referred to as the sample standard deviation of a time series. Note the σ value in this equation is not related to the σ value in the SVR formulation.

Figure 6.1 is an illustration of the difference between biased/unbiased and consistent/inconsistent examples of data sets. Consider a case where a single two dimensional data point is trying to be estimated. The blue X in Figure 6.1 illustrates the target. The red dots mark the given estimation process attempts to target the location X. The four separate panels in the figure represent the four qualitative possibilities of bias and consistency measures.

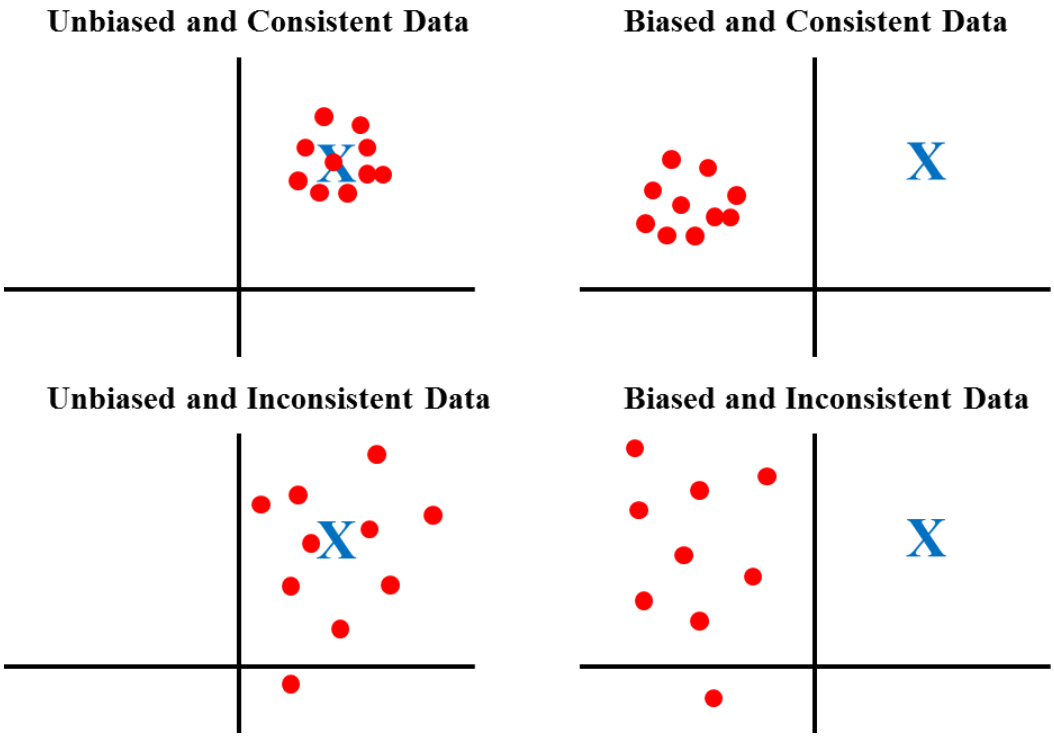

Figure 6.1: Illustration of Estimator Bias and Consistency

It is clear from Figure 6.1 that a good estimator will hit close to the target repeatedly, making it unbiased and consistent.

For the data series examined in this chapter, there are other measures of time series prediction effectiveness that need to be defined. First is the Mean Square Error (MSE) and subsequently the Root Mean Square Error (RMSE) defined in Equations 6.7 and 6.8, respectively.

$$\text{MSE: } \frac{1}{N_S}\sum_{i=1}^{N_S}(y_i - f(x_i))^2 \tag{6.7}$$

$$\text{RMSE: } \sqrt{\frac{1}{N_S}\sum_{i=1}^{N_S}(y_i - f(x_i))^2} \tag{6.8}$$

As seen in these two equations, both of these measures are focused on the consistency of the estimating function.

The next two measures are the Maximum Average Percent Error (MAPE) and Maximum Error (MAXE) shown in Equations 6.9 and 6.10, respectively.

$$\text{MAPE: } \frac{100}{N_S}\sum_{i=1}^{N_S}\left|\frac{y_i - f(x_i)}{y_i}\right| \tag{6.9}$$

$$\text{MAXE: } max(|y_i - f(x_i)|)_{i=1,\ldots,N_S} \tag{6.10}$$

These measures of effectiveness give an indication of bias. It should be noted that the MAPE value is scaled to 100 %.

We have now defined a set of measures of effectiveness for time series regression and estimation. A good estimator will tend to have close to zero values for all of the above represented measures of effectiveness.

6.2 Computing Environment

To evaluate CMPSO against selected time series applications, a computing environment was set up to implement CMPSO as well as measure the performance per the metrics outlined in Section 6.1. For this research, a single Windows based Personal Computer (PC) was used as the development and test platform. As noted earlier, computational performance is a function of PC hardware and many CMPSO parameter settings mentioned in Chapter 5 could be changed if more (or less) advanced computing hardware and software is available.

The computer platform used for this research is a 64 bit Windows 7 Ultimate based PC running an Intel Core i7 Quad Central Processing Unit (CPU) (model Intel Core i7 950) with a variable clock between 2.8 and 3.2 GHz. The PC also has a 1.0 terabyte (TB) 7200rpm Serial

Advanced Technology Attachment (SATA) III hard drive and 12 gigabytes (GB) of Double Data Rate (DDR) 3 Random Access Memory (RAM).

The CMPSO algorithm software development was done using a 64 bit version of MATLAB version R2010b (Version 7.11.0.584, win64). It should be noted that when CMPSO was executed on the PC, the MATLAB process affinity was set to all cores of the CPU and the MATLAB process priority was set to "high". In addition, some of the CMPSO functions were written in the C programming language and converted to "MATLAB Executable" or "MEX" files to decrease the overall processing time. They are generally implemented when large, repetitive processes are required in the computations. MEX files are essentially dynamically linked routines that MATLAB can call externally [132]. They have the ability to access (read from and write to) the MATLAB workspace where all of the computed data is stored for processing.

The C programs were developed and compiled using Microsoft Visual C++ 2010 Express. The compilation process is part of the MATLAB MEX file development process, but the C compiler is called externally. Figure 6.2 illustrates the software architecture for the CMPSO algorithm implementation.

6.3 Computational Efficiency

As stated in Section 5.5, the use of the duality gap as the PSO fitness function eliminates the need for a QP solver for every particle at every epoch. To prove this, a simple experiment is used to illustrate the computational efficiency gains of CMPSO vs. a more conventional, or "separated" PSO and SVR implementation. In the separated implementation, each particle's Lagrange multipliers are compute to reach a duality gap limit of 0.001 for every epoch.

Equation 6.11 represents an arbitrary exponentially weighted sinusoidal function with additive Gaussian noise that is used to measure CMPSO computational efficiency.

$$y = \sin(0.2\pi x)\, e^{0.1x} + 0.5N(0,1) \tag{6.11}$$

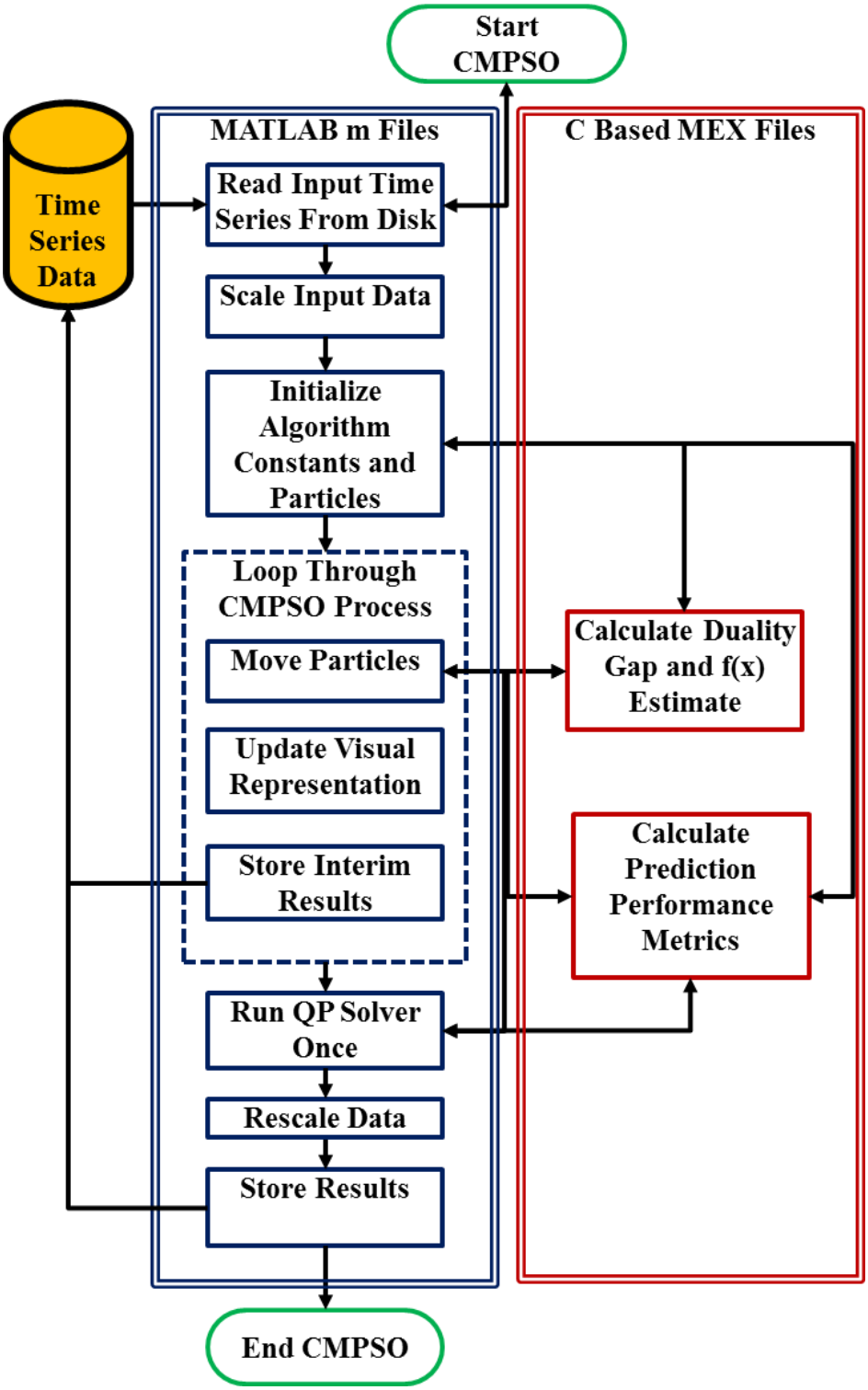

Figure 6.2: CMPSO Software Architecture

A 41 point data set was created based on Equation 6.11 and is shown in Figure 6.3.

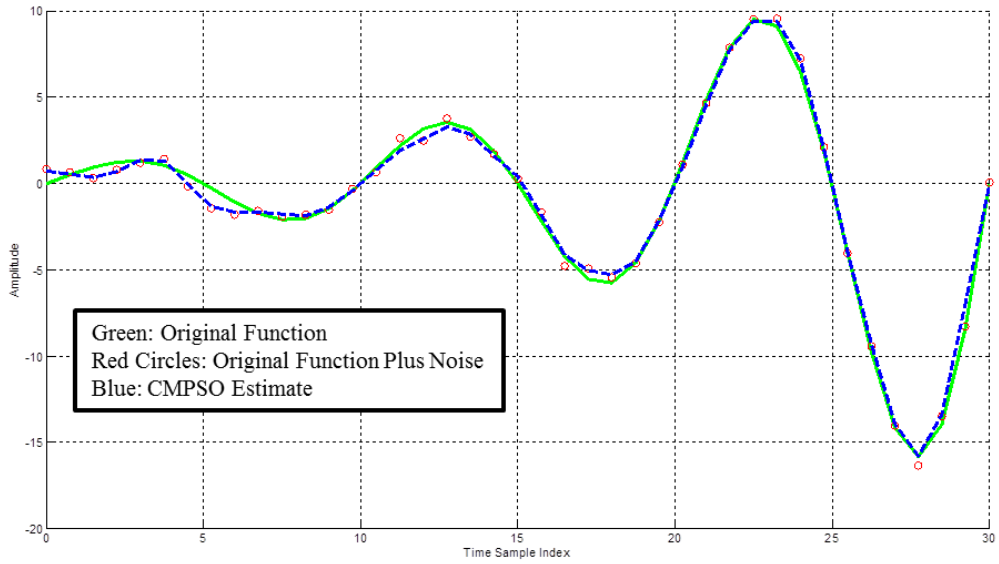

Figure 6.3: Example Time Series for Computational Efficiency Analysis

Note that the sample spacing between points in Figure 6.3 is 0.75.

For this experiment, a parallel PSO and SVR implementation was created to evaluate the effect of eliminating the QP process from the CMPSO iteration cycle. Figure 6.4 shows the block diagram of the two approaches considered.

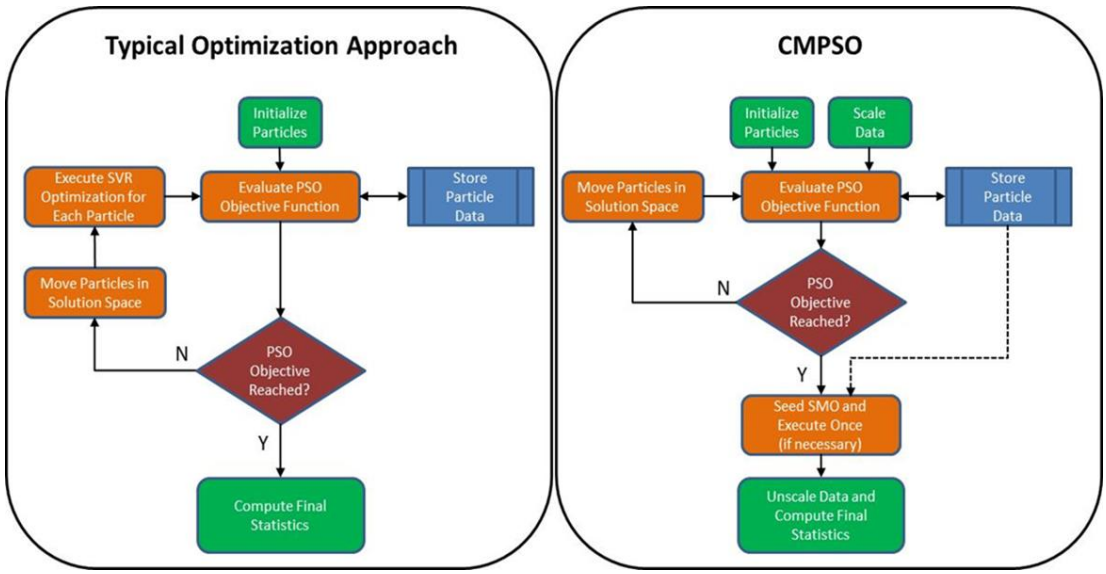

Figure 6.4: Comparison of CMPSO Architecture vs. Typical Optimization Setup

As seen in Figure 6.4, a typical optimization routine would require the use of a QP to optimize the SVR problem for every particle and for every step (epoch) in the process. Alternatively, the CMPSO process forces each potential SVR solution candidate to be feasible and will continue until the duality gap is reached, eliminating the need for executing a QP solver for every particle at every epoch.

