Examining the Impact of Experimental Design Strategies on the Predictive Accuracy of Quantile Regression Metamodels for Computer Simulations of Manufacturing Systems

by

Rishikesh Reddy Nimma

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2016 by the
Graduate Supervisory Committee:

Jennifer M. Bekki, Chair
Sharon L. Lewis
Jeremi S. London

ARIZONA STATE UNIVERSITY

August 2016

ABSTRACT

This thesis explores the impact of different experimental design strategies for the development of quantile regression based metamodels of computer simulations. In this research, the objective is to compare the resulting predictive accuracy of five experimental design strategies, each of which is used to develop metamodels of a computer simulation of a semiconductor manufacturing facility. The five examined experimental design strategies include two traditional experimental design strategies, sphere packing and I-optimal, along with three hybrid design strategies, which were developed for this research and combine desirable properties from each of the more traditional approaches. The three hybrid design strategies are: arbitrary, centroid clustering, and clustering hybrid. Each of these strategies is analyzed and compared based on common experimental design space, which includes the investigation of four densities of design point placements three different experimental regions to predict four different percentiles from the cycle time distribution of a semiconductor manufacturing facility. Results confirm that the predictive accuracy of quantile regression metamodels depends on both the location and density of the design points placed in the experimental region. They also show that the sphere packing design strategy has the best overall performance in terms of predictive accuracy. However, the centroid clustering hybrid design strategy, developed for this research, has the best predictive accuracy for cases in which only a small number of simulation resources are available from which to develop a quantile regression metamodel.

To my beloved parents Neeraja Gurrala and Jayasimha Reddy Nimma

ACKNOWLEDGMENTS

I would like to sincerely acknowledge and thank my advisor Dr. Jennifer Bekki, for her constant support and supervision throughout my master's program. Her expert guidance, prompt inspirations, timely suggestions, meticulous scrutiny, scholarly advice and scientific approach have helped me in every step to a very extent in accomplishing the master's thesis.

I would also like to thank Dr. Sharon Lewis and Dr. Jeremi London for serving in my committee. I am very thankful to them for their support and encouragement throughout the research. I also want to thank Dr. Rong Pan, for his valuable advice that has enabled me to complete my thesis. I thank profusely all the faculty and staff of engineering department, for their kind help and co-operation throughout my research and administrative work.

I reserve special thanks to my parents Neeraja Gurrala, Jayasimha Reddy Nimma and my sister Spandana Nimma for their love and support throughout my life. Finally, my thanks are also due to Vajra Sabhapathy Badha, Revanth Soma, Anirudh Varma, Prudhupal Jally, and Santosh Vemula for their moral support. I profusely thank many of my friends, whose names are not mentioned here, for making life memorable during my stay at ASU.

TABLE OF CONTENTS

APPENDIX Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1 - Introduction

Semiconductor manufacturing is a highly competitive industry with high volume and rapidly changing demand patterns. To keep up with the market demands and to sustain a competitive edge, semiconductor manufacturing companies concentrate on producing high quality products at a comparatively lower price at a faster speed. Accordingly, semiconductor manufacturing companies compete not only on the traditional metrics of product quality and price, but also increasingly on the basis of lead time and service level, which are both affected by the cycle time of the product. Improving customer service levels, or the probability of delivering on time, has become a very critical issue in this environment that greatly impacts customer satisfaction (Wang and Wang, 2007; Meyersdorf and Yang, 1997), and on-time delivery is also often noted as critical in predicting service levels (Boyaci and Ray, 2006).

Clearly, when focusing on the on-time delivery metric, cycle time plays a very important role (Pfund, Mason, and Fowler, 2006). In concert, accurate estimation of cycle time is crucial, and inaccurate cycle time estimate in semiconductor manufacturing planning can lead to substantial revenue loss (Ankenman *et al*. 2007). In order to obtain these estimates of cycle time, semiconductor manufacturing companies typically develop models.

Chung and Huang (2002) characterized four methods to model and predict cycle time: simulation, statistical analysis, analytical approaches, and hybrid approaches. Simulation models create digital prototypes to predict performance of the real world application. The statistical analysis method is applied to determine the relationship between cycle time and other related parameters with regression analysis or some other statistical analysis approaches. Analytical approaches are based primarily on queueing theory or some other mathematical model to derive the lot cycle time and its deviation. Finally, the hybrid method combines aspects of different approaches to produce a cycle time estimate. For example, the application of analytical methods and simulations in combination could be used to develop a dynamic cycle time estimation.

Atherton and Atherton (1995) argue that computer simulation is the best approach for modeling complicated processes. Researchers apply computer simulation to many areas, including circuit simulation, weather reporting, manufacturing environments, etc., (Johnson et al. 2008). Typically, a researcher performs a computer simulation experiment by making a number of systematic changes to a vector of inputs, $\mathbf{x}$, and observing the corresponding changes to an output measure of interest, y. The aim is to use the model to develop a relationship between $\mathbf{x}$ and y.

Discrete event simulation (DES) is a particular type of computer simulation that utilizes computational and mathematical techniques and tools to model and analyze the performance of systems (Babulak and Wang, 2010). More specifically, DES models represent the stochastic and temporal behavior of a system as it advances through a set of well-defined changes. It can be used to handle almost any level of system detail and can generate detailed performance reports giving almost any performance metric of interest.

The most common applications of DES are in advanced and hybrid manufacturing systems, service sectors like health care and hospitals, banking and finances services, logistics and transportations. Also, DES is used in public sectors like modeling of police emergency response, optimization of armed response vehicle deployment, re-engineering criminal investigation process. There are also opportunities to apply DES to applications in business intelligence and simulation-based education (Babulak and Wang, 2010). It is also commonly used to model the operations in semiconductor wafer fabrication facilities and to perform comparisons between competing current and potential operating policies.

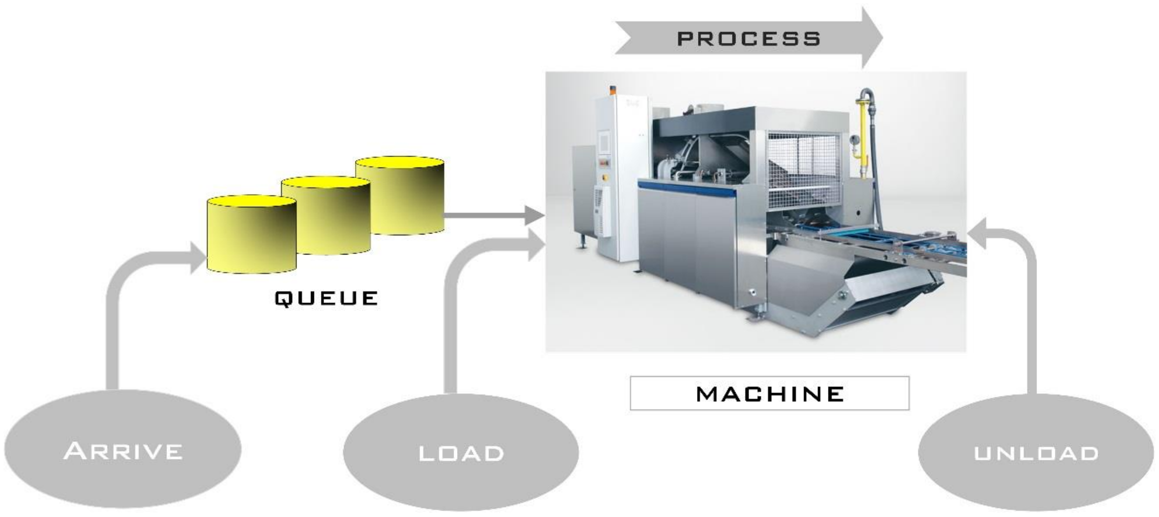


Figure 1. A simple single server system model.

Figure 1 (the figure appears before reference to it) depicts a simple manufacturing process model. It consists of a machine and a queue in a generic factory. The dynamics of the system are as follows. First, a job arrives to the system with the arrival time of t_a, and the job is loaded onto the machine, where it is processed for a time of t_p. The job is then unloaded. Meanwhile, other jobs continue to arrive and wait in the queue until the machine becomes available. After the job is done processing, a job is taken from the queue and begins the process of loading. This logic represents the events of arrival, loading, and unloading, which are typical in DES simulation models of manufacturing systems (Choi and Kang, 2013).

In a DES model of this type, total cycle time is calculated from the variables arrival time and processing time. The average time spent by a job in the system is calculated as the average across the time in system values from all individual jobs and is typically provided to the modeler in an output report. In this research, DES is used to estimate and predict cycle times of a semiconductor manufacturing system.

Semiconductor manufacturing is a very complicated process. Typical characteristics of semiconductor manufacturing systems include fluctuating demand, lots (i.e., groups of individual wafers processed together) with various product types and priorities, unbalanced resource capacity, reentrant flow to the bottleneck machines, hundreds of operation steps, batching, sequence-dependent set-up times, the use of secondary resources, etc. (Chen and Wang, 2009). Given these complexities, when DES models of semiconductor manufacturing systems are simulated and run, it can require substantial use of computer resources and time. The impact of this is even greater when *what-if* analyses are done to predict y, in our case cycle time, at differing levels of the input vector, **x,** that represent potential future states of the system**.** In response, researchers seek approaches that allow the estimation of cycle time at values of **x** without having to apply excessive simulation effort. One approach for this is to develop a mathematical relationship between **x** and y based on simulation output at a well selected set of values from **x**.

1. Metamodeling

Metamodels are literally models of models (Kleijen, 1987). They can be a physical, logical, or mathematical representation of another model or simulation, making them two layers of abstraction away from the real phenomena that they represent (Kerman *et al.* 2009).

