LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TR-224851

Regression Strategies for Parameter Space Exploration: A Case Study in Semicoarsening Multigrid and R

Benjamin C. Lee, Martin Schulz, Bronis R. de Supinski

September 29, 2006

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Regression Strategies for Parameter Space Exploration: A Case Study in Semicoarsening Multigrid and R

Benjamin C. Lee, Martin Schulz, Bronis R. de Supinski
bclee@deas.harvard.edu, {schulzm, bronis}@llnl.gov

Report No. UCRL-TR-224851

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

Division of Engineering and Applied Sciences
Harvard University

Abstract

Increasing system and algorithmic complexity, combined with a growing number of tuanble application parameters, pose significant challenges for analytical performance modeling. This report outlines a series of robust techniques that enable efficient parameter space exploration based on empirical statistical modeling. In particular, this report applies statistical techniques such as clustering, association, correlation analyses to understand the parameter space better. Results from these statistical techniques guide the construction of piecewise polynomial regression models. Residual and significance tests ensure the resulting model is unbiased and efficient We demonstrate these techniques in R, a statistical computing environment, for predicting the performance of semicoarsening multigrid. 50 and 75 percent of predictions achieve error rates of 5.5 and 10.0 percent or less, respectively.

1 Introduction

Parameter space exploration often requires constraining the size or granularity of the space before measuring the metric of interest for every point in the space. These constraints are most often imposed when per point measurements are computationally expensive. However, pruning the search space according to particular criteria prior to a study risks conclusions that simply reinforce the prior criteria and may not generalize to a broader space. Similarly, reducing granularity or resolution risks obscuring yet unknown trends.

In contrast, techniques in statistical inference enable a scalable exploration framework, modestly reducing detail for substantial gains in speed and tractability. This trade-off enables the analysis of large parameter spaces without constraint. Even for applications in which obtaining extensive measurement data is feasible, efficient analysis of this data often lends itself to statistical modeling. In particular, inference models are constructed from measurements of sparsely sampled points from the parameter space. This sparse sampling decouples the number of measurements from the size of the parameter space, thereby allowing arbitrarily large increases in the number and granularity of parameters considered.

Regression is particularly cost effective, modeling a response as a weighted sum of predictors. Models are formulated from measured points by solving a linear system in least squares (via Cholesky or QR decomposition) and unmeasured points are predicted by evaluating a linear system (via matrix multiplication). Well optimized numerical linear algebra libraries enable computationally efficient regression with models formulated in less than a second and thousands of predictions performed in a few seconds.

This tutorial provides a theoretical background for relevant statistics and regression theory (Section 2). The R statistical computing environment (Section 3) is used for a case study in model derivation for semicoarsening multigrid, SMG2000 (Section 4). The statistically rigorous derivation emphasizes the role of domain-specific knowledge when specifying the model's functional form, leading to models consistent with prior intuition about the domain. Furthermore, statistical significance testing prunes unnecessary and ineffective predictors to improve model efficiency. Specifically, we consider the following design process for regression modeling:

- **Hierarchical Clustering:** Clustering examines correlations between potential predictors and enables elimination of redundant predictors. Predictor pruning controls model size, thereby reducing risk of over-fitting and improving model efficiency during formulation and prediction.
- **Association Analysis:** Scatterplots qualitatively capture approximate trends of predictor-response relationships, revealing the degree of non-monotonicity or non-linearity. Scatterplots with low response variation as predictor values change may suggest predictor insignificance, enabling further pruning.
- **Correlation Analysis:** Correlation coefficients quantify the relative strength of predictor-response relationships observed in scatterplots of association analysis. These coefficients impact our choice in non-linear transformations for each predictor.
- **Model Specification:** Domain-specific knowledge is used to specify predictor interaction. The correlation analysis is used to specify the degree of flexibility in non-linear transformations. Predictors more highly correlated with the response will require more flexibility since any lack of fit for these

predictors will impact overall model accuracy more. Given the model's functional form, least squares determines regression coefficients.

- **Assessing Fit:** The R^2 statistic quantifies the fraction of response variance captured by the model's predictors. Larger R^2 suggests a better fit to training data. Normality and randomness assumptions for model residuals are validated using quantile-quantile plots and scatterplots. Residual normality and randomness are prerequisites for significance testing.
- **Significance Testing:** Hypothesis testing with the F-statistic assesses the statistical significance of terms in the model. Given a baseline model and a smaller model obtained by dropping terms from the baseline, hypothesis tests use p-values to determine the significance of dropped terms. Small p-values suggest dropped terms non-trivially improve model fit.

Although accurate predictive models might be obtained without the statistical analyses before and after model specification, the additional statistical rigor contributes to a better understanding of the parameter space and improved model efficiency. These approaches are complemented by parameter space sampling optimizations that could further improve model accuracy (Section 5). This modeling process is robust and has been effective in several domains, capturing trends in microarchitectural power-performance and parallel scientific computing performance (Section 6). Collectively, sparse spatial sampling and statistical inference via regression significantly improve the efficiency of parameter space exploration.

2 Statistics and Regression Theory

2.1 Hierarchical Clustering

Data clustering classifies N data elements into clusters based on a measure of similarity represented by a symmetric $N \times N$ matrix S where $S(i, j)$ quantifies the similarity between data elements i and j. Hierarchical clustering is an iterative approach that identifies successive clusters based on previously identified clusters. Specifically, the clustering algorithm implements the following steps:

- **Initialize:** Assign each element to its own cluster to obtain N single-element clusters
- **Merge:** Combine the most similar pair of clusters into a single cluster
- **Iterate:** Repeat the merge step until one N-element cluster is obtained

The similarity between two clusters A and B is the maximum similarity between elements of each cluster: $max\{S(x, y) : x \in A, y \in B\}$. We use the squared correlation coefficient to quantify similarity of two variables, enabling the modeler to identify potential redundancy in the data set. If multiple predictors are highly correlated, a single representative predictor may capture the cluster's impact on the response. Similarly, if multiple responses are highly correlated, a single representative response may be modeled since correlated responses will likely scale with the modeled response.

Pruning the number of predictors is important to control model size, not only by controlling the number of predictors, but also by controlling the number of potential interactions between predictors. Smaller models are preferable as they reduce the number of sampled observations required for model formulation. A number of studies in which models are validated on independent data sets have shown a fitted regression model is likely reliable (no over-fitting) when the number of samples is 20 times the number of model terms [1].

2.2 Association and Correlation

Scatterplots qualitatively represent the association between variables. Such plots are useful for examining association between predictors and the response, revealing potential non-monotonicity or non-linearity. Scatterplots could quickly identify more significant predictors by showing, for example, a clear monotonic relationship with the response. Conversely, plots that exhibit low response variation despite a changing

predictor value might suggest predictor insignificance. Overall, scatterplots allow the modeler to understand the parameter space quickly at a high level.

The relative strength of scatterplot relationships may be quantified by computing the correlation between variables. Pearson's correlation coefficient between two random variables is computed by Equation (1) where X,Y are random variables with expectations μ_x,μ_y and standard deviations σ_x,σ_y.

$$\rho = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X \sigma_Y} \tag{1}$$

In cases where the distribution of X, Y are unknown, non-parametric statistics may be more robust. We use the Spearman rank correlation coefficient ρ_{sp} that can quantify association independently of variable distribution. The computationally efficient approximation only requires d_i, the difference in ordinal rank of x_i in X and y_i in Y. Suppose $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$. Compute x_i's rank in X and y_i's rank in Y. Spearman rank correlation computes a coefficient using the n differences in rank.

$$\rho_{sp} = \frac{\sum_{i=1}^{N} X_i Y_i}{\sqrt{\sum_{i=1}^{N} X_i^2 \sum_{j=1}^{N} Y_j^2}} \approx 1 - 6 \sum_{i=1}^{N} \frac{d_i^2}{N(N^2-1)} \tag{2}$$

2.3 Regression

We apply regression modeling techniques to obtain empirical estimates for metrics of interest efficiently. We apply a general class of models in which a response is modeled as a weighted sum of predictor variables plus random noise. Since basic linear estimates may not adequately capture nuances in the response-predictor relationship, we also consider more advanced techniques to account for potentially non-linear relationships.

