# META-MODELING OF A CLUSTER TOOL SIMULATOR

David Ruppert, Lee Schruben, and Michael Freimer
School of Operations Research
and Industrial Engineering
Cornell Univesity
Ithaca, NY 14853
email: davidr@orie.cornell.edu
(corrected version: Dec 14, 2001)

**Key words and phrases.** Additive models, Latin hypercubes, modeling average cycle time, regression splines.

**Abstract:**

We develop regression spline meta-models to predict the output of a cluster tool simulator. By proper selection of the predictor variables, it is often possible to find simple spline meta-models that predict the cluster tool output with very little error. The statistical meta-models have two uses. First, they can be used as a rapidly-computed replacement of the cluster tool simulator in factory-level simulations. Second, their component functions can be plotted for visualization of the factors influencing cluster tool throughput.

By examining meta-model components, we were able to quantify several interesting characteristics of cluster tools. In particular, we found an extreme, perhaps chaotic, sensitivity of average tool cycle times to small perturbations in individual chamber processing times. The methodology presented in this paper can be used to quantify the domains of applicability for different cluster tool modeling techniques. The indication from this study is that the accuracy of statistical meta-models for cluster tool performance prediction falls somewhere between simple spread-sheet performance calculations or lookup tables and highly accurate direct simulation.

# 1 DESCRIPTION OF THE CLUSTER TOOL MODEL

In this section, we describe the cluster tool model that we studied. The meta-model we developed is an approximation only to this specific cluster tool model, but the motivation of our work was to develop meta-modeling techniques that can be applied to other cluster tool models.

Our cluster tool model has five chambers arranged in a circle. The load lock is next to chamber 1. The robot can move from the loading dock to chamber 1, then to chamber 2, etc., to chamber 5. However, the robot cannot directly move from chamber 5 to chamber 1. Thus, for the robot to move from chamber 5 to chamber 1 it must go back past chambers 4, 3, and 2. The distance between chamber $i$ and chamber $j$ is $(5/4)|i-j|$ with the load lock being denoted as chamber 0. By changing the distance matrix, the cluster tool simulator would allow other geometries, e.g., ones allowing the robot arm to move past chamber 5 to chamber 1. However, this is the only geometry that we study in this report. The robot moves at speed 1, so in effect the robot speed is used to define our unit of time. The result of this is that robot move times are equal to these distances. This arbitrary robot speed is not material to our study. In our experiments, we will vary chamber processing times over a broad range so that the fraction of the cycle time that a wafer spends waiting for the robot or in transit ranges from insignificant to excessive.

A routing specifies the number of steps that each wafer goes through, the processing time of each step, the primary chamber each step is performed in, and (possibly) alternative chambers for each step.

The total cycle time, TCT, was defined as the time from when the cassette is put in the load lock until the time when the last wafer is returned to the cassette. The average cycle time, ACT, of the wafers was defined as the ratio of TCT to the number of cassettes, which in our case was always 25, so that ACT = TCT/25.

The "predictor variables" that affect ACT (or TCT) were pump time, vent time, and the processing times of the steps. For simplicity, we assumed that the processing time of each step does not depend on the particular alternative chamber in which the step is performed. In principle, our method of meta modeling would be unchanged if we relaxed this assumption by increasing the number of predictor variables.

The goal of meta-modeling is to predict ACT from the routing and these predictor variables. Our approach to meta-modeling was to fit penalized regression splines, also known as P-splines, to experimental data obtained from the cluster tool simulator with the response variable being ACT. We also fit linear regression models and semiparametric models that combine a nonparametric P-spline model for main effects with a quadratic model for two-way interactions. These models were fit separately to each routing using data from the cluster tool. The predictor variables were varied according to Latin hypercube ex-

perimental designs (McKay, Beckman, and Conover, 1979). In a Latin hypercube design, the values of a single predictor are chosen to be spaced on an equally spaced grid, with grid spacing, $h$, say. Then each of these values is perturbed by added to them independent random numbers that are uniformly distributed on $[-h/2, h/2]$. Finally, these perturbed values are randomly assigned to the runs in the experiment. The random uniform perturbations and the random assignments are independent between variables.

In a cluster tool, the predictor variables can interact in very complex ways to determine ACT. The P-splines models that we used have only simple quadratic two-way interactions. The result was that P-splines in the *original* predictor variables did not predict ACT well. However, at least in for some routings, we could find functions of the original predictors that greatly improved the P-spline predictions. Specifically, when these functions were added as additional predictors, then a low-interaction model fit the cluster tool very well. For example, in routing 1 described below, when we added as a new predictor the maximum of the processing times for the five steps, then models with no interactions fit very well and a model with quadratic interactions could predict ACT with less than 0.1% error. However, there were routings for which we were unable to find new predictors that give a high quality predictive model.

For cluster tool move sequencing, we considered push, pull, and strict priority rules. In a pull policy the next wafer to process is the one with the fewest steps remaining. In a push policy the next wafer to process is the one with the most steps remaining. In a priority policy, each wafer has a priority, and next wafer to process is the one with the highest priority.

None of these three policies can prevent deadlock. To avoid the issue of deadlock, we only studied routings where deadlock is impossible. This restriction to no-deadlock routings limited the complexity of the routings we could study.

Another possible use of metamodeling is prediction of deadlock. In this application, the response is binary (deadlock or no deadlock), so methods of binary regression or classification are appropriate. Modeling of deadlock is outside the scope of the present study, but will be a topic of future research. There would be at least two possible uses of a meta model predicting deadlock. One would be to select routings were deadlock will not occur for the range of processing times that will be used. The second use would be to devise scheduling policies that avoid deadlock.