Typically, metamodels are simpler approximations of the relationship between **x** and y, constructed based on output from the original simulation models at well selected points within **x**. They are computationally more efficient than the original models themselves (Yin, Ng and Ng, 2010) and allow the prediction of model outcomes at points not simulated, making them a time and cost effective approach for executing *what-if* analyses.

When used for the prediction of system performance, it is important to select an approximation function, *g*, in the metamodeling process. This function allows the prediction of the desired output measure for a given vector inputs, **x.** The relationship between the input vector and output is given in Equation (1), where y represents the desired performance measure, and the randomness of the simulation model is represented by ε. The process of metamodeling involves finding ways to effectively model *g* and ε.

$$y = g(\mathbf{x}) + \varepsilon \qquad (1)$$

A variety of metamodeling techniques exist. The simplest approach is to fit a standard linear regression model based on simulation output at some points in **x** and then to use this model to predict y at **x** values not simulated. Along with linear regression, both response surface methodology and artificial neural networks methods are widely used metamodeling approaches. In the context of simulation–based optimization, Response Surface metamodels (RSM) and Kriging Metamodels (Hernández *et al.*, 2010) are also commonly used. Finally, other statistical techniques such as multivariate adaptive regression splines and radial basis function approximations are beginning to draw the attention of researchers (Jin, Chen and Simpson, 2001).

In all these metamodeling approaches, the focus is most typically on predicting a *mean* performance measure such as mean cycle time (CT). However, quantiles provide a more comprehensive understanding of the CT distribution and are of greater use to decision makers (Koenker and Hallock, 2001). Specifically, having access to quantiles from the CT distribution allows decision makers to assume a known level of risk when quoting lead times to customers. For example, if a customer is quoted the 0.80 quantile of the cycle time distribution as the lead-time, the decision makers can feel confident that the product will be delivered on time in 80% of the cases. *Quantile regression* (QR) metamodeling focuses on predicting such *quantiles*, and this research uses polynomial quantile regression metamodeling for determining the metamodel function, *g* to predict quantiles of the cycle time distribution.

1.1 ***Quantile Regression***

Regression is a statistical method to investigate the relationship between dependent (y) and independent variables (**x**). Standard linear regression provides an estimate of the conditional mean. Quantile regression (QR) (Koenker and Bassett, 1978) is also a statistical method to investigate the relationship between the y and **x**, but the response it predicts is a quantile from the distribution of interest. In other words, standard ordinary regression (least squares, linear, etc.) models the relationships between **x** and conditional mean of y, where QR models relationship between **x** and conditional quantiles of y. It has been shown to provide a comparatively complete and robust analysis of stochastic relationships among random variables by Kerman *et al.* (2008). Kerman *et al.* also compared 7 different UQ (Uncertainty Quantification) methods using five metrics and found ‘quantile regression’ metamodels to be superior to other 6 methods.

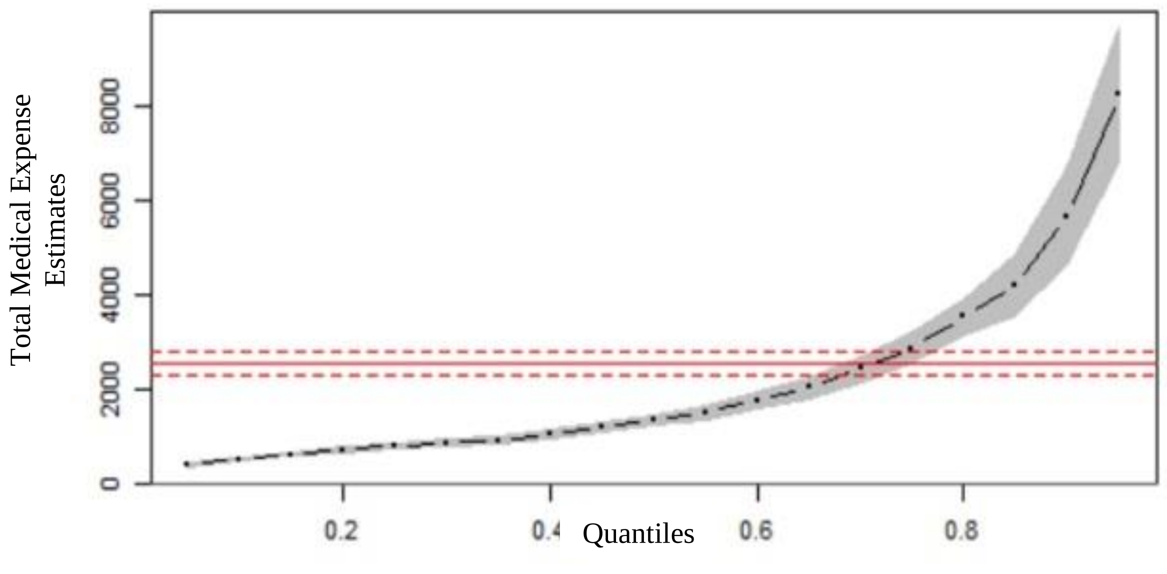


Figure 2. Comparison of TME estimates with least square & QR (Katchova, 2013).

Figure 2 (reference appears after the figure) provides an example comparison of OLS and quantile regression in the context of medical expenses when independent variable **x**, is the total number of chronic conditions. Here, the x-axis represents the quantiles, and the y-axis represents the dependent variable, total medical expenses. The red, solid, horizontal line represents the mean least square regression and dotted lines on both sides of this red line give a 95% confidence interval around the mean. The black curve represents the QR estimates of all quantiles from 0 to 1. For example, the total medical expense estimate for OLS (i.e., mean of distribution) and 0.75 quantile (approximately) of the QR distribution are equal.

In Figure 2, the quantile regression estimates sometimes lie outside the confidence intervals for the OLS regression, suggesting that the effects of these covariates may not be constant across the conditional distribution of the independent variable. The OLS regression confidence interval does a poor job of representing this range of disparities.

Figure 2 illustrates that, particularly for skewed distributions, a much richer picture can be obtained using quantile regression than OLS regression.

The mathematical representation of the metamodel function, *g*, at a given input vectors **x**, and the q^{th}-quantile of the CT output variable y for quantile regression is given in Equation (2). In Equation (2), the CT output variable $y^{[q]}$ is assumed to the distribution $F_{Y,}$ and the *q*-quantile is defined as $y^{[q]} = F_Y^{-1}(q) = \inf\{F_Y(y) \geq q\}$. A strength of the proposed method originates from the fact that no distributional assumptions are made for F_Y, and hence ε.

$$y^{[q]} = g(\mathbf{x}) + \varepsilon \qquad (2)$$

Once a quantile regression metamodel is fit, an equation such as that given in Equation (3) is generated. In Equation (3), βq is the vector of unknown parameters associated with the qth quantile, $\boldsymbol{x}$ is the vector of independent variables, and *y* is the dependent variable to be predicted. Equation (3) utilizes β_q instead of β to make it clear that different choices of *q* estimate different values of β. This research utilizes a polynomial form of QR, and *k* is used to refer to the order of the polynomial function used in fitting the QR.

$$y_i = x_i' \beta_q + e_i, \qquad (3)$$

QR uses linear programming methods to obtain the coefficient estimates. Specifically, it minimizes $\sum_i q|e_i| + \sum_i (1\text{-}q)\ |e_i|$, a sum that gives the asymmetric penalties $q|e_i|$ for underprediction and (1-*q*) $|e_i|$ for overprediction, (Katchova, 2013). The qth quantile regression estimator $\widehat{\beta_q}$ minimizes over β_q the objective function shown in Equation (4). In Equation (4), $0 < q < 1$, and x_i' is a row vector of covariates of the ith data point. The resulting regression fit, $x_i'\beta_q$, is an estimate of the q-quantile of the response variable y given **x**$_i$.

$$Q(\beta_q)=\sum_{i:y_i\geq x_i'\beta}^{N} q\left|y_i - x_i'\beta_q\right| + \sum_{i:y_i<x_i'\beta}^{N}(1-q)\left|y_i - x_i'\beta_q\right|, \tag{4}$$

In this research, QR metamodels are fit also utilizing the Lasso penalty, λ. In Equation (5), λ is the lasso penalty, also called shrinkage parameter, and the summation includes the coefficients of the regression model excluding the intercept (Tibshirani, 1996). The lasso penalty, λ, is not unique to QR; it can be applied to any regression model and imposes a penalty for the terms entering the model. For larger values of λ, many components of β_q are estimated to be zero. As λ shrinks to zero, the estimates of β_q move toward an unpenalized estimate. Of additional importance to this research is that the lasso penalty permits the inclusion of correlated predictor variables, which are not allowed in quantile regression without the lasso penalty. In the polynomial linear regression models used in this research, correlated predictor variables are used to predict quantiles of the CT distribution from semiconductor manufacturing systems. The lasso penalty, λ, with the QR accommodates the presence of these predictor variables.

$$\Gamma\,(\beta_q;\lambda) = Q\,(\beta_q) + \lambda \sum_{j=1}^{c}\left|(\beta_q)_j\right|, \tag{5}$$

This research includes some attention to the order of the polynomial function from the QR fit, *k*, and the lasso parameter, λ as key parameters in the QR metamodeling procedure. These parameters are important because, for the same set of data used to fit the QR model, model fits with different (k, λ) combinations produce metamodels of differing quality (i.e., metamodels with differing predictive accuracies).

2. Experimental design strategy

When building metamodels, verification to establish that the obtained predictions are sufficiently accurate is required. In other words, the metamodels should accurately predict the performance of the simulation model. Influencing the predictive accuracy of the metamodels is the placement of design points within the experimental region. The placement of design points over the design region is referred to as the experimental design strategy.