2.3.1 Notation

For a large universe of interest, suppose we have a subset of n observations for which values of the response and predictor variables are known. Let $y = y_1, \ldots, y_n$ denote the vector of observed responses. For a particular point i in this universe, let y_i denote its response variable and $x_i = x_{i,1}, \ldots, x_{i,p}$ denote its p predictors. These variables are constant for a given point in the universe. Let $\beta = \beta_0, \ldots, \beta_p$ denote the corresponding set of regression coefficients used in describing the response as a linear function of predictors plus a random error e_i as shown in Equation (3). Mathematically, β_j may be interpreted as the expected change in y_i per unit change in the predictor variable $x_{i,j}$. The e_i are assumed independent random variables with zero mean and constant variance; $E(e_i) = 0$ and $Var(e_i) = \sigma^2$.

$$\begin{aligned} f(y_i) &= \beta g(x_i) + e_i \\ &= \beta_0 + \sum_{j=1}^{p} \beta_j g_j(x_{ij}) + e_i \end{aligned} \tag{3}$$

Transformations f and $g = g_1, \ldots, g_p$ may be applied to the response and predictors, respectively, to improve model fit by stabilizing a non-constant error variance or accounting for non-linear correlations between the response and predictors.

Fitting a regression model to observations, by determining the $p+1$ coefficients in β, enables response prediction. The *method of least squares* is commonly used to identify the best-fitting model for a set of observations by minimizing the sum of squared deviations of the predicted responses given by the model from the actual observed responses. Thus, least squares finds the $p+1$ coefficients to minimize $S(\beta)$ by solving a system of $p+1$ partial derivatives of S with respect to β_j, $j \in [0, p]$. The solutions to this system, $\hat{\beta}_j$, are estimates of the coefficients in Equation (3). Furthermore, the solutions to this system of

linear equations may often be expressed in closed form. Closed form expressions enable using the statistical properties of these estimates to identify the goodness of fit (Section 2.4) and the significance of particular response-predictor correlations (Section 2.5).

$$S(\beta_0, \ldots, \beta_p) = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \tag{4}$$

2.3.2 Predictor Interaction and Non-Linearity

In some cases, the effect of two predictors $x_{i,1}$ and $x_{i,2}$ on the response cannot be separated; the effect of $x_{i,1}$ on y_i depends on the value of $x_{i,2}$ and vice versa. The interaction between two predictors may be modeled by constructing a third predictor $x_{i,3} = x_{i,1}x_{i,2}$ to obtain $y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,1}x_{i,2} + e_i$.

Basic linear regression models often assume the response behaves linearly in all predictors. This assumption is often too restrictive and several techniques for capturing non-linearity may be applied. The most simple of these techniques is a polynomial transformation on predictors suspected of having a non-linear correlation with the response. However, polynomials have undesirable peaks and valleys. Furthermore, a good fit in one region of the predictor's values may unduly impact the fit in another region of values. For these reasons, we consider splines a more effective technique for modeling non-linearity.

Spline functions are piecewise polynomials used in curve fitting. The function is divided into intervals defining multiple different continuous polynomials with endpoints called *knots*. The number of knots can vary depending on the amount of available data for fitting the function, but more knots generally leads to better fits. For example, a linear spline (*i.e.* piecewise linear function) on x with three knots at a, b, and c is given by Equation (5) where $(u)_+ = u$ if $u > 0$ and $(u)_+ = 0$ otherwise.

$$y = \beta_0 + \beta_1 x + \beta_2 (x-a)_+ + \beta_3 (x-b)_+ + \beta_4 (x-c)_+ \tag{5}$$

Linear splines may be inadequate for complex, highly curved relationships. Splines of higher order polynomials may offer better fits [1]. Unlike linear splines, cubic splines may be made smooth at the knots by forcing the first and second derivatives of the function to agree at the knots. For example, a cubic spline on x with three knots is given by Equation (6).

$$\begin{aligned} y &= \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \\ &\quad + \beta_4 (x-a)_+^3 + \beta_5 (x-b)_+^3 + \beta_6 (x-c)_+^3 \end{aligned} \tag{6}$$

Cubic splines may have poor behavior in the tails before the first knot and after the last knot [6]. Restricted cubic splines that constrain the function to be linear in the tails are often better behaved. A restricted cubic spline on x with k knots $t_1, \ldots, t_k$ is given by Equation (7) where $j = 1, \ldots, k-2$ [6].

$$\begin{aligned} y &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{k-1} x_{k-1} \\ x_1 &= x \\ x_{j+1} &= (x-t_j)_+^3 - (x-t_{k-1})_+^3 (t_k - t_j)/(t_k - t_{k-1}) \\ &\quad + (x-t_k)_+^3 (t_{k-1} - t_j)/(t_k - t_{k-1}) \end{aligned} \tag{7}$$

Figure 1 schematically illustrates a restricted cubic spline with five knots and linear tails. The choice and position of knots are variable parameters when specifying non-linearity with splines. Placing knots at fixed quantiles of a predictor's distribution is a good approach in most data sets, ensuring a sufficient number of points in each interval [6]. In practice, five knots or fewer are generally sufficient for restricted cubic splines. Fewer knots may be required for small data sets. As the number of knots increases, flexibility improves at the risk of over-fitting the data. In many cases, four knots offer an adequate fit of the model and is a good compromise between flexibility and loss of precision from over-fitting.

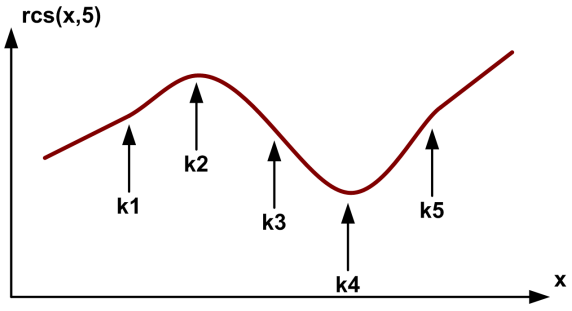

Figure 1: Schematic of restricted cubic spline with 5 knots

2.4 Assessing Fit

The model's fit to data used in formulation is quantified with the *multiple correlation statistic*, R^2, in Equation (10). This statistic quantifies regression error (SSE) as a fraction of total error (SST). From the equation, R^2 will be zero when model error is just as large as the error from simply using the mean to predict responses. Larger values of R^2 suggest better fits for the observed data. Thus, R^2 is the percentage of variance in the response captured by the predictors. However, a value too close to one may indicate over-fitting, a situation in which the model's worth is exaggerated and future observations may not agree with the modeled predictions predicted values. Over-fitting typically occurs when too many predictors are used to estimate relatively small data sets.