# 2 ROUTING 1: A SIMPLE ROUTING

We started with a simple routing where there are five chambers and five steps, the $i$th step being done only in the $i$th chamber, so that each wafer goes to chambers 1, 2, 3, 4, and 5 in that order. Scheduling was done by a pull policy.

First we studied the effects of varying the predictor variables one at a time. Each predictor was varied over 15 values between 1 and 100 with the other predictor variables held fixed at 50. The 15 values of the predictor being varied were chosen from a Latin hypercube design. For each of these 15 values, the cluster tool simulator was run and average time ACT was calculated. The results are in Figure 1. Notice that ACT increases linearly with pump time and vent time, but the increases are slow as is to be expected since pumping and venting occur once in the processing of an entire cassette and so are a small part of the total cycle time (TCT). ACT is a piecewise linear function of each processing time with a "kink," i.e., change in slope, at 50. Recall that as we vary each processing time, the other times are fixed at 50. So the change in slope occurs when a given processing time becomes the rate-limiting one. This result, of course, is exactly what one might expect. This result suggests that an important predictor is the maximum of the step processing times.

Next we simultaneously varied all seven predictor variables between 1 and 100 using a 500×7 Latin hypercube. Figure 2 shows each of the seven predictor variables and a new predictor, **Max15** which is the maximum of the processing times for step 1 through step 5, each plotted again ACT. As can be seen, ACT is nearly linear in **Max15**.

# 3 REGRESSION MODELS

We fit both linear regression models and spline models to the cluster tool data. The response for the $i$th run, $Y_i$, was ACT, and the predictors for the $i$th run, $X_{1,i}, \ldots, X_{K,i}$. were pump time, vent time, step 1 processing time, ..., step 5 processing time, and other predictor variables such as **Max15**. Here $K$ is the number of predictor variables. For the routing 1 data, the predictors were pump time, vent time, step 1 processing time, ..., step 5 processing time, and **Max15** so that $K = 8$.

We will introduce our models using general notation which will be applicable to the other routings that are discussed later. The simplest model is multiple linear regression, denoted here as LINEAR, where

$$\text{LINEAR}: \quad Y_i = \beta_0 + \sum_{k=1}^{K} \beta_k X_{k,i} + \epsilon_i.$$

Here $\epsilon_i$ is modeling error representing the inability of the meta-model to predict the cluster tool performance exactly. This error term is purely deterministic since we are modeling a simulator with set processing and robot move times. Nevertheless, cluster tool behavior is unpredictable enough to warrant using statistical modeling protocol and analysis.

A nonparametric alternative to LINEAR is the additive model, denoted here as ADDITIVE, which is of the form:

$$\text{ADDITIVE}: \quad Y_i = \beta_0 + \sum_{k=1}^{K} m_k(X_{k,i}) + \epsilon_i.$$

Here, $m_k(\cdot)$ is a univariate spline as described in the Appendix. The error $\epsilon_i$ will, in general, be smaller for ADDITIVE than for LINEAR, since ADDITIVE is a more flexible model and includes LINEAR as the special case where $m_k(x) = \beta_k x$.

Additive models, including LINEAR, are at least somewhat restrictive because they assume that $Y_i$ is the sum of functions of the predictors taken one at a time. In ADDITIVE the partial derivative of $Y_i$ with respect to any two or more distinct predictors is 0. We can augment these models with quadratic interaction terms. The linear regression model with quadratic interactions (QI) is

$$\text{LINEAR/QI}: \quad Y_i = \beta_0 + \sum_{k=1}^{K} \beta_k X_{k,i}$$
$$+ \sum_{k=1}^{K-1} \sum_{k'=k+1}^{K} \beta_{k,k'} (X_{k,i} - \overline{X}_k)(X_{k',i} - \overline{X}_{k'}) + \epsilon_i.$$

Here $\overline{X}_k$ is the sample mean of $\{X_{k,i}\}_{i=1}^n$. Centering $X_{K,i}$ at its mean does not change the mathematical form of the model by induces stability, both computational and statistical.

One can also add quadratic interactions to the additive model to get

$$\text{ADDITIVE/QI}: \quad Y_i = \beta_0 + \sum_{k=1}^{K} m_k(X_{k,i})$$
$$+ \sum_{k=1}^{K-1} \sum_{k'=k+1}^{K} \beta_{k,k'} (X_{k,i} - \overline{X}_k)(X_{k',i} - \overline{X}_{k'}) + \epsilon_i.$$

This is a *semiparametric* model since it combines the *nonparametric* additive model for the main effects with the *parametric* model for the two-way interactions.

More general nonparametric interactions are also possible. These would use bivariate splines to model 2-way interactions; see Ruppert and Carroll (1996).

## 3.1 Measuring the quality of a fit

The coefficients in LINEAR and LINEAR/QI can be estimated by ordinary least squares and the coefficients in ADDITIVE and ADDITIVE/QI can be estimated by penalized least squares; see the Appendix.

Once estimates of the coefficients are available, the $i$th predicted value, denoted by $\widehat{Y}_i$, is the right hand side of the model with the unknown coefficients replaced by their estimates and the unknown "error" $\epsilon_i$ deleted. Thus, for ADDITIVE $\widehat{Y}_i = \widehat{\beta}_0 + \sum_{k=1}^{K} \widehat{m}_k(X_{k,i})$. The residual, $r_i = Y_i - \widehat{Y}_i$, is the prediction error for the $i$th case. Small absolute residuals are an indication of a model that fits well. Residual plots, such as plots of the residuals against the fitted values or against the predictor variables, are useful for detecting regions of poor fit and perhaps suggesting reasons for lack of fit.

An indication of how well the model fits overall is the sum of squared residuals, also called the sum of squared errors (SSE), given by

$$\text{SSE} = \sum_{i=1}^{n} (Y_i - \widehat{Y}_i)^2 = \sum_{i=1}^{n} r_i^2.$$

SSE can be compared to the total sum of squares (SSTO),

$$\text{SSTO} = \sum_{i=1}^{n} (Y_i - \overline{Y})^2,$$

where $\overline{Y}$ is the mean of the response. The squared multiple correlation, $R^2$ is

$$R^2 = 1 - \frac{\text{SSE}}{\text{SSTO}}.$$

A value of SSE that is small relative to SSTO, which corresponds to a value of $R^2$ close to its upper bound of 1, indicates a good predictive model.

However, models can appear to fit well simply because they have a lot of parameters and so can be

adjusted to fit any data set. Fitting a large number of meta-model parameters to a particular set of data is called overfitting. Such meta-models can not provide valid error estimates for future predictions since they are tied too closely to idiosyncrasies in past data. Therefore, a large value of $R^2$ may indicative overfitting rather than a model with good predictive ability. Cross-validation (CV) and generalized cross-validation (GCV) are measures of predictive ability that penalize overfitting. Therefore, a model that is chosen to minimize either CV or GCV should have good predictive power.

Cross-validation is the process of fitting a model to a subset of the data and then testing the fit on the data not using in fitting. The "leave-one-out" cross-validation statistic is like SSE but $\widehat{Y}_i$ is from the fit that does not use $Y_i$.

Generalized cross-validition, which is an easily computed approximation to leave-one-out cross-validation, minimizes the criterion

$$\text{GCV} = \frac{\text{SSE}}{n(1 - \text{DF}/n)^2},$$

where DF is the number of degrees of freedom in the model. For fitting with OLS, DF is simply the number of parameters in the model. For penalized least-squares and other fitting methods which produce a smoother fit than OLS by penalizing overfitting, DF is defined differently. In such cases, DF need not be an integer and is smaller than the number of parameters in the model because of the overfitting penalty; see Hastie and Tibshirani (1990) and the Appendix. A small value of GCV indicates a good predictive model.

CV and GCV were developed for model selection when the errors are random and independent. In this research, we used GCV under deterministic errors. The results here suggest that GCV can be used successfully under deterministic errors, at least of the types we encountered.

# 4 FITTING THE MODELS TO THE ROUTING 1 DATA

Table 1 gives the results of fitting the four models to the routing 1 data. The most complex of the four models, ADDITIVE/Q1, fits best by all three criteria, SSE, $R^2$, and GCV. Its superiority according to SSE and $R^2$ is simply a result of having more parameters, but the fact that ADDITIVE/Q1 minimizes GCV indicates that it has good predictive power.

Figure 4 is a plot of the residuals from the ADDITIVE model versus the predictor variables. In that plot the residuals tend to be small relative to typical values of ACT, which are around 80. Also, there are no patterns in the residual plot. Patterns in the residual plots, if present, might suggest ways to improve the models, since residuals are the prediction errors.

Figure 3 plots components of the ADDITIVE model. Each subplot shows ACT as a function of one predictor with the other predictors held fixed at their mean values. If the other predictors were fixed at values other then their means, then the curves would shift up or down but otherwise would be unchanged. (This would not be true if the model had interactions.)

Notice in Table 1 that all models have very large values of $R^2$. The linear regression model with no interactions has the smallest $R^2$ value, yet its $R^2$ value is .9988 so there is about an overall 0.1% error in prediction.

# 5 ROUTING 2

Routing 2 has four steps. Step 1 is processed in chamber 1, step 2 in chamber 2, and step 3 in chamber 3. Step 4 can be processed in either chamber 4 or chamber 5. With this routing, step 4 is not a bottleneck unless its processing time is quite large, at least twice as large as the larest of the other processing step times. As in routing 1, a pull scheduling policy was used.

We simulated this routing 300 times with pump time, vent time, and the four step processing times varied between 1 and 100 according to a 300 row Latin hypercube design.

As potential predictor variables, besides the pump, vent, and step processing times, we considered **Max13** and **Max14** where **Max13** was the maximum of the processing times for steps 1 to 3 and **Max14** was the maximum of **Max13** and *half* the processing time for step 4. The processing time for step 4 was halved since this step can be processed in either of two chambers.

The raw data are shown in Figure 5; ACT is plotted against each of the predictor variables. One can see

the small effects of pump and vent time and that the processing times for steps 1 to 3 provide lower bounds for ACT. **Max13** alone is a good predictor of ACT except for a few simulation runs. These are the cases where **Max14** is larger than **Max13** because the step 4 processing time is over twice as large as the processing times of all other steps. Thus, these are the rare cases where step 4 is the bottleneck. **Max14** is an excellent predictor of ACT in all cases. We did not use **Max13** in modeling since **Max14** seemed to be a better predictor.

The summaries of the fits to the four models are in Table 2. Clearly all four models fit extremely well, all predicting ACT with at most 0.01% error. ADDITIVE and LINEAR give essentially the same fits. The addition of interactions improves the fits somewhat, but there really isn't much room for improvement over LINEAR.