At a very general level, the experimental design facilitates the investigation of the relationship between a response and the factors that influence the response in order to determine the underlying mechanism governing the process under study (Hunter and Naylor, 1970). It is also useful for finding the combination of factor levels at which the response variable is optimized. The main objective of an experimental design strategy is to explore and describe the response surface over the experimental region / space using the observations of the response at various factor levels of the data. Experimental designs are not only created to provide the economy in the required number of experimental trials (n), but also to maximize additional qualities such as producing minimum variance estimates of the response (Hunter and Naylor, 1970).

In the context of metamodeling, it is important to utilize an appropriate experimental design strategy so that information on the response variable obtained from the simulation variable be effectively and efficiently used to predict the response variable at points not simulated. Much previous research has evaluated experimental design strategies and analysis methods for computer simulations in the context of metamodeling. For example, Hunter and Naylor (1970) discussed various experimental designs in the context of computer simulation; Challeno (2013) considered the design of experiments for the validation of the fitted metamodel, and Yin *et al.,* (2010) proposed a 'Bayesian' metamodeling approach for stochastic simulations.

One of the more conventional design strategies is a factorial design in which all factors are varied by a set amount. Such design strategies are very inefficient (i.e., requires intensive simulation effort) and for high dimensions can be virtually impossible to carry out (Challenor, 2013). Another experimental design strategy is a space-filling design, which attempts to fill a high dimensional space in an efficient way. Many types of space filling designs have been proposed in the last 30 years (Johnson *et al*. 2008), including sphere-packing designs, Latin hypercube designs, and uniform designs. Finally, if experimenters are considering a polynomial model to describe the underlying relationship between y and **x**, then an optimal design such as a D-optimal or I-optimal design can also be used. Design strategies using the D-optimal criterion focus on minimizing the variances of the model coefficients, while design strategies using the I-optimal criterion focus on minimizing a measure of average prediction variance (Montgomery, 2009).

Johnson, Jones, Fowler, and Montgomery (2008) compared several experimental design strategies including, optimal, space filling and maximum entropy designs for computer simulation experiments. Specifically, Johnson *et al.,* (2008) found that space-filling designs exhibited high variability with respect to prediction variance performance across the design region, and sphere packing designs were generally the best space-filling designs in terms of prediction variance. They also noted that I-optimal designs had the best prediction variance properties among all the other designs.

However, when prediction *accuracy* (vs. variance) is considered, because of the placement of design points throughout the experimental region, space filling design strategies are expected to have better predictive accuracy than optimal designs. This points to the possibility of augmenting both space filling and optimal designs to generate hybrid experimental design. Highly related to the work presented here, Kennedy (2013) discusses different methodologies for combining these two popular design strategies (space-filling & optimal design strategies) to generate hybrid designs for use in the field of computer experiments. Kennedy also compared these composite designs to pure space-filling and optimal designs to analyze how positive properties of each design are retained while mitigating potential weaknesses. Kennedy's dissertation paper compares the designs with prediction variance metrics. The results conclude that, hybrid designs have performed better than space filling designs but performed less than I-optimal design. In general, there is an improvement in performance with hybrid designs. In a similar context, Johnson, Montgomery, Jones and Parker (2010) demonstrate that augmenting a space-filling design with optimal points can be effective in improving the prediction variance across the design region.

This research will investigate the effectiveness of sphere-packing, I-optimal and hybrid design strategies in the context of their impact on the predictive accuracy obtained with quantile regression metamodeling. Hybrid designs are augmentations of sphere packing and I-optimal experimental design strategies on their own and are designed to take advantage of the strengths of each of the two design strategies. Notably, while the quantile regression metamodeling technique is being used increasingly, this research is among the first which is interested in investigating the impact of various experimental design strategies on its performance as a metamodel for computer experiments. The remainder of this thesis will provide details about the semiconductor manufacturing simulation model used to conduct experimentation, the experimental plan and the methodology of the experiments, and the results obtained. Finally, discussions, conclusions, and directions for future work are given in Chapter 4.

Chapter 2 – EXPERIMENTAL PLAN AND METHODOLOGY

As discussed in the previous chapter, semiconductor manufacturing industries focus on accurate cycle time (CT) estimation, as it plays a very important role in the on-time delivery metric. In order to obtain these estimates of cycle time, semiconductor manufacturing companies typically develop discrete event simulation (DES) models. Discrete event simulation (DES) is utilized in this research to estimate and predict cycle times of a semiconductor manufacturing system. However, these models require substantial use of computer resources and time to execute. To overcome this, polynomial quantile regression metamodels are developed.

The placement of design points within the experimental region, or the experimental design strategy, affects the predictive accuracy of the developed metamodels. This research will investigate the effectiveness of sphere-packing, I-optimal, and hybrid[1] experimental design strategies in the context of the predictive accuracy obtained with quantile regression metamodeling. The overarching research questions addressed through this work are given next.

1. How does the placement of design points within the experimental region influence the predictive accuracy of quantile regression metamodels built based on simulation output generated at the design points?
2. How does the density of the design points within the experimental region (i.e., the number of design points included within the experimental region) influence the

[1] Hybrid designs are the augmentation of space-filling design strategies with I-optimal design strategies. See sub section 2.2.3 - Hybrid design for more detailed explanation

predictive accuracy of quantile regression metamodels built based on simulation output generated at design points?

3. Does a hybrid design, which is augmentation of standard I-optimal and sphere packing designs, work better than either sphere-packing or I-optimal design or both in terms of producing greater predictive accuracy for quantile regression metamodels?

The remainder of this chapter describes the simulation model used to generate an experimental testbed for the analysis and results to support the research questions, the experimental design strategies themselves, the quantile regression metamodeling process used to develop the resulting metamodels, and a description of the approach used to evaluate the predictive accuracy.

1. Simulation Model – Minifab Model

The Minifab model is a simulation model designed to imitate the key characteristics of a semiconductor manufacturing processes in a simple format. The model used in this research includes most of the components from the originally developed Minifab model, designed by Intel in collaboration with ASU (Dr. Kempf, 1994). This Minifab model has also been used by other researchers (Chen and Kelton (2012)) for evaluating different aspects of semiconductor manufacturing. In this research, the Minifab model is used specifically as a vehicle to evaluate the impact of different experimental designs on the predictive accuracy of quantile regression metamodeling for predicting quantiles of the cycle time distribution produced by the Minifab model. The simulation program ARENA, by Rockwell Automation, was used to create the model for this work, and the “. mod” and

“. exp” files, which provide the code used to execute the model, is contained within the appendix.

The Minifab model has two types of products flowing through the system, Type X and Type Y. The distribution surrounding the time between arrivals for both the part types follows an exponential distribution. For product Type X, various expected time-between-arrival values are utilized in experimentation, ranging from 0.002 to 0.005 parts per minute, and the expected arrival rate for product Type Y is set to 0.003 parts per minute.

There are three tool groups in the model. Tool Group 1 is similar to a diffusion oven in an actual semiconductor manufacturing setting, Tool Group 2 is similar to a photolithography stepper, and Tool Group 3 is similar to an ion implanter. Both the products flow through the system visiting each tool groups twice in the order shown in Figure 3. The numbers above the arrows represent the process step.

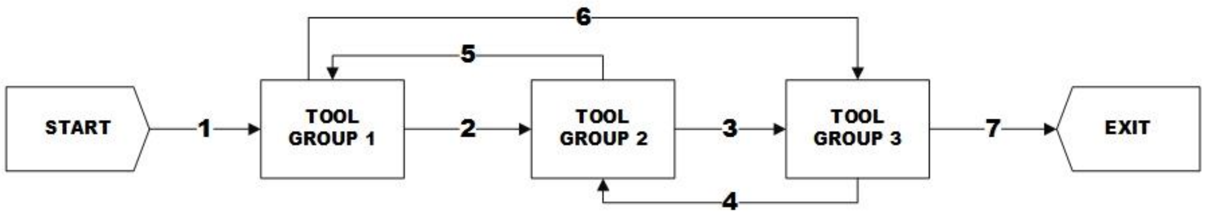


Figure 3. Product flow through the Minifab model

Tool Group 1 has two identical, parallel processing machines, named as Machine A and Machine B. Similarly, Tool Group 2 also has two identical parallel machines, named as Machine C and Machine D. Tool Group 3 has a single machine named as Machine E. At each tool group, there is a single operator who serves as a secondary resource and is utilized for loading, unloading and machine set-up operations. Also, after completing processing at Tool Group 2, 2% of the products fail quality and require rework to strip off the photoresist. For this rework, there is a single rework station and a single operator to handle all the rework operations. The processing time at the rework machine is 50% of the processing time in Tool Group 2 machine. Tool group 1 also requires parts to be batched before processing. The batch size is three, and batching is done based on the following rules:

- Parts arriving for Step 1 processing: Different part types (Type X, Type Y) can be mixed together to form a batch.
- Parts arriving for Step 5 processing: Different part types cannot be mixed together. Separate batches are to be formed for two part types.
- Parts waiting for Step 1 and Step 5 can never be batched together.
- After processing, all the batches are separated into original parts.

Table 1 shows the processing time, batch size, and duration of loading and unloading operations at the machines in each particular tool group. The processing times at each machine follow a normal distribution, with an assigned coefficient of variation. The operator has to load the part (or batch) into the machine and unload it as soon as the part (or batch) has been processed. Only after the processed part (or batch) is unloaded can a new part (or batch) be loaded into the machine.

Machine E requires sequence-dependent setup when changing between product types or between steps. The length of setup time varies based on the process step and part type. Setup times are modeled with a normal distribution with a 0.5 coefficient of variation. When two sequential parts to be processed are the same part type, but they are on different processing steps, the expected setup time is 10 minutes. When two sequential parts are different part types but are on the same processing step, then the expected setup time is 5 minutes. When two sequential parts are both different part types and on different processing steps, then the expected setup time is 12 minutes. If the two sequential parts are there for the same step and are the same part type, no setup is required (i.e., the setup time is zero).

Table 1. Characteristics of the Minifab Model (Distribution included is Normal (Mean, Standard deviation))

Process Step	Tool Group	Processing Time (min)	Batch Size (parts)	Load Time (min)	Unload Time (min)
1	1 (A,B)	Normal (225, 11.25)	3	Normal (20, 2)	Normal (40, 4)
2	2 (C,D)	Normal (30, 1.5)	1	Normal (15, 1.5)	Normal (15, 1.5)
3	3 (E)	Normal (55, 2.75)	1	Normal (10, 1)	Normal (10, 1)
4	2 (C,D)	Normal (50, 2.5)	1	Normal (15, 1.5)	Normal (15, 1.5)
5	1 (A,B)	Normal (255, 12.75)	3	Normal (20, 2)	Normal (40, 4)
6	3 (E)	Normal (10, 0.5)	1	Normal (10, 1)	Normal (10, 1)

All the machines in the Minifab model require preventative maintenance every 7 days, and each preventative maintenance session takes 1 hour. Also, all machines require a condition check every 30 days, and each check takes 6 hours. The machine in Tool Group 3 also requires emergency maintenance. The time between failures of Tool Group 3 is modeled with an exponential distribution with an expected value of 50 days. The time to repair the machine after a failure is modeled with a gamma distribution with a scale parameter of 864 minutes and a shape parameter of 0.25. The First-In-First-Out (FIFO) dispatching policy is employed at all workstations.

Finally, the following assumptions regarding the model are also made during the construction of the model.

1. The time taken to transport a job from one tool group to other tool group is considered negligible and is modeled as taking zero minutes.
2. The operators work continuously.

Finally, due to the stochastic nature of the Minifab model, two different runs with the same input parameter specifications will yield two different results. As a result, output measures (e.g., cycle time) from the Minifab model are also stochastic, random variables. To account for the assumption that dependent measures are independent, identically distributed, which is commonplace among standard statistical analysis approaches, both bias due to initial empty and idle starting conditions and autocorrelation caused by the nature of queueing systems in the model need to be accounted for. Accounting for initialization bias helps ensure that the output measures are identically distributed, and accounting for the auto-correlation ensures that the output measures are independent.

To remove the impact of initialization bias, 200,000 time units of data are truncated at the beginning of every simulation run. This truncation point was determined based on plots of the cumulative moving average of cycle time vs. the simulation run time. To address autocorrelation, a lag of 300 between data points collected for analysis is utilized (i.e., only every 300th cycle time observations is used in analysis). This lag value was determined based on a plot of auto-correlation vs. lag length.

2. Methodology

2.1. Simulation Experiments

To address the proposed research question, three simulation experiments were conducted based in the Minifab model. In all the simulation experiments, the output variable Y is q^{th}-quantile of cycle time distribution, and the input variables, **x**, are controllable factors of the Minifab model. This research considered four input variables: 1) throughput, controlled by TBA (Time Between Arrivals), 2) COV (coefficient of variance) of the unloading operations at all the machines, 3) MTBF (Mean Time Between Failures) for emergency maintenance at Machine E, which is at the Tool Group 3 and 4) MTTR (Mean Time to Repair) for emergency maintenance at Machine E, which is at the Tool Group 3. These factors were selected because they are known to influence the cycle time distribution and represent factors that could be controlled or influenced by a production manager.

In each of the three simulation experiments, two input variables were manipulated to generate the simulation metamodel. In all three, $\mathbf{x_1}$ is the time between arrivals (TBA) of Part A entities, while $\mathbf{x_2}$ represents one of the other input variables. Table 2 describes which factors were considered for each simulation experiment, while Table 3 shows the range of values examined for each input variable. These ranges were selected to align with a published proof of concept for the quantile metamodeling approach as applied to computer simulation (Bekki, Chen, and Batur, 2014) and represent the boundaries of the experimental region.

Table 2. Factors considered for each simulation experiment.

	TBA	**COV**	**MTBF**	**MTTR**
Simulation Experiment-1	**x_1**	**x_2**	**-**	**-**
Simulation Experiment-2	**x_1**	**-**	**x_2**	**-**
Simulation Experiment-3	**x_1**	**-**	**-**	**x_2**

Table 3. Range and average values of each factor.

Factor	**Range**	**Average**	**Units**
TBA	[0.002-0.005]	0.0035	Parts per min
COV	[0.1,0.9]	0.5	
MTBF	[10,40]	25	days
MTTR	[108,324]	216	min

2.2. *Experimental Design Strategies*

The location and density of design points and prediction points in the experimental design region is an important consideration in the development of a metamodel. Five specific experimental design strategies were through this work: sphere packing, I-optimal, and three hybrid designs, which combine features from the I-optimal and sphere packing design strategies. In addition, for each experimental design strategy, the number of included design points included in the experimental design, N, was varied to be N= 6, 10, 15, or 25. Details on each of the design strategies are given next.

2.2.1. *Sphere Packing Design*

Space-filling design strategies are often thought to be particularly appropriate for deterministic computer models because they spread the design points out nearly uniformly throughout the experimental region to maximize the distance between any two-design points (Myers, Montgomery and Anderson, 2010). The sphere packing design strategy is a space filling design strategy that maximizes the minimum distance between pairs of design points. This maximization helps in spreading the design points all over the design region.

The software tool JMP was used to generate the experimental designs for the sphere packing design strategy. Using JMP, a table with the appropriate number design points is easily generated. For example, for simulation experiment-1, Tables 4 gives the sphere packing design strategy for 6 design points. Tables 1-11 in the appendix shows the sphere packing design points at all densities (N = 6, 10, 15, 25) for each of the three simulation experiments. Figure 4 displays the distribution of the 10 design points of the sphere packing design in the experimental region for simulation experiment -1.

Table 4. Design points of sphere packing design for simulation experiment-1 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00297	0.69176
2	0.00395	0.9
3	0.005	0.7137
4	0.00403	0.1
5	0.002	0.1
6	0.002	0.9

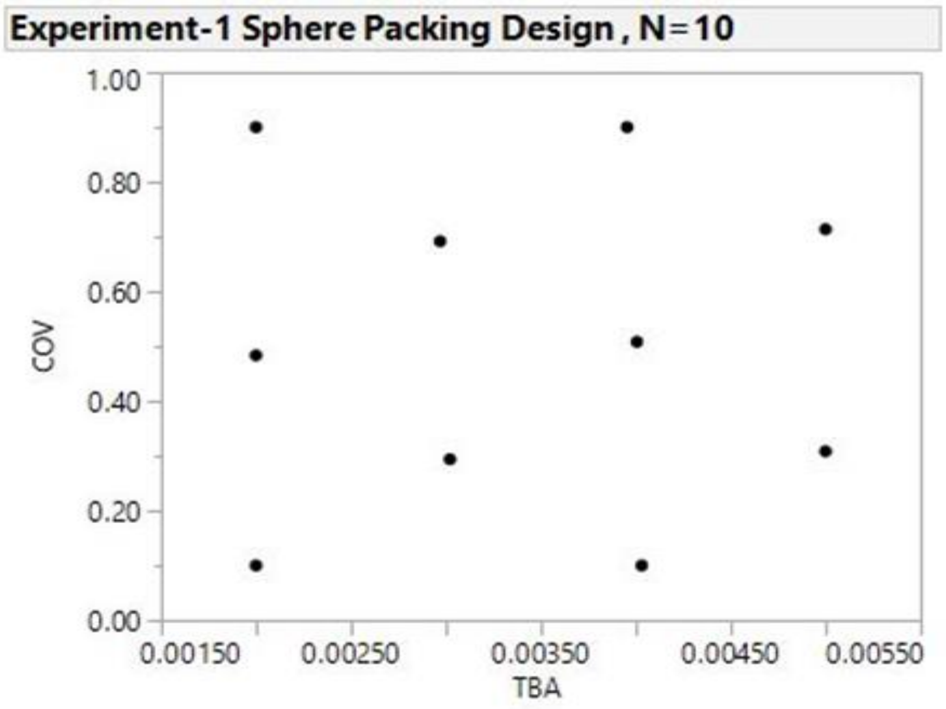


Figure 4. Distribution of 10 design points of sphere packing design for simulation experiment -1.

2.2.2. *I-Optimal Design*

I-optimal design strategies minimize the prediction variance in a design space. This design strategy is more commonly applied when a form of the model is already known, and a benefit of I-optimal design strategies is that they have the best prediction variance properties of any design (Johnson *et al.*, 2008). The JMP software tool was used to generate the design points for I-optimal design strategy. Tables 5 gives the I-optimal design strategy at 6 design points for simulation experiment-1. Tables 12-22 in the appendix shows the I-optimal design points at all densities (N = 6, 10, 15, 25) for each of the three simulation experiments. Figure 5 displays the distribution of 10 design points of the I-optimal design in the experimental region for simulation experiment -1.

Table 5. Design points of I-optimal design for simulation experiment-1.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0035	0.5
2	0.002	0.9
3	0.0035	0.1
4	0.005	0.9
5	0.005	0.1
6	0.002	0.212

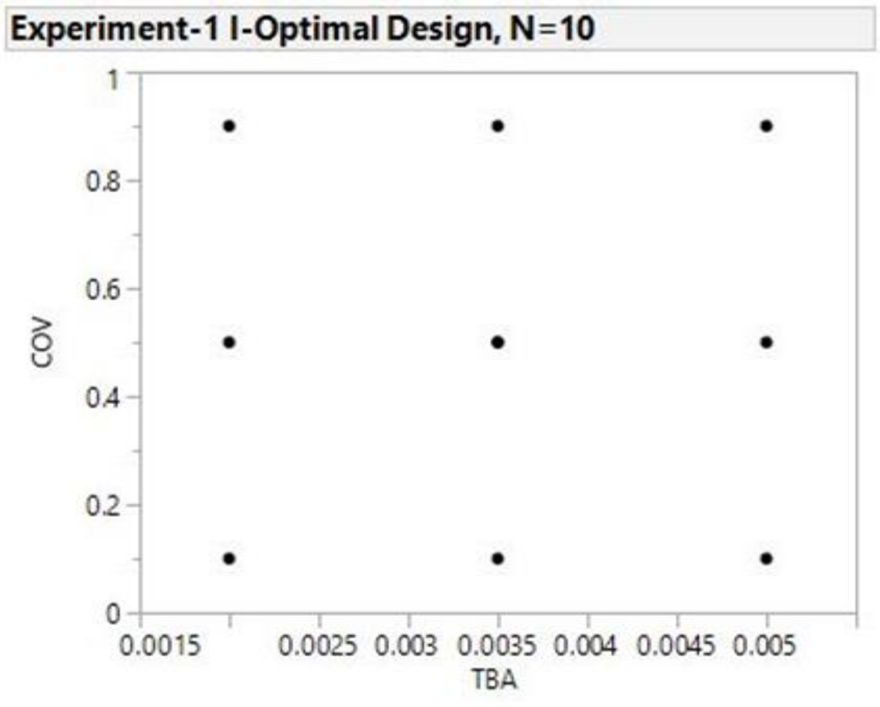


Figure 5. Distribution of 10 design points of I-optimal design for simulation experiment1.

2.2.3. *Hybrid Designs*

Augmentation of space-filling designs with I-optimal points is appropriate when initial modeling indicates that the computer simulation model can be adequately approximated by a polynomial. (Kennedy, 2013). The concept behind such hybrid designs is to combine the beneficial qualities of the space filling and optimal design strategies within a single design strategy. The design points in the hybrid design strategies used in this research represent a combination of sphere-packing and I-optimal design points.

There are number of methods by which the design points from the original design strategies could be combined into a hybrid design. An example for a hybrid design is shown in Figure 6, which shows the placement of design points for a hybrid design with 15 design points. Specifically, the design points are for simulation experiment - 1 with TBA on x-axis and COV on y-axis. The blue points are those that were selected from the original sphere packing design points, and orange points are those that were selected from the original I-optimal design points. I-optimal design strategy points are distributed at the four corners of the design space and on center points of line that is drawn connecting any two points of the four corner points. The design points of space filling design strategy, on the other hand, are distributed all over the design region, approximately uniformly. This research examines three approaches for generating a hybrid design, each of which will be discussed next in more detail: arbitrary hybrid method, clustering hybrid method, and centroid clustering method.

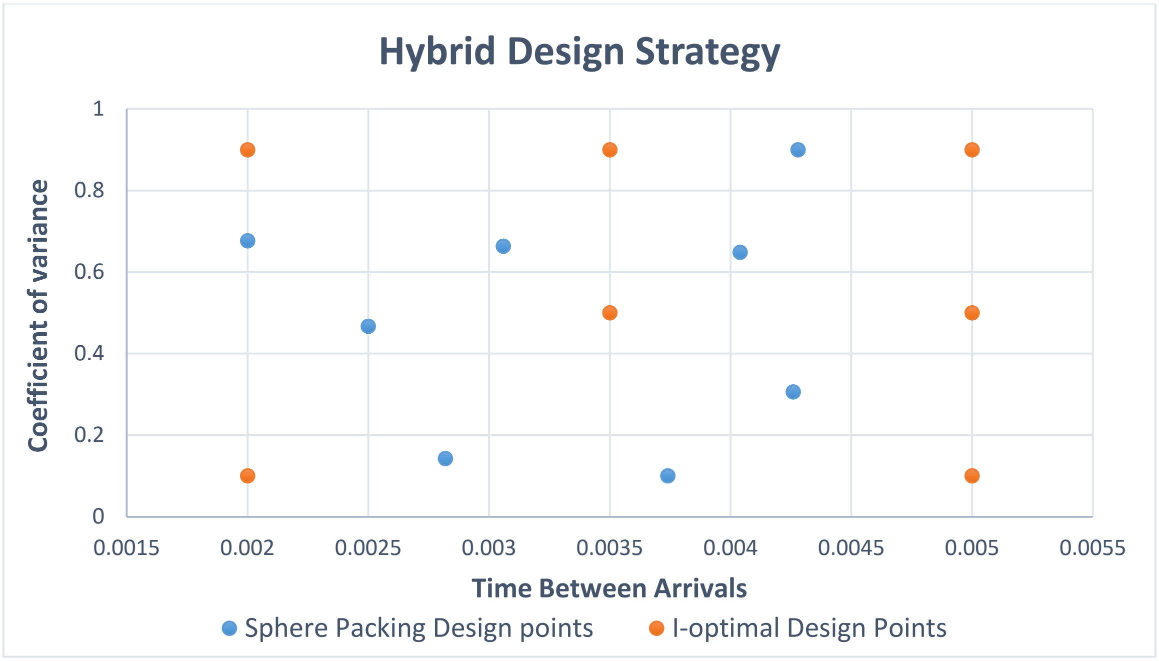


Figure 6. Scatter plot of hybrid design strategy with N = 15 design points.

2.2.3.1. *Arbitrary Hybrid Design Strategy*

To implement the arbitrary hybrid design strategy, all the points from the sphere packing and I-optimal design strategies were plotted on the common design space. Then, the repeated design points were first eliminated. Next, the points that were very near to the optimal design strategy points were eliminated until the required number of design points remained. The ratio of the number of design points from sphere packing to the number of design points from the I-optimal design considered for the arbitrary hybrid design depends on the total number of design space. In majority of the cases, the ratio is more than one, as points in the I-optimal design are repeated. Tables 6 gives the arbitrary hybrid design strategy at 6 design points for simulation experiment-1. Tables 23-33 in the appendix shows the arbitrary hybrid design points at all densities (N = 6, 10, 15, 25) for each of the three simulation experiments. Figure 7 displays an example distribution of 10 design points of arbitrary hybrid design in the experimental region for simulation experiment -1.

Table 6. Design points of arbitrary hybrid design for simulation experiment-1.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.004	0.5
2	0.003	0.9
3	0.005	0.9
4	0.005	0.1
5	0.0035	0.1
6	0.002	0.212

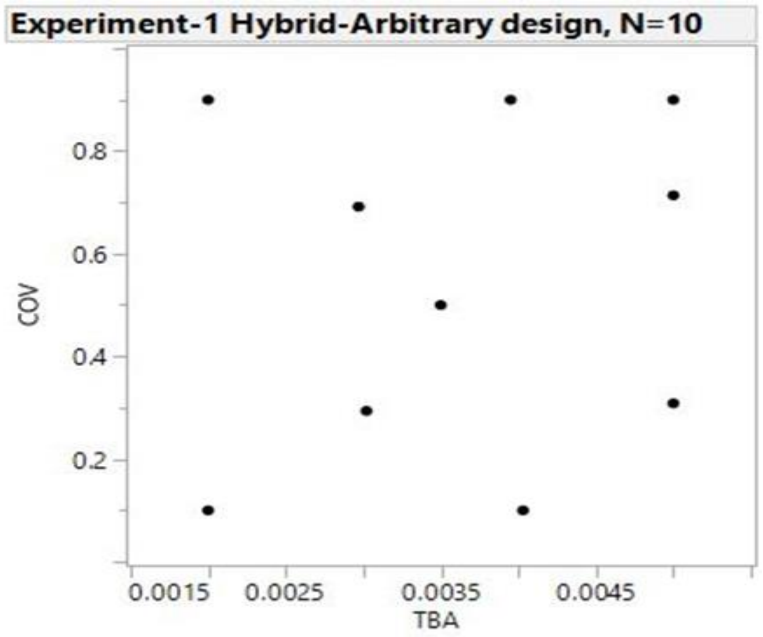


Figure 7. Distribution of 10 design points of arbitrary hybrid design for simulation experiment - 1.

2.2.3.2. *Clustering Hybrid Design Strategy*

To implement the clustering hybrid design strategy, all the points from the space filling and optimal design strategies were first considered. Based on this set of design points, clusters of design points are first created, in which the number of clusters was equal to the desired number of design points, N. The 'simple K-Mean' clustering module in the Weka tool was used to create the clusters. Weka is a collection of machine learning algorithms for data mining tasks. Here, K indicates the number of clusters. After creating the clusters, the design point closest to the centroid of the cluster was chosen as the representative design point for that cluster. An example of the design point placement for a clustering hybrid design with N = 6 is shown in the Figure 8. Each of the colors in Figure 8 represent a separate cluster, and the enlarged design point is the point from that cluster (i.e., from among those points of the same color) that was selected for inclusion in the experimental design. Tables 7 gives the clustering hybrid design strategy at 6 design points for simulation experiment-1. Tables 34-44 in the appendix shows the clustering hybrid design points at all densities (N = 6, 10, 15, 25) for each of the three simulation experiments.

Table 7. Design points of clustering hybrid design for simulation experiment-1.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00403	0.1
2	0.005	0.9
3	0.00395	0.9
4	0.002	0.212
5	0.002	0.1
6	0.0035	0.5

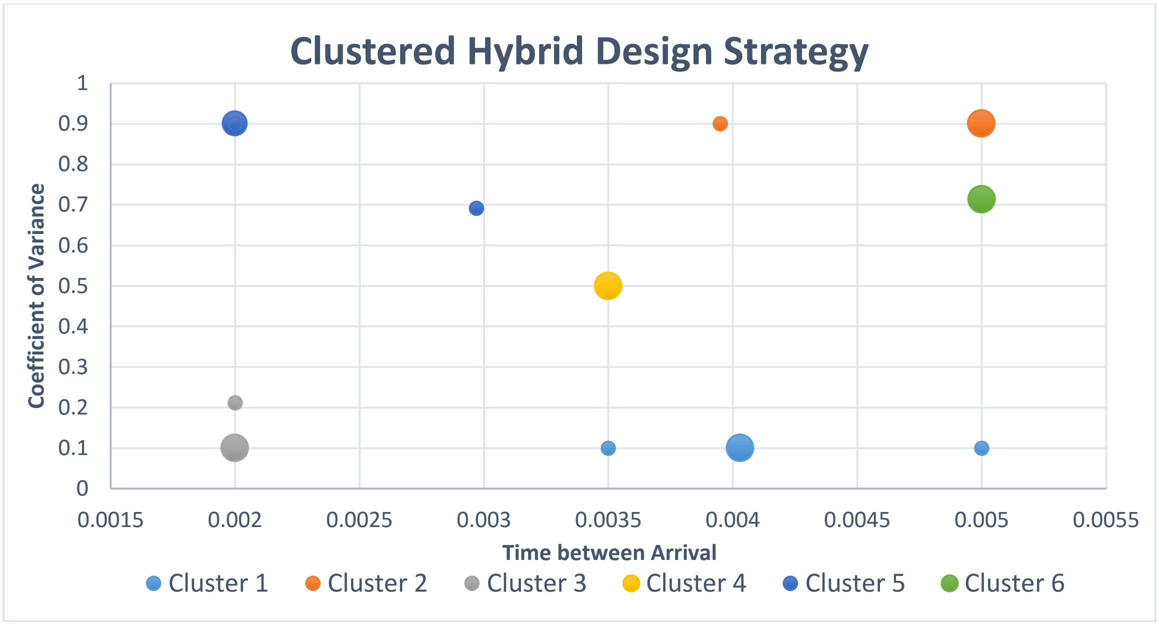


Figure 8. Scatter plot of clustered hybrid design strategy with 6 representing design points.

2.2.3.3. *Centroid Clustering Hybrid Design Strategy*

The centroid clustering experimental design strategy is similar to the clustering method. To implement the design strategy, all the points from the space filling and optimal design strategies were again first considered. The simple K-Mean clustering method was also again used to form the clusters. However, in contrast to the clustering approach, in the centroid clustering approach, the actual centroid point of the cluster is considered as the design point (vs. the original design point that is closest to the cluster). The number of clusters is equal to the number of design points, so the centroids of each of the clusters become the design points for the centroid clustering hybrid design. Tables 8 gives the centroid clustering hybrid design strategy at 6 design points for simulation experiment-1. Tables 44-54 in the appendix shows the centroid clustering hybrid design points at all densities (N = 6, 10, 15, 25) for each of the three simulation experiments. As an example, Figure 9 displays the distribution of 10 design points of centroid clustering hybrid design in the experimental region for simulation experiment -1.

Table 8. Design points of centroid clustering hybrid design for simulation experiment-1.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0042	0.1
2	0.0045	0.9
3	0.002	0.156
4	0.0035	0.5
5	0.005	0.7137
6	0.0023	0.8306

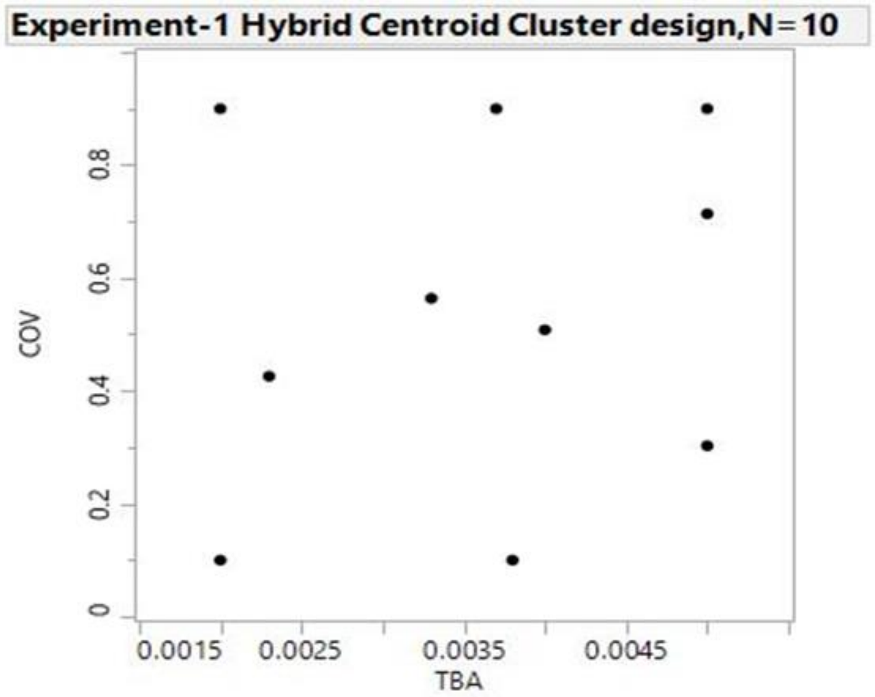
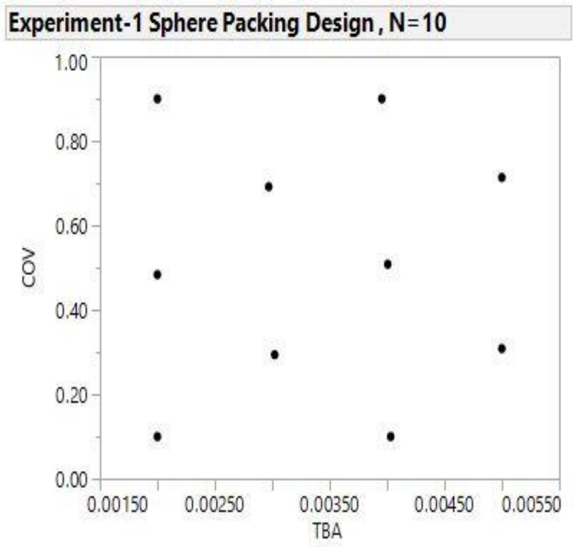
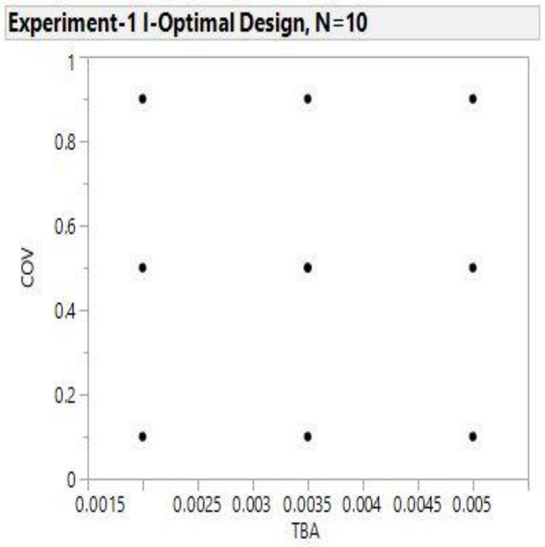


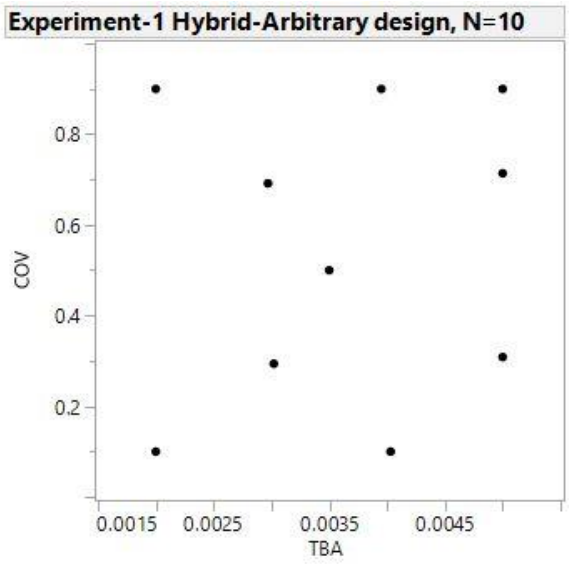
Figure 9. Distribution of 10 design points of centroid clustering hybrid design for simulation experiment - 1.

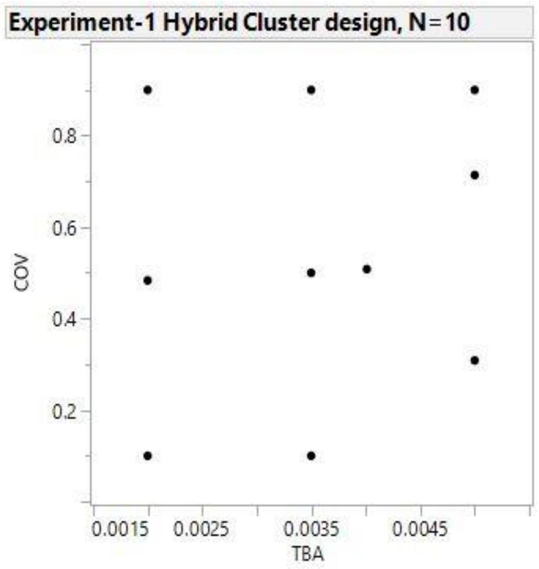
2.2.4. *Experimental Design Strategy Summary*

Figure 10 gives an illustration of how the placement of design points differs across the experimental design strategies. On all charts in this figure, the two factors (time between arrivals and coefficient of variance of the unloading operations at all the machines are from simulation experiment 1. The horizontal axis gives the time between arrivals (TBA), while the vertical axis gives the coefficient of variance at unloading operations of machines. In all plots shown in Figure 12, N = 10.









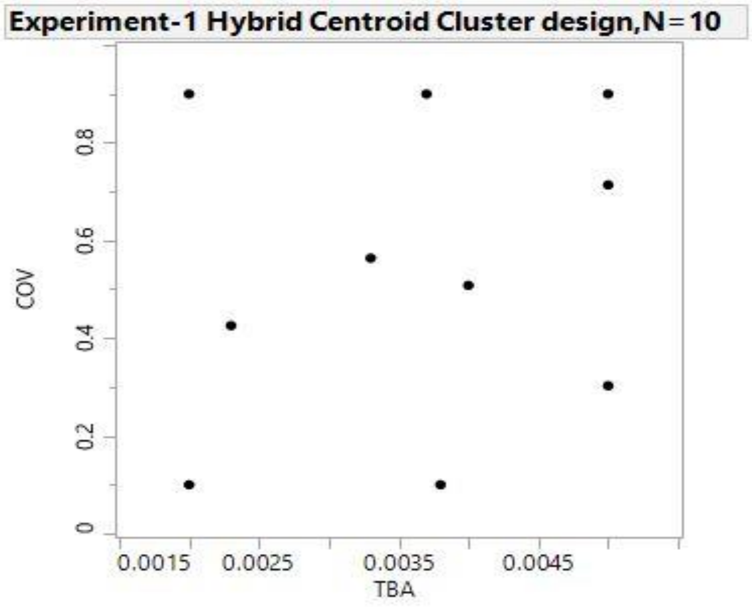


Figure 10. Placement of design points in the design region for different designs.

2.3. *Other Experimental Factors*

Along with conducting three simulation experiments using each of the five experimental design strategies, the k and λ parameters need to be specified when generating a quantile regression (QR) metamodel. As was pointed out previously, with the same experimental design strategy, different (k, λ) combinations result in quantile regression metamodel fit of different qualities. Based on previous work (Bekki, et al., 2014), the following four (k, λ) combinations were considered in this work: (2,100), (2,0.1), (3,100), (3, 0.1). To understand the impact of the design strategies on the predictive accuracy of quantile regression metamodels developed to varying quantiles of the cycle time distribution, metamodels were developed to predict the following specific quantiles of the cycle time distribution from the Minfab model: 0.5, 0.8, 0.9, and 0.95.

The overall experimental design plan, then, includes three simulation experiments. For each of these, five experimental design strategies were utilized to create experimental designs with N = 6, 10, 15, and 25. Consequently, for each computer simulation, there were twenty different experimental designs (five different experimental design strategies x four different numbers of design points). Additionally, quantile regression models were fit for each of the 20 experimental designs at four different (k, λ) combinations: (2, 100), (2, 0.1), (3, 100), and (3, 0.1). Finally, for each (k, λ) combination at each experimental design, quantile regression fits were obtained to predict the 0.5, 0.8., 0.9, and 0.95 quantile. Therefore, for each simulation experiment, 320 quantile regression fits were made (20 experimental designs x four (k, λ) combinations x four quantiles). Figure 11 provides an illustration of this experimental plan. In Figure 11, the experimental hierarchy is provided for only a single pathway. However, the same procedure is followed at each experimental design strategy and each simulation experiment.

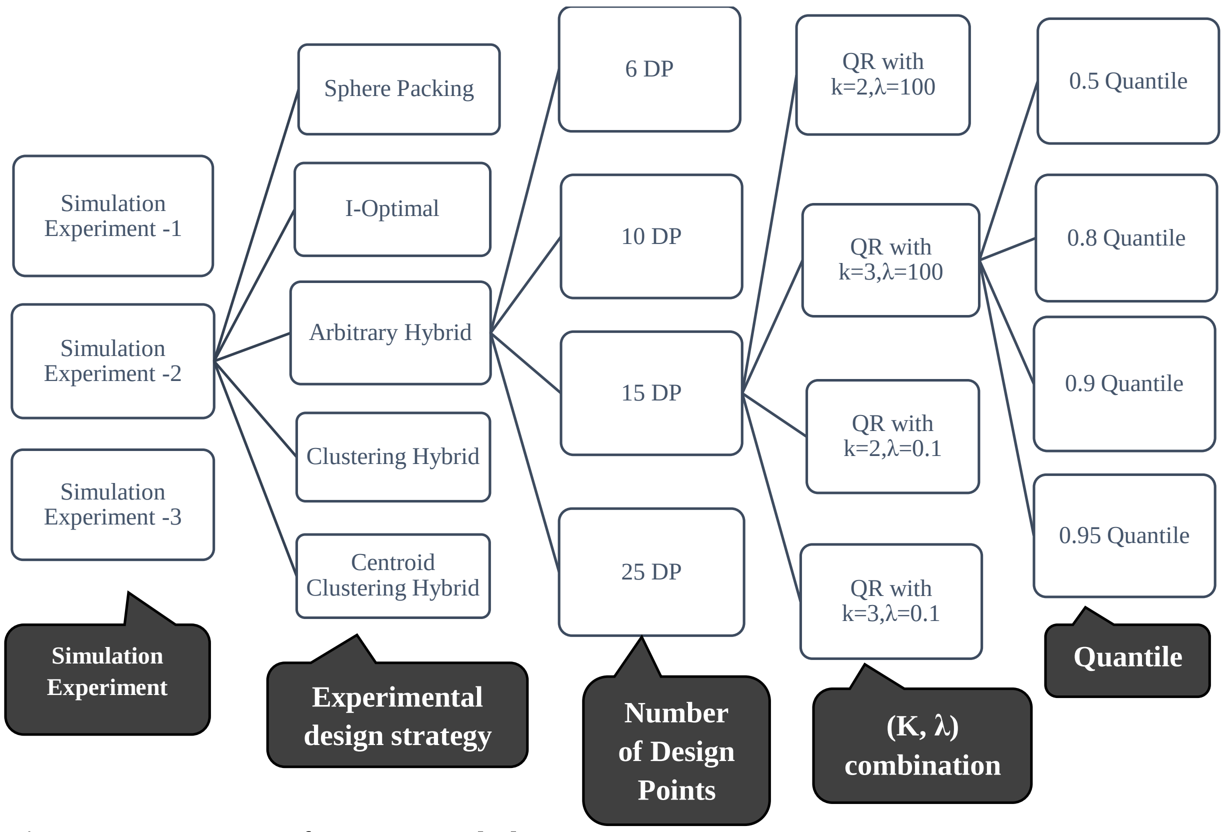


Figure 11. Summary of experimental plan.

2.4. *Generating the Quantile Regression Metamodel*

In order to be able to evaluate the efficacy of each of the experimental design strategies, quantile regression metamodels need to be fit, and their predictive accuracy evaluated. This subsection describes how these processes were conducted for this research.

The first step in fitting a quantile regression (QR) is to decide on the location of the design points and to execute the simulation model at these points. At each design point, across all the experimental design strategies, 5000 cycle time observations were obtained from the Minifab model. The value of 5000 was determined by observing the output quantile variance at different number of cycle time observations, ranging from 500 – 1M. After 5000 cycle time observations were obtained, the quantile variance became negligible, indicating that the collection of additional cycle time observations would not add additional value to the cycle time quantile predictions.

After executing the simulation model, the input variables (e.g., throughput and COV) and output variable (i.e., cycle time) data was standardized so that each variable had a mean of zero and a standard deviation of one. Next, the quantile to be estimated (i.e., 0.5, 0.8, 0.9 or 0.95) was determined. With the simulation data and QR modeling parameters, then, the "quantreg" package within the R programming tool was used calculate the coefficients of QR equation. As described previously, four (k,λ) combinations were investigated for this work: (2, 100), (2, 0.1), (3, 100) and (3, 0.1). The R program calculates the coefficients of the quantile regression model for each of the four, associated equations. The equations for these second order and third order quantile regression equations are given in Equations 6 and 7, respectively. In these equations, y is the cycle time output, β_i are coefficients obtained from the experimental design, $\mathbf{x_1}$ and $\mathbf{x_2}$ are the two input variables, and ε is the constant. The R-code used to execute the "quantreg" package can be found in the appendix.

$$\mathbf{y = \beta_0\, x_1 + \beta_1\, x_2 + \beta_2\, x_1\, x_2 + \beta_3\, x_1^2 + \beta_4\, x_2^2 + \varepsilon,} \tag{6}$$

$$\mathbf{y = \beta_0\, x_1 + \beta_1\, x_2 + \beta_2\, x_1\, x_2 + \beta_3\, x_1^2 + \beta_4\, x_2^2 + \beta_5\, x_1^2\, x_2 + \beta_6\, x_1\, x_2^2 + \beta_7\, x_1^3 + \beta_8\, x_2^3 + \varepsilon,} \tag{7}$$

At the conclusion of this process, quantile regression metamodels (one for each (k, λ) combination) are obtained for a particular quantile. By inserting values of the input variables into one of these equations, an estimate for the desired cycle-time quantile, y, can be obtained.

2.5. *Assessing the Predictive Accuracy of a Quantile Regression Metamodel*

In order to address the research questions posed in this thesis, an approach for evaluating the predictive accuracy of a QR metamodel (generated from a particular experimental design strategy) is required. To facilitate this, a measure of the "true" cycle time quantiles needs to be obtained to which the estimates of those same quantiles generated by the quantile regression metamodel can be compared. In such a situation, QR models that make predictions closer to the "true" cycle time quantiles are considered superior.

Prediction points serve as these estimates of the 'true' cycle time quantiles values to which this research will compare the values predicted by the QR metamodels. The estimates obtained using prediction points are based on order statistics-based estimates from very long simulation runs, which is a standard approach for estimating accurate quantile estimates. To obtain these estimates, the simulation output data is first sorted in ascending order. Then, at a particular quantile, cycle time estimates are drawn from the data. For example, 0.5 quantile is obtained by pulling out 50% value, i.e., median of the output sorted data.

Notably, this approach is time and resource intensive, but provides a valid estimate to which the quantile regression-based estimates can be compared. A set of 50 prediction points was generated for each of the three simulation experiments. The points were distributed uniformly over the experimental region at points not included in any of the five experimental design strategies for a particular simulation experiment. At each prediction point, the Minifab model was run for 100,000 replications to accurately find the quantile estimates. Figures 12 - 14 illustrates the location of the 50 prediction points, shown as blue dots, in the experimental design region of each of the simulation experiments. In each of the figures, the small black dots represent the design points across all the experimental design strategies.

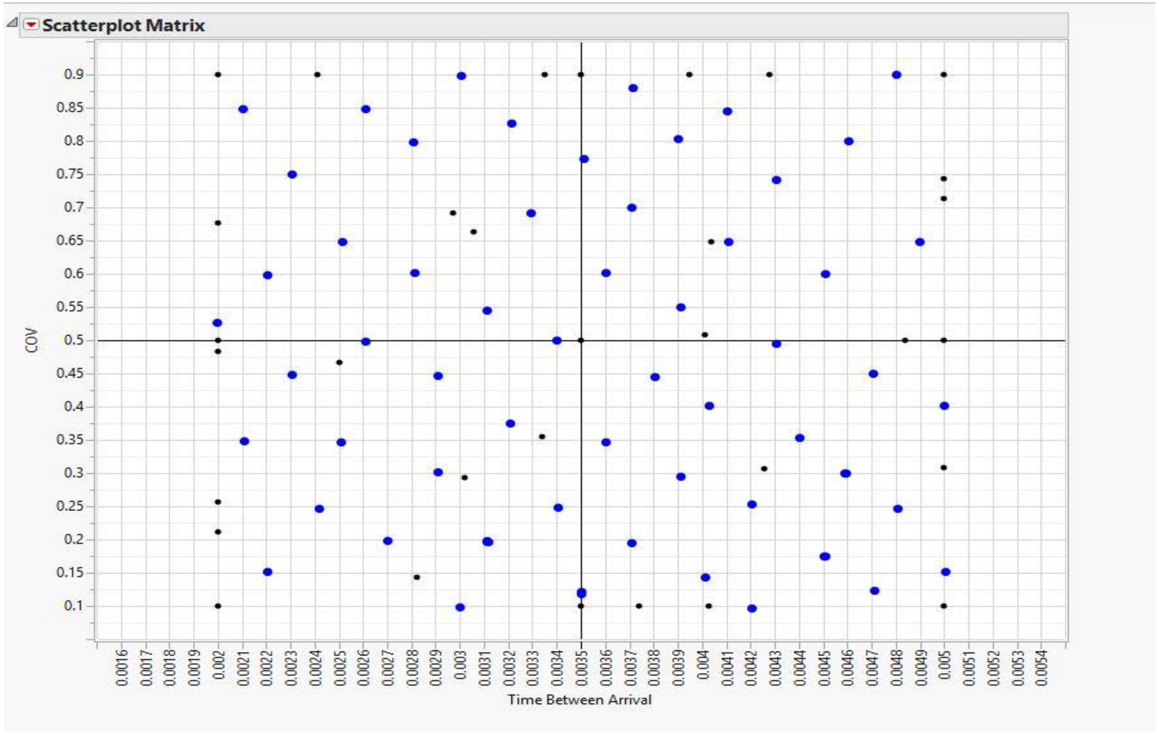


Figure 12. Scatter plot of prediction points over the design region for experiment - 1.

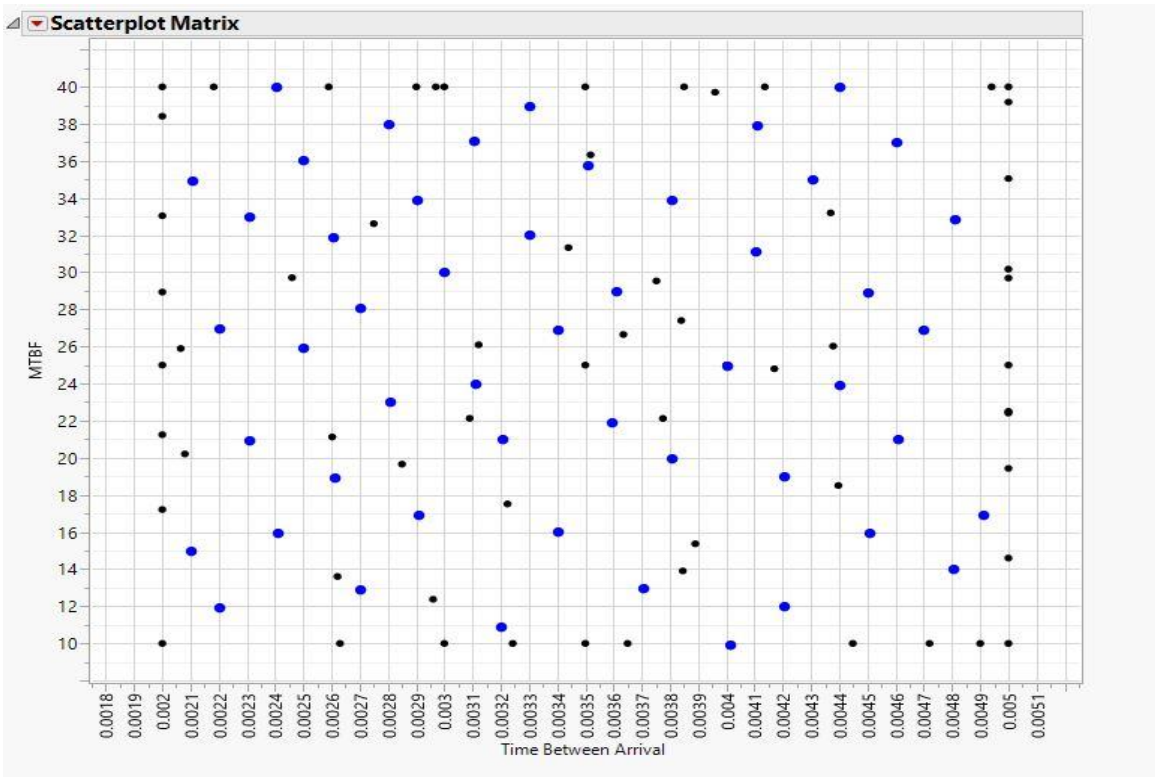


Figure 13. Scatter plot of prediction points over the design region for Experiment -2.

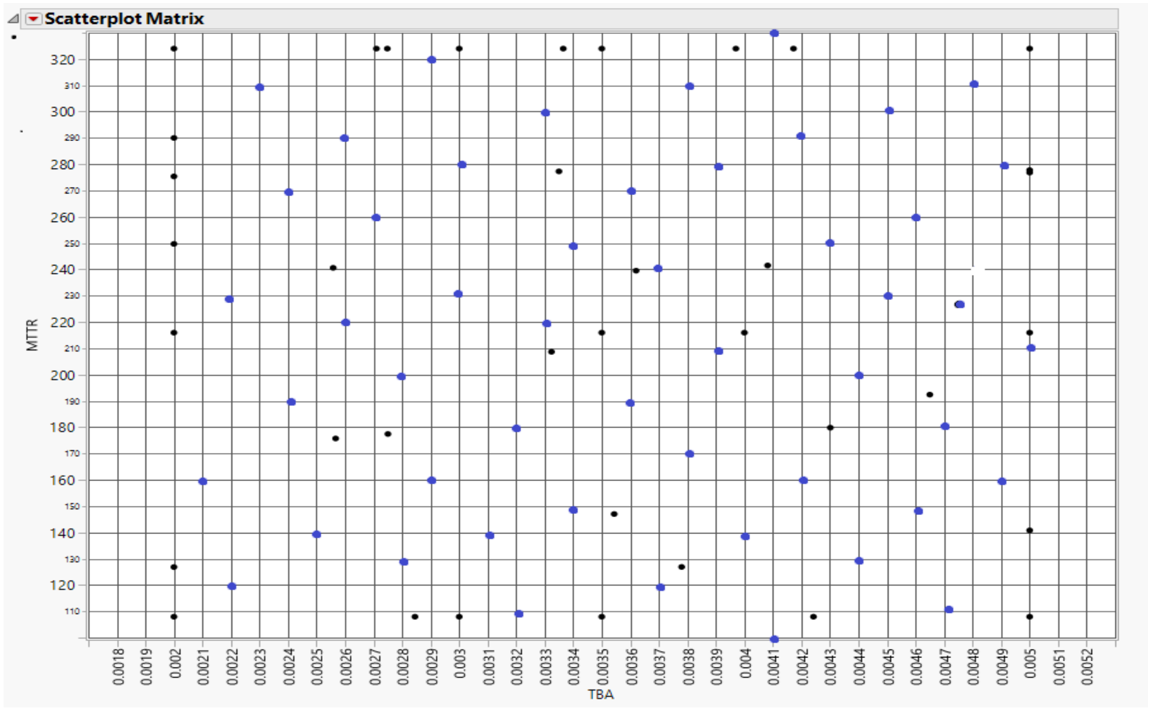


Figure 14. