1-1-2013

The Inference Engine

Nate Phillips

The inference engine

By

Nathaniel C Phillips

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Computer Science
in the Department of Computer Science and Engineering

Mississippi State, Mississippi

May 2013

The inference engine

By

Nathaniel C Phillips

Approved:

Name: Nathaniel C Phillips

Date of Degree: May 10, 2013

Institution: Mississippi State University

Major Field: Computer Science

Major Professor: J. Edward Swan, II

Title of Study: The inference engine

Pages in Study: 43

Candidate for Degree of Master of Science

Data generated by complex, computational models can provide highly accurate predictions of hydrological and hydrodynamic data in multiple dimensions. Unfortunately, however, for large data sets, running these models is often time-consuming and computationally expensive. Thus, finding a way to reduce the running time of these models, while still producing comparable results, is of notable interest.

The Inference Engine is a proposed system for doing just this. It takes previously generated model data and uses them to predict additional data. Its performance, both accuracy and running time, has been compared to the performance of the actual models, in increasingly difficult data prediction tasks, and it is able, with sufficient accuracy, to quickly predict unknown model data.

DEDICATION

To all those who have helped along the way.

ACKNOWLEDGEMENTS

I would like to express sincere thanks to the following individuals and organizations for their contributions to this research:

Bill McAnally

Phil Amburn

Sarab Prasad

Vladmir Alarcon

Department of Computer Science & Engineering

Northern Gulf Institute

Northern Gulf Coastal Hazards Collaboratory

National Science Foundation

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ADCIRC Model	A (Parallel) Advanced Circulation Model for Oceanic, Coastal, and Estuarine Waters
BBN	Bayesian Belief Network
EFDC	Model Environment Fluid Dynamics Code Model
m	meters
MARSplines	Multivariate Adaptive Regression Splines
mm	millimeters
SVR	Support Vector Regression

CHAPTER I

INTRODUCTION

Background

Throughout history, natural disasters have had devastating consequences. From the ancient volcanic eruption at Pompeii to the tsunamis and hurricanes of more recent memory, natural disasters are often unexpected and terrifyingly powerful. Thankfully, in the modern era, better predictive and analytical tools have been and are being developed that can track developing natural disasters and provide advanced warning to those who may be effected.

One of the uses of these predictive tools is, given a developing natural disaster, predicting the eventual extent and severity of the natural disaster. The tools which do this create a model of the selected system, usually based on some number of mathematical equations, and use it to predict how the system will behave. These tools are, unsurprisingly, referred to as models. Models are often large and complex and can generally generate information in multiple dimensions. Unfortunately, these models are often computationally expensive and take significant amounts of time to run. As an example, a given model might be expected to run for 10 days to generate a year's worth of data, according to Vladimir Alarcon, assistant research professor at the Geophysical Research Institute of Mississippi State University. Thus, it is desirable to find a way to produce similar results, using less time and computational resources. This, then, is what the Inference Engine is designed to do.

Objectives and Hypothesis

The goal of this research is to determine whether the proposed Inference Engine is effective at predicting hydrological data. If the Inference Engine does prove effective, this research shall also consider the limitations on the types of data that the Inference Engine can effectively predict. The hypothesis, then, for this research is that the Inference Engine will be able to effectively predict hydrological data, within certain limitations.

Scope

The computational procedure examined here has been developed to serve as one of a collection of model data analysis tools within Sulis, an informatics services system developed by the Northern Gulf Institute to support resource management assessments. Specifically, it will serve as a component of the Inference Engine, a program that evaluates user requests, fetches data, performs analyses, and generates results for the user. As a simplification, the proposed computational procedure discussed in this research shall be referred to as the Inference Engine, even though this computational procedure represents only a part of the full functionality of a completed Inference Engine.

To lower the costs of running computational models, the Inference Engine takes previously generated model output data and uses them to estimate model results at a different time or under different conditions. Estimating additional data in this way allows users to gain the benefit of additional data without having to resort to extra model runs.

However, for the Inference Engine to be successful, it must estimate data at or above a certain threshold of accuracy.

This paper, then, seeks to set forth a computational procedure, referred to as the Inference Engine, and determine if this computational procedure fulfills all the requirements for efficiently and accurately expanding geophysical model results. The first requirement for a successful Inference Engine is that using it must be significantly easier and quicker than running a comparable computational model. The Inference Engine must also be able to estimate results at an appropriate level of accuracy, or, otherwise, the entire exercise is pointless. Initially, this means ensuring that results are at least as effective as linear regression.

CHAPTER II

REDUCED ORDER MODELING

Numerical Models

Before analyzing the Inference Engine, it may be constructive to further examine the sorts of data some of these models are used to predict. It may also be helpful to consider how the data are structured and how each model goes about producing its predictions. In understanding this background information, a better vantage point may be reached from which to consider the problem.

First off, this research focuses primarily on hydrological and hydrodynamic models. Even within the realm of hydrological and hydrodynamic models, however, there are still more models than this paper can reasonably consider. Therefore, this research only addresses a small number of these models. Extrapolating results from a small selection of models allows this research to focus on examining and extending the capabilities of the Inference Engine, but also creates a risk that some hydrological models, or some subset of hydrological models, will be overlooked. Thus, there may be some types of models that the Inference Engine is ineffective at predicting, but those models which are tested provide a starting point for understanding how well data can be predicted outside of computational models.

The computational models that are examined in this research include the Environmental Fluid Dynamics Code (EFDC) Model and the ADCIRC Coastal

Simulation and Storm Model. The EFDC model is a publicly available modeling system that simulates hydrodynamics, water quality, and sediment concentrations. It is used to model rivers, lakes, coastal regions, wetlands, and estuaries, generating time series predictions in up to three spatial dimensions [1]. This paper makes use of data generated by the EFDC Model for the Mobile Bay region. These data include a large number of timesteps, each containing information on 1,758 different, irregularly-distributed x, y, z positions. Each timestep, of course, contains the same positions as all the other time steps. Each data point, in turn, contains data on the water depth, salinity, flow velocity (vector), and temperature, for a given position and timestep.

The ADCIRC model used here is a model that solves time-dependent problems in two dimensions, using a flexible, unstructured grid coordinate system [2]. ADCIRC is most notable for modeling near-shore marine circulation based on tides and wind patterns and storm surges and floods caused by hurricanes. Specifically, this paper makes use of data from Hurricane Ivan provided by the Louisiana State University. These data cover the whole of the eastern seaboard in an irregular grid. The data are composed of 92 timesteps and 1,090,530 positions. Each data point contains information on the water surface elevation, for the given timestep and position.

Reduced Order Modeling Methods

Generally, the design and use of methods for reducing the running costs of complex computational models is known as reduced order modeling. The purpose of model order reduction is to quickly capture the essential elements of a given system or model [18]. In most reduced order modeling techniques, this involves simplifying the underlying mathematical functions, normally by reducing the number of variables used to describe the system [15]. There are a broad variety of different methods that have been used to do this, but most research deals with automatically finding and implementing simplifications within the functions used to model the system [18]. These simplifications often use complex mathematical tools to find good approximations for the function. Examples include algorithms such as Asymptotic Waveform Evaluation, in which a Padé approximation is used to model the function and Laguerre Methods, in which Laguerre functions are used to calculate an approximation of the function [18].

While many reduced order modeling methods are based on finding solutions to the mathematical functions from the model itself, this does not necessarily have to be the case. In particular, it is possible to use information on the past behavior of the model to predict future model behavior. This methodology allows the Inference Engine to predict new information more quickly and easily than a model run alone.

There are a large number of tools specifically designed for predicting various types of data. Notable examples of these include LIBSVM [3] and MARSplines [5], both of which are used within the Inference Engine itself. There are, of course, many other examples, based on a variety of techniques, including neural networks, genetic algorithms, support vectors, or even combinations of these three methods [6, 7]. However, these are only prediction methods. They do not address the underlying question of what can be predicted, with accuracy, or the guidelines for insuring accurate predictions. The Inference Engine, on the other hand, is a system that takes prediction requests, evaluates them, and then predicts the requested hydrological data.

There are many other works that deal with related problems, some even in related ways. Generally, however, these are targeted at a very specific subject area or problem.

As an example, X. Wang et al. [8] used support vector regression (SVR) as part of an algorithm to predict water quality for the Weihe River in Shannxi Province, China. In another instance, Wu et al. [9] used SVR to predict travel time between several different points. There are fewer examples of MARSplines techniques being used for prediction, but they exist, as well. C. Wang and Liu [10] compared MARSplines to several other techniques in an attempt to examine the link between urban development and several other factors.

It is worth noting that quite a few other works refer to themselves as inference engines. Many of these deal with processing logical statements from semantic input [11, 12]. Others are designed to process fuzzy logic, for various applications [13]. In general, these papers have little to do with this research, other than the shared name. However, there are other inference engine papers that are very similar to this research. One paper by Bishop et al. [14] proposes a software system to solve a large number of probabilistic problems at once. This system, called VIBES, is based on Bayesian Belief Networks (BBNs) and uses a drawing tool to allow users to define a graph representing the BBN. VIBES provides prediction capabilities for a whole class of probabilistic models and, as such, can be used to model any problem within its domain. Unlike the Inference Engine, though, this inference engine is restricted to predicting and solving probabilistic data and models [14]. In contrast, the Inference Engine predicts data that can be modeled by non-probabilistic functions. So, while these two endeavors are very similar, they are targeted at fundamentally different problems.

However, the body of work that is most similar to the Inference Engine is, unsurprisingly, reduced order modeling. As mentioned previously, researchers in reduced order modeling take complex computational models and develop algorithms for finding nearly optimal solutions to them with dramatically reduced computational costs [15]. As an example of this, Larimore [16] proposes a reduced order modeling method and discusses a situation where his method was used to create a reduced-order model of stream-flow dynamics. In another example, Burkardt et al. [17] show how several reduced order modeling techniques can be used to generate low cost solutions to the Navier-Stokes system. These examples, while focused on the same task as this research, take something of a different approach. They seek to simplify the underlying problem by changing the model itself. The Inference Engine, on the other hand, seeks to simplify the problem by taking model results and using them to predict additional data.

CHAPTER III

INFERENCE ENGINE

Inference Engine Overview

In operation, the Inference Engine takes in some input data, generated by computational models, and uses them to predict additional data. The data used in the Inference Engine will have one dependent variable and one or more independent variables. The input data will be referred to, here, as the "training set," and the data which the Inference Engine is predicting will be referred to as the "test set." The training set and the test set will be very similar, with the only distinction being that the values for the dependent variable are not required to be included in the test set.

As to particulars, the Inference Engine makes use of support vector regression (SVR), Multivariate Adaptive Regression Splines (MARSplines), and linear regression. Linear regression is not intended for use in the final implementation of the Inference Engine; rather, it is used to provide a baseline comparison to ensure that SVR and MARSplines are effective prediction methods. If either method is routinely less accurate than linear regression, this is an indication that the method is less effective than desired.