Figure 6 shows the components of the ADDITIVE fit to the routing 2 data. Clearly, **Max14** alone has a very large influence on ACT. The residuals from the ADDITIVE fit are plotted in Figure 7. Most absolute residuals are less that 0.5, which is reasonably small compared to the typical value of ACT which is around 80.

Figure 7 shows some evidence that the absolute residuals are larger when **Max14** is small. However, the size of the absolute residuals is difficult to ascertain when only the raw residuals are plotted. Moreover, regions of high data density often appear more dispersed. Carroll and Ruppert (1988) recommend that to check for variable dispersion, one plot *absolute* residuals and fit a smooth curve through them. In Figure 8 the absolute residuals are plotted and a spline fit to the absolute residuals is superimposed. In that plot, there is clear evidence that the absolute residuals are larger when **Max14** is smaller. In other words, throughput is more difficult to predict when **Max14** is small.

Figure 8 illustrates the effect of a distinctive characteristic of cluster tools. In a cluster tool, individual wafer cycle times are dependent on one another because they may need to wait for the shared robot(s). A relatively fast robot would usually be available when a wafer is ready to be moved; queueing is minimal. In this situation, the total wafer cycle times are nearly independent and easier to predict. In fact, if there were no queueing for the robot, individual wafer cycle times would simply be the sum of all their processing and move times. On the other hand, when the processing times are relatively short, as reflected here by a small value of **Max14**, then waiting for the robot can account for a substantial portion of a wafer's total cycle time. This results in the wafer cycle times being smaller, dependent, and harder to predict. They are no longer just the sum of their individual processing and move times. This is further complicated by the potentially chaotic movement of the robot when it is very busy. A more complete analysis of absolute residuals like the one presented in Figure 8 should help quantify the domain of applicability for statistical meta-models for actual cluster tool performance. It is reasonable to conjecture that for a specific tool configuration and recipe there is a critical ratio of robot speed to maximum chamber processing time below which only direct simulation can provide accurate predictions.

# 6 ROUTING 3

Routing 3 has four steps. For $i = 1, \ldots, 4$, the $i$th step can be processed in either the $i$th chamber or in the fifth chamber. Routing 3 was studied using the same design as used in routing 2; this was possible since each routing has four steps. As in routings 1 and 2, a pull scheduling policy was used.

Table 3 gives the goodness-of-fit statistics for the four models. ADDITIVE/Q1 fits best, but no model predicts nearly as well as in routing 1 and 2.

Figures 9, 10, and 11 show the raw data, components of the ADDITIVE fit, and residuals, respectively. The residual plot shows that prediction errors as large as 20, approximately a 25% relative error, are frequent.

The predictor variable **Max14** here is the maximum of the four step processing times. It certainly is a useful predictor. However, as can be seen in Figure 9 it provides an upper bound for ACT, but not a good prediction for ACT. Even with the other predictor variables and their interactions added to the model, the best $R^2$ is .88 indicating that typical prediction errors are around 12%.

Since the cluster tool is deterministic, in principle ACT can be predicted perfectly. What seems needed here is a better understanding of the routes that the wafers take through the cluster tool. This information would help us understand how the complex interactions between the step processing times and robot movements influence ACT.

To investigate the behavior of this routing, we

looked at the effects of step 3 processing time, but restricting the other predictor variables to the range $\{50, 51, 52\}$ while the step 3 processing time varied from 1 to 100. The raw data are plotted in Figure 12. When the step 3 processing is less than 50, ACT increases slowly and linearly in that time. When the step 3 processing time exceed 52, the graph of ACT as a function of that time bifurcates. One fork continues the slow linear increase, while the other fork increases more quickly and is less linear. The second fork contains a smaller percentage of the data than the first. What causes some observations to fall on the upper fork with large ACT values? The problem is that the pull policy is not optimal. Small changes in timings can affect which wafer is processed next by a pull policy. Such a change in routing can have large effects on throughput as the pull policy either becomes closer or less close to the optimal policy. Notice as well that in the left, middle row plot, ACT versus step 2 processing time, we see that the upper fork occurs only if the step 2 processing time is 51 or 53, not 52. Thus, we have the paradoxical behavior that throughput can decrease if the step 2 processing time is increased from 51 to 52. This odd behavior occurs because wafer move sequencing policies are only optimal over a narrow range of processing times. An optimal sequencing policy for one set of processing times may become very poor when there are small perturbation in these times. This indicates that the robustness of a cluster tool dispatching policy may be much more important than its optimality for a particular recipe.

In particular, having seen an example of extreme, perhaps chaotic, sensitivity of cycle times to processing times in this study, it may be more important for wafer sequencing rules to have uniformly good performance over a wide range of processing times than to be truly optimal over a narrower range. The methodologies employed in this study can be used to quantify the domain of applicability of statistical meta-models or static look-up tables for use in factory-level simulations. The indication from this study is that the accuracy of statistical meta-models for cluster tool performance prediction falls somewhere between simple spread-sheet performance calculations or lookup tables (valid only with fast or numerous robots) and highly accurate direct simulation.

# 7 DISCUSSION

Parametric regression and P-spline meta-models can be very effective for at least some cluster tool models. However, neither parametric regression nor P-splines can be used blindly. It is necessary to choose the "right" predictor variables. In some cases, such as routing 2 and 3, the right predictor variables are rather obvious. In other cases, e.g., routing 3, we were unable to find highly effective predictor variables and ACT prediction error average about 10% with the meta-models we developed.

Meta models such as we have developed have many possible applications. The most obvious is to replace the cluster tool itself when modeling factories. This can only be done when the meta model is highly accurate such as with Routings 1 and 2. Another use of meta models is as a visualization tool to provide insight into the behavior of the cluster tool and to generate new research questions.

# APPENDIX: P-SPLINES

## Univariate P-splines

The material in this Appendix is adopted from Ruppert and Carroll (1996 and 2000).

Suppose that we have data $(X_i, Y_i)$ where $X_i = X_{1,i}$ is the sole predictor variable,

$$Y_i = m(X_i) + \epsilon_i, \tag{1}$$

and $m$ is a smooth function giving the conditional mean of $Y_i$ given $X_i$. The $\epsilon_i$'s are errors. To estimate $m$ we can let $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p, \beta_{p+1}, \ldots, \beta_{p+N})^T$ and use a regression spline model

$$m(x; \boldsymbol{\beta}) = \beta_0 + \beta_1 x + \cdots + \beta_p x^p + \sum_{l=1}^{N} \beta_{p+l} (x - \kappa_l)_+^p. \tag{2}$$

where $p \geq 1$ is an integer, $(u)_+^p = u^p I(u \geq 0)$, and $\kappa_1 < \ldots < \kappa_N$ are fixed knots.

Model (2) is a piecewise polynomial, since $m(x; \boldsymbol{\beta})$ is a $p$th degree polynomial between any two adjacent knots. Moreover, $m(x; \boldsymbol{\beta})$ has $(p-1)$ continuous derivatives at each knot. If the number of knots, $N$, is large, then the ordinary least squares fit of (2) to data will tend to interpolate the data, i.e., the fit will follow the noise and will have a rough appearance.

This phenomenon of fitting the noise is called overfitting.

The traditional method of avoiding overfitting is knot selection, that is, selection of a subset of the knots such that the ordinary least squares fit follows the signal in the data but not the noise. In this report we use a different approach by allowing $N$ to be large and retaining all knots, but using a roughness penalty on $\{\beta_{p+k}\}_{k=1}^{N}$ which is the set of jumps in the $p$th derivative of $m(x;\boldsymbol{\beta})$. We could view this method as a penalty on the $(p+1)$th derivative of $m(x;\boldsymbol{\beta})$ where that derivative is a generalized function. We recommend $N$ between 5 and 40 and letting $\kappa_k$ be the $k/(N+1)$th sample quantile of the $X_i$'s—we call this choice of knots "equally-spaced sample quantiles."

We define $\widehat{\boldsymbol{\beta}}(\alpha)$ to be the minimizer of

$$\sum_{i=1}^{n}\left\{Y_i - m(x;\boldsymbol{\beta})\right\}^2 + \alpha\sum_{k=1}^{N}\beta_{p+k}^2, \qquad (3)$$

where $\alpha$ is a smoothing parameter. Because $\alpha$ controls the amount of smoothing, the value of $N$ is not crucial. For typical mean functions, $N = 10$ and $N = 40$, say, produce very similiar estimates, provided that $\alpha$ is selected appropriately for each $N$ and that $p \geq 2$. Selection of $\alpha$ will be by GCV.

Let $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$ and $\boldsymbol{X}$ be the "design matrix" for the regression spline so that the $i$th row of $\boldsymbol{X}$ is

$$\boldsymbol{X}_i = (1, X_i, \cdots, X_i^p, (X_i-\kappa_1)_+^p, \cdots, (X_i-\kappa_N)_+^p). \qquad (4)$$

Also, let $\boldsymbol{D}$ be a diagonal matrix whose first $(1+p)$ diagonal elements are 0 and whose remaining diagonal elements are 1. Then simple calculations show that $\widehat{\boldsymbol{\beta}}(\alpha)$ is given by

$$\widehat{\boldsymbol{\beta}}(\alpha) = \left(\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{D}\right)^{-1}\boldsymbol{X}^T\boldsymbol{Y}. \qquad (5)$$

This is a ridge regression estimator that shrinks the regression spline towards the least-squares fit to a $p$th degree polynomial model (Hastie and Tibshirani, 1990, Section 9.3.6).

Computing (5) is extremely quick, even for a relatively large number, say 30, values of $\alpha$. The computational time for the matrices $\boldsymbol{X}^T\boldsymbol{X}$ and $\boldsymbol{X}^T\boldsymbol{Y}$ is linear in $n$, but these matrices need only be computed once. As Eilers and Marx (1996) mention, after these matrices are computed, only $N \times N$ matrices need to be manipulated. This allows rapid selection of $\alpha$ by techniques such as minimizing $C_p$ or generalized cross-validation when $\widehat{\boldsymbol{\beta}}(\alpha)$ is calculated over a grid of values of $\alpha$.

For a linear model fit by ordinary least squares, the degrees of freedom (DF) of the model is defined to be the number of parameters in the model. Thus, the DF of the polynomial spline is $1+p+N$ if fit by ordinary least squares. If we drop all of the piecewise polynomial terms and fit by ordinary least squares, then DF is only $1+p$. For penalized least squares fitting, the concept of degrees of freedom does not apply exactly. However, there is the idea of the "effective degrees of freedom" and this is defined as

$$\mathrm{DF}(\alpha) = \mathrm{Tr}\Big(\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \alpha\boldsymbol{D})^{-1}\boldsymbol{X}^T\Big),$$

where $\mathrm{Tr}(\boldsymbol{A})$ is the trace of a square matrix $\boldsymbol{A}$. It can be shown that $\mathrm{DF}(0) = 1+p+N$ as it should since $\alpha = 0$ gives an ordinary least squares fit to the spline model. Similarly, $\mathrm{DF}(\infty) = 1+p$ since $\alpha = \infty$ gives an ordinary least squares fit to a $p$th degree polynomial model. For $\alpha$ between 0 and $\infty$, $\mathrm{DF}(\alpha)$ is between $1+p$ and $1+p+N$ and need not be an integer.

## Multivariate splines

Recall that $(X_{1,i}, \ldots, X_{K,i})^T$ is the vector of predictor variables. Suppose now that $K > 1$. As we will see, a full multivariate model for $m$ can be constructed using tensor-product regression splines. For $k = 1, \ldots, N$, let $\{\kappa_{k,j}\}_{j=1}^{N}$ be a set of knots for the $k$th predictor. In practice, $N$ could vary with $k$, but for ease of notation $N$ will be independent of $k$ in this exposition. The basis functions for regression splines in the $k$th predictor, $X_k$, are

$$\mathcal{B}(k) = \{\mathbf{1}, x_k, \ldots, x_k^p, (x_k-\kappa_{k,1})_+^p, \ldots, (x_k-\kappa_{k,N})_+^p\} \qquad (6)$$

Here $\mathbf{1}$ is the function that is identically 1. The tensor-product regression spline basis is

$$\mathcal{B}(1,\ldots,K) \equiv_{def} \mathcal{B}(1)\otimes\cdots\otimes\mathcal{B}(K),$$

i.e., the set of all products $b(1)\cdots b(K)$ where $b(k) \in \mathcal{B}(k)$. We use the notation "$a \equiv_{def} b$" to mean that $a$ equals $b$ by definition of $a$. The dimension of this basis, $(1+p+N)^K$, grows geometrically in $K$.

## Additive models

As just described, when $K$ is large, the number of tensor-product basis functions is enormous, a prob-

lem often called the "curse of dimensionality." To overcome this difficulty, we can use an appropriate subset of the tensor-product spline basis giving, for example, an additive model or a low-order interaction model. The idea is analogous to setting interactions, or at least higher order interactions, to 0 when fitting a factorial model. For simplicity, in this report we have only used additive spline models. Any interactions were modeled parametrically, not as splines. In this section, we describe additive spline models.

A function $m$ of $\boldsymbol{x} = (x_1, \ldots, x_K)^T$ is said to be additive if $m(\boldsymbol{x}) = \sum_{k=1}^{K} m_k(x_k)$ for univariate functions $m_k, k = 1, \ldots, K$. An additive model restricts $m$ in (1) to be an additive function. Additive models can be fit using the basis

$$\mathcal{B}(1) \cup \cdots \cup \mathcal{B}(K), \tag{7}$$

that is, the union of the the univariate bases. Notice that the dimension of $\mathcal{B}(1) \cup \cdots \cup \mathcal{B}(K)$ is $1 + K(p+N)$ which grows only linearly, not exponentially, in the number of predictors $K$. Thus, additive models do not suffer from the curse of dimensionality.

The simplest method of penalization for additive models is to use the same value of the penalty parameter $\alpha$ for each predictor variable. This approach works well when all of the component functions, $m_k$, have roughly the same amount of curvature as one would expect for cluster tool meta modeling. This is the penalization method used in this report. In other situations, where some component functions have much more curvature than others, the value of $\alpha$ should vary with $k$. In this case, $K$ values of $\alpha$ must be chosen by GCV. See Ruppert and Carroll (2000).

Once the coefficients of the additive model basis functions have been estimated by penalized least squares, the component functions of $m(\boldsymbol{x}) = \beta_0 + m_1(x_1) + \cdots + m_K(x_K)$ can be estimated by $\widehat{\beta}_0$ and

$$\widehat{m}_k(x_k) = \sum_{l=1}^{p} \widehat{\beta}_{k,l} x_k^l + \sum_{l=1}^{N} \widehat{\beta}_{k,p+l} (x_k - \kappa_{k,l})_+^p, \qquad k = 1, \ldots, K,$$

where $\widehat{\beta}_{lk}$ is the estimated coefficient of $x_k^l$, etc.

# REFERENCES


Carroll, R. J., and Ruppert, D. (1988). Transformation and Weighting in Regression, Chapman & Hall: New York.

Eilers, P.H.C., and Marx, B.D. (1996), "Flexible smoothing with B-splines and penalties (with discussion)," *Statistical Science*, 11, 89–121.

McKay, M.D., Beckman, R.J., and Conover, W.J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21, 239–245.

Hastie, T. and Tibshirani, R. (1990). Generalized additive models. Chapman and Hall: New York.

O'Sullivan, F. (1986), "A statistical perspective on ill-posed inverse problems (with discussion)," *Statistical Science*, 1, 505–527.

O'Sullivan, F. (1988), "Fast computation of fully automated log-density and log-hazard estimators," *SIAM Journal of Scientific and Statistical Computation*, 9, 363–379.

Ruppert, D., and Carroll, R.J. (1996), "A simple roughness penalty approach to regression spline estimation," Technical Report #1167, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY.

Ruppert, D., and Carroll, R.J. (2000). "Spatially-adaptive penalties for spline fitting," *Australian and New Zealand Journal of Statistics*, 42, 205-223.

| Model | $R^2$ | SSE | GCV | DF |
|---|---|---|---|---|
| LINEAR | .9988 | 127.6 | 0.265 | 9 |
| LINEAR/Q1 | .9990 | 106.4 | 0.235 | 24 |
| ADDITIVE | .9991 | 101.7 | 0.226 | 25.4 |
| ADDITIVE/Q1 | .9994 | 61.7 | 0.152 | 50.0 |

Table 1: Routing 1. Summary of fits to four models.

| Model | $R^2$ | SSE | GCV | DF |
|---|---|---|---|---|
| LINEAR | .9999 | 14.05 | 0.049 | 8 |
| LINEAR/Q1 | .9999 | 12.74 | 0.048 | 18 |
| ADDITIVE | .9999 | 14.05 | 0.049 | 8.00 |
| ADDITIVE/Q1 | .9999 | 13.57 | 0.094 | 18.00 |

Table 2: Routing 2. Summary of fits to four models.

| Model | $R^2$ | SSE | GCV | DF |
|---|---|---|---|---|
| LINEAR | .76 | $2.81 \times 10^4$ | 99.0 | 8 |
| LINEAR/Q1 | .84 | $1.92 \times 10^4$ | 72.6 | 18 |
| ADDITIVE | .82 | $2.17 \times 10^4$ | 82.7 | 19.6 |
| ADDITIVE/Q1 | .88 | $1.37 \times 10^4$ | 55.6 | 30.9 |

Table 3: Routing 3. Summary of fits to four models.

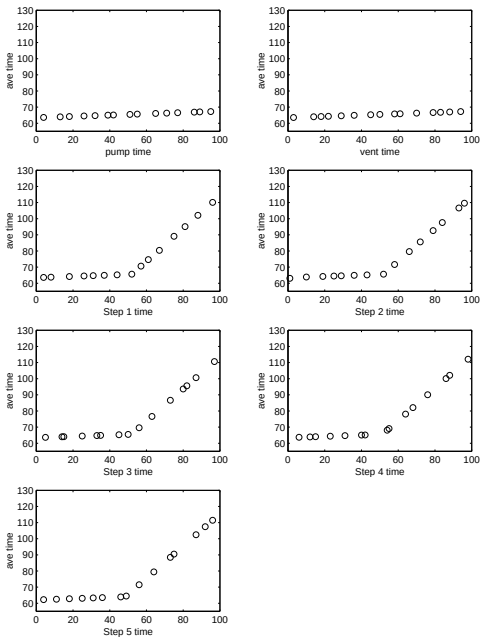


Figure 1: *Routing #1. Plot of average cycle time (ACT) versus pump time, vent time, and each of five processing times. In each plot, the variable on the horizontal axis is varied while the other variables are fixed at 50 time units.*

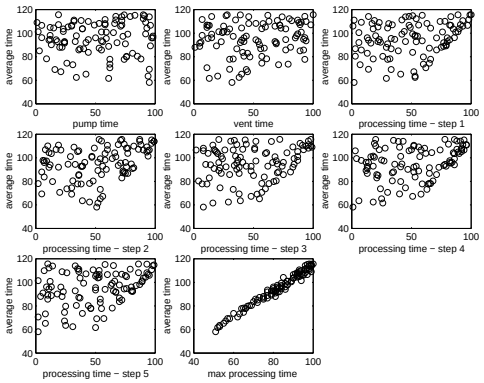


Figure 2: *Routing #1. Plot of average cycle time (ACT) versus pump time, vent time, and each of five processing times and versus the maximum processing time,* **Max15**.

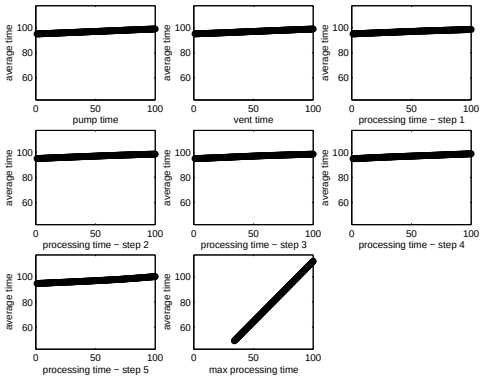


Figure 3: *Routing #1. Plot of additve spline fit to average cycle time (ACT) with the following predictors: pump time, vent time, each of five processing times, and the maximum processing time,* **Max15**.

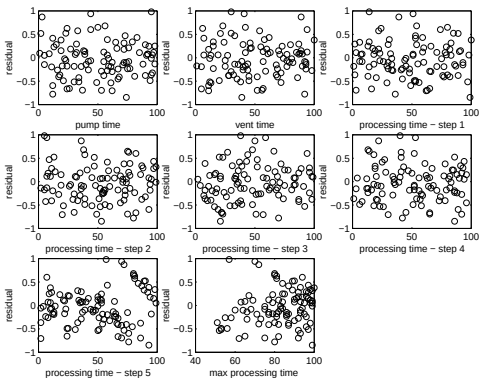


Figure 4: *Routing #1. Plot of residuals versus pump time, vent time, and each of five processing times and versus the maximum processing time.*

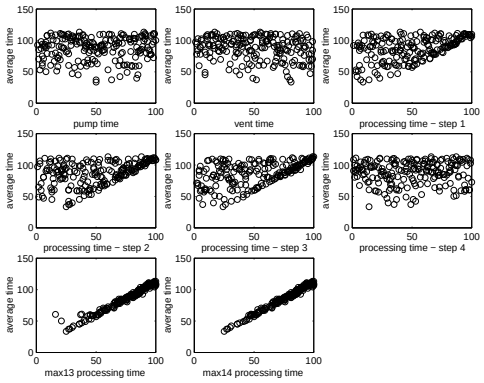


Figure 5: *Routing #2. Plot of average cycle time (ACT) versus pump time, vent time, each of four processing times,* **Max13***, and* **Max14***.*

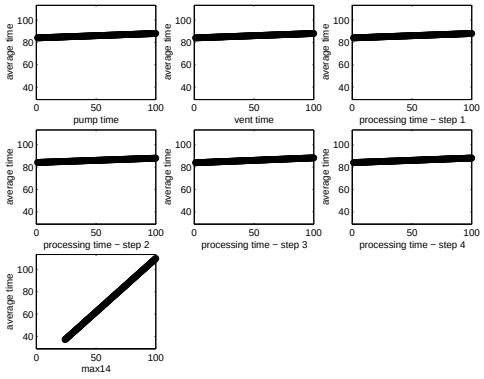


Figure 6: *Routing #2. Plot of additve spline fit to average cycle time (ACT) with the following predictors: pump time, vent time, each of four processing times, and* **Max14***.*

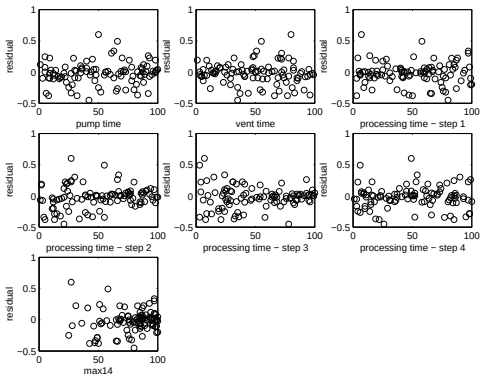


Figure 7: *Routing #2. Plot of residuals versus pump time, vent time, each of four processing times and* **Max14***.*

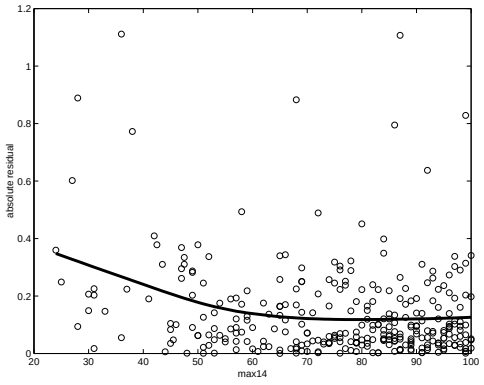


Figure 8: *Routing #2. Plot of absolute residuals versus* **Max14** *with a spline fit.*

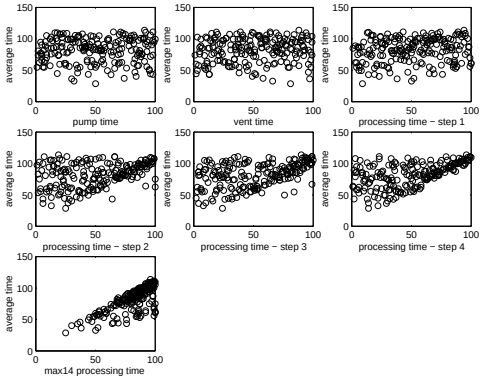


Figure 9: *Routing #3. Plot of average cycle time (ACT) versus pump time, vent time, each of four processing times, and* **Max14***.*

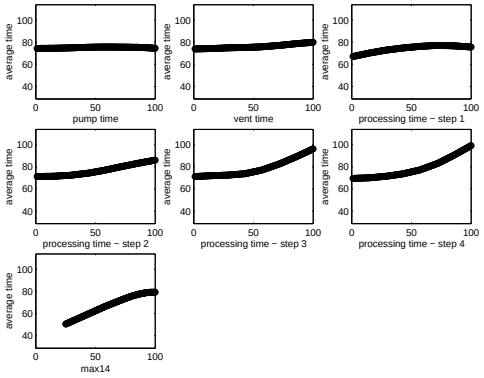


Figure 10: *Routing #3. Plot of additve spline fit to average cycle time (ACT) with the following predictors: pump time, vent time, each of four processing times, and* **Max14***.*

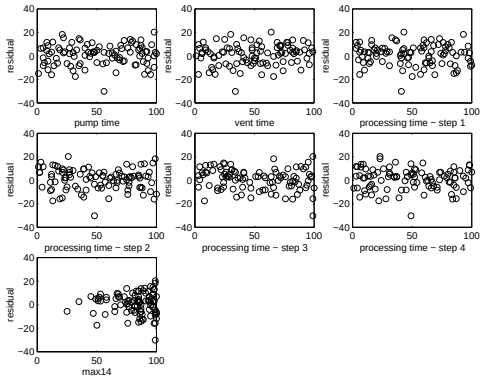


Figure 11: *Routing #3. Plot of residuals versus pump time, vent time, each of four processing times, and* **Max14**.

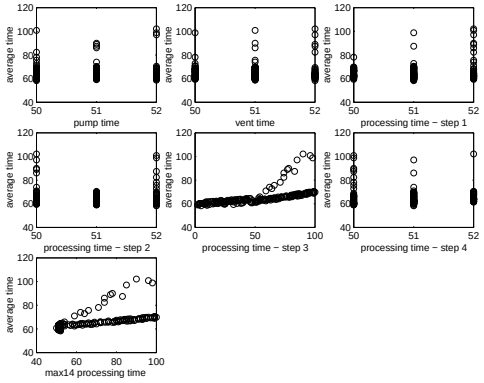


Figure 12: *Routing #3. Plot of average cycle time (ACT) versus pump time, vent time, each of four processing times, and* **Max14**. *In this data set, all predictors except the step 3 processing time are restricted to the set* $\{50, 51, 52\}$.