To show the computational benefit CMPSO has vs. the other typical optimization approach, the data set shown in Figure 6.3 was run 300 times for CMPSO and for a combined PSO/SVR implementation. Table 6.1 shows the difference between the two approaches for comparison.

Table 6.1: CMPSO vs. Alternative Optimization Implementation

Function or Parameter	**CMPSO**	**Alternative**
Number of Particles	CMPSO uses 500 particles regardless of application.	Maximum of 25.
SVR Optimization Technique	Uses PSO and at most one pass of an SMO based QP algorithm.	Uses same SMO based QP algorithm for every particle at every epoch.
SVR Parameter Optimization	PSO based.	PSO based to only determine ε, C, and σ.
Algorithm Stopping Criteria	CMPSO halts when fitness function (duality gap) γ is less than or equal to 0.001.	Same as CMPSO.

The alternative algorithm is only replacing the PSO part of CMPSO. The data is still scaled and rescaled at the end of the process. This is necessary to ensure the solution space for both approaches is the same. It is also important to note for this experiment that the exact same code elements shown in Figure 6.2 used for CMPSO are used for the alternative method. This is necessary to have a meaningful comparison between the two approaches.

The 41 point time series defined in Equation 6.11 was generated 300 times, with different values of noise added for each individual set. Each of the 300 time series sets were input to both CMPSO and the alternative (PSO/SVR separated) optimization approach. Note the two algorithms were not run concurrently. A set of timing metrics was recorded for each of the 300 runs for CMPSO and the alternative approach. At the time either of the algorithms starts, a time stamp is recorded in MATLAB. When the optimization routine is complete, another time stamp is recorded and the difference between the two is calculated and stored. The mean value of the time to execute (time difference) for each algorithm over the 300 data sets was then computed.

In addition to the timing performance of each algorithm, other performance metrics were calculated. First is the percentage of data sets where either of the two approaches converged to the duality gap limit. The second is the NMSE estimation metric described in Section 6.1. Table 6.2 shows the performance results of the two approaches.

Table 6.2: CMPSO Performance vs. Alternative Optimization Implementation

Measured Parameter	CMPSO	PSO + SVR	Notes
Mean Algorithm Execution Time (sec)	7.27	39.08	PSO + SVR used minimum of 25 particles
Percentage of Runs that Converged	99.0%	0.0	PSO + SVR never converged to dual gap limit
NMSE	Typically less than 0.005	{N/A}	Since PSO + SVR never converged, NMSE was not calculated

The experimental results show that 99% of the time CMPSO converged to a solution in approximately seven seconds, typically with a very small NMSE of less than 0.005. By contrast, running PSO and SVR separately, the solution never converged with the minimum number of

particles (25) and it took more than five times longer for the PSO+SVR method to settle on incorrect results. This means that the swarm in that approach essentially stagnated to a suboptimal solution. If more particles are used in the alternative method to try and improve the results, the processing time would only increase. This simple experiment shows that not only is CMPSO a viable algorithm for approximating time series data, it is also more computationally efficient than alternatives that use SVR and PSO separately.

6.4 Arbitrary Function Analysis

The CMPSO formulation is adaptable to many different kinds of data series. The first example shown above is approximating an exponentially weighted sinusoidal function. This section illustrates the utility of CMPSO for other several other arbitrary time series functions.

6.4.1 SINC Function

The SINC function is a common mathematical function and is defined in Equation 6.12.

$$y = \frac{\sin(x)}{x} \tag{6.12}$$

CMPSO was run against this function for 101 data samples. Figure 6.5 shows the results for this example. The CMPSO optimized the fit for this curve in 93 epochs. Note the blue curve is the estimate after the first phase of CMPSO is complete where the γ value is just under 0.25. The blue curve is a close approximation of the curve, but this approximation is clearly good enough to stop iterating over the ε, C, and σ values as seen by the final result shown by the green curve. The final statistical measures between the original data set and the CMPSO approximation show an error mean of -0.015, an NMSE of 0.00026, an RMSE of 0.0053 and a MAXE of 0.0065. All of these measures show excellent matching between the original data and the CMPSO approximation.

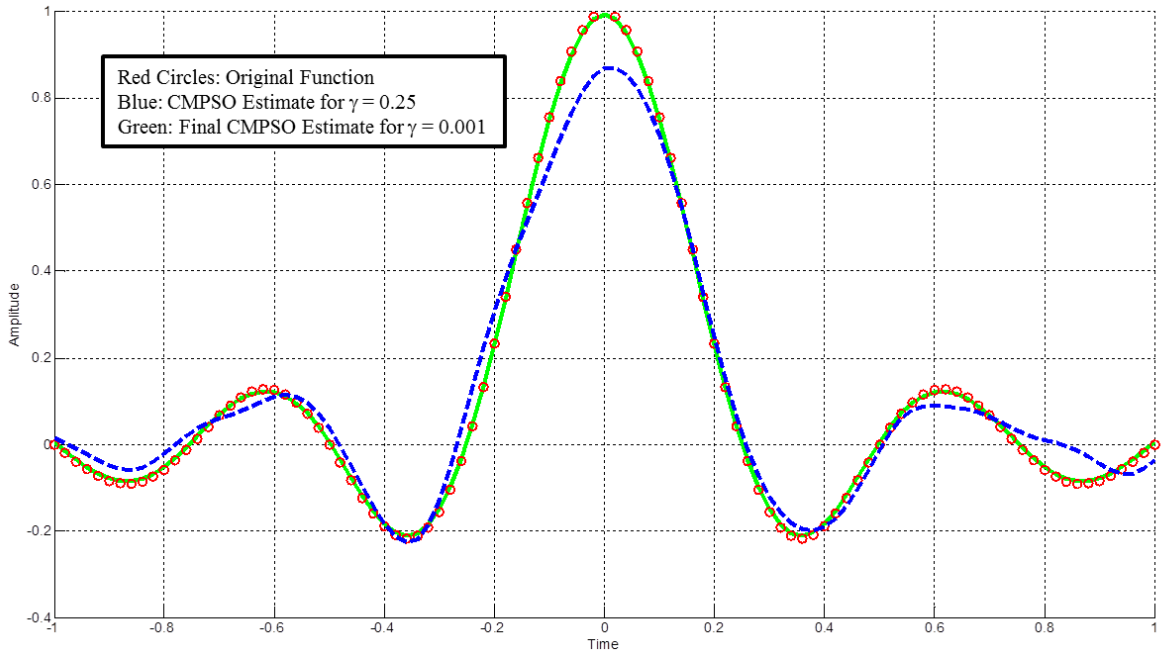

Figure 6.5: CMPSO Approximation of SINC Function

6.4.2 SINC Function with Missing Data

Another CMPSO utility is the ability of the algorithm to interpolate functions where data is missing over certain regions. Using the same SINC function defined in Equation 6.12, we now recreate the same 101 point data sample, but omit the data points where $-0.35 < x < -0.25$. This represents about 5% of the data missing. Figure 6.6 shows the CMPSO performance. Although there was not a significant amount of data missing, the error statistics show that the approximation over the whole curve was still good. The error mean is -0.0017, the NMSE and RMSE are 0.00026 and 0.0054, respectively, and the MAXE 0.0065. These values are very similar to the previous example. The data presented here is for illustration purposes showing the utility of CMPSO. A more challenging data set with missing data is illustrated in Section 5.4.3.

6.4.3 S&P 500 Data

The last three data examples showed CMPSO performance against arbitrary sinusoidal and exponential based functions. CMPSO performed well for the regression and data

interpolation applications including the estimation of missing data. We now examine CMPSO performance against a real world application.

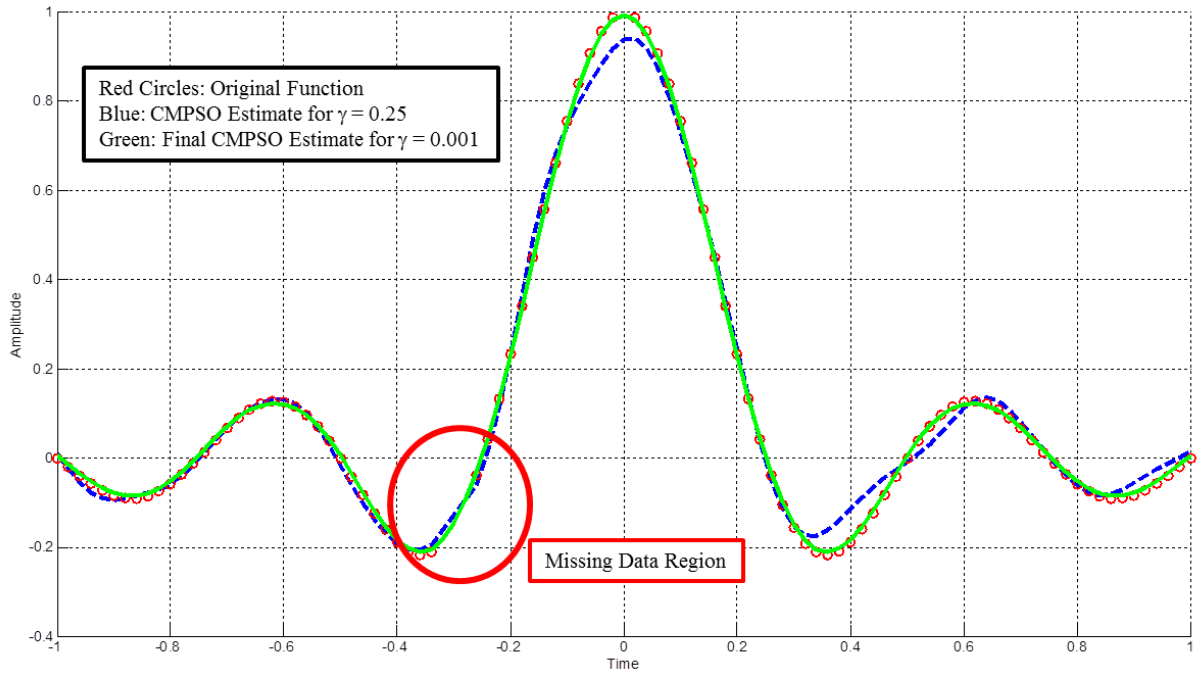

Figure 6.6: CMPSO Approximation of SINC Function with Missing Data

The next data set is an example of a common stock market index: the S&P 500 Index. The S&P 500 is a stock market index based on the performance of 500 large companies. We selected 75 samples of S&P 500 daily closing price from mid-October 2014 through the end of January 2015 for analysis, as this data is publically available for download [133]. This is very challenging data set to estimate as there are many variations and trends in the data as shown in Figure 6.7, thus the reason these data samples were chosen for analysis. The statistical results for this analysis show an error mean of -0.2192, NMSE and RMSE values of 0.0247 and 8.3060 respectively, and a MAXE of -25.0894. We will examine other real world data examples in Section 6.4.2.

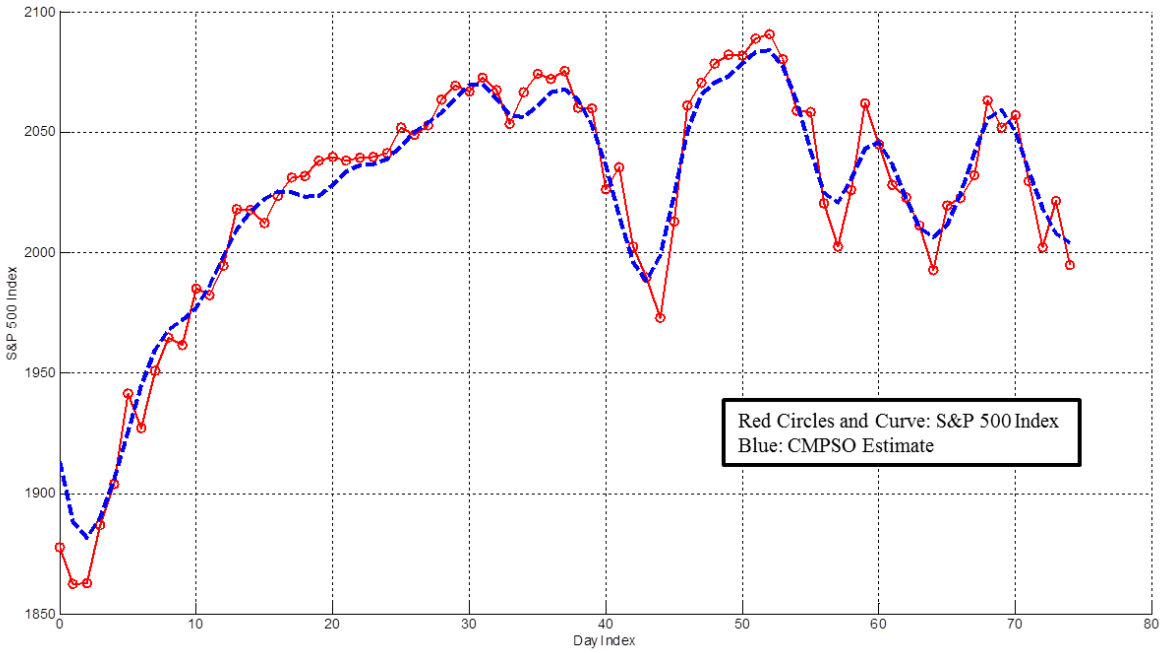

Figure 6.7: CMPSO Approximation of S&P 500 Data

6.5 Benchmark and Competition Data Performance

The previous examples in Sections 6.2 and 6.3 showed several examples of CMPSO performance against arbitrary functions as well as one real world application. The results showed that CMPSO performed well against relatively easy and somewhat more complex data sets with many trends and variations. We now explore CMPSO performance further by estimating standard benchmark data sets that are commonly used to evaluate time series prediction methods.

6.5.1 Mackey-Glass Benchmark Data

As seen in [1], Mackey-Glass benchmark data is a common artificial (synthesized) time series benchmark by which time series prediction algorithms are evaluated, including SVR. Mackey-Glass data is a nonlinear time delay differential equation which has both periodic and chaotic dynamics [61]. The Mackey-Glass equation is represented in Equation 6.13.

$$\frac{dx(t)}{dt} = -\mathrm{b}x(t) + \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} \tag{6.13}$$

where x(t) is the time series representation, t is time, τ is a delay factor and a and b are real valued scale factors. There is another factor, Δt, not expressed in Equation 6.13 but it is typically used for generating the samples of time series data. As published, the standard values for these parameters are a = 0.2, b = 0.1, τ = 17, x(0) = 1.2 (initial condition), and Δt = 0.1. In order to generate the time series data, one must solve the differential equation. Figure 6.8 is an example output of the function for the first 1000 seconds.

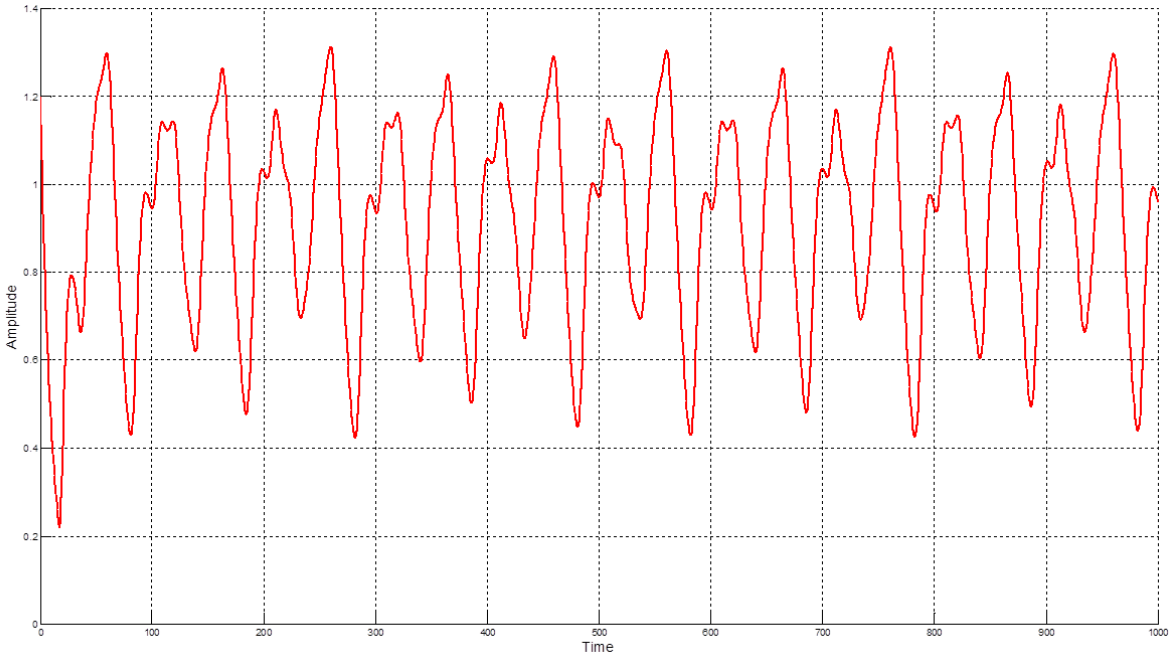

Figure 6.8: Mackey-Glass Data Example (First 1000 Seconds)

CMPSO performance has been compared to other published time series prediction techniques [108] and has performed well. For comparison, there are two specific experiments using this data that are presented in this research.

1. Regression: The first 2000 data points of the Mackey-Glass data, sampled every 15 samples (Δt = 0.1) are used to evaluate CMPSO regression capabilities.

2. Time series prediction: Starting with the first 2000 data samples, sampled every 15 samples ($\Delta t = 0.1$), the CMPSO was trained and the following 1 to 30 sample prediction horizons were estimated and compared to truth data. This process was repeated 50 times, moving the starting sample training sample point by 100. The resulting ensemble of prediction horizon data is then measured to evaluate CMPSO accuracy (bias) and consistency.

Figure 6.9 illustrates the 2000 point Mackey-Glass data set used for evaluation.

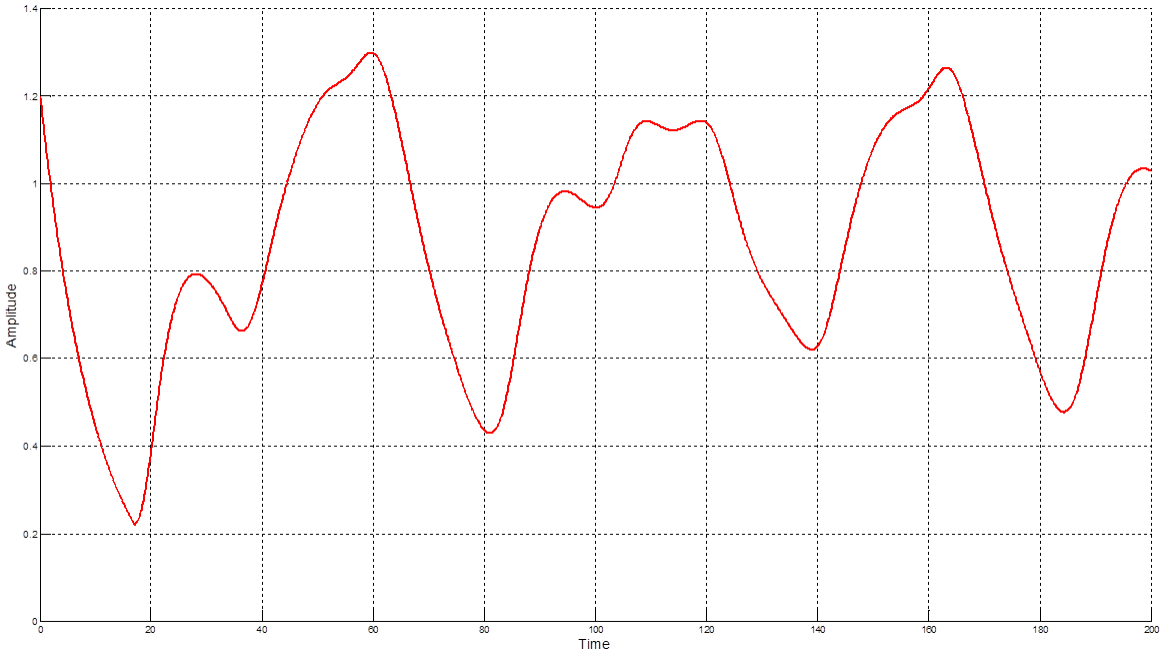

Figure 6.9: Mackey-Glass Data Example (First 200 Seconds Used for Evaluation)

CMPSO comparison is against other SVR/SVM related techniques also using the Mackey-Glass benchmark data for evaluation. The first comparison is to a (Regularized) Orthogonal Least Squares algorithm that is SVM based [28]. This technique uses a least squares method to reduce the number of training values similar to the strategy used for CMPSO in this example. The performance stated in this reference is only for prediction for a +1, +5, and +10 sample prediction horizon, where each sample has a Δt of 0.1.

The second is an SVR based approach using a radial basis function similar to CMPSO [25]. In this example, the data is interpolated and there is no prediction application associated with the publication. The third approach is also SVM based and again uses an SVR based approach like CMPSO [34]. This SVR based application only uses the Mackey-Glass data to evaluate performance for a one-step-ahead prediction horizon (+1 data sample). The fourth CMPSO comparison is made against a prediction model called Kernel Dynamical Modeling which is partially based on the use of kernel functions, similar to the SVR method used in CMPSO [88]. This particular application uses the Mackey-Glass data set to again evaluate performance for a one-step-ahead prediction horizon (+1 data sample).

Additionally, more recent approaches using SVR and Mackey-Glass benchmark data have been examined and compared. One method is called Accurate Online Support Vector Regression (AOSVR) [89]. This approach uses an SVR based algorithm, but also uses a technique that improves the training efficiency and also varies the SVR user defined variables. AOSVR also has a user defined parameter that bounds the regularization constant and it has to be set by the user *a priori*.

Another recent SVR method for Mackey-Glass estimation is the Multiple Output SVR (M-SVR) [90]. This technique uses multi-dimensional input/output training sets along with a quadratic loss function to estimate future values of the function. The algorithm uses five different models and uses PSO for model validation and selection. It should be noted that only the MAPE metric was the only similar metric to what was calculated and used in other publications.

The last current SVR based estimation is based on an iterative approach also using multiple support vector regression models [91]. It basically predicts one step ahead and feeds the

current predicted value (prediction horizon of +1) into another training set with one less sample. The framework shown in this reference uses multiple SVR models, one for each prediction horizon and then combines the results for a given prediction horizon. The results shown are for a prediction horizon of +20.

Figure 6.10 shows the CMPSO estimation (interpolation) of the Mackey-Glass data set.

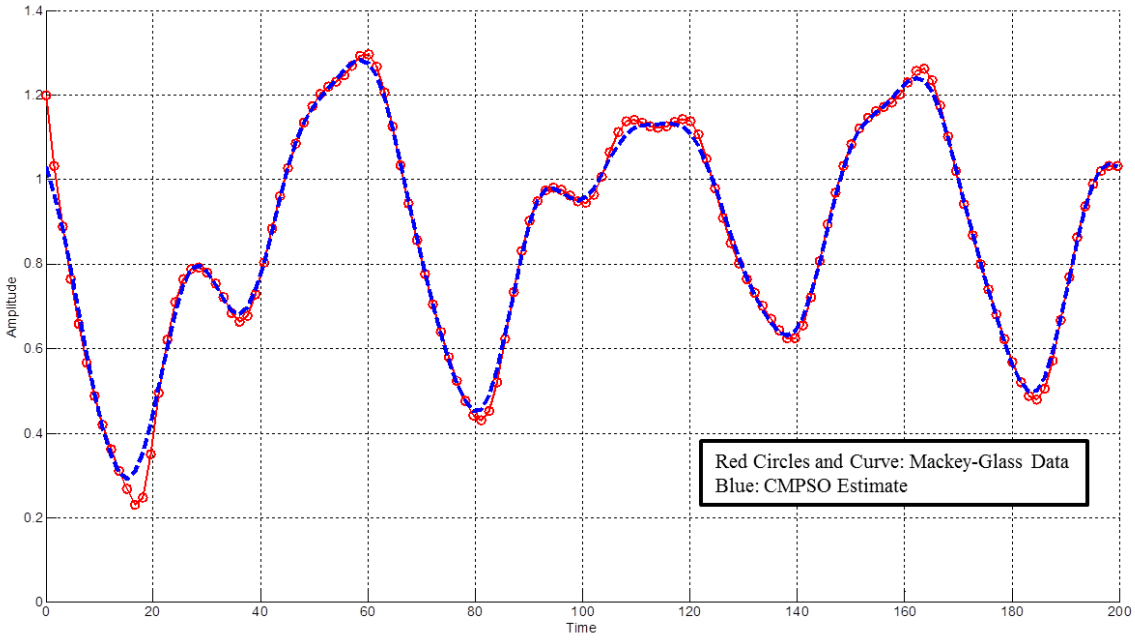

Figure 6.10: CMPSO Mackey-Glass Data Estimation

The performance metrics for interpolation show that the CMPSO estimation had an error mean of -0.00017, an NMSE and RMSE of 0.0081 and 0.0242, respectively and a MAXE of 0.1690.

To calculate statistics for the prediction problem, a set of 100 Monte Carlo runs were made, each with different random initialization points for CMPSO. The first 30 data points after the last Mackey-Glass data set point were estimated 100 times and collected for analysis (prediction horizon of [1,…,30]). The CMPSO prediction performance results are shown in Table 6.3.

Table 6.3: CMPSO Prediction Performance for Mackey-Glass Data

Prediction Horizon	Performance Metric		
	MSE	NMSE	RMSE
Interpolation	0.0006	0.0055	0.0199
+1 Sample	0.0006	0.0120	0.0254
+5 Samples	0.0014	0.0268	0.0379
+10 Samples	0.0032	0.0610	0.0569
+15 Samples	0.0062	0.1176	0.0787
+20 Samples	0.0104	0.1988	0.1018
+30 Samples	0.0214	0.4184	0.1461

One example run of the CMPSO estimate is shown in Figure 6.11 for the +1 through +30 samples prediction horizon.

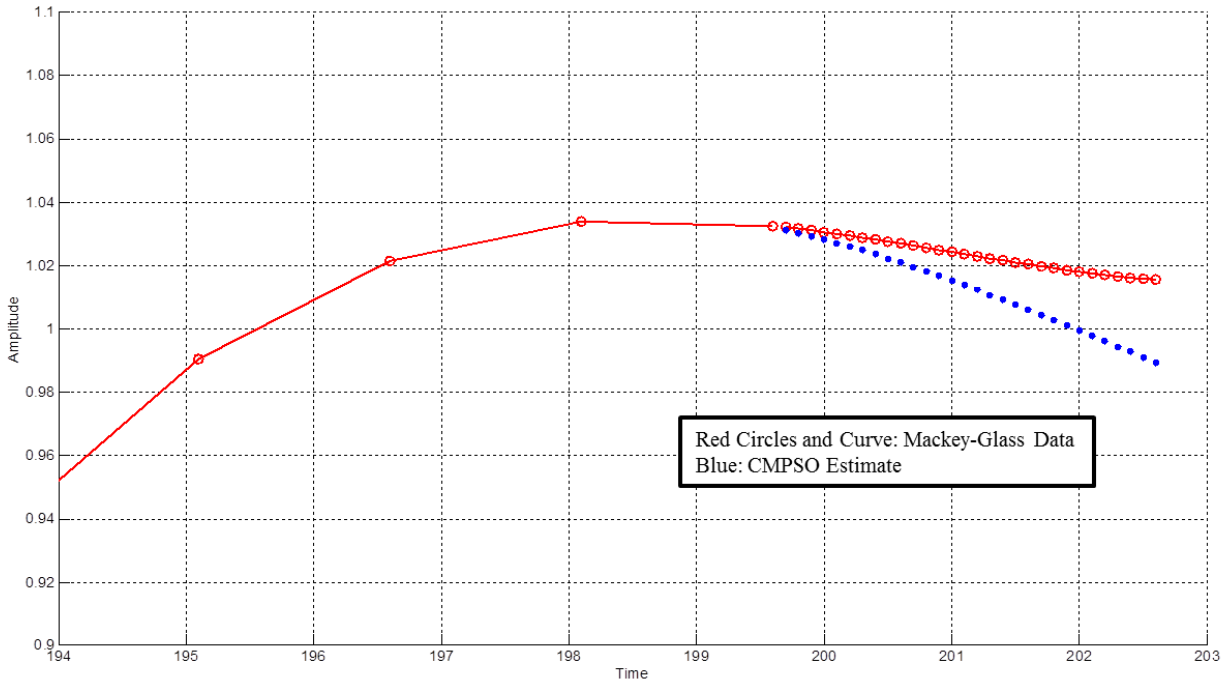

Figure 6.11: CMPSO Mackey-Glass Data Prediction

We now compare CMPSO prediction performance to the other five published time series regression and prediction methods listed in this section. It should be noted that the published

data used the MSE metric for comparison, or in one case MAPE. Table 6.4 is a summary of CMPSO performance vs. the other methods.

Table 6.4: CMPSO Prediction Performance (MSE) vs. Other Techniques

Technique	**Regression**	**Prediction Horizon (Samples)**			
		+1	**+5**	**+10**	**+20**
CMPSO (MAPE)	0.0006	0.0006 (2.57)	0.0014	0.0032	0.0104
ROLS [28]	N/A	0.1064	0.1696	0.3248	N/A
SVR [25]	0.0004	N/A	N/A	N/A	N/A
SVM+RT [34]	N/A	0.5158	N/A	N/A	N/A
KDM [88]	N/A	0.0066	N/A	N/A	N/A
AOSVR [89] (two results that depend on a user defined parameter setting)	N/A	0.0011 or 0.00005	N/A	N/A	N/A
M-SVR [90] (MAPE)	N/A	N/A (0.748)	N/A	N/A	N/A
Iterative M-SVR [91]	N/A	N/A	N/A	N/A	0.0065 +/- 0.0003

CMPSO performed well as compared to the published results using SVR based methods. In at least one recent case, user defined parameters had to be adjusted for the given algorithm to exceed CMPSO performance. More than half the cases CMPSO exceeded other relevant SVR based time series prediction techniques as published.

6.5.2 EUNITE Competition Data

SVR based solutions have been examined for many real world applications including Electrical Utility Load forecasting [134 to 148]. Coincidently, another common benchmark data set used to evaluate time series prediction and regression algorithms is based on real world electrical utility load data. In 2001, the European Network on Intelligent Technologies for Smart

Adaptive Systems (EUNITE; see ref [62]) initiated a competition [63] to predict maximum electrical loads for the East-Slovakia Power Distribution Company. The goal is to estimate maximum daily power loads for the month of January 1999 based on the daily values of 1997 and 1998 of electrical load and temperature. The measures of performance for this problem were both MAPE and MAXE (as defined in Section 6.1). It should be noted that the number of samples used for performance estimation is 31 which corresponds to the number of days in January.

Unlike the previous data sets analyzed in this chapter, this particular time series estimation problem has multiple inputs: time, load power, and daily temperature. Clearly a strategy will be needed to address all these factors and a way to implement CMPSO to estimate the January 1999 maximum power loads.

The first observation is the relationship between day of the week and the maximum power load. Figure 6.12 shows a plot of the mean of the maximum power load for 1997 (red curve) and 1998 (green curve) as a function of the day of the week.

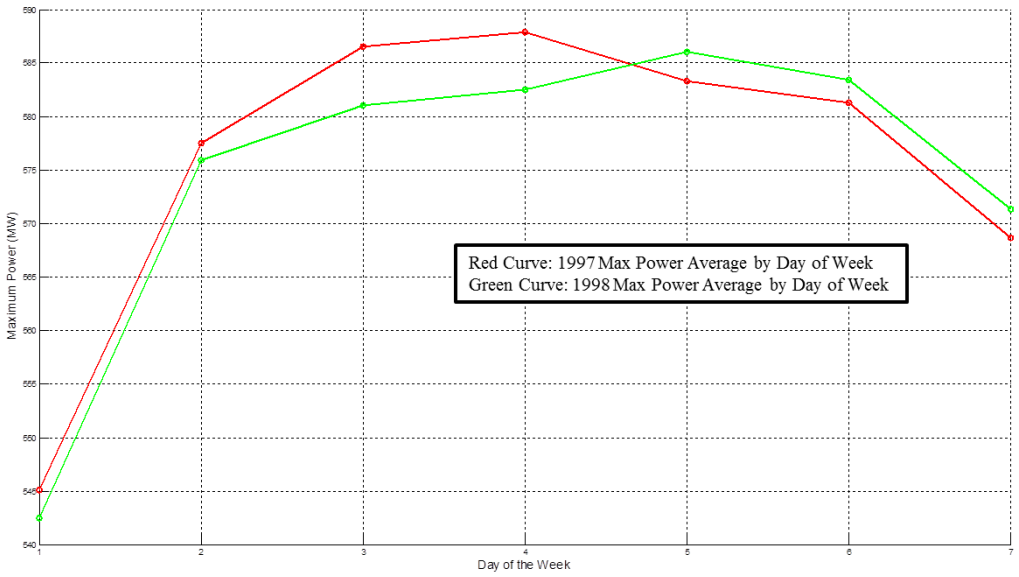

Figure 6.12: Mean of Maximum Power by Day of Week for 1997 and 1998

It is clear from Figure 6.12 that the weekend days (Saturday, Sunday; Day 7 and Day 1 in Figure 6.12) tend to draw less power than the rest of the working week and is apparently valid year to year. Figure 6.13 is a view of the January 1997 (red curve) and January 1998 (green curve) maximum power loads, time aligned such that the two data sets are aligned by day of the week. There are trends surrounding the weekends vs. the work week, as noted by the relative maximums of five days duration vs. the general minimums of two days duration that are clearly visible.

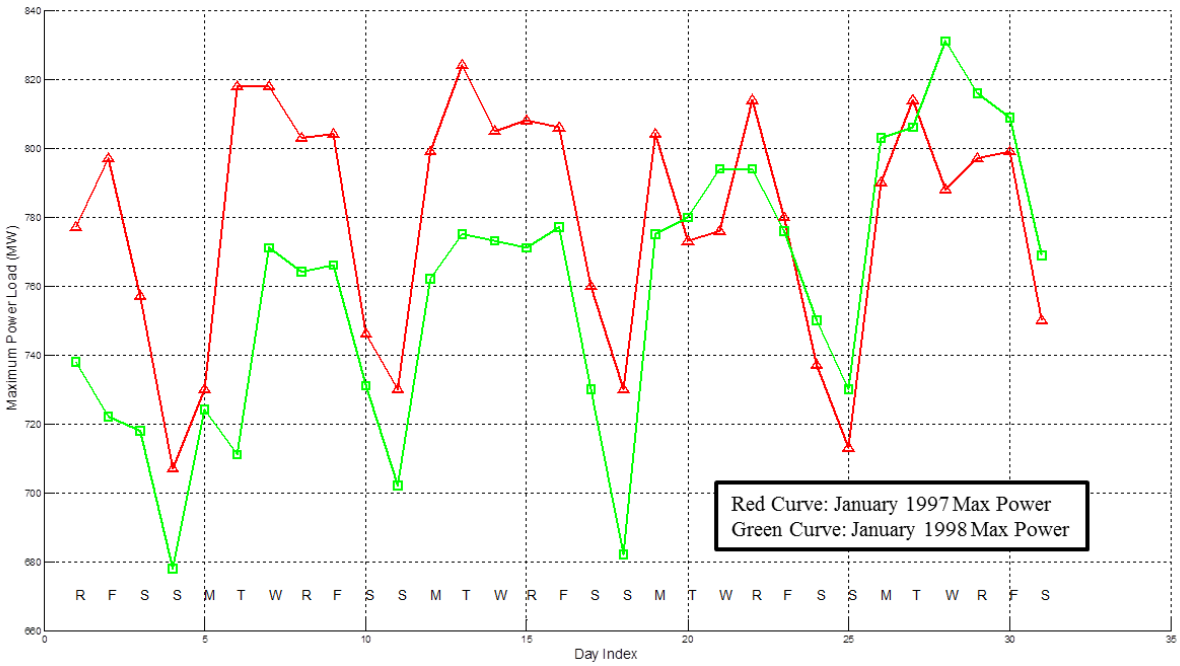

Figure 6.13: Maximum Power by Day of the Week for January 1997 and January 1998

There is also a relationship between temperature and maximum power consumption for any given day. This can be seen by looking at the day-of-the-week aligned difference in power and temperature between 1997 and 1998. Figure 6.14 illustrates 360 sample differences between 1997 and 1998, time aligned to the day of the week.

There appears to be an inverse relationship between temperature and maximum power as seen in Figure 6.14. This can be quantified by calculating a correlation coefficient between the two data sets as shown in Equation 6.14.

$$\rho_{x,y} = \frac{E[(x - u_x)(y - u_y)]}{\sigma_x \sigma_y} \tag{6.14}$$

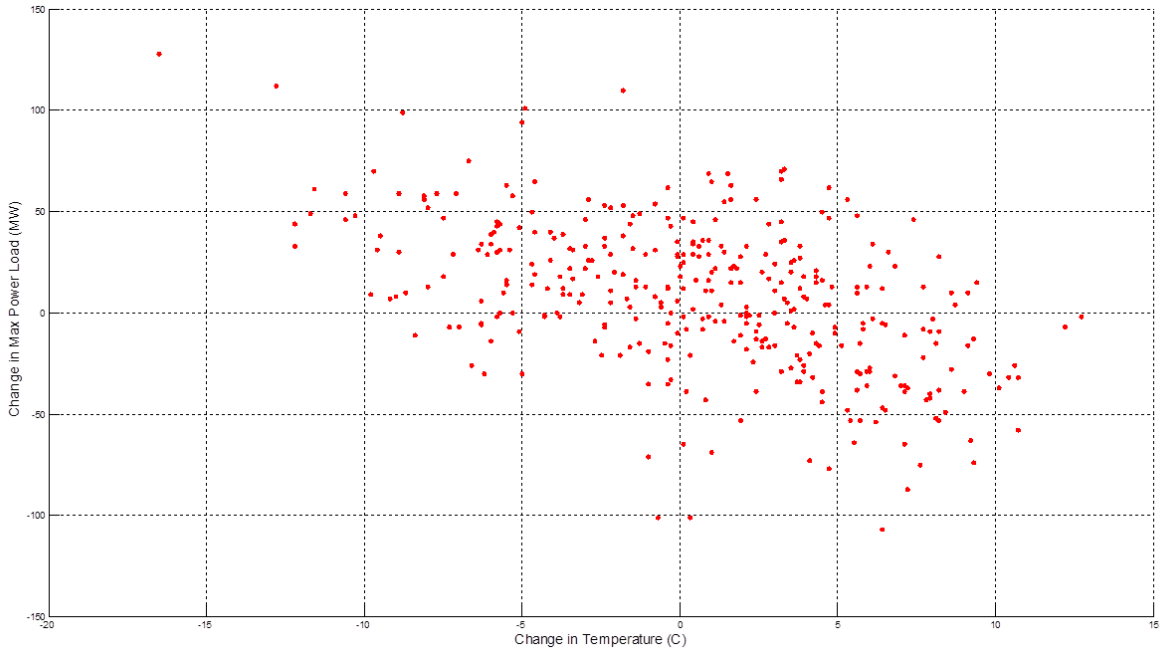

Figure 6.14: Scatter Plot of the Difference in Power and Difference in Temperature for 360 Days in 1997 and 1998 (Time Aligned by Day of the Week)

In this case, x and y are the change in temperature and change in maximum power load respectively. The μ and σ values are the means and standard deviations of the two data sets respectively. For the data sets shown in Figure 6.14, a correlation coefficient ρ was calculated (Equation 6.14) to be -0.5041 over the 360 data points. This clearly shows that there is an inverse relationship between maximum power and temperature.

Given this relationship, we would like to construct a simple linear approximation between the change in maximum power and the change in temperature shown in Figure 6.14. This approximation is defined in equation 6.15.

$$\Delta_P = -3.4668\Delta_T + 9.9573 \tag{6.15}$$

where Δ_P and Δ_T are the change in power and change in temperature respectively.

For this data set, we are going to formulate two different strategies to estimate the maximum power load for January 1999:

1. The mean value of the January 1997 and January 1998 data sets, time aligned to the day of the week, will be used to estimate a regression curve using CMPSO.
2. Assuming the temperature for January 1999 is known (which was provided as part of the EUNITE competition), we can now offset the amount of power (Δ_P) based on the difference in temperature between January 1998 and January 1999 and use the same CMPSO strategy listed in approach 1).

Table 6.5 shows the CMPSO results for the EUNITE competition, with and without the linear approximation to the change in temperature and power relationship.

Table 6.5: CMPSO EUNITE Prediction Performance

Measured Parameter	**CMPSO**	**CMPSO with Linear Offset**	**Notes**
MAPE	3.176	2.628	% Error
MAXE	84.931	74.663	Max Error in MW

As shown in Table 6.5, there is significant improvement by using the temperature data to offset the values. Figure 6.15 shows the final results of this experiment. As can be seen in

Figure 6.15, CMPSO estimates for the January 1999 maximum power load are very close to the actual values, for both prediction strategies.

In addition, Figure 6.16 is where CMPSO (and the offset addition) would have ranked in the competition based on the values shown in Table 6.6 (referenced from [62]). As shown in this figure, an attempt was made to rank CMPSO and CMPSO with the maximum power offset with the other competitors (in the absence of the actual result values from each competitor). There are two sets of highlighted text boxes and lines for each of the CMPSO results in this figure. Based on MAPE, it appears CMPSO alone would have ranked at least sixth. In the case where the January 1999 temperature is included, CMPSO (plus the offset) would have ranked no worse than third based on MAPE. Note that the MAXE values were also close to the other competitors (refer to Table 6.5 and Figure 6.16).

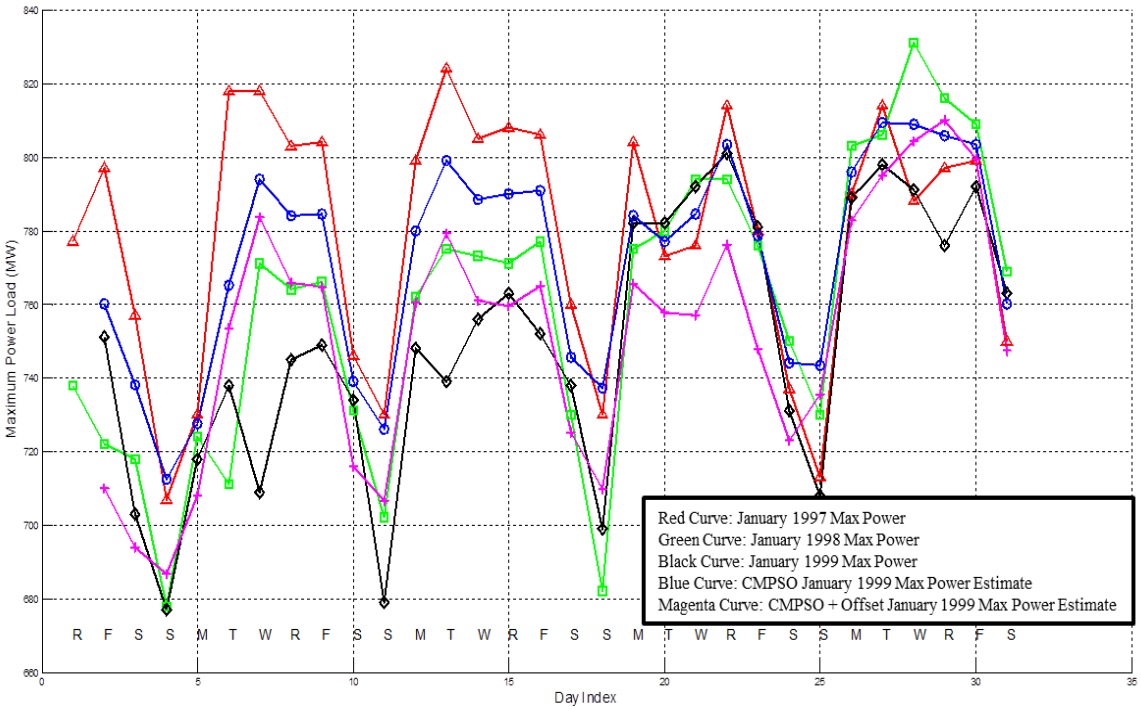

Figure 6.15: CMPSO EUNITE Prediction Results

The winners of this competition, Chen, Chang, and Lin [92] used an SVM/SVR approach with multi-dimensional inputs to predict load. They also took into consideration additional attributes such as if the day being predicted is a holiday. The authors further reported details of their approach in [93], noting that there is no added value in using temperature data to predict the load. Although the weighting of electrical load for a holiday was not taken into consideration for this CMPSO benchmark evaluation, there was clearly an improvement in CMPSO prediction performance by modeling temperature and including the offset in the results. For reference, the next eight winners are listed in references [94 through 101]. There are several neural network based approaches as well as fuzzy systems applied to this problem. It is interesting to note that SVR was not considered in any of the top competitors, except for the winner of the competition and CMPSO.

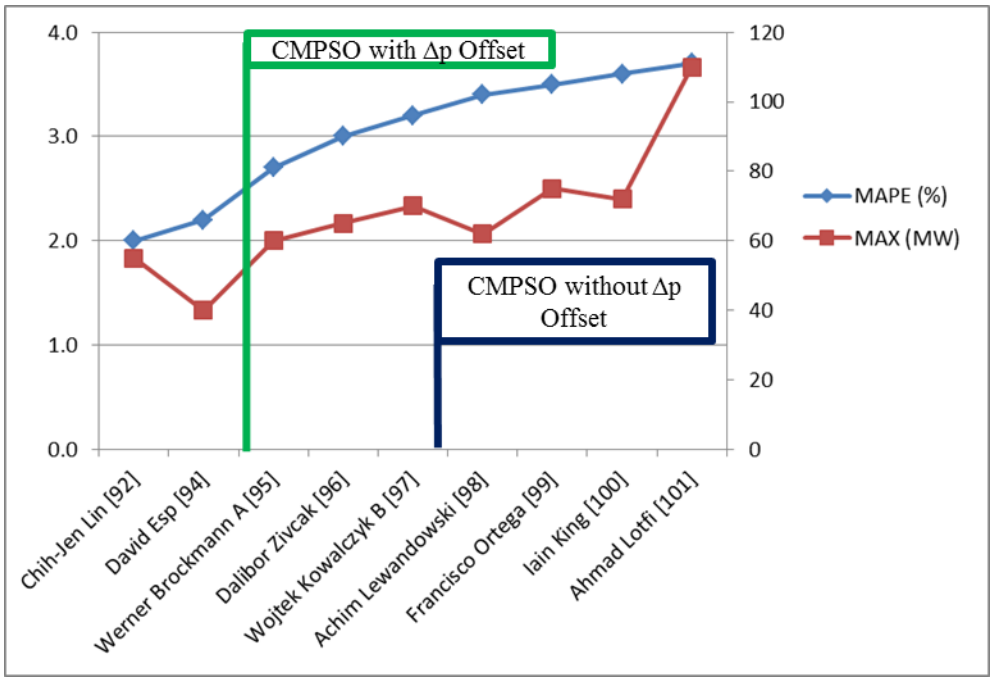

Figure 6.16: CMPSO EUNITE Results as Compared to Top 9 Competitors (from [62])

There are several SVM/SVR based research efforts since the competition was held that use EUNITE data as benchmark data for evaluation. These methods are used as comparison to

CMPSO, with and without the simple delta power offset strategy. Table 6.6 summarizes the results.

The first method uses an Empirical Mode Decomposition (EMD) and SVR hybrid method [102]. The EMD decomposes the data into smaller subsets at which point SVR is applied. Another SVM based method is described in [103], where a Least Squares SVM (LS-SVM) approach is used along with a grid search based model as well as a Bayesian framework. Also a grid based pattern search is used to evaluate the SVR user defined parameters.

The next reference compares many different models including ANNs and SVMs and applies them to the EUNITE data [104]. As it turns out, the best approach presented here was an Auto-Regressive type model (AR). Another SVM based approach is given in [105], where the past seven days are used to predict the next day ahead. Again, a Radial Basis function is used for the kernel. Another LS-SVM based approach is given that uses a k-nearest neighbors approach to help identify training patterns for the SVM [106]. Recall the LS-SVM uses a different loss function in the formulation as compared to CMPSO's ε-insensitive loss function.

6.5.3 CATS Competition Data

The last benchmark data set used to evaluate CMPSO is based on an artificially generated time series. In 2004 at the International Joint Conference on Neural Networks, a competition was organized to compare different prediction methods using a proposed CATS (Competition on Artificial Time Series) benchmark time series data of 5000 samples, among which 100 samples are missing [64]. The goal of the competition was to predict five sets of 20 missing data points each from the given data of 5000 samples and compare the methods using two criteria. The first was the MSE for the four missing data gaps at sample indices 981 to 1000, 1981 to 2000, 2981 to 3000, and 3981 to 4000 in addition to the last omitted 20 data points with sample indices 4981 to

5000. This error is denoted as E1 in the competition literature. The second was the MSE for the first four missing data gaps only. This error is denoted as E2 and was meant to only evaluate the effectiveness of the interpolation of a given algorithm versus the prediction estimate in E1 (the last 20 omitted samples). Figure 6.17 shows the entire CATS data set.

Table 6.6: CMPSO EUNITE Prediction Performance vs. Post Competition SVM Methods

Method	**Metric**			
	MAPE	**MAXE**	**MSE**	**Notes**
LS-SVM with k-nearest neighbor [Ref 106]	1.71	40.99	N/A	
LS-SVM [Ref 103]	1.97	39.778	N/A	Bayesian Approach
EMD-SVR [Ref 102]	1.98	N/A	284.3	
CMPSO with Linear Offset	2.628	74.663	594.3	Offset Applied for Jan 1999 Temperature Data
CMPSO	3.176	84.931	908.8	
SVM Model [Ref 105]	3.67	N/A	N/A	
AR Model [Ref 104]	6.69	N/A	N/A	AR Type Model Showed Better Result than SVM

The strategy for adapting CMPSO was straight forward for this particular application. For each of the first four data gaps, every tenth data sample was used. Half of the samples were before the gap and the other half after the gap. In addition, ten consecutive samples were used just before and after the data gap for a total of 94 data points per set. For the last data set, a similar strategy of every 10 data samples was selected prior to the end of the data set along with the last 20 sequential samples for a total of 138 samples. Every sample was estimated between the first and last index of the data set and compared to the given results for evaluation.

Table 6.7 shows the CMPSO results as compared to the top ten results in descending order of E1 MSE. From the results, CMPSO has the lowest E1 and E2 MSE scores in the competition. Since the competition concluded, there have been other time series prediction methods that have used the CATS benchmark for evaluation. These are also included in Table 6.7 and are marked with an asterisk (*). As seen in the table, CMPSO still outperformed all of the other more recent methodologies.

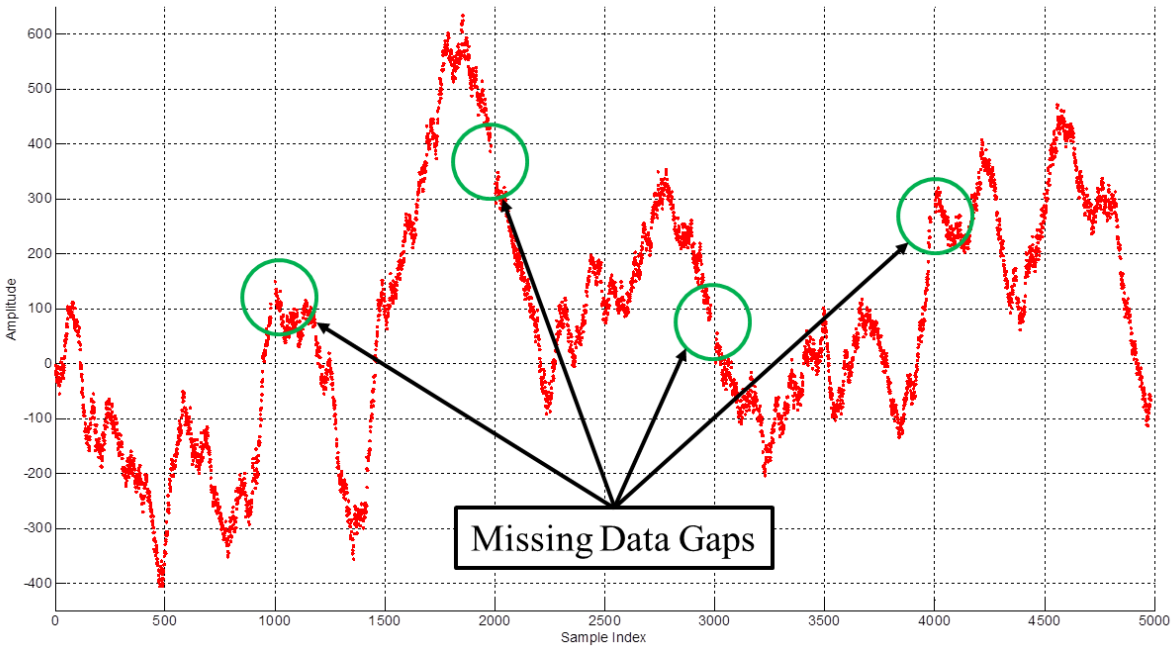

Figure 6.17: CATS Data

Figures 6.18 through 6.22 show the CMPSO results against the four missing gaps of data as well as the last extrapolation data set. The red curve is the original data set with the red circles being the actual missing data points (truth data). The blue curve (and blue circles) is the CMPSO estimate. As seen from these figures, there is very good agreement between the actual CATS data and the CMPSO results.

Table 6.7: CMPSO CATS Prediction Performance vs. Top Ten Entries and Recent Publications

E1 MSE	E2 MSE	Model
113	140	CMPSO
143	129	ANN and AdaBoost* [118]; Single Global Model
262	239	ANN and AdaBoost* [118]; Multiple Local Models
287	N/A	Variance Minimization LS-SVM* [120]
390	288	Dynamic Factor Graphs* [121]
408	346	Kalman Smoother [107]
441	402	Recurrent Neural Networks [108]
502	418	Competitive Associative Net [109]
530	370	Weighted Bidirectional Multi-stream Extended Kalman Filter [110]
577	395	SVCA Model [111]
644	542	MutliGrid-Based Fuzzy System [112]
653	351	Double Quantized Forecasting Method [113]
660	442	Time-reversal Symmetry Method [114]
676	677	BYY Harmony Learning Based Mixture of Experts Model [115]
725	222	Ensemble Models [116]
1215	979	Deep Belief Network with Boltzmann Machines* (best value) [119]
2510	2450	ANN Based Approach* (best value) [117]

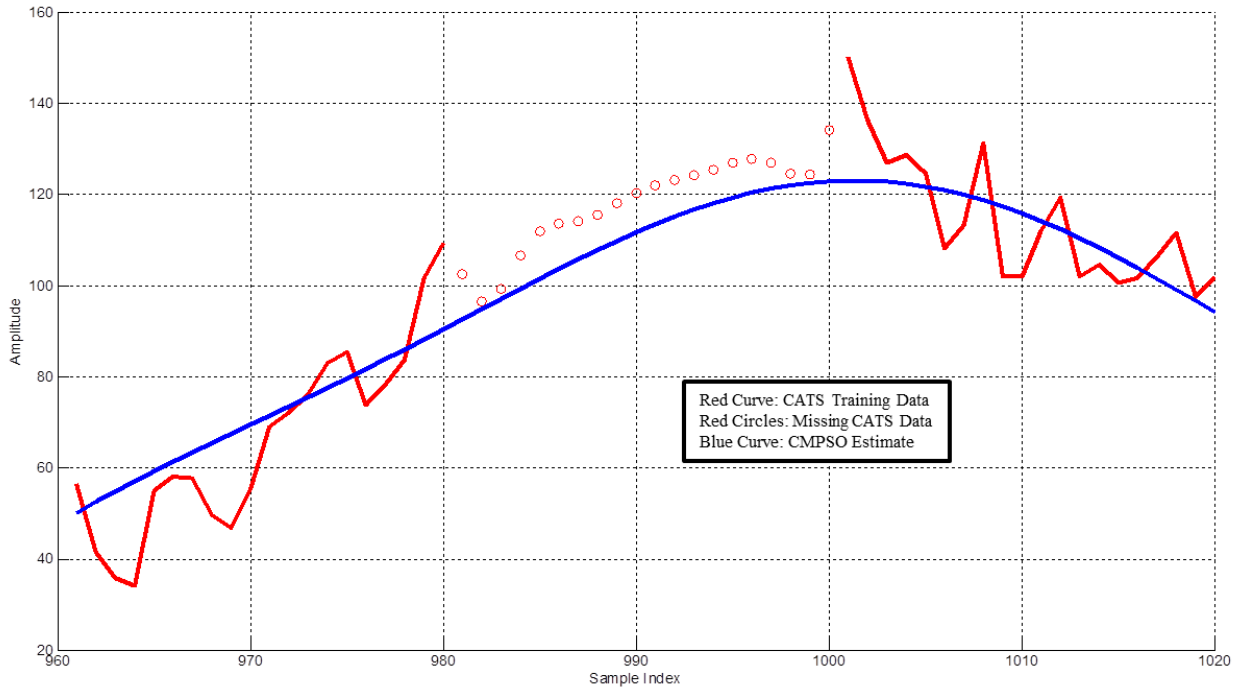

Figure 6.18: CMPSO Estimate for CATS Data Set 1

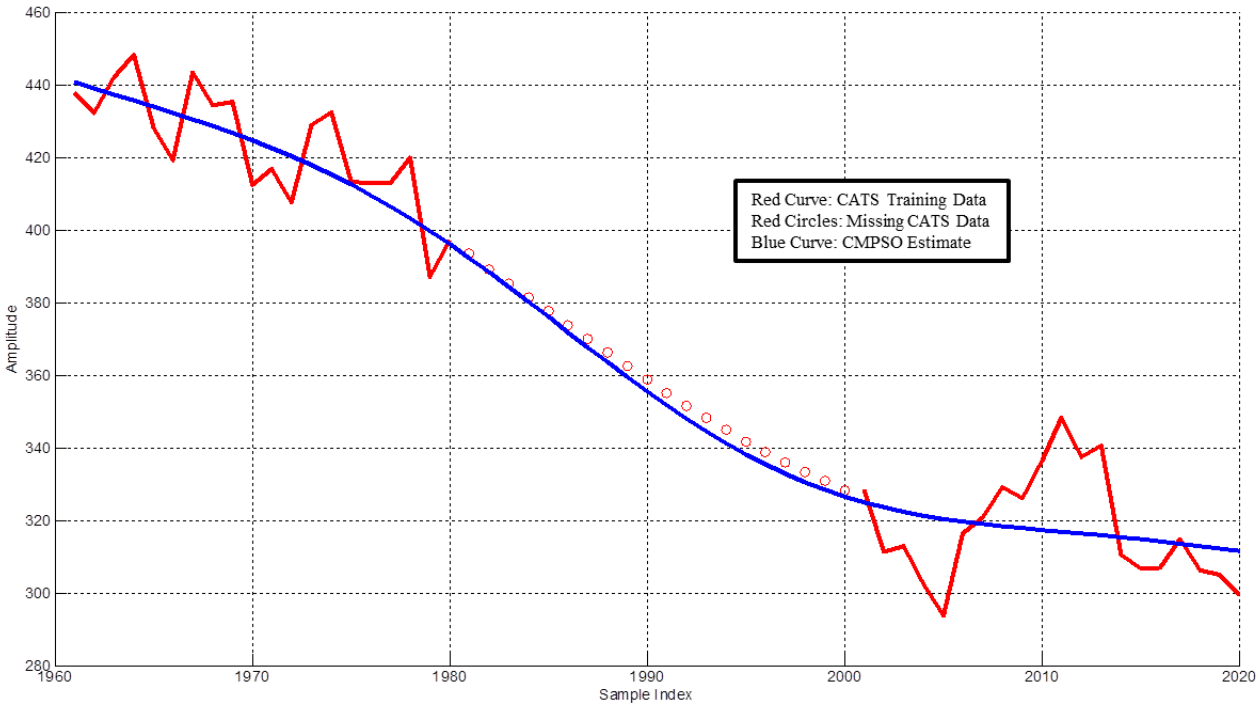

Figure 6.19: CMPSO Estimate for CATS Data Set 2

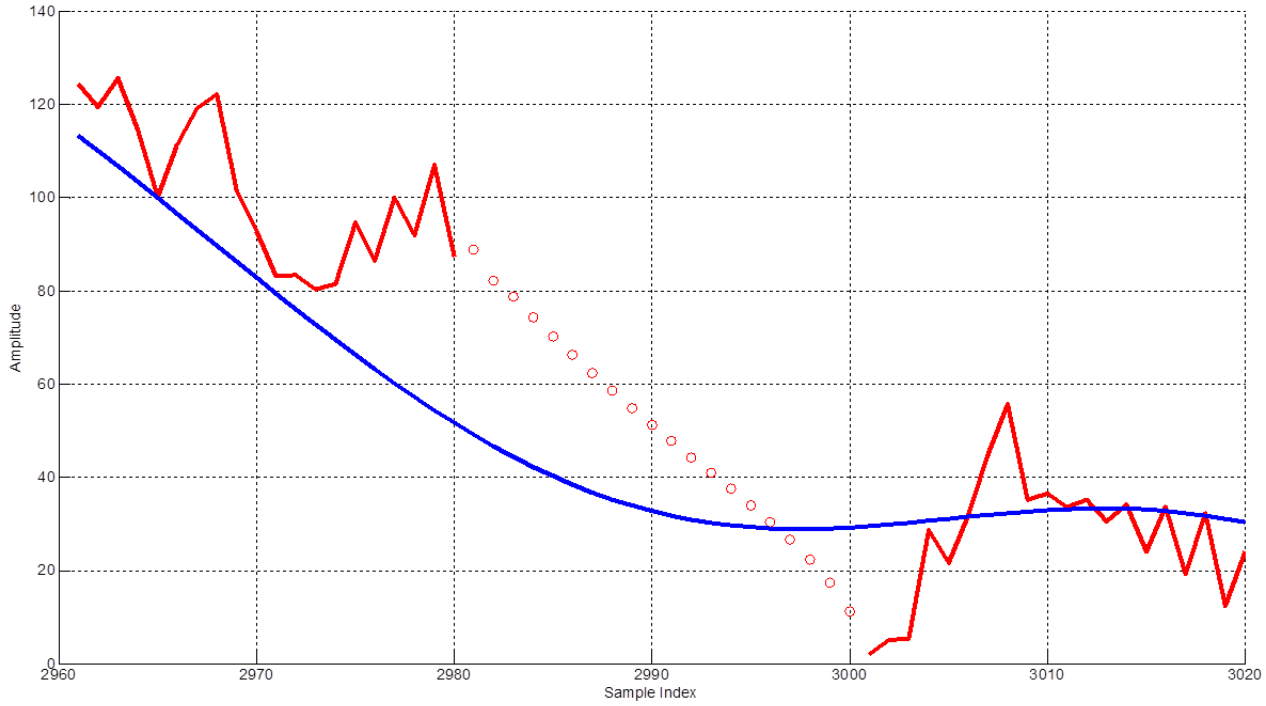

Figure 6.20: CMPSO Estimate for CATS Data Set 3

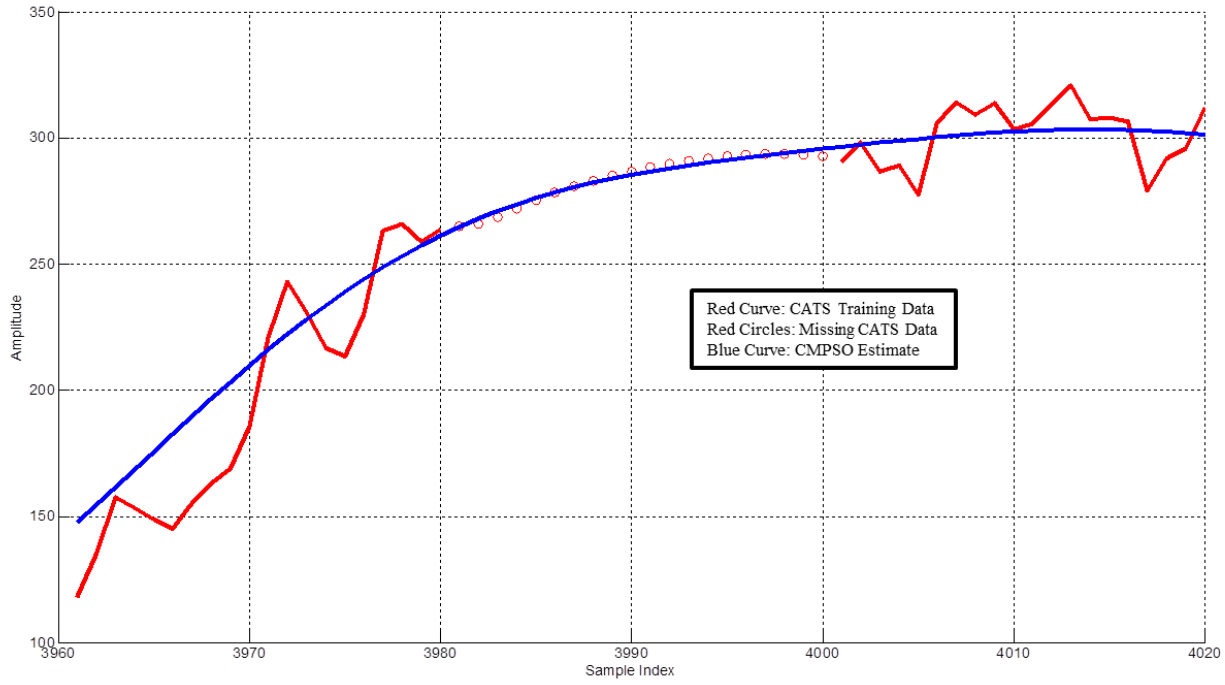

Figure 6.21: CMPSO Estimate for CATS Data Set 4

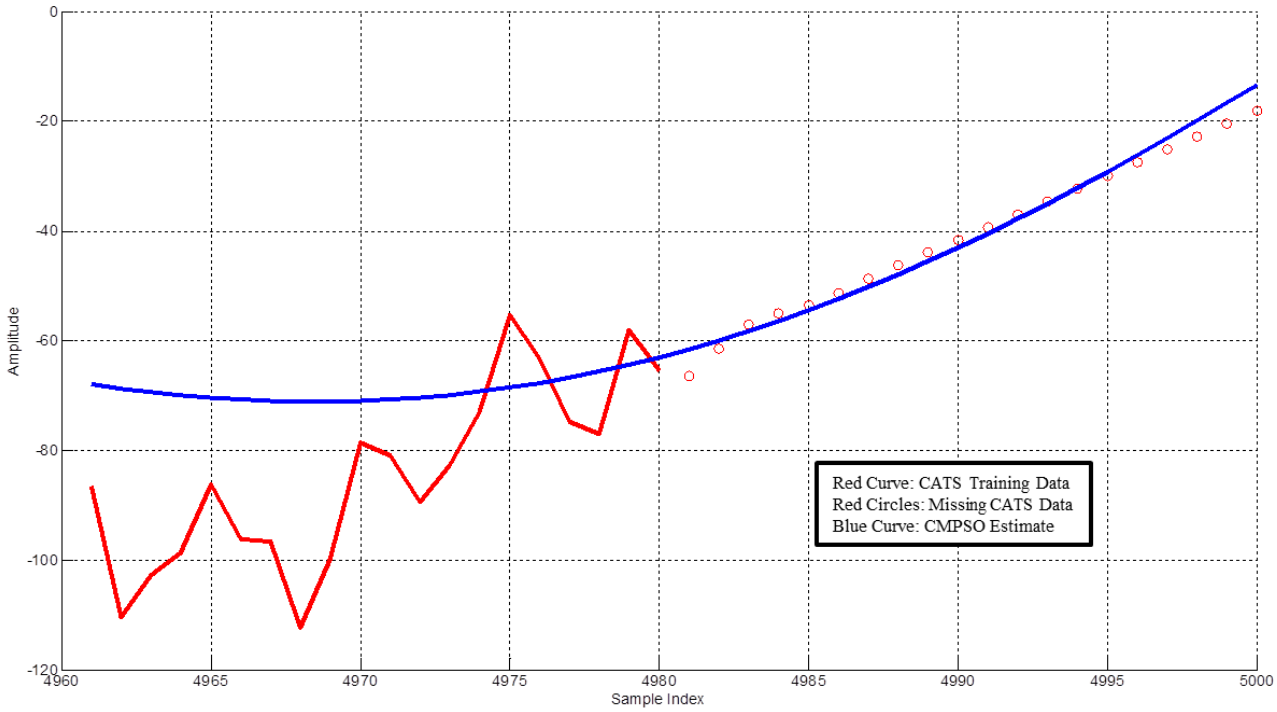

Figure 6.22: CMPSO Estimate for CATS Data Set 5

CHAPTER 7: CONCLUSIONS AND FUTURE RESEARCH

7.1 Conclusions

Support Vector Machines (SVMs) are powerful learning mechanisms that have been developed and matured over the last 15+ years. They provide a method for predicting and estimating time series for a myriad of applications including financial market forecasting, weather and environmental parameter estimation, electrical utility loading prediction, machine reliability forecasting, various signal processing and control system applications, and several other applications. Non-traditional time series prediction methods are necessary for these types of applications due to the highly non-linear aspects of the data and processes.

Support Vector Regression (SVR) research continues to be a viable approach in the prediction of time series data in non-linear systems. Many methods and alternatives exist in the design of SVRs and a great deal of flexibility is left to the designer in its implementation. The underlying motivation behind the development of CMPSO is to help the designer in defining implementation details and in providing a mechanism by which to tune SVR for a wide variety of time series prediction and regression problems.

CMPSO is a unique fusion of Support Vector Regression and Particle Swarm Optimization algorithms that attempts to optimize SVR free parameters ε, C, and σ, while minimizing the need for excess computational resources by reducing the required solution space. This research provides detailed analysis of the CMPSO technique and also provides proof of

performance by testing CMPSO performance against many different and diverse data sets. In all the presented sample cases, CMPSO has met or exceeded the performance of many other different time series regression and prediction algorithms while not requiring any algorithm tuning or modification. No user intervention is required to execute CMPSO. Based on results shown in this research, CMPSO is a viable SVR based time series regression and estimation approach as compared to other comparable SVM/SVR approaches as well as Neural Network based algorithms.

CMPSO has also demonstrated good performance against artificial data series as well as real world data. CMPSO has demonstrated good performance against financial market data such as the S&P 500 example shown in chapter 6. Against Mackey-Glass non-linear data, CMPSO performed well against other similar algorithms in the MSE sense. As compared to real world applications such as electrical load forecasting, CMPSO would have placed at least third in the EUNITE competition. Its performance based on more recent publications using EUNITE data shows that it is at least within one percentage point from a MAPE performance measure standpoint. CMPSO would have won the CATS competition, not only against the applicants in the original contest, but also against more recent time series regression and prediction algorithms.

7.2 Future Work

There are several topics of advance research that can be considered beyond the CMPSO formulation:

- Kernel Function Selection: The presented research used a Radial Basis Function as the Kernel for the SVR implementation, but as stated in Chapter 3, there are other functions that could be used or possible a combination of Kernel Functions.

- PSO Parameter Tuning: The computational performance aspects of the CMPSO software implementation were not in scope for this research, but alternative implementations (software hosting) on other computing platforms may result in other PSO parameter settings.
- CMPSO Distributed Computing: The use of multiple computing platforms to simultaneously employ different swarms could be considered to further increase the computational efficiency of the process and possibly handle much larger data sets in a smaller amount of time.
- Selection of Alternative Loss Functions: The SVR formulation in Chapter 3 assumed one kind of loss function for the dual objective function. Other functions exist and could potentially be used as part of the CMPSO framework. This was evident in the EUNITE competition data that used a least squares type function.
- Multi-Dimensional Input Data: The focus of this research has been the two dimensional case where there is only one independent variable. CMPSO and the SVR implementation can also be altered to handle multi-dimensional input vectors, where a vector could be a set of other quantized information or data regarding the dependent variable or underlying process.
- Pattern Recognition Applications: This research is for time series prediction and regression applications. The same methodology could be used for SVM pattern recognition applications.

REFERENCES

[1] N. Sapankevych and R. Sankar, "Time Series Prediction Using Support Vector Machines: A Survey", IEEE Computational Intelligence, vol. 4, no. 2, pp. 24-38, May 2009.

[2] V. N. Vapnik, " The Nature of Statistical Learning Theory", Springer, 1995.

[3] V. N. Vapnik, "Statistical Learning Theory", John Wiley and Sons, 1998.

[4] R. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory," in Proc. 6th Int. Symp. Micro Machine and Human Science (MHS '95), pp. 39-43, 1995.

[5] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in IEEE Proceedings on the Internation Conference of Neural Networks, vol. 4, pp. 1942-1948, 1995.

[6] J. Kennedy and R. C. Eberhart, "Swarm Intelligence", Morgan Kaufmann, 2001.

[7] J. Kennedy, "The Behavior of Particles," Proceedings of the 7th Annual Conference on Evolutionary Programming (EP-98), Lecture Notes in Computer Science, vol. 1447, pp. 581-589, Mar. 1998.

[8] R. Poli, J. Kennedy, and T. Blackwell, "Particle Swarm Optimization: An Overview", Swarm Intelligence, vol. 1, pp. 33-57, 2007.

[9] R. Poli, "An Analysis of Publications on Particle Swarm Optimization Applications", University of Essex, UK, Department of Computer Science, Technical Report CSM-469, May 2007 (Revised Nov. 2007).

[10] J. Robinson and Y. Rahmat-Samii, "Particle Swarm Optimization in Electromagnetics", IEEE Transactions on Antennas and Propagation, vol. 52, no. 2, pp. 397-407, Feb. 2004.

[11] D. W. Boeringer and D. H. Werner, "Particle Swarm Optimization Versus Genetic Algorithms for Phased Array Synthesis", IEEE Transactions on Antennas and Propagation, vol. 52, no. 3, pp. 771-779, Mar. 2004.

[12] S. J. Orfanidis, "Optimum Signal Processing: An Introduction – Second Edition", MacMillan Publishing Company, 1988.

[13] R. D. Yates, D. J. Goodman, “Probability and Stochastic Processes”, John Wiley and Sons, 2002.

[14] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems”, Transactions of the ASME – Journal of Basic Engineering, vol. 82 (Series D), pp. 35-45, 1960.

[15] S. S. Blackman, “Multiple-Target Tracking with Radar Applications”, Artech House, 1986.

[16] G. Minkler, J. Minkler, “Theory and Application of Kalman Filtering”, Magellan Book Company, 1993.

[17] R. G. Brown, P. Y. C. Hwang, “Introduction to Random Signals and Applied Kalman Filtering – Second Edition”, John Wiley and Sons, 1992.

[18] S. Haykin, “Neural Networks Second Edition”, Prentice Hall, 1999.

[19] A. K. Jain, J. Mao, “Artificial Neural Networks: A Tutorial”, Computer, vol. 29, issue 3, Mar. 1996.

[20] B. Schölkopf, A. J. Smola, and C. Burges, “Advances in Kernel Methods – Support Vector Learning”, MIT Press, Cambridge, MA, 1999.

[21] N. Cristianini and J. Shawe-Taylor, “An Introduction to Support Vector Machines and other Kernel-Based Learning Methods”, Cambridge University Press, 2000.

[22] A. J. Smola and B. Schölkopf, “A Tutorial on Support Vector Regression”, NeuroCOLT Technical Report, Royal Holloway College, London, UK, 1998.

[23] B. Schölkopf, A. J. Smola, and C. Burges, “Advances in Kernel Methods – Support Vector Learning”, MIT Press, Cambridge, MA, 1999.

[24] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, V. Vapnik, “Predicting Time Series with Support Vector Machines”, Proceedings of the International Conference on Artificial Neural Networks, Springer, 1997.

[25] S. Mukherjee, E. Osuna, and F. Girosi, “Nonlinear Prediction of Chaotic Time Series Using Support Vector Machines”, Proceedings of the 1997 IEEE Workshop – Neural Networks for Signal Processing VII, pp. 511-520, Sep. 1997.

[26] N. de Freitas, M. Milo, P. Clarkson, M. Niranjan, and A. Gee, “Sequential Support Vector Machines”, Proceedings of the 1999 IEEE Signal Processing Society Workshop – Neural Networks for Signal Processing IX, pp. 31-40, Aug. 1999.

[27] S. Rüping, “SVM Kernels for Time Series Analysis”, CS Department, AI Unit, University of Dortmund, 44221 Dortmund, Germany, 2001.

[28] K. L. Lee and S. A. Billings, “Time Series Prediction Using Support Vector Machines, the Orthogonal and Regularized Orthogonal Least-Squares Algorithms”, International Journal of Systems Science, vol. 33, no. 10, pp. 811-821, 2002.

[29] L. Cao and Q. Gu, “Dynamic Support Vector Machines for Non-Stationary Time Series Forecasting”, Intelligent Data Analysis, vol. 6, no. 1, pp. 67-83, 2002.

[30] J.-Y. Zhu, B. Ren, H.-X. Zhang, and Z.-T. Deng, “Time Series Prediction via New Support Vector Machines”, Proceedings of the First International Conference on Machine Learning and Cybernetics, vol. 1, pp. 364-366, Nov. 2002.

[31] S. Rüping and K. Morik, “Support Vector Machines and Learning About Time”, Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), vol. 4, pp. 864-867, Apr. 2003.

[32] L. J. Cao, “Support Vector Machine Experts for Time Series Prediction”, Neurocomputing, vol. 51, pp. 321-339, Apr. 2003.

[33] Y. Mei-Ying and W. Xiao-Dong, “Chaotic Time Series Prediction Using Least Squares Support Vector Machines”, Chinese Physics, vol. 13, no. 4, pp. 454-458, Apr. 2004.

[34] J. M. Górriz, C. G. Puntonet, M. Salmerón, R. Martin-Clemente, and S. Hornillo-Mellado, “Using Confidence Interval of a Regularization Network”, Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (MELECON 2004), pp. 343-346, May 12-15, 2004.

[35] A. Hornstein and U. Parlitz, “Bias Reduction for Time Series Models Based on Support Vector Regression”, International Journal of Bifurcation and Chaos, vol. 14, no. 6, pp. 1947-1956, 2004.

[36] J. M. Górriz, C. G. Puntonet, M. Salmerón, J. J. G. de la Rosa, “A New Model for Time Series Forecasting Using Radial Basis Functions and Exogenous Data”, Neural Computing and Applications, vol. 13, no. 2, pp. 101-111, Jun. 2004.

[37] C. Cortes and V. Vapnik, “Support-Vector Networks”, Machine Learning, pp. 273-297, 1995.

[38] Support Vector Machines: http://www.support-vector.net/index.html (Jan 2015)

[39] Kernel Machines: http://www.kernel-machines.org/index.html (Jan 2015)

[40] International Neural Network Society: http://www.inns.org/ (Jan 2015)

[41] Neural Computation: http://www.mitpressjournals.org/loi/neco (Jan 2015)

[42] European Neural Network Society: http://www.snn.ru.nl/enns/ (Jan 2015)

[43] Asia Pacific Neural Network Assembly: http://www.apnna.net/apnna (Jan 2015)

[44] Japanese Neural Network Society: http://www.jnns.org/english/index.php (Jan 2015)

[45] Journal of Artificial Intelligence Research: http://www.jair.org/ (Jan 2015)

[46] S. Rüping, http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html (Jan 2015)

[47] R. Collobert and S. Bengio, "SVMTorch: Support Vector Machines for Large-Scale Regression Problems", Journal of Machine Learning Research, vol. 1, pp. 143-160, Sep. 2001.

[48] K. Pelckmans, J. A. K. Suykens, T. Van Gestel, J. De Brabanter, L. Lukas, B. Hamers, B. De Moor, and J. Vandewalle
http://www.esat.kuleuven.be/sista/lssvmlab/tutorial/lssvmlab_paper0.pdf (Jan 2015)

[49] A. Schwaighofer,
http://www.princeton.edu/~kung/ele571/571-MatLab/571svm/svmtrain.m
(Jan 2015)

[50] SMO Matlab Based Algorithm Implementation:
http://www.codeforge.com/read/131255/svm_SMO.m__html (Jan 2015)

[51] X. Wang, J. Lu and J. Liu, "Wavelet Transform and PSO Support Vector Machine Based approach for Time Series Forecasting", Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI), pp. 46-50, Nov. 2009.

[52] J. Hu, P. Gao, Y. Yao, and X. Xie, "Traffic Flow Forecasting with Particle Swarm Optimization and Support Vector Regression", Proceedings of the 17th International Conference on Intelligent Transportation Systems (ITSC), pp. 2267-2268, Oct. 2014.

[53] Y. Wen, Y. Chen, "Modified Parallel Cat Swarm Optimization in SVM Modeling for Short-term Cooling Load Forecasting", Journal of Software, vol. 9, no. 8, pp. 2093-2104, Aug. 2014.

[54] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble Deep Learning for Regression and Time Series Forecasting", Proceedings of the IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL), pp. 1-6, Dec. 2014.

[55] C. Rajeswari, B. Sathiyabhama, S. Devendiran, and K. Manivannan, "Bearing Fault Diagnosis Using Wavelet Packet Transform, Hybrid PSO, and Support Vector Machine", Proceedings of the 12th Global Congress on Manufacturing and Management (GCMM), Procedia Engineering, vol. 97, pp. 1772-1783, 2014.

[56] P. Shinde and T. Parvat, "Analysis on: Intrusions Detection Based on Support Vector Machine Optimized with Swarm Intelligence", International Journal of Computer Science and Mobile Computing, vol. 3, no. 12, pp. 559-566, Dec. 2014.

[57] W. Wenhai and D. Jiandong, "Short-term Wind Power Forecasting Based on Maximum Correntropy Criterion", Proceedings of the International Conference on Power System Technology (POWRCON), pp. 2800-2805, Oct. 2014

[58] Z. Hu, Q. Liu, Y. Tian, and Y. Liao, "A Short-term Wind Speed Forecasting Model Based on Improved QPSO Optimizing LSSVM", Proceedings of the International Conference on Power System Technology (POWERCON), pp. 2806-2811, Oct. 2014.

[59] H. Dong, X. Zhu, Y. Liu, F. He, and G. Huo, "Iris Recognition Based on CPSO Algorithm for Optimizing Multichannel Gabor Parameters", Journal of Computational Information Systems, vol. 11, no. 1, pp. 333-340, 2015.

[60] J. F. L. de Oliveira and T. B. Ludermir, "A Distributed PSO-ARIMA-SVR Hybrid System for Time Series Forecasting", Proceedings of the IEEE International Conferences on Systems, Man, and Cybernetics (SMC), pp. 3867-3872, Oct. 2014.

[61] http://www.scholarpedia.org/article/Mackey-Glass_equation (Jan 2015)

[62] http://www.eunite.org (Jan 2015)

[63] http://neuron-ai.tuke.sk/competition (Jan 2015)

[64] Lendasse, E. Oja, O. Simula, and M. Verleysen, "Time Series Prediction Competition: The CATS Benchmark", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1615-1620, Jul. 2004.

[65] T. B. Trafalis and H. Ince, "Support Vector Machine for Regression and Applications to Financial Forecasting", Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN), vol. 6, pp. 348-353, Jul. 2000.

[66] F. E. H. Tay and L. J. Cao, "Application of Support Vector Machines in Financial Time Series Forecasting", Omega, vol. 29, pp. 309-317, 2001.

[67] T. Van Gestel, J. A. K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, and J. Vandewall, "Financial Time Series Prediction Using Least Squares Support Vector Machines Within the Evidence Framework", IEEE Transactions on Neural Networks, vol. 12, no. 4, pp. 809-821, Jul. 2001.

[68] F. E. H. Tay and L. J. Cao, "Improved Financial Time Series Forecasting by Combining Support Vector Machines with Self-Organizing Feature Map", Intelligent Data Analysis, vol. 5, no. 4, pp. 339-354, 2001.

[69] F. E. H. Tay and L. J. Cao, "Modified Support Vector Machines in Financial Time Series Forecasting", Neurocomputing, vol. 48, pp. 847-861, Oct. 2002.

[70] F. E. H. Tay and L. J. Cao, "ε-Descending Support Vector Machines for Financial Time Series Forecasting", Neural Processing Letters, vol. 15, no. 2, pp. 179-195, 2002.

[71] H. Yang, I. King, and L. Chan, "Non-Fixed and Asymetrical Margin Approach to Stock Market Prediction using Support Vector Regression", Proceedings of the 9th International Conference on Neural Information Processing (ICONIP), vol. 3, pp. 1398-1402, Nov. 2002.

[72] H. Yang, L. Chan, and I. King, "Support Vector Machine Regression for Volatile Stock Market Prediction", Proceedings of the Third International Conference on Intelligent Data Engineering and Automated Learning (IDEAL), Springer, pp. 391-396, 2002.

[73] A. Abraham, N. S. Philip, and P. Saratchandran, "Modeling Chaotic Behavior of Stock Indices Using Intelligent Paradigms", International Journal of Neural, Parallel, and Scientific Computations, vol. 11, nos. 1 and 2, pp. 143-160, Mar. 2003.

[74] H. Yang, "Margin Variations in Support Vector Regression for the Stock Market Prediction", Ph. D. Thesis, Chinese University of Hong Kong, Jun. 2003.

[75] P. Ongsritrakul and N. Soonthornphisaj, "Apply Decision Tree and Support Vector Regression to Predict the Gold Price", Proceedings on the International Joint Conference on Neural Networks (IJCNN), vol. 4, pp. 2488-2492, Jul. 2003.

[76] L. J. Cao and F. E. H. Tay, "Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting", IEEE Transaction on Neural Networks, vol. 14, no. 6, pp. 1506-1518, Nov. 2003.

[77] Y. Liang and Y. Sun, "An Improved Method of Support Vector Machine and its Application to Financial Time Series Forecasting", Progress in Natural Science, vol. 13, no. 9, pp. 696-700, 2003.

[78] A. Abraham and A. A. Yeung, "Integrating Ensemble of Intelligent Systems for Modeling Stock Indices", Lecture Notes in Computer Science, vol. 2687, pp. 774-781, 2003.

[79] K.-J. Kim, "Financial Time Series Forecasting Using Support Vector Machines", Neurocomputing, vol. 55, pp. 307-319, 2003.

[80] Y. Bao, Y. Lu, and J. Zhang, "Forecasting Stock Price by SVMs Regression", Lecture Notes in Computer Science, vol. 3192, pp. 295-303, 2004.

[81] H. Yang, K. Huang, L. Chan, I. King, and M. R. Lyu, "Outliers Treatment in Support Vector Regression for Financial Time Series Prediction", Proceedings of the 11th International Conference on Neural Information Processing (ICONIP) - Lecture Notes in Computer Science, vol. 3316, pp. 1260-1265, 2004.

[82] W. Huang, Y. Nakamori, S.-Y. Wang, "Forecasting Stock Market Movement Direction with Support Vector Machine", Computers and Operations Research, vol. 32, no. 10, pp. 2513-2522, Oct. 2005.

[83] Y.-K. Bao, Z.-T. Liu, L. Guo, and W. Wang, "Forecasting Stock Composite Index by Fuzzy Support Vector Machines Regression", Proceedings of the Fourth International Conference on Machine Learning and Cybernetics , pp. 3535-3540, Aug. 2005.

[84] D.-Z. Cao, S.-L. Pang and Y.-H. Bai, "Forecasting Exchange Rate Using Support Vector Machines", Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, pp. 3448-3452, Aug. 2005.

[85] T. Z. Tan, C. Quek, and G. S. Ng, "Brain-inspired Genetic Complimentary Learning for Stock Market Prediction", Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 2653-2660, Sep. 2005.

[86] N. Sapankevych and R. Sankar, "Constrained Motion Particle Swarm Optimization and Support Vector Regression for Non-Linear Time Series Regression and Prediction Applications", Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA), pp. 473-477, Dec. 2013.

[87] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble Deep Learning for Regression and Time Series Forecasting", Proceedings of the IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL), pp. 1-6, Dec. 2014.

[88] L. Ralaivola and F. d'Alche-Buc, "Dynamical Modeling with Kernels for Nonlinear Time Series Prediction", Proceedings of the Neural Information Processing Systems (NIPS), pp. 8-13, Dec. 2003.

[89] O. A. Omitaomu, M. K. Jeong, and A. B. Badiru, "Online Support Vector Regression With Varying Parameters for Time-Dependent Data", IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, vol. 41, no. 1, pp. 191-197, Jan. 2011.

[90] Y. Bao, T. Xiong, Z. Hu, "Multi-step-ahead Time Series Prediction Using Multiple-Output Support Vector Regression", Neurocomputing, vol. 129, pp. 482-493, 2014.

[91] L. Zhang, W.-D. Zhou, P.-C. Chang, J.-W. Yang, and F.-Z. Li, "Iterated Time Series Prediction with Multiple Support Vector Regression Models", Neurocomputing, vol. 99, pp. 411-422, 2013.

[92] M.-W. Chang, B.-J. Chen, and C.-J. Lin, "EUNITE Network Competition: Electricity Load Forecasting", Nov. 2001.

[93] B.-J. Chen, M.-W. Chang, and C.-J. Lin, "Load Forecasting Using Support Vector Machines: A Study on EUNITE Competition 2001", IEEE Transactions on Power Systems, vol. 19, no. 4, pp. 1821-1830, Nov. 2004.

[94] D. Esp, "Adaptive Logic Networks for East Slovakian Electrical Load Forecasting", EUNITE Competition, 2001 (see [62]).

[95] W. Brockmann and S. Kuthe, "Different Models to Forecast Electricity Loads", EUNITE Competition, 2001 (see [62]).

[96] S. Zivcac, "Electricity Load Forecasting using ANN", EUNITE Competition, 2001 (see [62]).

[97] W. Kowalczyk, "Averaging and data enrichment: two approaches to electricity load forecasting", EUNITE Competition, 2001 (see [62]).

[98] A. Lewandowski, F. Sandner, and P. Protzel, "Prediction of electricity load by modeling the temperature dependencies", Report for EUNITE 2001 Competition, 2001 (see [62]).

[99] F. Ortega et. al., "An Hybrid Approach to Prediction of Electric Load with MARS and Kohonen Maps", EUNITE Competition, 2001 (see [62]).

[100] I. King and J. Tindle, "Storage of Half Hourly Electric Metering Data and Forecasting with Artificial Neural Network Technology", EUNITE Competition, 2001 (see [62]).

[101] A. Lotfi, "Application of Learning Fuzzy Inference Systems in Electricity Load Forecast", EUNITE Competition, 2001 (see [62]).

[102] B. Bican and Y. Yaslan, "A Hybrid Method for Time Series Prediction Using EMD and SVR", Proceedings of the 6th International Conference on Communications, Control, and Signal Processing (ISCCSP), pp. 566-569, May 2014.

[103] X. Yang, "Comparison of the LS-SVM Based Load Forecasting Models", Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), vol. 6, pp. 2942-2945, Aug. 2011.

[104] I. Fernandez, C. E. Borges, and Y. K. Penya, "Efficient Building Load Forecasting", Proceedings of the IEEE 16th Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8, Sep. 2011.

[105] B. E. Turkay and D. Demren, "Electrical Load Forecasting Using Support Vector Machines", Proceedings of the 7th International Conference on Electrical and Electronics Engineering (ELECO), vol. I, pp. 49-53, Dec. 2011.

[106] T. T. Chen and S. J. Lee, "A Weighted LS-SVM Based Learning System for Time Series Forecasting", Information Sciences, vol. 299, pp. 99-116, Apr. 2015.

[107] S. Sarkka, A. Vehtari, and J. Lampinen, "Time Series Prediction by Kalman Smoother with Cross Validated Noise Density", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1653-1657, Jul. 2004.

[108] X. Cai, N. Zhang, G. Venayagamoorthy, and D. Wunsch, "Time Series Prediction with Recurrent Neural Networks Using a Hybrid PSO-EA Algorithm", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2 pp. 1647-1652, Jul. 2004.

[109] S. Kurogi, T. Ueno, and M. Sawa, "Batch Learning Competitive Associative Net and Its Application to Time Series Prediction", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1591-1596, Jul. 2004.

[110] X. Hu and D. Wunsch, "IJCNN 2004 Challenge Problem: Time Series Prediction with a Weighted Bidirectional Multi-stream Extended Kalman Filter", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1641-1645, Jul. 2004.

[111] F. Palacios-Gonzalez, "A SVCA Model for The Competition on Artificial Time Series", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 4, pp. 2777-2782, Jul. 2004.

[112] L. J. Herrera Maldonado, H. Pomares, I. Rojas, J. Gonzalez, and M. Awad, "MultiGrid-Based Fuzzy Systems for Time Series Forecasting: CATS Benchmark IJCNN Competition", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1603-1608, Jul. 2004.

[113] G. Simon, J. A. Lee, M. Verleysen, and M. Cottrell, "Double Quantization Forecasting Method for Filling Missing Data in the CATS Time Series", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1635-1640, Jul. 2004.

[114] P. F. Verdes, P. M. Granitto, M. I. Szeliga, A. Rebola, and H. A. Ceccatto, "Prediction of the CATS benchmark exploiting time-reversal symmetry", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1631-1634, Jul. 2004.

[115] H.-W. Chan, W.-C. Leung, K.-C. Chiu, and L. Xu, "BYY Harmony Learning Based Mixture of Experts Model for Non-stationary Time Series Prediction", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), Jul. 2004.

[116] J. Wichard and M. Ogorzalek, "Time Series Prediction with Ensemble Models", Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), vol. 2, pp. 1625-1630, Jul. 2004.

[117] D. Samek and D. Manas, "Comparison of Artificial Neural Networks Using Prediction Benchmarking", Proceedings of the 13th WSEAS International Conference on Automatic Control, Modelling and Simulation (ACMOS), pp. 152-157, 2011.

[118] Y. Dong, J. Zhang, and J. M. Garibaldi, "Neural Networks and AdaBoost Algorithm Based Ensemble Models for Enhance Forecasting of Nonlinear Time Series", Proceedings of the International Joint Conference on Neural Networks (IJCNN), vol. 1, pp. 149-156, Jul. 2014.

[119] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi, "Time Series Forecasting Using a Deep Belief Network with Restricted Boltzmann Machines", Neurocomputing, pp. 47-56, 2014.

[120] R. Ormandi, "Applications of Support Vector-Based Learning", Research Group on Artificial Intelligence of the University of Szeged and the Hungarian Academy of Sciences, Ph. D. Dissertation, 2013.

[121] P. Mirowski, "Time Series Modeling with Hidden Variables and Gradient-Based Algorithms", Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, Ph. D. Dissertation, 2011.

[122] http://mathworld.wolfram.com/Regression.html (Jan 2015)

[123] S. Boyd and L. Vandenberghe, "Convex Optimization", Cambridge University Press, 2004.

[124] D. Bertsimas and J. N. Tsitsiklis, "Introduction to Linear Optimization", Athena Scientific, 1997.

[125] U. Paquest and A. P. Engelbrecht, "Training Support Vector Machines with Particle Swarms", Proceedings of the International Joint Conference on Neural Networks (IJCNN), pp. 1593-1598, 2003.

[126] M. Reyes-Sierra and C. A. C. Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," Proceedings of the International Journal of Computational Intelligence Research, vol. 2, no. 3, pp. 287-308, 2006.

[127] J. Fieldsend, “Multi–Objective Particle Swarm Optimization Methods”, Technical Report No. 419, Department of Computer Science, University of Exeter, 2004.

[128] W.-C. Hong, “Chaotic Particle Swarm Optimization Algorithm in a Support Vector Regression Electric Load Forecasting Model”, Energy Conversion and Management, vol. 50, no. 1, pp. 105-117, 2009.

[129] X. C. Guo, J. H. Yang, C. G. Wu, C. Y. Wang, and Y. C. Liang, "A Novel LS-SVMs Hyper-Parameter Selection Based on Particle Swarm Optimization", Neurocomputing, vol. 71, no. 16-18, pp. 3211-3215, 2008.

[130] H. Yuan, Y. Zhang, D. Zhang, and G. Yang, “A Modified Particle Swarm Optimization Algorithm for Support Vector Machine Training”, Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA), vol. 1, pp. 4128-4132, Jun. 2006.

[131] U. Paquest and A. P. Engelbrecht, “A New Particle Swarm Optimizer for Linearly Constrained Optimization”, Proceedings of the IEEE Congress on Evolutionary Computation (CEC), vol. 1, pp. 227-233, Dec. 2003.

[132] http://www.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html (Jan 2015)

[133] https://www.quandl.com/YAHOO/INDEX_GSPC-S-P-500-Index (Jan 2015)

[134] M. Mohandes, “Support Vector Machines for Short-Term Load Forecasting”, International Journal of Energy Research, vol. 26, no. 4, pp. 335-345, Mar. 2002.

[135] D. C. Sansom and T. K. Saha, “Energy Constrained Generation Dispatch based on Price Forecasts Including Expected Values and Risk”, IEEE Power Energy Society General Meeting, vol. 1, pp. 261-266, Jun. 2004.

[136] L. Tian and A. Noore, “A Novel Approach for Short-Term Load Forecasting Using Support Vector Machines”, International Journal of Neural Systems, vol. 14, no. 5, pp. 329-335, Aug. 2004.

[137] B. Dong, C. Cao, and S. E. Lee, “Applying Support Vector Machines to Predict Building Energy Consumption in Tropical Region”, Energy and Buildings, vol. 37, no. 5, pp. 545-553, May 2005.

[138] Z. Bao, D. Pi, and Y. Sun, “Short Term Load Forecasting Based on Self-organizing Map and Support Vector Machine”, Proceedings of the First International Conference on Natural Computation (ICNC), Advances in Natural Computation, Lecture Notes in Computer Science (LNCS), vol. 3610, pp. 688-691, Aug. 2005.

[139] P.-F. Pai and W. C. Hong, "Forecasting Regional Electricity Load Based on Recurrent Support Vector Machines with Genetic Algorithms", Electric Power Systems Research, vol. 74, no. 3, pp. 417-425, 2005.

[140] Y. Ji, J. Hao, N. Reyhani, and A. Lendasse, "Direct and Recursive Prediction of Time Series Using Mutual Information Selection", Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN), Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science (LNCS), vol. 3512, pp. 1010-1017, Jun. 2005.

[141] M.-G. Zhang, "Short-Term Load Forecasting Based on Support Vector Machine Regression", Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, vol. 7, pp. 4310-4314, Aug. 2005.

[142] X. Li, C. Sun, and D. Gong, "Application of Support Vector Machine and Similar Day Method for Load Forecasting", Proceedings of the First International Conference on Natural Computation (ICNC), Advances in Natural Computation, Lecture Notes in Computer Science (LNCS), vol. 3611, pp. 602-609, 2005.

[143] P.-F. Pai and W.-C. Hong, "Support Vector Machines with Simulated Annealing Algorithms in Electricity Load Forecasting", Energy Conversion and Management, vol. 46, no. 17, pp. 2669-2688, Oct. 2005.

[144] H.-S. Wu and S. Zhang, "Power Load Forecasting with Least Squares Support Vector Machines and Chaos Theory", Proceedings of the International Conference on Neural Networks and Brain (ICNN&B), vol. 2, pp. 1020-1024, Oct. 2005.

[145] M. Espinoza, J. A. K. Suykens, and B. De Moor, "Load Forecasting Using Fixed-Size Least Squares Support Vector Machines", Proceedings of the 8th International Work-Conference on Artificial Neural Networks (IWANN), Computational Intelligence and Bioinspired Systems, Lecture Notes in Computer Science (LNCS), vol. 3512, pp. 1018-1026, 2005.

[146] C.-C. Hsu, C.-H. Wu, S.-C. Chen, and K.-L. Peng, "Dynamically Optimizing Parameters in Support Vector Regression: An Application of Electricity Load Forecasting", Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS), vol. 2, pp. 1-8, Jan. 2006.

[147] M. Espinoza, J. A. K. Suykens, and B. De Moor, "Fixed-size Least Square Support Vector Machines: a Large Scale Application in Electrical Load Forecasting", Computational Management Science, vol. 3, no. 2, pp. 113-129, Apr. 2006.

[148] Y. He, Y. Zhu, and D. Duan, "Research on Hybrid ARIMA and Support Vector Mchine Model in Short Term Load Forecasting", Proceedings of the 6th International Conference on Intelligent Systems Design and Applications (ISDA), vol. 1, pp. 804-809, Oct. 2006.