SVR is a prediction technique that maps a vector of training values onto a higher dimensional space and then finds the best separating hyperplane for the mapping. This hyperplane is then used to create a function to model the training vector. The regression is found by minimizing the following function [19]:

$$\boldsymbol{\Phi}(\boldsymbol{w}, \boldsymbol{\varepsilon}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + \boldsymbol{C}\sum_i \boldsymbol{\varepsilon}_i^- + \boldsymbol{\varepsilon}_i^+ \tag{1}$$

$\boldsymbol{C}$ is the input cost parameter, and **ε** is the input slack parameter, or, in simpler terms, **ε** controls the amount of deviation that can occur without penalty, and $\boldsymbol{C}$ controls the fittedness of the resulting hyperplane. As such, $\boldsymbol{\varepsilon}_i^- + \boldsymbol{\varepsilon}_i^+$ represents the boundary at which there is no penalty for prediction error. Both $\boldsymbol{\varepsilon}$ and $\boldsymbol{C}$are set by the user in this implementation of SVR and can be optimized for better performance. $\boldsymbol{\Phi}(\boldsymbol{w}, \boldsymbol{\varepsilon})$ is the separating hyperplane, and $||\boldsymbol{w}||$ is the magnitude of the weight vector of the hyperplane. The weight vector of the hyperplane, in turn, is the normal vector of the hyperplane plus an intercept parameter. In essence, this function finds the separating hyperplane by minimizing the sum of the magnitude of the hyperplane's weight vector and C multiplied by the sum of all points less than $\boldsymbol{\varepsilon}_i^-$ and greater than $\boldsymbol{\varepsilon}_i^+$ [19].

In this paper, linear regression is approximated by using SVR with a linear kernel. The kernel is a function whose purpose is to allow operations to be performed in input dimensional space, as opposed to the space of the found hyperplane [19]. For SVR, this means that the kernel determines the complexity of the regression function. Normally, this implementation of SVR uses a Gaussian radial basis kernel function, as it is generally more accurate and can effectively predict even non-linear relationships. The linear kernel, in comparison, only finds linear relationships [3]. While this setup would not be optimal for repeated usage, it works well as an approximation for linear regression. Thus, SVR with a linear kernel is used to provide an estimate for linear regression that can be compared to standard SVR and MARSplines.

MARSplines is a technique for solving regression problems developed by Thomas Friedman [4]. MARSplines posits a relationship between a dependent variable and a set of independent variables (equation 2). To find this relationship, MARSplines produces a set of basis functions and coefficients, created from the input data. As an example, (t - x)+ and (x - t)+ is one set of basis functions. The variable t is the knot, or the boundary between the two basis functions, and the + sign indicates that only positive results are considered. When expanded to more than two basis functions, the values for the knot parameter are, of course, different for each boundary between basis functions. The equation for finding the regression with MARSplines is as follows [4]:

$$\boldsymbol{f(x) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(x)} \tag{2}$$

In this equation, $\boldsymbol{x}$ is the vector of independent variables, $\boldsymbol{\beta_0}$ is the intercept parameter, and $\boldsymbol{M}$ is the number of basis functions considered. The variable $\boldsymbol{h_m(x)}$ is the m-th basis function, for a given vector of input variables, and $\boldsymbol{\beta_m}$ is the weight applied to it. $\boldsymbol{f(x)}$ represents the value of the dependent variable, for a given vector of input variables. In essence, this equation states that the value of the dependent variable is equal to an intercept parameter plus the sum of the basis functions times their weight, for a given vector of independent variables. The resulting basis functions and coefficients form a model of the input data and allow MARSplines software to predict additional data. This implementation of MARSplines makes use of the earth package from the R programming language, produced by Stephen Milborrow [2].

MARSplines is also notable for its pruning abilities. The MARSplines software considers each of the basis functions it produces and evaluates whether the basis function makes a sizable contribution to model accuracy [2]. If the basis function is not significant, it is pruned. This capability helps MARSplines avoid producing over-fitted models and cuts down on prediction time.

It is important to note that the Inference Engine is not meant to replace the results of computational models; instead, it is meant to extend model results, so that more information can be produced from a given model run.

Procedure

To predict new information, the Inference Engine must be presented with a training set and a test set. The training set should contain the dependent variable and all the related independent variables; the test set, on the other hand, requires only the independent variables. Regardless of the method used, the Inference Engine will build a model based on the training set and use it to predict the dependent variable in the test set.

To ensure maximum accuracy, both the training set and the test set must be scaled before being used within the Inference Engine. This includes two components: each attribute in both data sets must be set to the same scale and that scale must be either from 0 to 1 or from -1 to 1. Each attribute must also have its median at the middle point of the chosen scale.

To test the accuracy of each of the prediction methods within the Inference Engine, the results from the test sets are compared to the values generated by the full model and used to calculate the mean squared error. It should be noted, however, that the Inference Engine will find the value for the mean squared error of the scaled data set. Thus, the scale of the data set should be considered when dealing with the accuracy results. For example, a mean squared error of .01 in a data set scaled from 0 to 1 would be equivalent to a mean squared error of .04 in a data set scaled from -1 to 1. This is because a value within the data set scaled from 0 to 1 is twice as large as the same value within the data set scaled from -1 to 1. Since the input data are scaled to these sizes and then squared to find the mean squared error, this means that mean squared errors within the -1 to 1 data set will be four times larger than mean squared errors within the 0 to 1 data set.

Finally, it is possible to increase the accuracy of SVR by finding the optimum kernel parameters. Finding these parameters, however, generally takes significantly longer than simply using support vector regression to solve the problem with the default parameters. Thus, for this paper, optimized SVR and unoptimized SVR will be considered separately.

Testing

The Inference Engine is targeted toward three use cases: interpolation within a data set, extrapolation within a data set, and extrapolation from multiple data sets (see Figures 1 and 2 for more information). Each of these cases naturally builds upon the previous case; if interpolation within a data set is inaccurate or implausible, it follows that extrapolation will most likely also be inaccurate or implausible. If either of the first two cases are not successful, it is likely that extrapolation between multiple data sets will also not be successful. Thus, this paper will examine each of these cases, in order, to determine if the Inference Engine can achieve appropriate levels of accuracy and efficiency for each of these use cases.

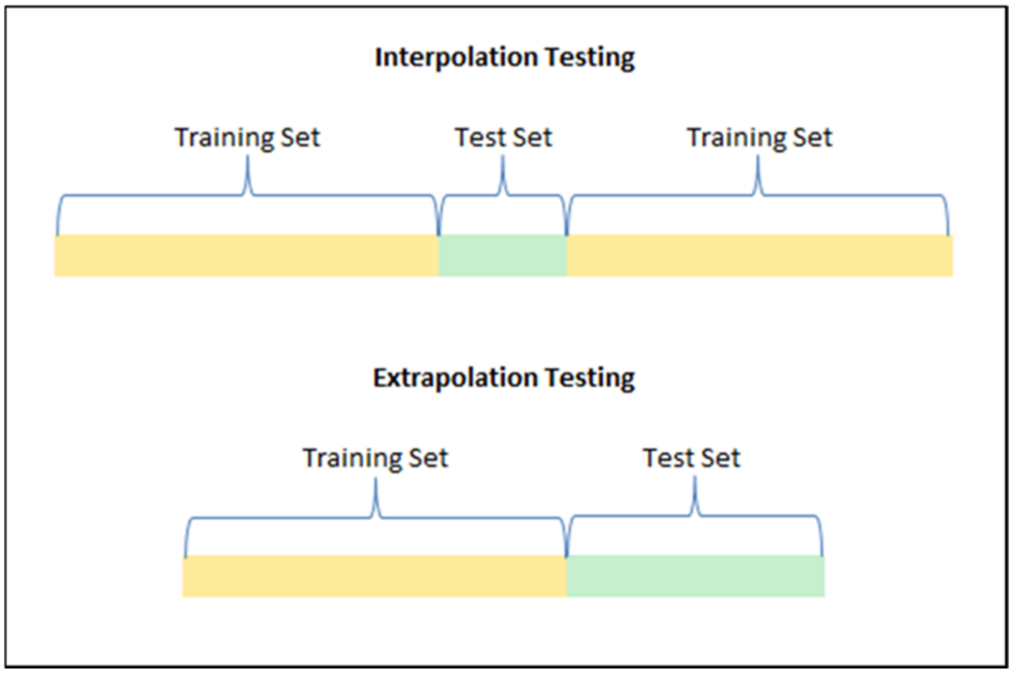


Figure 1 Comparison of Interpolation and Extrapolation Testing

Note that this figure is meant to impart the general concept and is not meant to be an exact representation of the size of the training sets or test sets.

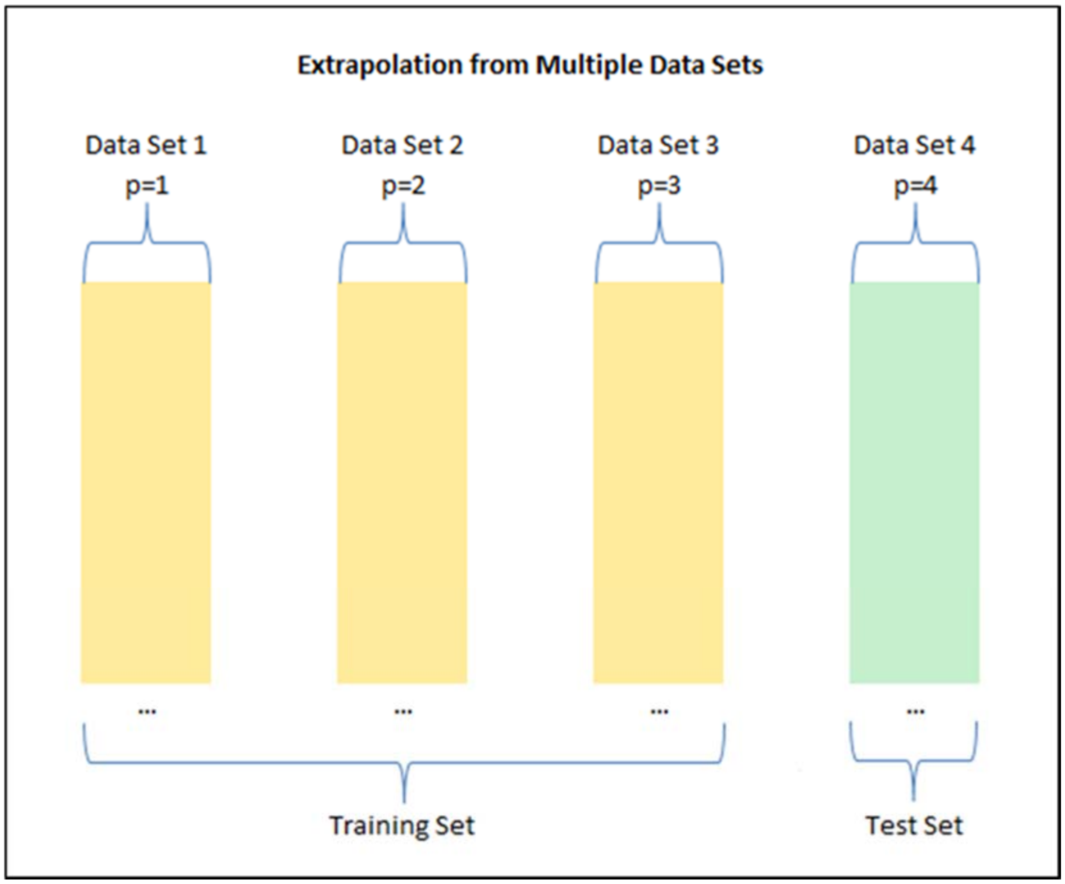


Figure 2 Graphic of Extrapolation from Multiple Data Sets

This represents an example scenario for extrapolation across multiple data sets, where p is the independent variable of interest.