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{8}$$

$$SST = \sum_{i=1}^{n}\left(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i\right)^2 \tag{9}$$

$$R^2 = 1 - \frac{SSE}{SST} \tag{10}$$

The model residuals should be examined to ensure predictions exhibit no systematic bias. Significance tests (*e.g.*, T-tests, F-tests) also make assumptions regarding the distribution of residuals in Equation (11). In particular, we must validate the following assumptions prior to significance testing:

1. the residuals are not correlated with the predicted response
2. the residuals' randomness should be the same for all predicted responses
3. the residuals have a normal distribution with zero mean and constant variance

The first two assumptions are typically validated with scatterplots of residuals against predicted responses since such plots may reveal systematic deviations from randomness. The third assumption is usually validated by quantile-quantile plots in which the quantiles of one distribution are plotted against another. Practically, this means ranking the n residuals $\hat{e}^{(1)}, \ldots, \hat{e}^{(n)}$, obtaining n ranked samples from the normal distribution $s^{(1)}, \ldots, s^{(n)}$, and producing a scatterplot of $(\hat{e}^{(i)}, s^{(i)})$ that should appear linear if the residuals follow a normal distribution.

$$\hat{e}_i = y_i - \hat{\beta}_0 - \sum_{j=0}^{p} \hat{\beta}_j x_{ij} \tag{11}$$

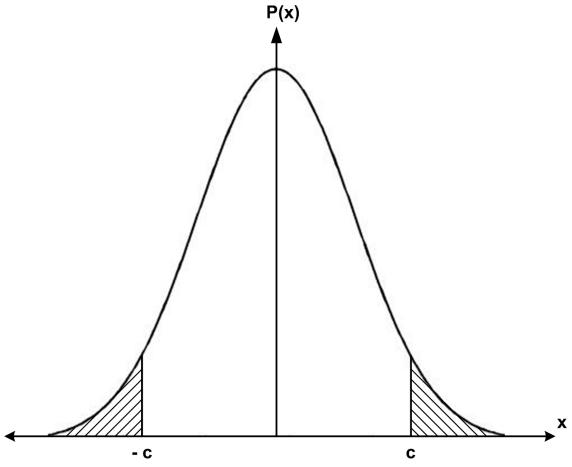

Figure 2: Schematic of p-value computation

2.5 Significance Testing

Suppose we formulate a model with p predictors that allows for interaction and non-linearity. There are three contributions to a response estimate: (1) primary effects (*e.g.* x_i), (2) interaction effects (*e.g.* $x_i x_j$), and (3) non-linear effects (*e.g.* a restricted cubic spline of x_i with k knots, $rcs(x_i, k)$). The significance of the association between the response and each of these contributions may be quantified with standard statistical tests. Relatively insignificant predictors may be pruned to improve model efficiency.

2.5.1 T-Tests

If the errors e_i in Equation (3) are independent, normal random variables, then the estimated coefficients $\hat{\beta}_j$ are also normally distributed. This normality assumption leads to the relationship in Equation (12) that states the standardized coefficient estimate follows a t distribution with $n - p - 1$ degrees of freedom. The estimates $\hat{\beta}_j$ are obtained in closed form by solving a linear system and their standard deviations $s_{\hat{\beta}_j}$ may be obtained analytically from the closed form estimates.

$$\frac{\hat{\beta}_j - \beta_j}{s_{\hat{\beta}_j}} \sim t_{n-p-1} \tag{12}$$

A commonly tested null hypothesis states the j-th term in the model has no association with the response ($H_0 : \beta_j = 0$). This hypothesis is tested by evaluating Equation (12) under the null hypothesis to obtain the *t-statistic*, $\hat{\beta}_j / s_{\hat{\beta}_j}$. The t-statistic is computed for coefficient estimate $\hat{\beta}_j$ as a first step toward determining significance of the j-th term.

The *p-value* is defined as $2P(X \geq |c|)$ for a random variable X and a constant c. This computation is illustrated schematically in Figure 2 where the shaded region corresponds to the p-value. In our analyses, $X \sim t_{n-p-1}$ and c is the t-statistic, $\hat{\beta}_j / s_{\hat{\beta}_j}$. The p-value may be interpreted as the probability a t-statistic value greater than or equal to the value actually observed would occur by chance if the null hypothesis were true. If this probability is extremely small, either the null hypothesis holds and an extremely rare event has occurred or the null hypothesis is false. Thus, a small p-value for $\hat{\beta}_j$ casts doubt on the hypothesis $\beta_j = 0$ and suggests the effect from the j-th term in the model is statistically significant in predicting the response.

2.5.2 F-Tests

Although T-tests are often used for assessing the significance of individual terms, it is often more useful to assess a group of terms simultaneously. Consider a model $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + e$. Testing the significance of x_1 requires testing the null hypothesis $H_0 : \beta_1 = \beta_3 = 0$ with two degrees of freedom. More generally, performing significance tests on a subset of the β's is preferable to t-tests for complex models with interaction and non-linear terms. The *F-test* compares two nested models (*e.g.* a full model and a subset of the full model) using their multiple correlation statistic, R^2. Given R^2 for the full model and R^2_* for a model constructed by dropping a number of terms from the full model, define the *F-statistic* by Equation (13) where p is the number of coefficients in the full model excluding the intercept β_0 and k is the difference in degrees of freedom between the models.[1] Since the F-statistic follows the F-distribution with parameters k and $n - p - 1$, a p-value may be calculated as shown in the previous section for T-tests. Smaller p-values suggest terms dropped to obtain the smaller model were significant and should have been retained.

$$F_{k,n-p-1} = \frac{R^2 - R^2_*}{k} \times \frac{n - p - 1}{1 - R^2} \tag{13}$$

2.6 Prediction

Once regression coefficients are determined by least squares, evaluating Equation (3) for a given x_i will give the expectation of y_i and, equivalently, the estimate $\hat{y}_i$ for y_i of Equation (14). This result follows from observing the additive property of expectations, the expectation of a constant is the constant, and random errors have mean zero.

$$\begin{aligned} \hat{y}_i &= E\big[y_i\big] \\ &= E\big[\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\big] + E\big[e_i\big] \\ &= \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} \end{aligned} \tag{14}$$

3 R Statistical Computing Environment

R is a statistical computing environment consisting of a language, run-time interface, scripting functionality, and statistical libraries. The standard R distribution contains functionality for statistical procedures including linear and generalized linear models, non-linear regression models, time series analysis, classical parametric and non-parametric tests, clustering and smoothing [7].

3.1 Obtaining and Installing R

Sources, binaries, and documentation for R may be obtained via CRAN, the "Comprehensive R Archive Network", at http://www.r-project.org/. If R is already installed, it may be started by typing "R" at the shell prompt, assuming the executable is in the path. If binaries are available for the target platform, simply follow the instructions accompanying them.

Instructions for compiling and installing R are included in the install file accompanying the distribution. In the simplest case, untar the R source code, change to the created directory, and issue the following commands at the shell prompt.

[1] The degrees of freedom for a model is a function of the number of predictors and splines in the model. This is computed in any standard statistics package.

```
$ ./configure
$ make
$ make check
```

If these commands execute successfully, the R binary and shell script front-end called `R` are created and copied to the `bin` directory.

Add-on packages supplement the baseline R distribution. We use two particular packages for regression: Design and Hmisc. Design is a collection of functions to assist and streamline regression modeling. Hmisc includes functions useful for data analysis such as variable clustering. These packages may be downloaded from the same mirror as the baseline. The add-on packages on CRAN come as gzipped tar files of the form `pkg_version.tar.gz`. The following commands install the packages to the default library tree.

```
$ R CMD INSTALL <path>/<pkg_version>.tar.gz
```

Available library packages are displayed with the `library()` command at the R prompt. An installed package `pkg` may be loaded with the `library(pkg)` command.

3.2 Using R

To use R interactively, start the program with the command `R` and quit the program with the command `q()`. Documentation for any library function may be obtained with the command `help(function)`. To have R run a script of R commands named `commands.R`, diverting the output to `commands.Rout`, invoke the following command:

```
\$ R CMD BATCH commands.R
```

Manuals providing a broad introduction to R commands are available online at R-project website (www.r-project.org). This report will discuss commands and features as they are introduced for use in regression.

4 Model Design and Construction

4.1 Preliminaries

Load the necessary libraries for regression and graphics support. The `lattice` library provides support for trellis graphics. If necessary, specify a file for output redirection using the `sink` function.

```
library(Hmisc,T);
library(Design,T);
library(lattice);
sink(``output.txt'');
```

4.2 Data Organization

We consider regression modeling for semicoarsening multigrid (SMG2k) with its performance predicted by the per processor working set size and processor topology. In particular denote the size of local working sets by nx, ny, nz in three dimensions. Also denote the processor topology by px, py, pz in three dimensions for a total processor count of $px \times py \times pz$ (held constant at 512 in our case study). Execution times for each configuration are broken into initialization ($tinit$), setup ($tset$), and solve ($tsolve$).

Use `read.table` to read measured performance into a data frame, a structure type that enables named fields. Column names are specified as an argument in `read.table` and named fields are accessed with the `$` operator. Noting that processor counts are measured in powers of 2, we apply a log transformation on these columns to improve data linearity.

```
---------------------------------------------------------------------------
tsolve
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90
   3358       0    3358   23.66   5.138   7.534  13.300  22.214  32.365  41.751
    .95
 47.698

lowest :  1.353  1.529  1.603  1.713  1.943
highest: 65.171 65.326 66.028 66.731 70.064
---------------------------------------------------------------------------
```

Figure 3: Solve Time Data Description

```
smg.df <- read.table("./data/smg2000-bgl.txt", sep="\t", header=F,
     col.names=c("nx","ny","nz","px","py","pz","tinit","tset","tsolve"),
     row.names=NULL);

smg.df$px <- log(smg.df$px, 2);
smg.df$py <- log(smg.df$py, 2);
smg.df$pz <- log(smg.df$pz, 2);

describe(smg.df);
dd <- datadist(smg.df); ### uses information about data set distribution
options(datadist='dd'); ### to set options for other Design, Hmisc functions
```

Invoking `describe` provides summary statistics of variables in the data frame. In Figure 3, the entry for tsolve specifies the number of data observations (3358, 0 missing, all unique). The mean and various quantiles provide a sense of the data distribution. In this case, the mean is close to the median, suggesting a symmetric distribution. Lastly, five outliers at both extremes are listed.

4.3 Sampling

Variable `n` contains the number of rows in our data frame and variable `pcount` contains the number of predictions we wish to use for validation. The `sample` function takes `x`, a sequential vector of integers from one to n representing row indices for our data frame, and stores samples obtained uniformly at random into `idx`. Since the number of samples equals the size of the set `x`, this function effectively produces a random permutation of row indices for our data frame. We will take subsets of `idx` for training data. For example, if we want to train with 1,000 random samples, we simply take the first 1,000 values of `idx`. Similarly, we obtain `pcount` samples uniformly at random from `x` without replacement and store them into `psample`. These random samples will be used for model validation.

```
n <- dim(smg.df)[1];
pcount <- 100;
idx <- sample(x=1:n, size=n, replace=FALSE);
psample <- sample(x=1:n, size=pcount, replace=FALSE);
```

This code allows overlapping data sets for training and validation. Given the size of the space and the small number of samples obtained, the probability of significant overlap is small. The possibility of overlap may be eliminated by taking both training and validation samples from `idx` without replacement.

```
Similarity matrix (Spearman rho^2)
-----------------------------------------------------------
         nx   ny   nz   px   py   pz tinit tset tsolve
nx     1.00 0.19 0.19 0.21 0.05 0.05  0.00 0.01   0.00
ny     0.19 1.00 0.19 0.05 0.21 0.05  0.05 0.00   0.00
nz     0.19 0.19 1.00 0.05 0.05 0.21  0.06 0.06   0.20
px     0.21 0.05 0.05 1.00 0.25 0.25  0.00 0.00   0.00
py     0.05 0.21 0.05 0.25 1.00 0.25  0.00 0.00   0.01
pz     0.05 0.05 0.21 0.25 0.25 1.00  0.00 0.00   0.02
tinit  0.00 0.05 0.06 0.00 0.00 0.00  1.00 0.89   0.77
tset   0.01 0.00 0.06 0.00 0.00 0.00  0.89 1.00   0.79
tsolve 0.00 0.00 0.20 0.00 0.01 0.02  0.77 0.79   1.00
-----------------------------------------------------------
```

Figure 4: Variable Clustering Data

4.4 Hierarchical Clustering

Hierarchical clustering examines correlations between a large number of potential predictors to ensure redundant predictors are not included in the model. We perform clustering with `varclus` on the potential predictors and responses, specifying variables and the data frame where the variables are defined. The $\sim$ operator specifies a relationship between $x \sim y + z$ where x is the response and y, z are the predictors. For `varclus`, no response is needed and the sum of terms on the RHS says we want to examine pairwise correlations in a similarity matrix. Storing the clustering results to `v` allows us to print the numerical results of the similarity matrix in Figure 4. Entries are the pairwise squared correlation coefficients between variables.

```
v <- varclus(~ nx + ny + nz + px + py + pz + tinit + tset + tsolve, data=smg.df);
print(v);
```

The correlations between variables are more easily observed in a clustering figure. Such a figure is generated by creating a trellis device with the `trellis.device` function, specifying the figure format, file name, and figure dimensions. The clustering data `v` is augmented with additional information about font size (1.5x the default size) in the `plot` function. Figure 5 plots elements from the similarity matrix. Close the device with the `dev.off` function after plotting the data. The correlation between the $p*$ in three dimensions are a result of the constant product $px \times py \times pz = 512$. The correlation between components of execution time are significant. In particular, solve time is highly correlated with initialization and setup time, suggesting a model that only predicts solve time would also be representative of the other two smaller components of total time.

```
trellis.device("pdf", file="./varclus.pdf", width=9, height=6);
plot(v, cex=1.5, cex.main=1.5, cex.axis=1.5, cex.lab=1.5, cex.sub=1.5);
dev.off();
```

Thus, we use the clustering analysis to identify redundant responses (*tinit* and *tset*). For predictors, the clustering figure is less interesting for this particular data set since the $p*$ and $n*$ variables were chosen for variation in this parameter space study and we are unlikely to eliminate them from consideration. More generally, such a figure may reveal redundant predictors to be eliminated. This clustering analysis would be useful if predictor significance were not obvious or small sample sizes required smaller models with fewer predictors to prevent over-fitting.

4.5 Association Analysis

Hierarchical clustering computes pairwise variable correlation to prune the number of predictors or identify a response for our model. However, we also qualitatively consider each predictor's association against the

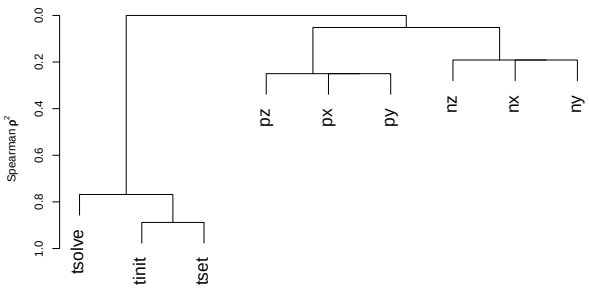

Figure 5: Variable Clustering Plot

tsolve N=3358

		N	tsolve
nx	10	1003	24.59354
	[30, 90)	909	23.49165
	[90,170)	655	23.50827
	[170,510]	791	22.79670
ny	10	1003	23.59006
	[30, 90)	909	23.34427
	[90,170)	655	23.81386
	[170,510]	791	23.98546
nz	10	1003	16.18646
	[30, 90)	909	22.59320
	[90,170)	655	27.05688
	[170,510]	791	31.55101
Overall		3358	23.66032

Figure 6: Association Data

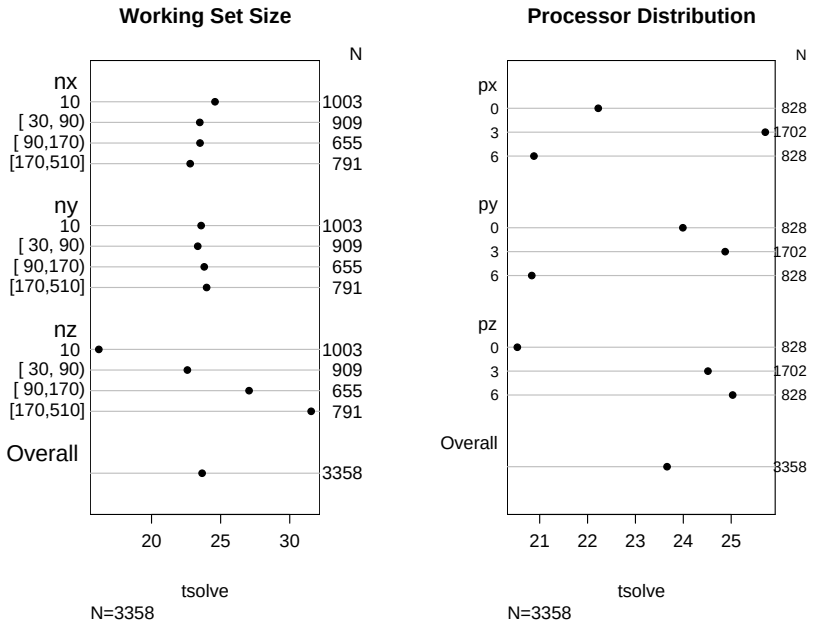

Figure 7: Association Plots

response using scatterplots. The `summary` function takes a relationship specified by the $\sim$ operator. For example, $tsolve \sim nx + ny + nz$ says we wish to consider the relationship between solve time and working set sizes. The `summary` function divides the predictor domain into intervals. For each interval, it computes the average response of all samples in the interval. The `print` command for each summary output `s.*` produces a data table relating predictor intervals to response. Figure 6 is the table for the $n*$ predictors. For example, nz equals 10 for 1003 of 3358 input combinations. The average solve time for these combinations is 16.18 seconds.

```
s.n <- summary(tsolve ~ nx + ny + nz, data=smg.df);
s.p <- summary(tsolve ~ px + py + pz, data=smg.df);

print(s.n);
trellis.device("pdf", file="./sum_n.pdf", width=4, height=6);
plot(s.n, cex=1, cex.main=1, cex.axis=1, cex.lab=1, cex.sub=1, main='Working Set Size');
dev.off();

print(s.p);
trellis.device("pdf", file="./sum_p.pdf", width=4, height=6);
plot(s.p, cex=1, cex.main=1, cex.axis=1, cex.lab=0.8, cex.sub=1, main='Processor Distribution');
dev.off();
```

Figure 7 suggests processor count and working set size in the z-dimension are particularly good predictors with strictly monotonic response-predictor relationships. For the x- and y-dimensions, working set sizes do not appear to strongly impact solve time and processor counts appear to have non-monotonic relationships with solve time. These figures will be useful when determining the most significant predictors to include in the model. This figure may also enable further pruning of predictors from the model to improve efficiency.

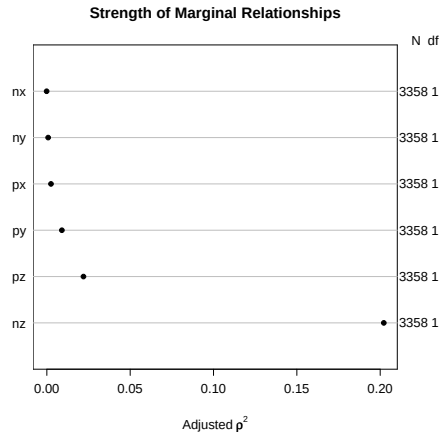

Figure 8: Strength of Marginal Relationships

4.6 Correlation Analysis

Although we qualitatively observe marginal relationships between each predictor and the response in the previous scatterplots, we have not quantified the strength of these relationships. For example, we can observe a stronger relationship between nz and solve time relative to the relationship between pz and solve time, but we do not quantify the relative strength. For this reason, we must also compute the correlation coefficients. We rank predictors by their correlation coefficients to assess relative significance. These rankings guide our choice of relative knot counts in our restricted cubic splines. Predictors with higher rankings will use more knots than those with lower rankings since any lack of fit for highly ranked predictors will negatively impact overall model accuracy more consequentially.

```
sp <- spearman2(tsolve ~ nx + ny + nz + px + py + pz, data=smg.df);
print(sp);
trellis.device("pdf", file="./sp2.pdf", width=6, height=6);
plot(sp, main='Strength of Marginal Relationships');
dev.off();
```

Given, a predictor-response relationship, the `spearman2` function computes the squared spearman rank correlation coefficient, a robust measure of correlation. Figure 8 confirms the qualitative observations of the association scatterplots. nz ($\rho^2 = 0.20$) much more strongly impacts solve time than pz ($\rho^2 = 0.02$). All other predictors are less strongly correlated, if at all, with performance.

4.7 Model Specification and Fit

We specify the regression model, predicting solve time from processor topology and working set size. Restricted cubic splines are specified by the `rcs` function that takes the predictor and the number of knots. Each cubic spline must use at least three knots. Interaction between linear terms is specified by the colon operator `:`. The `%ia%` operator allows for restricted interactions between cubic splines by removing doubly non-linear terms in the polynomial product. This operator is necessary to control the number of terms in a model with many polynomial interactions. The interactions are specified with domain-specific knowledge. For example, we would expect interaction between `pz` and `rcs(nz,5)` because processor count in the z-dimension may impact solve time differently for different working set sizes in the z-dimension. The model specification is assigned to variable `mf`. Lastly, we use the `update` function to create variants of model `mf`, transforming only the response and leaving the right side of the relationship unchanged.

```
mf <- (tsolve ~ rcs(nx,4) + rcs(ny,4) + rcs(nz,5) +
        rcs(nx,4) %ia% rcs(ny,4) +
        rcs(nx,4) %ia% rcs(nz,5) +
        rcs(ny,4) %ia% rcs(nz,5) +
        rcs(nx,4) %ia% rcs(ny,4) %ia% rcs(nz,5) +
        py + pz +
        py : pz +
        py %ia% rcs(ny,4) +
        pz %ia% rcs(nz,5)
);
mg <- update(mf, log(tsolve) ~ .);
mh <- update(mf, sqrt(tsolve) ~ .);
```

4.8 Residual Analysis

As noted in Section 2.4, there are several ways to assess a model's fit to the training data. After models are formulated, the distribution of residuals should be checked for randomness to ensure no systematic bias in the model. Furthermore, if the modeler wants to use parametric inferential methods (e.g., F-tests) on the least squares parameter estimates (e.g., regression coefficients) and wants to ensure these estimates are efficient in the number of predictors, assumptions about residual randomness and normality must be validated. We formulate models with ordinary least squares using the `ols` function and obtain the residuals with the `resid` function. These residuals are per observation differences between modeled and observed performance in the training set.

Residuals are plotted against the fitted values (i.e., regression predicted values) to ensure a lack of correlation between residuals and predictions, validating assumptions 1 and 2 in Section 2.4. The `xYplot` command uses the `quantile` method to stratify fitted values into groups of ten elements (`nx=10`). The figure plots the median, lower and upper quantiles of the residuals for these groups of ten observations. This grouping is necessary if there are too many observations for a standard scatterplot. Figure 9 presents the residual distribution for models `f` and `g`. Aside from a few trends at the tails, there are no significant deviations from randomness that suggest obvious biases. These trends can be re-examined if the modeler suspects biases when the model is applied to actual studies.

```
rf <- resid(f);
trellis.device("pdf", file="./residf.pdf", width=6, height=4);
xYplot(rf ~ fitted(f), method='quantile', nx=10,
       pch=8, type='p',
       #ylim=c(-20,20), xlim = c(-0.5,2.5),
       abline=list(h=0, lwd=0.5, lty=2),
       aspect='fill',
       xlab="Fitted Values",
       ylab="Residuals",
       main="Baseline Residual Distribution",
       cex=1.5, cex.axis=1.5, cex.lab=1.5, cex.sub=1.5);
dev.off();
```

As noted in Section 2.4, we validate assumption 3 (residual normality) by plotting ranked residuals against ranked samples from the normal distribution to produce a quantile-quantile plot. The `qqnorm` function automatically generates this plot. The resulting plot should appear linear if the residuals follow a normal distribution. The `qqline` function draws a line through the 25-th and 75-th quantile of the residuals to aid the analysis. Figure 10 presents the quantile-quantile plots for the baseline model (`f`) and the log-transformed model (`g`). The deviation from linearity in the baseline's right tail suggests the residual distribution extends farther than the postive tail of the normal distribution. This deviation is mitigated by the log transformation on the response.

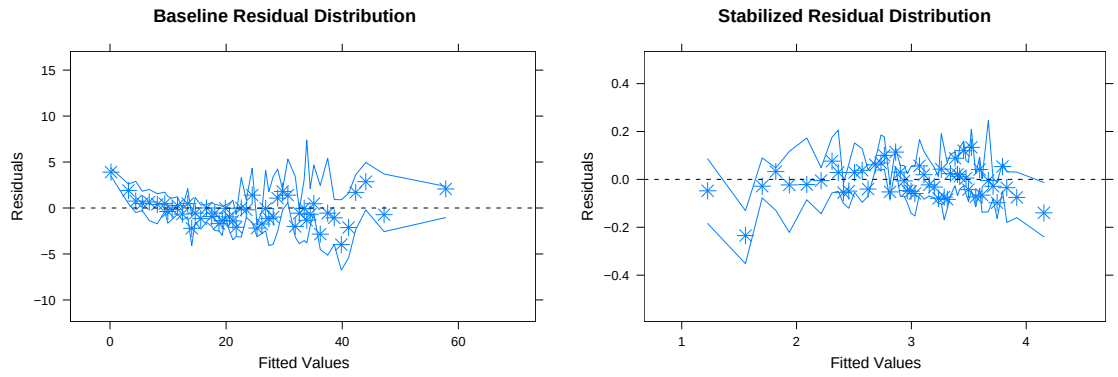

Figure 9: Residual Plots for model `f` (left) and `g` (right)

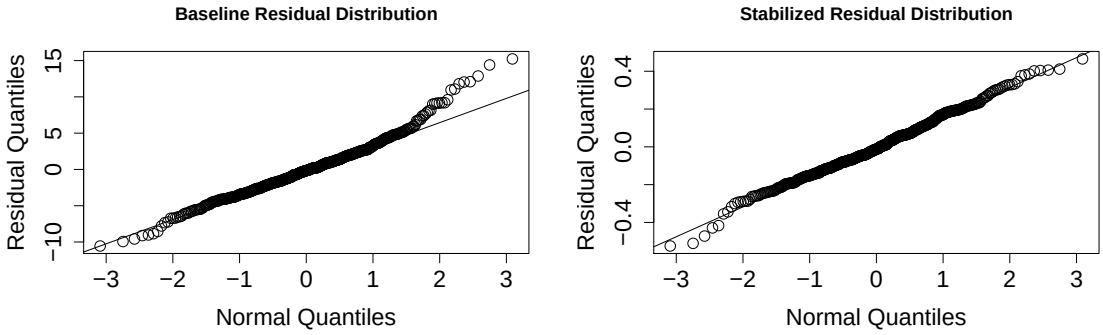

Figure 10: Residual Distribution for model `f` (left) and `g` (right)

```
rf <- resid(f);
trellis.device("pdf", file="./qqnormf.pdf", width=6, height=4);
qqnorm(rf,
       xlab="Normal Quantiles",
       ylab="Residual Quantiles",
       main="Baseline Residual Distribution",
       cex=1.5, cex.axis=1.5, cex.lab=1.5, cex.sub=1.5);
qqline(rf);
dev.off();
```

4.9 Significance Testing

After validating residual randomness and normality assumptions, we apply F-tests to assess predictor significance in our model. While the previous correlation analyses suggested significant predictors for our model, F-tests check predictor contribution to model accuracy. The `anova` function tests the hypothesis that removed terms do not significantly contribute to model accuracy. A small p-value casts doubt on this hypothesis and suggests removed terms are statistically significant in predicting the response. For example, to check the significance of `nx` in the model, we compare the original model against a reduced model that removes all terms containing `nx`. The resulting analysis computes a p-value of less than $2.2e-16$, suggesting

```
--------------------------------------------------------------------------
Analysis of Variance Table

Model 1: log(tsolve) ~ rcs(nx, 4) + rcs(ny, 4) + rcs(nz, 5) + rcs(nx,
    4) %ia% rcs(ny, 4) + rcs(nx, 4) %ia% rcs(nz, 5) + rcs(ny,
    4) %ia% rcs(nz, 5) + rcs(nx, 4) %ia% rcs(ny, 4) %ia% rcs(nz,
    5) + py + pz + py %ia% rcs(ny, 4) + pz %ia% rcs(nz, 5) +
    py:pz
Model 2: log(tsolve) ~ rcs(ny, 4) + rcs(nz, 5) + rcs(nx, 4) %ia% rcs(nz,
    5) + rcs(ny, 4) %ia% rcs(nz, 5) + py + pz + py %ia% rcs(ny,
    4) + pz %ia% rcs(nz, 5) + -rcs(nx, 4) %ia% rcs(nz, 5) + py:pz
  Res.Df     RSS  Df Sum of Sq      F    Pr(>F)
 1    459  12.475
 2    474  44.529 -15   -32.054 78.624 < 2.2e-16 *
--------------------------------------------------------------------------
```

Figure 11: F-Tests: ANOVA Analysis

significance for the group of removed terms. This analysis can also be applied to consider the significance of particular terms in the model (e.g., the third-order interaction between `nx`, `ny`, and `nz`).

```
g <- lm(mg, data=smg.df[idx[1:500],]);
g.nx <- update(g, . ~ . -rcs(nx,4)
        - rcs(nx,4) %ia% rcs(ny,4)
        - rcs(nx,4) %ia% rcs(nz,5)
        - rcs(nx,4) %ia% rcs(ny,4) %ia% rcs(nz,5));
a.nx <- anova(g, g.nx);
print(a.nx);
```

4.10 Prediction

We construct a loop that formulates regression models for varying sample sizes and uses the models to predict the validation set. We first initialize pointers to arrays that will contain error summaries. The loop considers sample sizes `s` ranging from 100 to 1000 in increments of 100. For each sample size, the `ols` function uses the model specification and the training data to determine regression coefficients and construct a model.

Variable `o` contains the observed true values of our validation points, extracting measured solve times of the points in `psample`. The `predict` function takes a regression model object produced by the `ols` function and a set of new data for prediction specified as a subset of the rows in the original data frame. Expression `smg.df[psample,]` returns all columns of the rows indexed by `psample`. In general, it is not necessary to have the training and prediction data frames formatted identically. The data frame provided as new data only needs the necessary field names required by the regression model. Note that inverse transformations on the predictions are necessary for models `g` and `h` to produce `pg` and `ph` (i.e., `exp` for `log` and square for square root). Lastly, the relative errors $e = |y - \hat{y}|/y$ are computed for the predictions.

```
summary_eg <- NULL; summary_eh <- NULL;
for(s in seq(100,1000,100)) {
  ## Formulate model with s random samples (ordinary least squares)
  f <- ols(mf, data=smg.df[idx[1:s],]);
  g <- ols(mg, data=smg.df[idx[1:s],]);
  h <- ols(mh, data=smg.df[idx[1:s],]);
```

```
-------------------------------------------------------------
         s      Min. 1st Qu. Median   Mean 3rd Qu.   Max.
 [1,]  100 1.754e-05 0.05521 0.1294 0.1803  0.2444 1.4980
 [2,]  200 4.719e-05 0.04542 0.1008 0.1205  0.1711 0.9580
 [3,]  300 2.598e-04 0.05256 0.1099 0.1326  0.1887 1.0700
 [4,]  400 4.211e-05 0.05097 0.1005 0.1202  0.1701 0.9067
 [5,]  500 1.644e-04 0.05325 0.1059 0.1297  0.1840 1.0880
 [6,]  600 2.330e-04 0.05486 0.1066 0.1311  0.1823 1.1520
 [7,]  700 1.036e-05 0.05199 0.1028 0.1284  0.1806 0.9968
 [8,]  800 1.378e-04 0.05414 0.1026 0.1286  0.1807 1.0750
 [9,]  900 4.065e-05 0.05298 0.1021 0.1284  0.1801 1.0500
[10,] 1000 1.229e-04 0.05345 0.1004 0.1282  0.1811 1.0540
-------------------------------------------------------------
```

Figure 12: Error Summary for Model g (summary_eg)

```
  ## Predict and compute errors
  o <- smg.df$tsolve[psample];
  pf <- predict(object=f, newdata=smg.df[psample,]);
  pg <- exp(predict(object=g, newdata=smg.df[psample,]));
  ph <- (predict(object=h, newdata=smg.df[psample,]))^2;
  ef <- abs(o-pf)/o;
  eg <- abs(o-pg)/o;
  eh <- abs(o-ph)/o;

  ## Aggregate results and write to file
  results_f <- cbind(o,pf,ef);
  results_g <- cbind(o,pg,eg);
  results_h <- cbind(o,ph,eh);
  write.table(results_f, file=sprintf("pred_smg_f[%d].txt", s), sep="\t", row.names=FALSE,
       col.names=c("observed", "predicted", "error"));
  write.table(results_g, file=sprintf("pred_smg_g[%d].txt", s), sep="\t", row.names=FALSE,
       col.names=c("observed", "predicted", "error"));
  write.table(results_h, file=sprintf("pred_smg_h[%d].txt", s), sep="\t", row.names=FALSE,
       col.names=c("observed", "predicted", "error"));

  ## Aggregate error statistics
  summary_eg <- rbind(summary_eg, cbind(s, t(summary(eg))));
  summary_eh <- rbind(summary_eh, cbind(s, t(summary(eh))));
}
```

The `cbind` function concatenates column vectors of true values, predicted values, and errors to produce a matrix with three columns and `pcount` rows. These matrices are written to a file with `write.table`. For each sample size, we compute summary statistics, append the sample size to the front with `cbind`, and append the data as a new row of a summary variable with `rbind`. Figure 12 presents data of `summary_eg`.

4.11 Accuracy Plots

Although the table of Figure 12 provides precise error statistics of model predictions, we may also graphically present the error distributions using cumulative distribution functions (CDF's) or boxplots. To produce CDF's, we plot sorted errors in `x` against fractions of predictions in `y`, a vector $(1/n, \ldots, k/n, \ldots, 1)$. CDF's

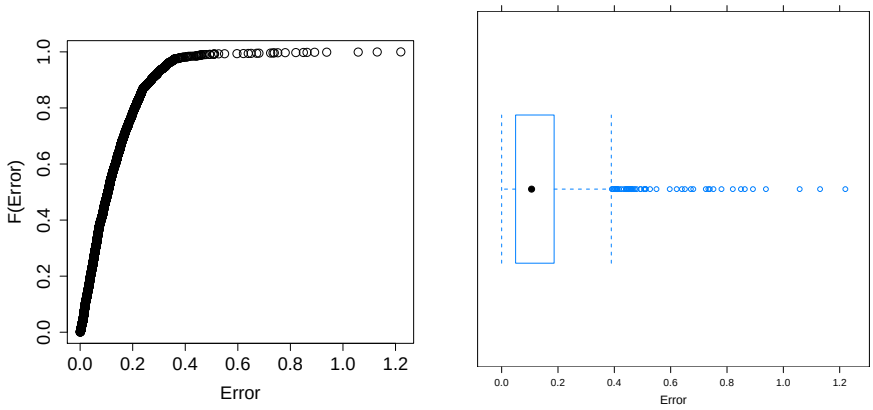

Figure 13: Cumulative distribution function (right) and boxplot (left) of model g errors

are interpretated by taking a particular error rate on the x-axis and identifying the percentage of predictions that achieve that error rate or less on the y-axis. Thus, CDF's provide an empirical approximation to the cumulative distribution function of our regression errors.

```
pred_bsl_g <- read.table("./pred_smg_bsl_g[500].txt", sep="\t", header=T,
col.names=c("obs","pred","err"), row.names=NULL);

d <- dim(pred_bsl_g)[1];
x <- sort(pred_bsl_g$err);
y <- (1:d)/d;

trellis.device("pdf", file="./cdf_bsl.pdf", width=6, height=6);
plot(x, y, xlab="Error", ylab="F(Error)",
     cex=1.5, cex.main=1.5, cex.axis=1.5, cex.lab=1.5, cex.sub=1.5);
dev.off();

trellis.device("pdf", file="./box_bsl.pdf", width=6, height=6);
bwplot(x);
dev.off();
```

Boxplots are graphical displays of data that measure location (median) and dispersion (interquartile range), identify possible outliers, and indicate the symmetry or skewness of the distribution. Boxplots are constructed by

1. solid circle at the median
2. vertical lines at the upper, lower quartiles to construct box
3. horizontal lines drawn right/left from the upper/lower quartile to the most extreme data point within 1.5 IQR of the upper/lower quartile with vertical lines to mark the end of the horizontal lines [2]
4. circles beyond the ends of horizontal lines to denote outliers

Figures such as those in Figure 13 enable graphical analysis of basic error statistics. Furthermore, boxplots also facilitate comparisons between multiple distributions (not shown here) as comparing boxplots often reveals similarities more readily than comparing tables of error statistics.

[2]IQR: interquartile range is the difference between first and third quartile.

5 Sampling Optimization

Figure 12 indicates the log-transformed model achieves modest accuracy with median errors between 10 and 12 percent. However, we observe significant maximum outlier errors between 91 and 150 percent. We propose *regional sampling* to reduce outlier error. For each query, we compute the euclidean distance between the query and every available sample. We then use 25 percent of these samples with the smallest distances, effectively excluding samples very different from the query.

We first define a function, `compED`, to compute the euclidean distance between two vectors `x` and `y`. To ensure unbiased distance calculations, we normalize and weight the data. We create a new data frame `smg_nw.df` based on the existing frame. For each field/variable in the frame, we normalize by subtracting the mean and dividing by the standard deviation. We scale the normalized value by the absolute value of the correlation between the predictor and the response (*i.e.*, solve time). Scaling by correlation coefficients ensures highly correlated predictors are given greater emphasis in the distance calculation.

```
compED <- function(x, y) {
  return(sqrt(sum((x-y)^2)));
}

smg_nw.df <- smg.df;
for(i in 1:dim(smg.df)[2]) {
  x <- smg.df[,i];
  smg_nw.df[,i] <- abs(cor(x,smg.df$tsolve))*(x-mean(x))/sd(x);
}
```

We initialize pointers to various data structures. In particular, `summary_e*` will contain error summaries for three model variants (baseline, log-transformed, sqrt-transformed), `o` contains observed response values, `p_*` contains predicted response values, and `e_*` contains prediction errors. We specify $s = 500$ samples from random permutations of sample indices contained in `idx`. We construct regions containing 125 samples, 25 percent of the full set of 500 samples.

```
summary_ef <- NULL; summary_eg <- NULL; summary_eh <- NULL;
o <- NULL; pf <- NULL; pg <- NULL; ph <- NULL;
naflag <- NULL; ef <- NULL; eg <- NULL; eh <- NULL;

## Sample size from which region is constructed
s <- 500; sidx <- idx[1:s];
```

We determine new regression coefficients based on relevant samples identified for each query. `p` indexes the i-th query's entry in the data frame. A loop iterates over the 500 samples, computing the euclidean distance between each sample and the current query. The result is stored in a vector of distances `ed`.[3] We construct a subset of the 500 samples by considering only samples with distances in the 25-th percentile using the command `which(ed < quantile(ed,0.25))` to obtain the region's indices in `ssidx`.

Once the region is identified, the model is formulated as before using ordinary least squares. Because we are performing least squares on the same model specification with substantially fewer samples, the model may be over-specified. In such an event, one or more regression coefficients will take the value `NA`, indicating a missing value. We check the coefficients of model `f` for missing values and store into `y` a boolean vector with length corresponding to the number of coefficients. If any element of `y` is true, we reduce the number of terms in the model by eliminating the interaction between `nx` and `nz`. Since these variables both use restricted cubic splines, removing this interaction will remove a large number of terms from the model and eliminate risk of model over-specification.

[3] A performance optimization not yet implemented is vectorized or parallelized distance calculation instead of the current iterative calculation

```
for(i in 1:pcount) {
  # Index into smg.df for i-th query
  p <- psample[i];
  cat(sprintf("Prediction %d, Configuration %d\n", i, p));

  ## Identify region of 125 points around query
  ed <- NULL;
  for(j in 1:s) {
    ed <- rbind(ed, compED(smg_nw.df[p,], smg_nw.df[sidx[j],]))
  }
  ssidx <- sidx[which(ed < quantile(ed,0.25))];

  ## Formulate model with region
  f <- ols(mf, data=smg.df[ssidx,]);
  g <- ols(mg, data=smg.df[ssidx,]);
  h <- ols(mh, data=smg.df[ssidx,]);
  y <- sapply(f$coefficients, 'is.na');
  if(length(subset(y, y==TRUE))) {
    naflag <- rbind(naflag, smg.df[p,]);
    f <- update(f, . ~ . - rcs(nx,4) %ia% rcs(nz,5));
    g <- update(g, . ~ . - rcs(nx,4) %ia% rcs(nz,5));
    h <- update(h, . ~ . - rcs(nx,4) %ia% rcs(nz,5));
  }

  ## Predict with regional model
  o <- rbind(o, smg.df$tsolve[p]);
  pf <- rbind(pf, predict(object=f, newdata=smg.df[p,]));
  pg <- rbind(pg, exp(predict(object=g, newdata=smg.df[p,])));
  ph <- rbind(ph, (predict(object=h, newdata=smg.df[p,]))^2);
  ef <- rbind(ef, abs(o[i]-pf[i])/o[i]);
  eg <- rbind(eg, abs(o[i]-pg[i])/o[i]);
  eh <- rbind(eh, abs(o[i]-ph[i])/o[i]);
}
print(summary(eg)); print(summary(eh));

## Aggregate and write predictions to file
results_f <- cbind(o,pf,ef)[order(o,pf,ef, na.last=NA),];
results_g <- cbind(o,pg,eg)[order(o,pg,eg, na.last=NA),];
results_h <- cbind(o,ph,eh)[order(o,ph,eh, na.last=NA),];
write.table(results_f, file="pred_smg_reg_f.txt", sep="\t", row.names=FALSE,
col.names=c("observed", "predicted", "error"));
write.table(results_g, file="pred_smg_reg_g.txt", sep="\t", row.names=FALSE,
col.names=c("observed", "predicted", "error"));
write.table(results_h, file="pred_smg_reg_h.txt", sep="\t", row.names=FALSE,
col.names=c("observed", "predicted", "error"));
```

Given the model, we predict and organize the results as before. Figure 14 presents regional model accuracy. Compared to baseline model accuracy of Figure 12, the regional models reduce median error from 10.6 to 5.5 percent for model g for 500 samples. Overall, every summary statistic from min to max is reduced by regional sampling. Figure 15 presents the cumultative distribution function and boxplot of the regional model error distribution. Comparing these figures with those for the baseline model in Figure 13, we find the number of outliers is drastically reduced. A comparison of boxplots indicates a reduction in median

```
------------------
       V1
 Min.   :0.002554
 1st Qu.:0.034311
 Median :0.054637
 Mean   :0.073933
 3rd Qu.:0.100312
 Max.   :0.803297
------------------
```

Figure 14: Error Summary for Regional Model `g` (`summary_eg`)

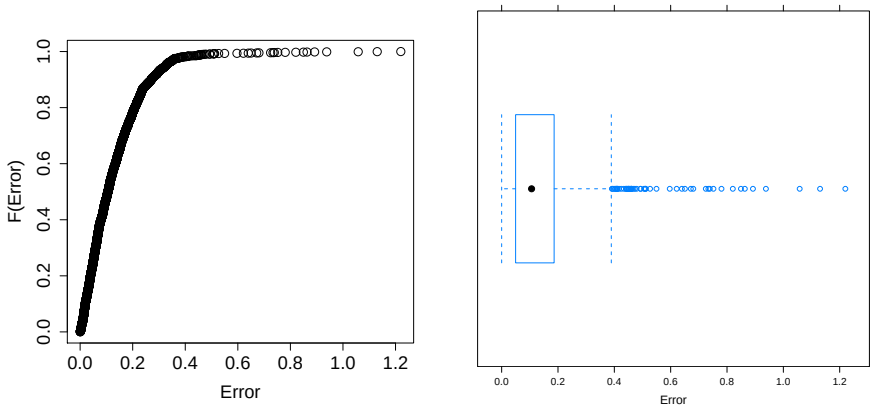

Figure 15: Cumulative distribution function (right) and boxplot (left) of regional model `g` errors

error and dispersion (i.e., smaller IQR). While other sampling techniques such as weighted regression and cross-validation may further improve accuracy, the need for these techniques will depend on model usage.

6 Further Reading

The presented regression strategies have been applied more extensively for the microarchitectural parameter space. Lee and Brooks derive piecewise polynomial regression models to predict microprocessor performance and power [3, 4], validate and assess model sensitivity to sampling techniques [2], and demonstrate model effectiveness in practical design space studies [5] Harrell describes more general regression strategies with techniques for logistic regression and survival analysis in addition to splines and linear models [1]. Background material for the R statistical computing environment is a summary of more comprehensive documentation available on the R project website [7]. This website includes manuals and frequently asked questions.

7 Conclusion

This report details a series of techniques in statistical analysis and inference for modeling a parameter space. In particular, we illustrate the application of hierarchical clustering, association and correlation analysis, piecewise polynomial regression, residual analysis, and significance testing. We detail the scripting of these techniques in R, a statistical computing environment. Within this framework, we construct effective regression models that predict semicoarsening multigrid performance with 50 percent and 75 percent of

predictions achieving error rates of 5.5 and 10.0 percent or less, respectively. The described model design process is robust and has been effectively applied to several different parameter spaces with similar accuracy. Although the parameter spaces may be drawn from different domains, the emphasis on domain-specific knowledge and statistical rigor is common to all models developed within this framework.

References

[1] F. Harrell. *Regression modeling strategies*. Springer, New York, NY, 2001.

[2] B. Lee and D. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *Proceedings Architectural Support for Programming Languages and Operating Systems (ASPLOS XII)*, October 2006.

[3] B. Lee and D. Brooks. Regression modeling strategies for microarchitectural performance and power prediction. Technical Report TR-08-06, Harvard University, March 2006.

[4] B. Lee and D. Brooks. Statistically rigorous regression modeling for the microprocessor design space. In *ISCA-33: Workshop on Modeling, Benchmarking, and Simulation*, June 2006.

[5] B. Lee and D. Brooks. Illustrative design space studies with regression. In *Proceedings High Performance Computer Architecture (HPCA 13)*, February 2007.

[6] C. Stone and C. Koo. Additive splines in statistics. In *Proceedings of the Statistical Computing Section ASA*, Washington, DC, 1985.

[7] www.r project.org. *R Language Definition*.