Before considering these tests, it is important to understand how extrapolation between multiple data sets might be used to predict data. As an example, a client might be interested in the water level of a river or bay and might want to know the expected water level, given several possible magnitudes of rainfall. A model might be used to find the water level, at low and medium rainfall magnitudes, providing data that would then allow the Inference Engine to predict the water level at high rainfall levels. This use case is the most challenging, in terms of both the Inference Engine and testing. For this use case, the Inference Engine must predict a large amount of data with less information up front, as compared to interpolation. To test this use case, several suitable data sets must also be generated and then tested. As a result, this use case is the most difficult to test and use but is also the most beneficial of the three. Having this functionality would make the Inference Engine an exceptionally versatile system.

Note that, in the figures, test set size is measured in number of timesteps and training set size is also measured in number of timesteps. In Figure 5, the data are scaled from -1 to 1, and, in Figures 9 and 10, the data are scaled from 0 to 1. In all the other figures, the mean squared error is expressed in terms of the real units squared. The reason that some figures use scaled units is either because, in the case of figure 5, the data are only comparable when using the scaled units or, in the case of figures 9 and 10, the exact scale value was not available.

It is also worth mentioning that, for each test, the data sets used in the predictions are randomly selected from all the appropriate possibilities. As a result of this randomness, there are occasionally anomalous fluctuations in the figures, possibly causing peaks or depressions in otherwise well-behaved graphs. It is, of course, possible that such unusual variations do indeed reflect the average behavior for a given test or prediction, but, without running ever larger numbers of predictions, it is difficult to be sure. While these fluctuations may make it harder to see the overall trend of each graph, the relationships between the prediction methods remain clear. Thus, it remains reasonable to draw conclusions about the relative effectiveness of each prediction, regardless of any unusual fluctuations within the figures.

The first use case that should be considered in measuring the effectiveness of the Inference Engine is interpolation within a data set. The first series of tests makes use of an EFDC model run of Mobile Bay. These tests use two groups of timesteps to predict a certain number of additional timesteps of information. The predicted data, or test set, are surrounded on both sides by the training set. In the first example, the temperature at the bottom of Mobile Bay is being predicted, given the position and the timestep. Each data point in Figures 3 and 4 represents twenty sample results, each based on a randomly selected starting point within the data; however, each prediction method is tested with the same twenty samples as all the others. Notably, MARSplines and unoptimized SVR significantly outperform linear regression in both cases.

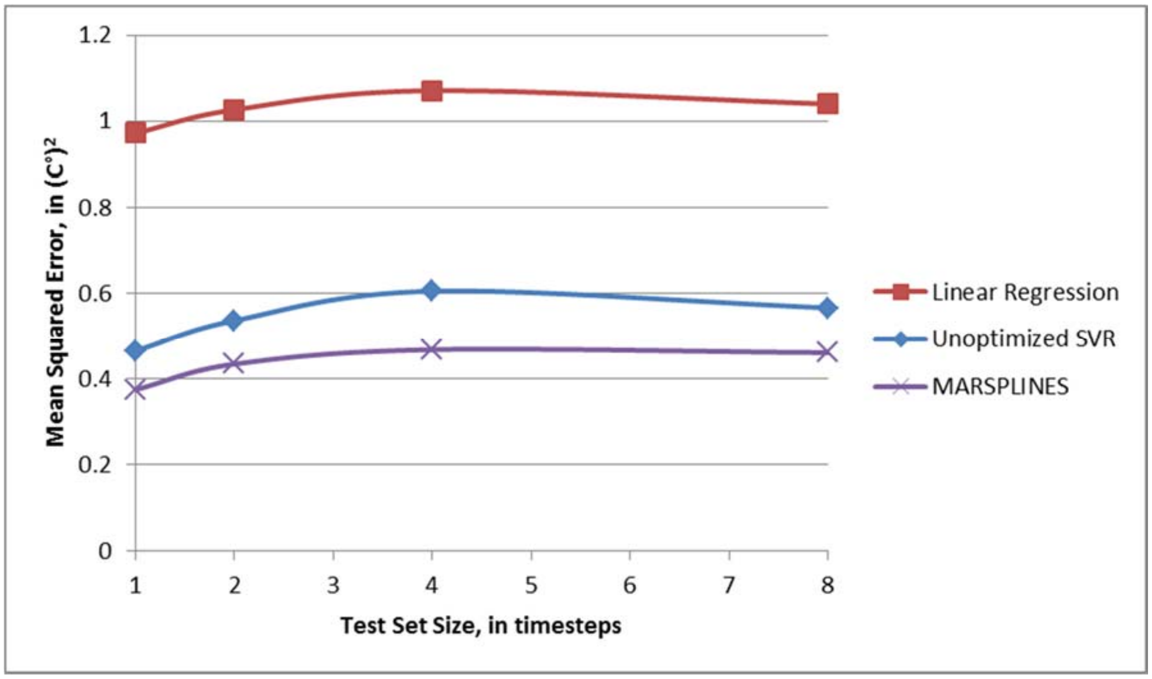


Figure 3 Mean Squared Error in Relation to Test Set Size, as Measured by Interpolation

This figure shows the mean squared error with respect to the size of the test set for the interpolation test. The variable being measured is the temperature at the bottom of Mobile Bay. Dependent variables are position (x, y, and z) and time step.

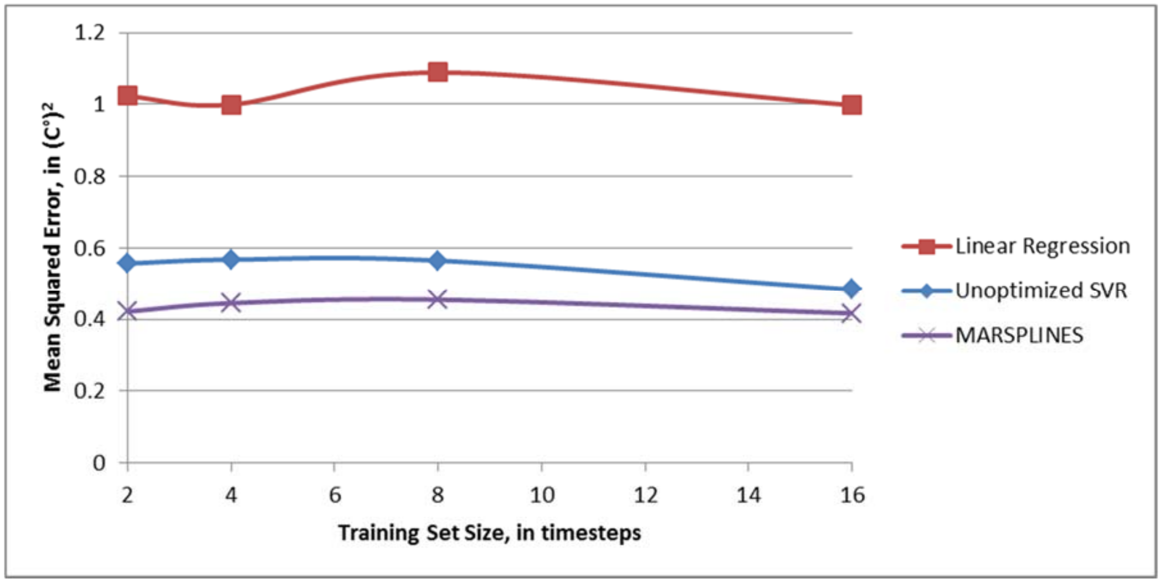


Figure 4 Mean Squared Error in Relation to Training Set Size, as Measured by Interpolation

This figure shows the mean squared error with respect to the size of the training set for the interpolation test. The variable being measured is the temperature at the bottom of Mobile Bay. Dependent variables are position (x, y, and z) and timestep.

Figure 5 illustrates the same test, but applied to each of the appropriate variables in the data set. In this case, each data point represents five different, randomly-selected samples. While the bottom temperature variable is among the variables that are predicted with the most accuracy, its predictability is by no means unique. As such, it is reasonable to conclude that accurate interpolation within a data set is entirely possible, at least for this sort of data.

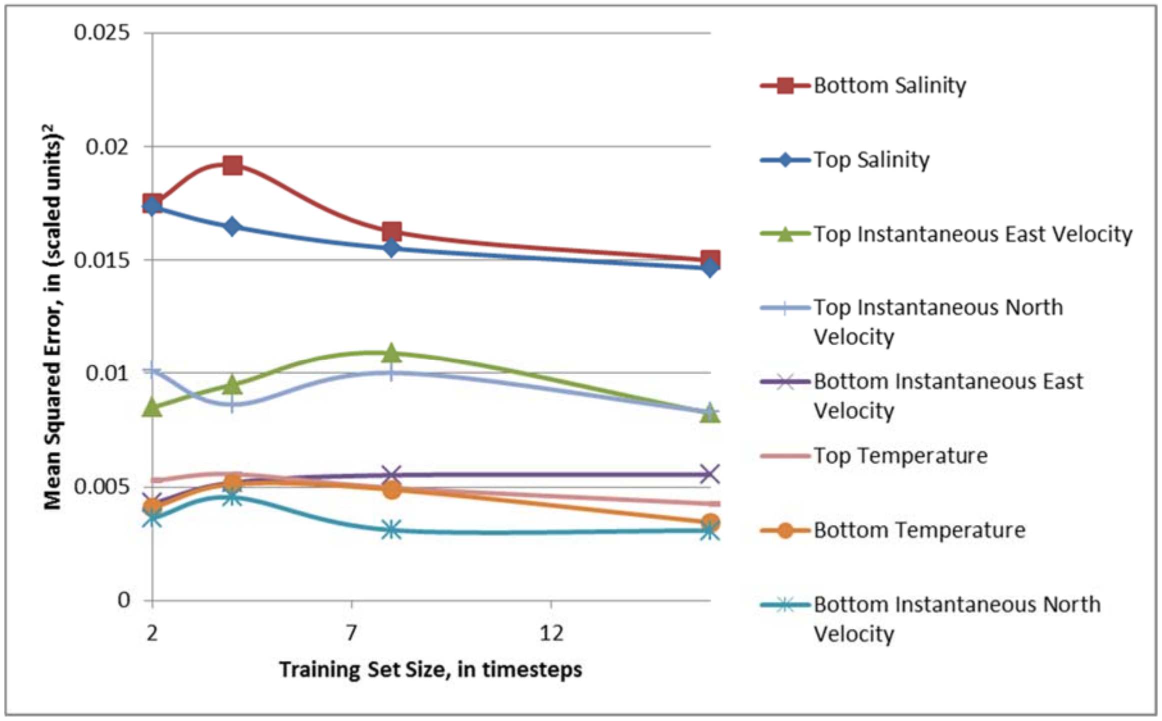


Figure 5 Mean Squared Error in Relation to Training Set Size for Several Different Variables, as Measured by Interpolation

This figure shows the mean squared error with respect to the size of the training set for the interpolation test. The variables being measured are the temperature, salinity, north/south flow velocity, and east/west flow velocity at both the surface and the bottom of Mobile Bay. Dependent variables are position (x, y, and z) and timestep.

The second use case that should be considered is extrapolation within a data set. These tests also make use of the same EFDC model run of Mobile Bay. As before, in Figures 6 and 7 the temperature at the bottom of Mobile Bay is predicted, and each data point represents twenty samples. These tests, in contrast to the previous tests, use several consecutive timesteps to predict a certain number of additional timesteps of information. Again, unoptimized SVR and MARSplines both significantly outperform linear regression.

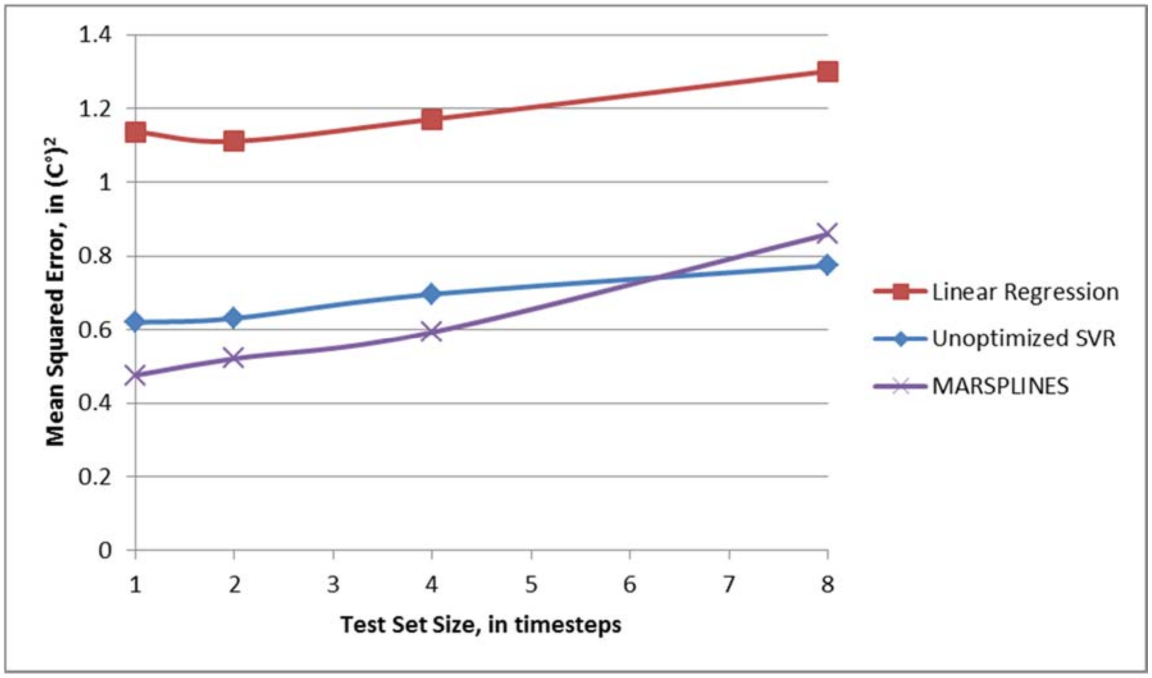


Figure 6 Mean Squared Error in Relation to Test Set Size, as Measured by Extrapolation

This figure shows the mean squared error with respect to the size of the test set for the extrapolation test. The variable being measured is the temperature at the bottom of Mobile Bay. Dependent variables are position (x, y, and z) and timestep.

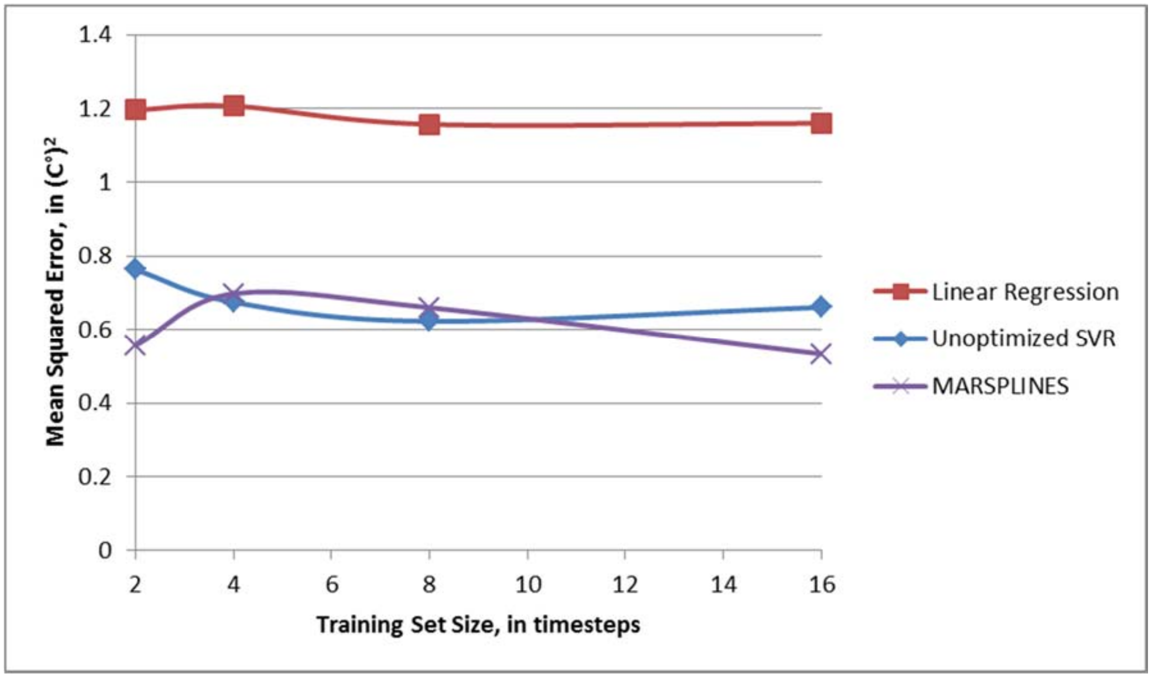


Figure 7 Mean Squared Error in Relation to Training Set Size, as Measured by Extrapolation

This figure shows the mean squared error with respect to the size of the training set for the extrapolation test. The variable being measured is the temperature at the bottom of Mobile Bay. Dependent variables are position (x, y, and z) and timestep.

This next test is based on the same principle, but uses the ADCIRC data for the east coast of the US. In this case, the water surface elevation is being predicted, given the position and timestep. Once more, each data point represents twenty samples. Figure 8 indicates that, for this data at least, SVR is much more stable at long-term predictions, while MARSplines is more effective for short-range predictions.

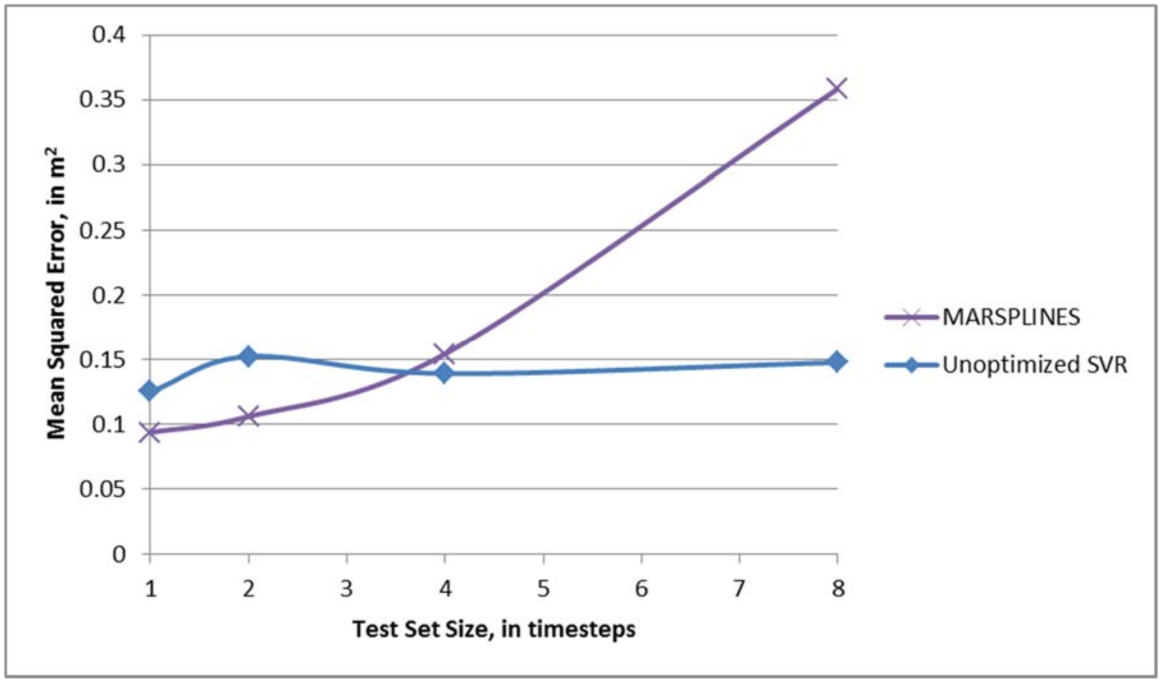


Figure 8 Mean Squared Error in Relation to Test Set Size for ADCIRC Data, as Measured by Extrapolation

This figure shows the mean squared error with respect to the size of the test set for the extrapolation test. The variable being measured is the water elevation on the eastern coast of the United States. Dependent variables are position (x, y, and z) and timestep.

The third and final use case is extrapolation between multiple data sets. For this test, a data set was specifically created, using the EFDC model, to be used as a trial. This data set models the flow of Fish River in Alabama, given rainfall and timestep. In addition to this, eight data points were produced, also using the EFDC model, containing information on discharge water depth vs. flow. A quadratic best-fit equation was generated for the new data and, using this equation, discharge depth was added into the main data set. This test takes the discharge water depth and uses it to predict the flow of Fish River. Since this relationship is much simpler than the relationships in previous tests, the results are notably different, as seen in Figures 9 and 10. First, linear regression and unoptimized SVR are much more competitive when predicting the Fish River data than in any of the other tests. The likely reason for this is that the Fish River data are much closer to a linear distribution than the other data sets and, so, linear regression is more effective. Second, MARSplines significantly outperforms both of the other prediction methods. This may be because MARSplines is a more effective predictor when given a simple, non-noisy quadratic distribution. Regardless, these data do indicate that accurate and efficient extrapolation between data sets is quite possible.

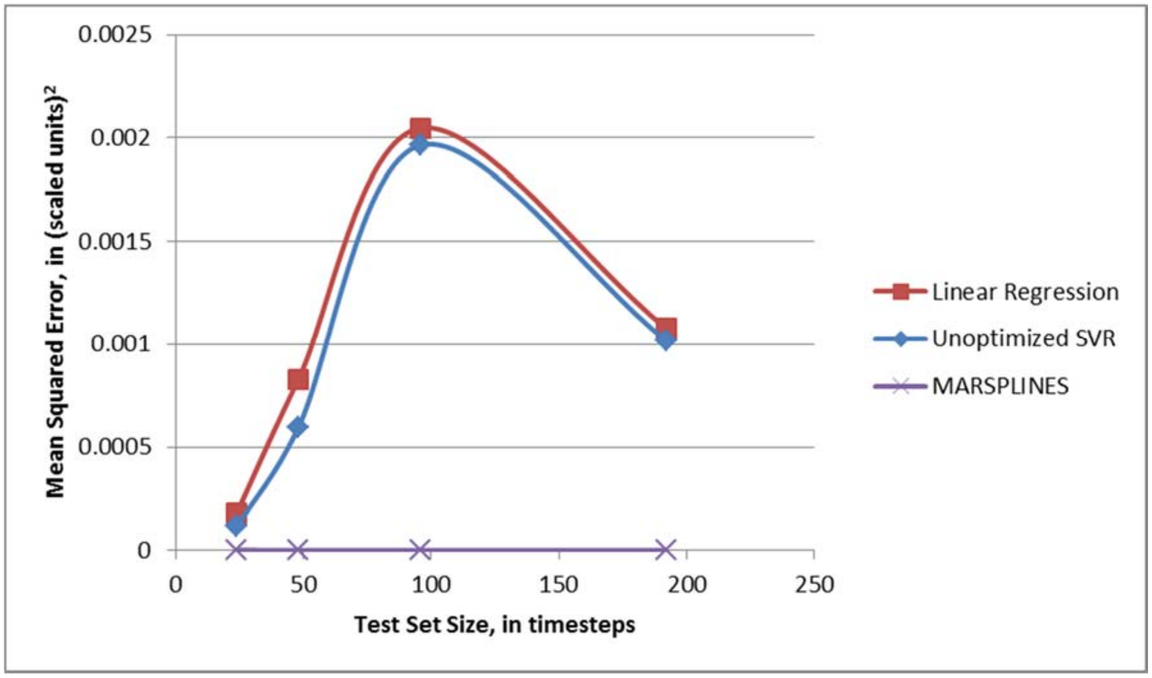


Figure 9 Mean Squared Error in Relation to Test Set Size, as Measured Using Extrapolation between Multiple Data Sets

This figure shows the mean squared error with respect to the size of the test set, based on extrapolation between multiple data sets. The variable being measured is the flow at a given point on the Fish River. The dependent variable is the depth of the Fish River.

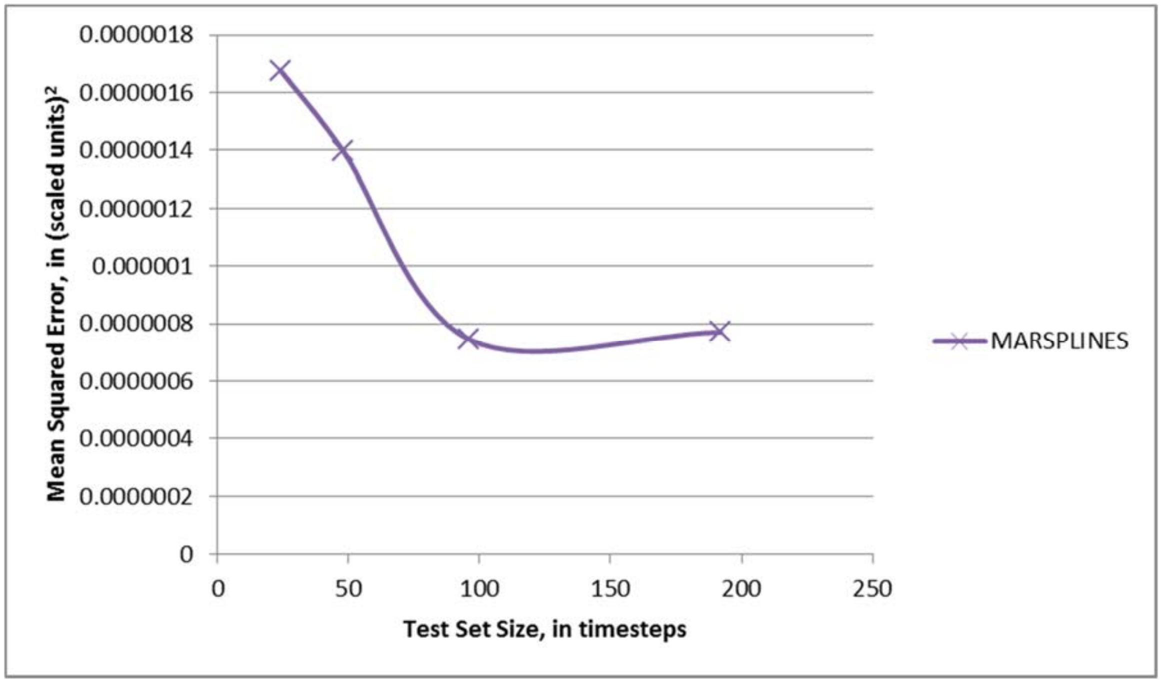


Figure 10 Detailed View of MARSplines Results from Extrapolation Between Multiple Data Sets

This figure shows the mean squared error with respect to the size of the test set, based on extrapolation between multiple data sets. The variable being measured is the flow at a given point on the Fish River. The dependent variable is the depth of the Fish River.

Finally, it should be mentioned that all of these predictions were generated very quickly. Instead of taking hours or days, this system was able to finish each of its predictions in less than three minutes, using a normal personal computer. Most predictions took even less time than that. And, while these tests are not necessarily equivalent to full model runs, they still represent a notable portion of one. To provide a specific example, it took slightly under two minutes, using SVR, to train a model on124,000 data points, based on the Mobile Bay EFDC data set. This is equivalent to between 70 and 71 timesteps worth of information. While the magnitude of this example prediction is likely smaller than most computational model runs, this does constitute a significant speed increase. Admittedly, in cases where the mathematical model is simple or the magnitude of the prediction is very large, this speed increase may become less significant. These, however, are special cases which can be easily be remedied by doing a normal model run. Thus, it seems reasonable to conclude that the Inference Engine represents an excellent method for quickly supplementing model results.

Improvements

Optimized Support Vector Regression

It is worth noting here that optimized SVR is far more effective than unoptimized SVR. Figure 11 is based on the same procedure as the interpolation test, except that each data point makes use of five samples, instead of twenty, and optimized SVR is included in the graph. This test shows that optimized regression does significantly better at prediction than each of the three other methods. Of course, this increase in accuracy is associated with a significant increase in the time and computational costs of the prediction. As an example, for a prediction that takes about a second to run normally, it took about two hours and fifteen minutes to find the optimized parameters needed for optimized SVR. After this, it then took about a second to run the prediction using the optimized parameters. It is likely that this increase in running time could be reduced through various methods, such as finding the optimized parameters for only a small subset of the data or by modifying the base algorithm itself. Either of these methods, though, could themselves represent another trade-off between time and accuracy. Optimized SVR, then, while possibly useful as a method of increasing prediction accuracy, also represents a significant increase in running time.

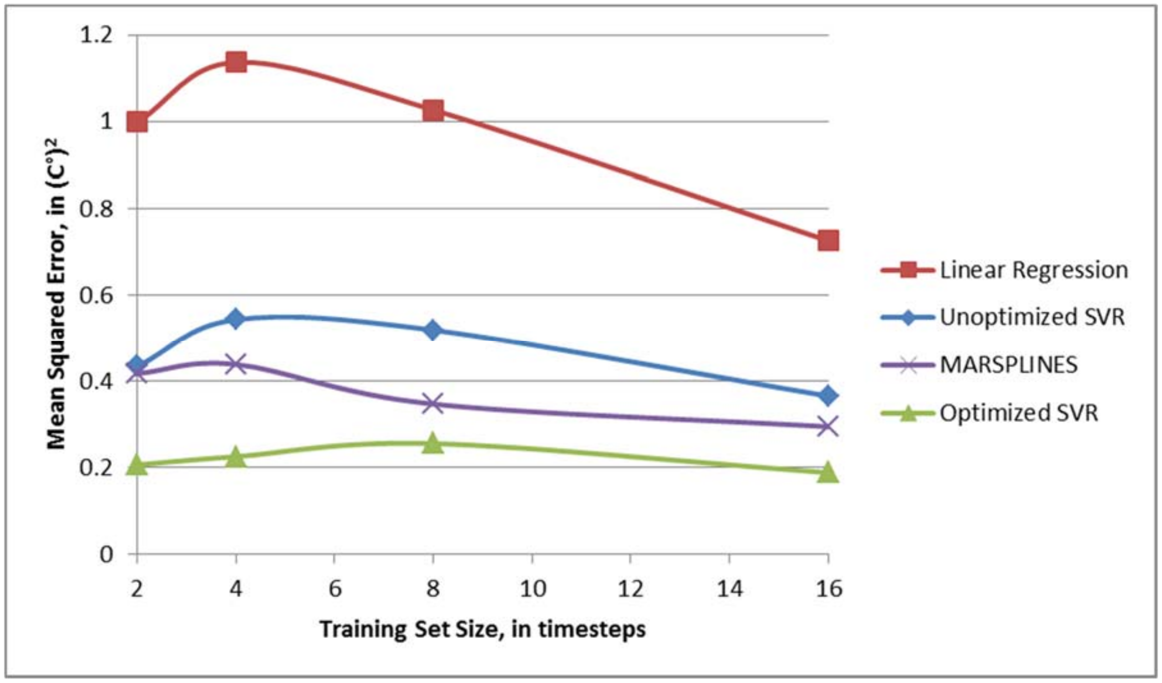


Figure 11 Interpolation Testing with Optimized SVR

This figure shows the mean squared error with respect to the size of the training set, based on interpolation. The variable being measured is the temperature at the bottom of Mobile Bay. The dependent variables are position (x, y, and z) and timestep.

Boundaries on Prediction

Much of the work in producing this Inference Engine is targeted not toward what the Inference Engine can effectively predict, but the cases and exceptions where the Inference Engine can not predict results accurately. In examining the limitations of the Inference Engine, it becomes easier to more clearly distinguish between problems the Inference Engine can solve effectively and problems it can not. This has a clear usefulness for producing and implementing the Inference Engine.

The first test dealing with the limitations of the Inference Engine is based on the interpolation tests previously discussed. However, instead of straight-up standard interpolation, this test introduces a gap between the training set and the test set (see

Figure 12). This gap is varied from zero to eight, and the mean squared error is calculated for each gap value. Each data point is based on twenty samples. Figure 13 shows that, as the size of the gap increases, the mean squared error also increases. As expected, the mean squared error rises slowly at first and becomes more notable as the gap becomes larger. This seems to indicate that SVR, at least, becomes more inaccurate as the temporal distance between the training set and the test set increases. While this may not be true for all types of data or in all cases, it is a useful consideration to make before attempting a prediction using the Inference Engine.

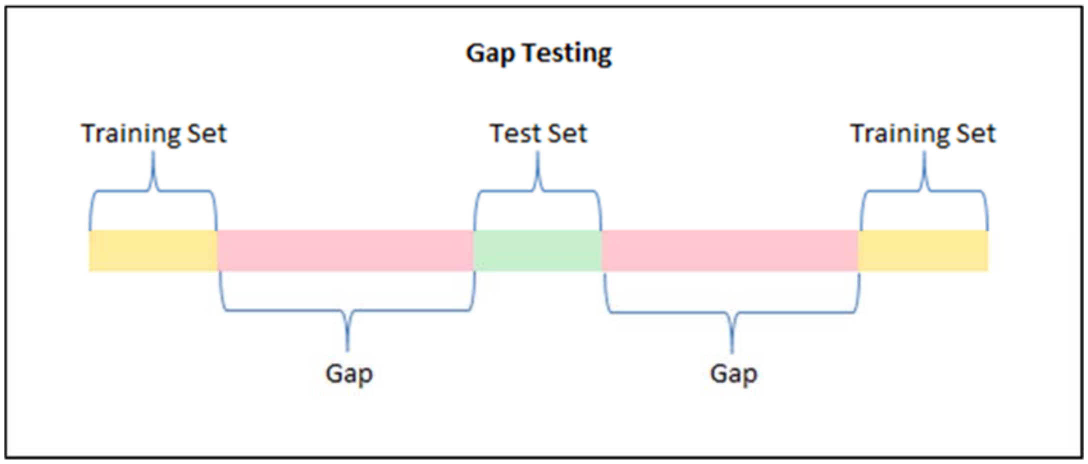


Figure 12 Graphic of Gap Testing

This figure illustrates the setup of the gap tests.

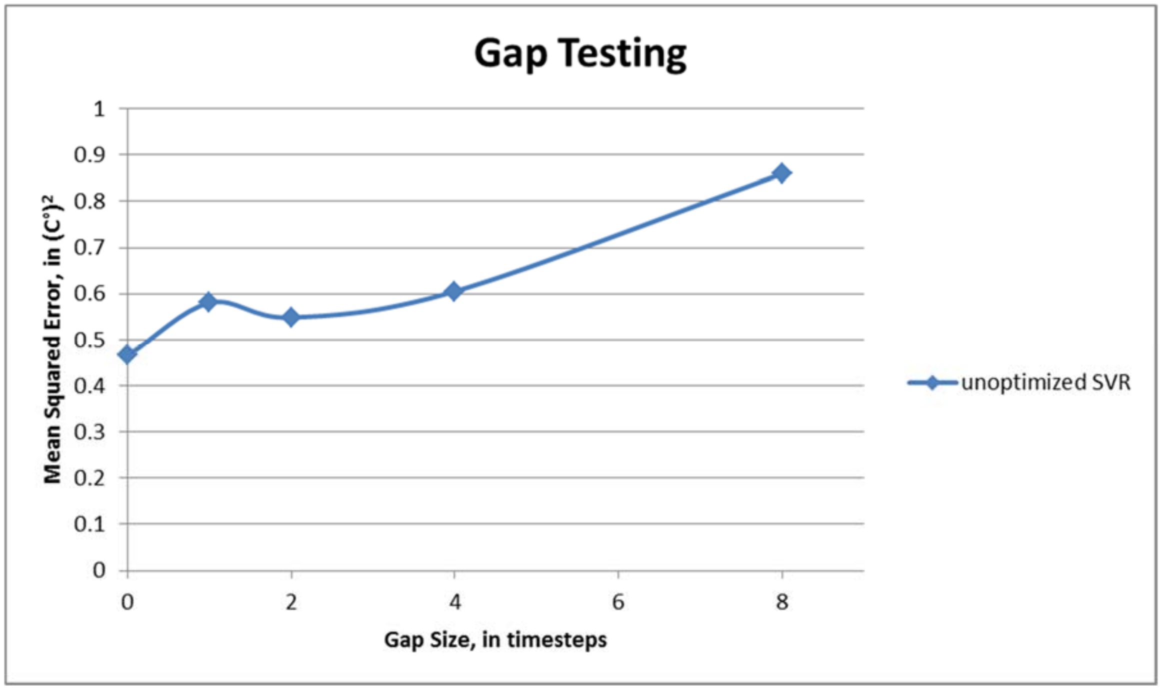


Figure 13 Mean Squared Error in Relation to Gap Size

This figure shows the mean squared error with respect to the size of the gap, based on gap testing. The variable being measured is the temperature at the bottom of Mobile Bay. The dependent variables are position (x, y, and z) and timestep.

There are several other factors that can decrease the accuracy of a prediction. These factors are not necessarily tied to a single, specific test; instead, they come from abandoned tests or concepts. First, it is necessary to choose an appropriate variable to predict. It would be difficult, if not impossible, to predict an x-coordinate, given any other variable or variables. This applies to many other variables, such as timestep and y-position, and different data sets may have more or less of these types of variables. It may be that there exists some way to perform such predictions, but the Inference Engine is not meant to be used for them.

Second, it is important to include all supporting variables for a given dependent variable. This may seem obvious, but it does have important ramifications for data prediction; independent variables with weak relationships, or no relationship, to the corresponding dependent variable will not produce accurate results. Further, the effectiveness of the Inference Engine and its prediction methods decreases as the amount of noise in the data increases. Noise, in this case, refers to variation in the dependent variable that is not accounted for by the independent variables. In many natural systems, noise would likely be present regardless of the variables used in the prediction, but, in the domain of model results, significant amounts of noise should only occur when variables used as input in the model itself are omitted. If the omitted variable or variables play a significant role in determining the independent variable, the inaccuracy will, unsurprisingly, be correspondingly greater. This is to be expected, and, unless important independent variables are left out of the input data, the effect on the prediction accuracy should not be devastating.

However, it is worth noting that increasing the number of independent variables and information in the training and test sets may not necessarily increase prediction accuracy. To test this, two different data sets were generated, using the EFDC Model. Both data sets use flow at a single point on the Fish River in Alabama as a dependent variable. Precipitation and timestep are both used as independent variables. One data set uses precipitation information for every hour and flow information for every day. The other, simpler data set sums up the hourly precipitation events and represents them all as a single, daily precipitation event. Figure 14 shows the results for the simplified data, and Figure 15 shows the results for the complex data. The data are separated by rainfall level, with 1, 2, and 3 representing very low, low, and high rainfall data, respectively. Figures 16 and 17 compare the accuracy of the simplified data with the accuracy of the more complex data. Each data point represents five samples and, as previously, each prediction method is tested with the same samples. The same samples are also used in both the simple and detailed data.

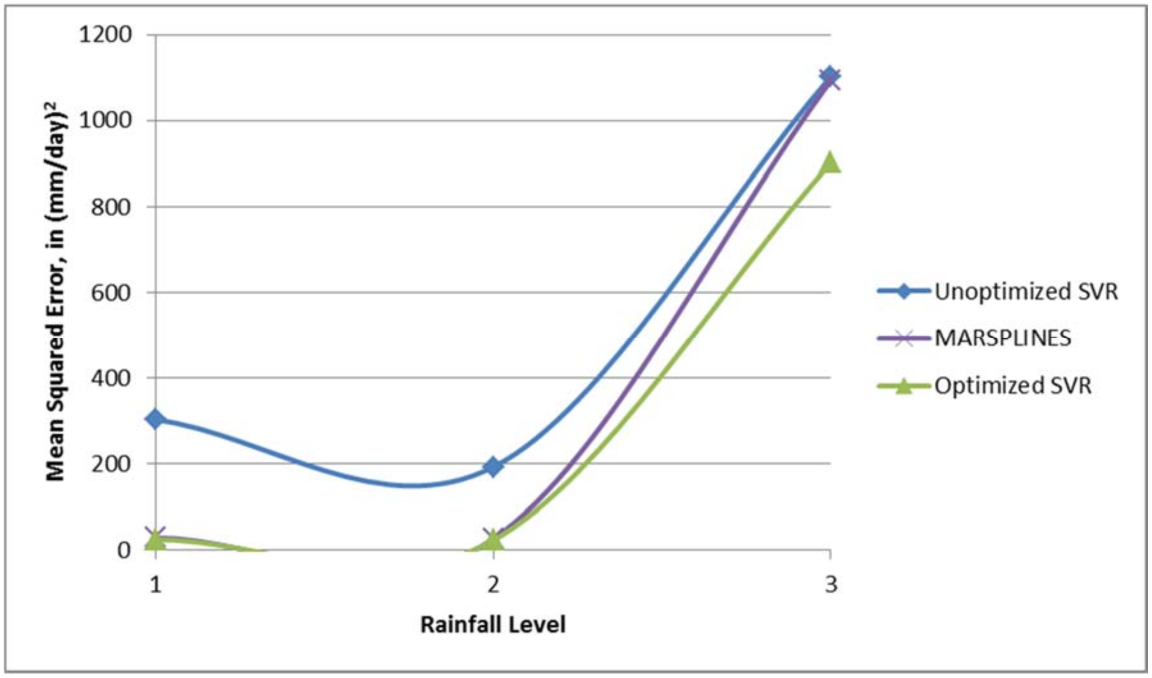


Figure 14 Simple Data Test

This figure shows the mean squared error with respect to the rainfall level, generated with extrapolation testing. The variable being measured is the flow at a certain point on the Fish River. The dependent variables are precipitation and timestep. These results are based on a simplified version of the data used in Figure 15.

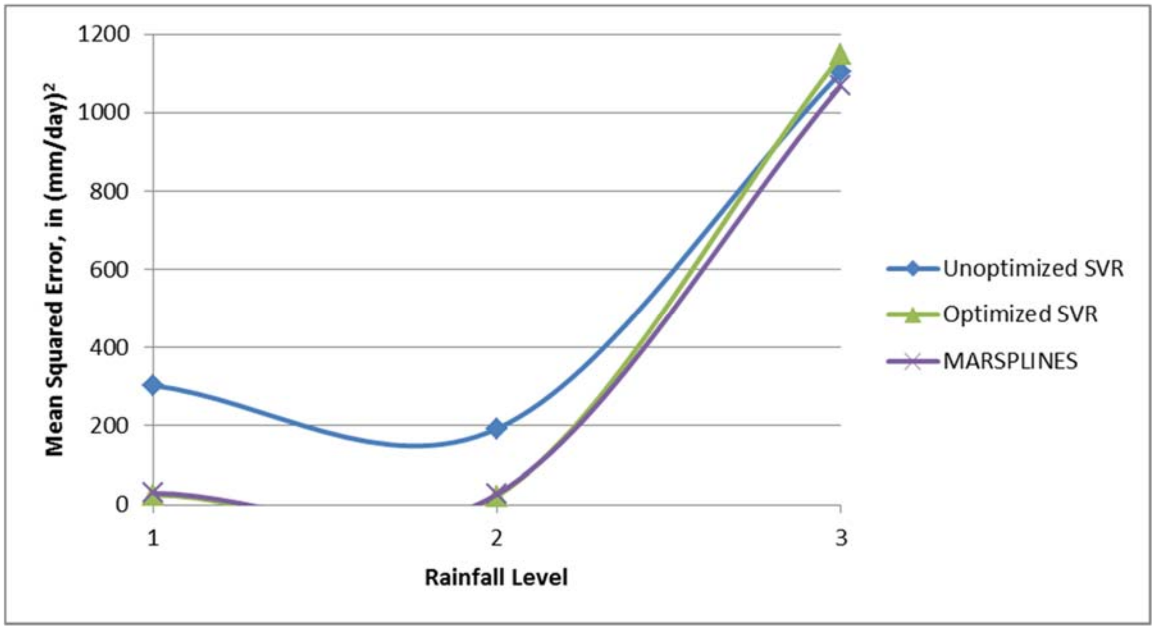


Figure 15 Detailed Data Test

This figure shows the mean squared error with respect to the rainfall level, generated with extrapolation testing. The variable being measured is the flow at a certain point on the Fish River. The dependent variables are precipitation and timestep. These results are based on the full, more detailed version of the data used in Figure 14.

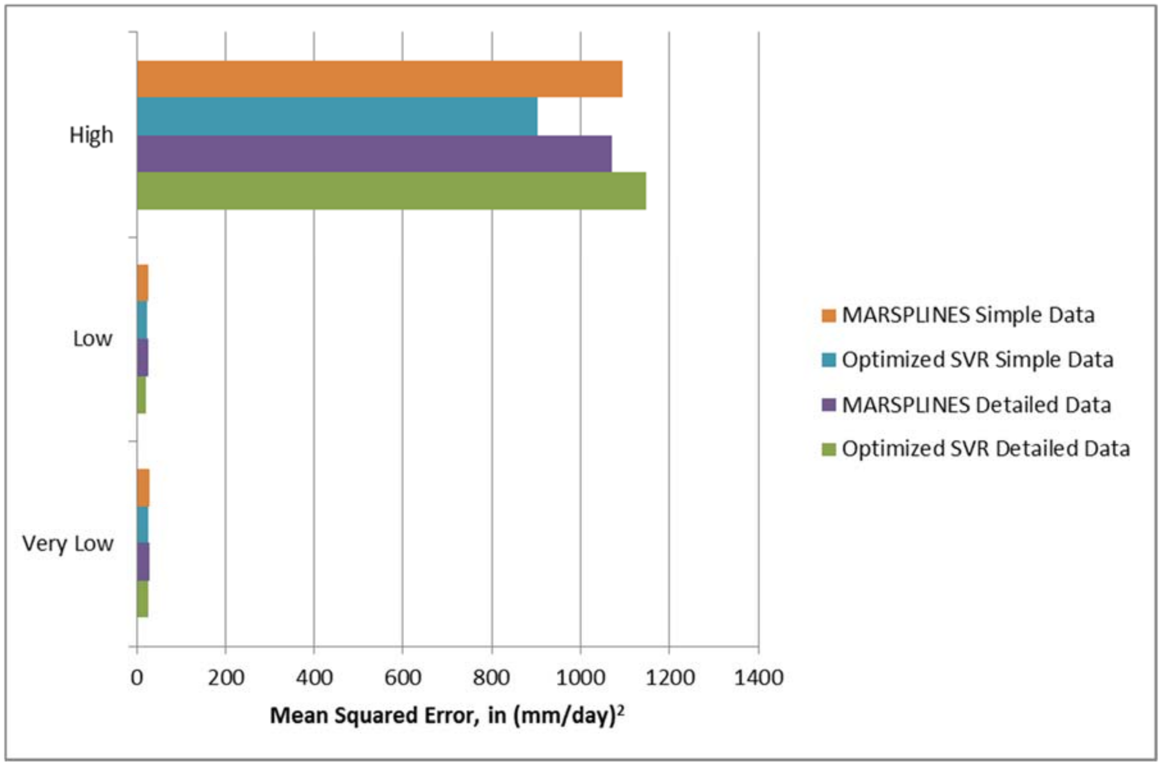


Figure 16 Comparison of Simple and Detailed Results

This figure compares the mean squared error of the simple data tests and the detailed data tests, from Figures 14 and 15, respectively.

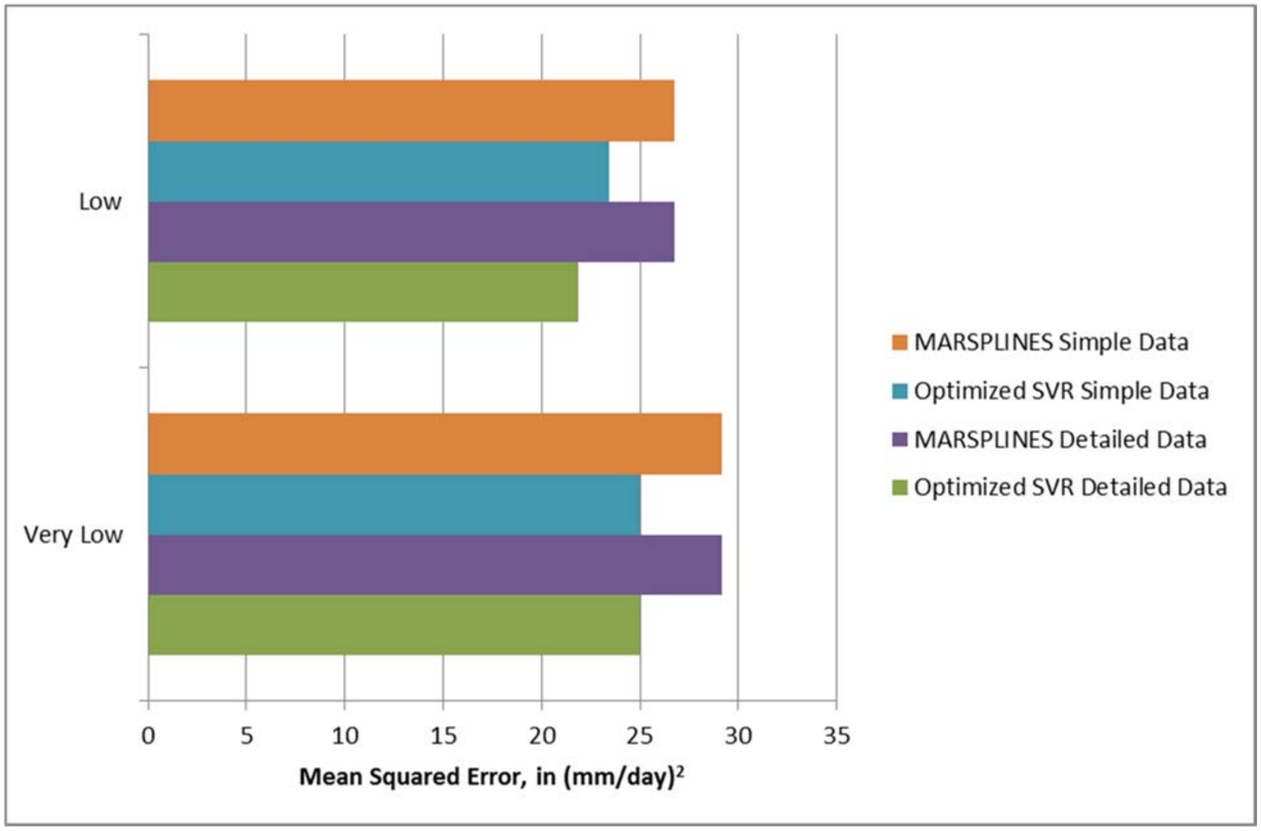


Figure 17 Comparison of Simple and Detailed Results, Magnified View

This figure compares the mean squared error of the simple data tests and the detailed data tests, from Figures 14 and 15, respectively. Note that only the very low and low rainfall results are listed here.

Examining the results, it is clear that the very low and low rainfall events have significantly less error than the high rainfall events. This is to be expected, since high rainfall events tend to generate significant flow variability. Also, both MARSplines and optimized SVR generally out-perform unoptimized SVR. Looking at Figures 16 and 17, which allow a more direct comparison of the results, it is easy to see that neither the predications made with simple data nor the predictions made with detailed data are significantly more accurate than the other.

This highlights the previous point: even if one data set provides more information than another, otherwise similar data set, there will not necessarily be a difference in prediction accuracy, unless the additional information significantly affects the dependent variable. In some cases, this may seem counter-intuitive, especially if the additional information seems like it should affect the dependent variable. In this example, it seems like more detailed precipitation data should allow for more accurate flow predictions, but testing shows that it makes little difference. Determining the ideal amount of information to include in a prediction is an important, and not necessarily trivial, task.

CHAPTER IV

CONCLUSION

Based on the results from the previously listed tests, the Inference Engine has been shown to be a suitable method for predicting hydrological data from the EFDC and ADCIRC models. This methodology does have some significant drawbacks as compared to simply running the various models, such as its reliance on previous model data and its decreasing accuracy over long stretches of time. However, these drawbacks are not enough to counter-act its usefulness as a means of quickly and accurately expanding model results. While the Inference Engine may not necessarily be the best choice for all data prediction tasks, as a means of expanding hydrological and hydrodynamic model data, it performs rather well.

Further, this research may be applicable to other fields besides hydrology and hydrodynamics. It stands to reason that, if the Inference Engine can accurately predict data for hydrological models, it may also work for other types of models. Thus, this concept should be tested in other areas where data prediction is an important topic.

The prediction accuracy during the tests of all three use cases indicates that the Inference Engine will be able to effectively predict hydrological data based on the EFDC and ADCIRC models. Throughout these tests, both prediction methods have been shown to be more effective than linear regression. Both have been tested in each of the three use cases and, in each case, effectively predicted the appropriate information. Thus, the hypothesis for this research is, indeed, true, as the Inference Engine is thoroughly capable of accurately predicting hydrological data. While there are certainly situations or data sets to which the Inference Engine is not applicable, this still leaves a broad range of possible applications. As such, the Inference Engine provides an excellent basis for extending the results of complex models, allowing quick, accurate prediction of hydrological data and, possibly, other data.

REFERENCES

[1] L. Feng, L. Ruijie, and H. Peng, “3D Hydrodynamic and Water Quality Simulation of North Jiangsu offshore Sea Based on EFDC,” *The 2nd International Conference on Bioinformatics and Biomedical Engineering (ICBBE 2008)*, May 2008, pp. 2878-2881.

[2] R. A. Luettich, J. J. Westerink, and N. W. Scheffner, “ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts, and Estuaries,” US Army Corps of Engineers, Washington DC, 1992.

[3] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, “A Practical Guide to Support Vector Classification,” National Taiwan University, Taiwan, 2010.

[4] T. Hill and P. Lewicki, *Statistics: Methods and Applications: a Comprehensive Reference for Science, Industry, and Data Mining*, Statsoft, Inc., Tulsa, Oklahoma, 2006, pp.327-334.

[5] S. Milborrow, “earth: Multivariate Adaptive Regression Spline Models,” R package version 3.2-3, http://cran.r-project.org/web/packages/earth/earth.pdf, (accessed on July 25, 2012).

[6] A. J. Cannon, “Neural Networks for Probabilistic Environmental Prediction; Conditional Density Network Creation and Evaluation (CaDENCE) in R,” *Computers and Geosciences*, vol. 41, April 2012, pp. 126-135.

[7] K.-Y. Chen and C.-H. Wang, “Support Vector Regression with Genetic Algorithms in Forecasting Tourism Demand,” *Tourism Management*, vol. 28, issue 1, Feb. 2007, pp. 215-216.

[8] X. Wang, L. Ma, and X. Wang, “Apply Semi-Supervised Support Vector Regression for Remote Sensing Water Quality Retrieving*,” IEEE International Geoscience and Remote Sensing Symposium (IGARSS) 2010*, Jul. 2010, pp. 2757-2760.

[9] C.-H. Wu, J.-M. Ho, and D. T. Lee, “Travel Time Prediction with Support Vector Regression,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, Dec. 2004, pp. 276-281.

[10] C. Wang and P.-S. Liu, “Data Mining and Hotspot Detection in an Urban Development Project,” *Journal of Data Science*, vol. 6, no. 3, Jul. 2008, pp. 389-414.

[11] V. Haarslev and R. Möller, “Racer: A Core Inference Engine for the Semantic Web,” *Proceedings of the Second International Workshop on Evaluation of Ontology-based Tools*, Oct. 2003, pp. 27-36.

[12] Y. Zou, T. Finin, and H. Chen, “F-OWL: An Inference Engine for the Semantic Web,” *Formal Approaches to Agent-Based Systems*, vol. 3228, Springer-Verlag, Berlin, Germany, 2005, pp. 238-248.

[13] T. Yamakawa, “A Fuzzy Inference Engine in Nonlinear Analog Mode and its Application to a Fuzzy Logic Control,” *IEEE Transactions on Neural Networks*, vol. 4, issue 3, May 2003, pp. 496-522.

[14] C. M. Bishop, D. Spiegelhalter, and J. Winn, “VIBES: A Variational Inference Engine for Bayesian Networks,” *Advances in Neural Information Processing Systems*, vol. 15, Jan. 2002, pp. 793-800.

[15] Z. Bai, P. M. Dewilde, and R. W. Freund, “Reduced Order Modeling,” *Handbook of Numerical Analysis Vol. XIII, Numerical Methods in Electromagnetics*, W. H. A. Schilders and E. J. W. ter Maten, eds., 2005, pp. 825-895.

[16] W. E. Larimore, “System Identification, Reduced-order Filtering and Modeling via Canonical Variate Analysis,” *American Control Conference 1983*, San Francisco, California, Jun. 1983, pp. 445-451.

[17] J. Burkardt, M. Gunzburger, and H.-C. Lee, “POD and CVT-based reduced-order modeling of Navier–Stokes flows*,” Computer Methods in Applied Mechanics and Engineering*, vol. 196, issues 1-3, Dec. 2006, pp. 337-3

[18] W.H.A. Schilders, “Introduction to Model Order Reduction,” *Model Order Reduction: Theory, Research Aspects and Applications*, W.H.A. Schilders, H.A. van der Vorst, J. Rommes, eds., Springer, Berlin, 2008, pp. 3-32.

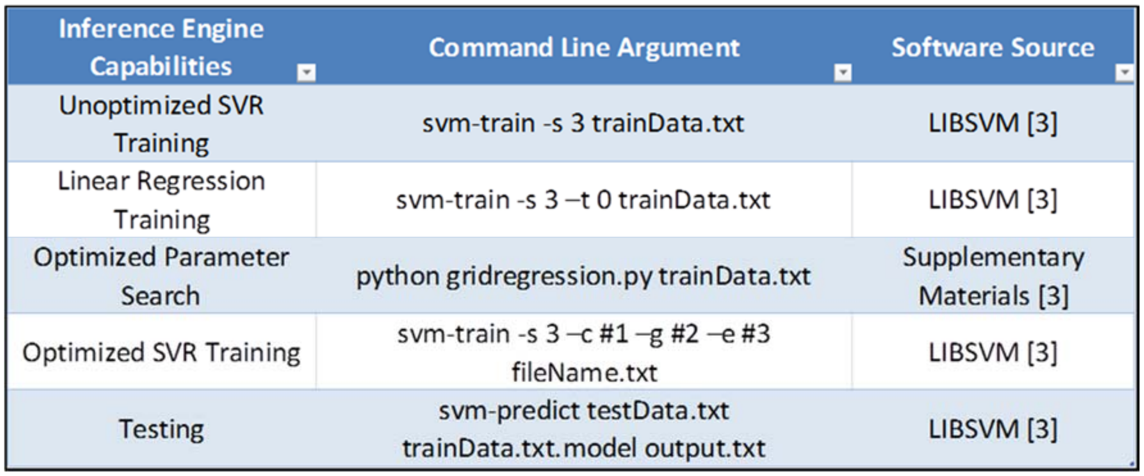
[19] S.R. Gunn, S*upport Vector Machines for Classification and Regression*, technical report, University of Southampton, Southampton, 1998.

APPENDIX A

INFERENCE ENGINE COMMANDS

It seems appropriate to provide some additional information about the software and commands used to run the Inference Engine. Table 1 includes information about SVR, linear regression, and optimized regression that may help future users reproduce and further examine the results produced in this paper. For each of these prediction methods, first a model file is generated that is specific to the prediction type. Then, regardless of which prediction function was used, the command used for testing remains the same. In optimized SVR, improved kernel parameters must first be generated and then used in training before testing can begin.

Table 1 Inference Engine Software and Commands



Inference Engine Capabilities	Command Line Argument	Software Source
Unoptimized SVR Training	svm-train -s 3 trainData.txt	LIBSVM [3]
Linear Regression Training	svm-train -s 3 –t 0 trainData.txt	LIBSVM [3]
Optimized Parameter Search	python gridregression.py trainData.txt	Supplementary Materials [3]
Optimized SVR Training	svm-train -s 3 –c #1 –g #2 –e #3 fileName.txt	LIBSVM [3]
Testing	svm-predict testData.txt trainData.txt.model output.txt	LIBSVM [3]

This figure shows the software and commands for several Inference Engine components. Note that MARSplines is not included in this list.

For MARSplines, a similar process is also used. However, instead of being based on command line arguments, MARSplines is used within the R programming language [5]. As such, I wrote a script that would allow me to use MARSplines to create a model and then predict data. Specifically, this script trains a model using trainData.txt, uses the model to predict the data in testData.txt, and then returns the mean squared error. This script must be slightly modified when predicting different variables, as it builds a model based on the name of the variable included in the data file. Currently, it is being used to predict the temperature at the bottom of Mobile Bay. The script is as follows:

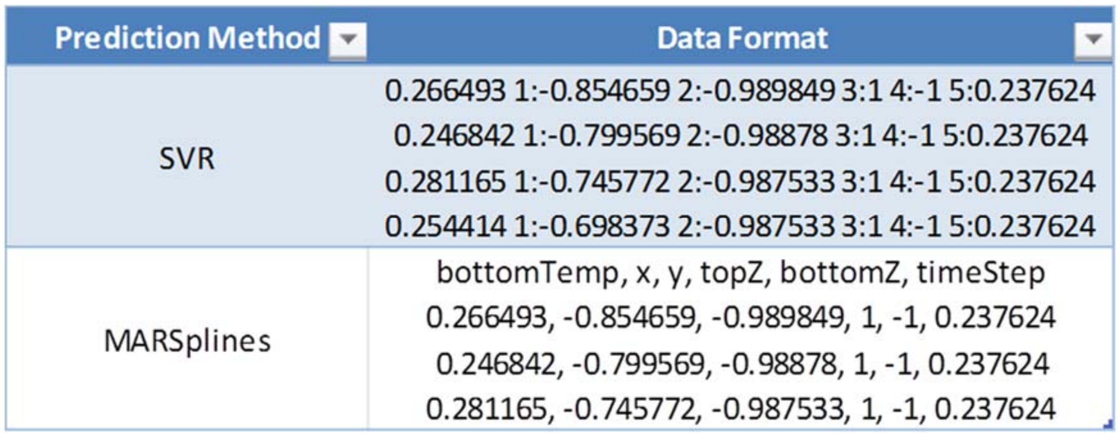
```
#This code uses the R earth class to predict bottom temperature, given a
trainData.txt and a testData.txt at the given file path.
#Based off code from http://cran.r-
project.org/web/packages/earth/vignettes/earth-notes.pdf , page 50.

library("earth")

trainData <- read.table("C: /trainData.txt", header=TRUE, sep=",")
testData <- read.table("C: /testData.txt", header=TRUE, sep=",")


train.subset <- trainData
test.subset <- testData
fit <- earth(bottomTemp ~ ., data = trainData)
yhat <- predict(fit, newdata = testData)
y <- testData$bottomTemp

rsquared <- (1 - sum((y - yhat)^2) / sum((y - mean(y))^2)) # find R-
Squared
vary <- var(y)
print((1 - rsquared) * vary) #print mean squared error, formula source:
http://www.duke.edu/~rnau/rsquared.htm
write(((1 - rsquared) * vary), "C: /results.txt")
```

APPENDIX B

Inference Engine Format Requirements

Both MARSplines and SVR require their own special data formatting before they can be used to predict data. In the case of MARSplines, this data format is fairly simple. The first line of a MARSplines-formatted data set must begin with the names of each of the variables, separated by commas. Variable names, unsurprisingly, are not allowed to have commas and must each be unique. Data is listed on each following line, with each variable separated by commas and listed in the same location as its variable name on the first line. For SVR, the data formatting is slightly different. In SVR, there is no line which names the variables being used in the prediction. Instead, the first variable is always the one being predicted. Each later variable is numbered consecutively with a number and a colon until the end of the line. Linear regression, unoptimized SVR, and optimized SVR all use this data format. Table 2 provides some examples of both types of data formats.

Table 2 Inference Engine Data Formats



Prediction Method	Data Format
SVR	0.266493 1:-0.854659 2:-0.989849 3:1 4:-1 5:0.237624 0.246842 1:-0.799569 2:-0.98878 3:1 4:-1 5:0.237624 0.281165 1:-0.745772 2:-0.987533 3:1 4:-1 5:0.237624 0.254414 1:-0.698373 2:-0.987533 3:1 4:-1 5:0.237624
MARSplines	bottomTemp, x, y, topZ, bottomZ, timeStep 0.266493, -0.854659, -0.989849, 1, -1, 0.237624 0.246842, -0.799569, -0.98878, 1, -1, 0.237624 0.281165, -0.745772, -0.987533, 1, -1, 0.237624

This figure shows the data formats for both SVR and MARSplines. This data is taken from a test predicting the temperature at the bottom of Mobile Bay, based on timestep and position (x, y, and z). Note that linear regression, unoptimized SVR, and optimized SVR each use the SVR data format.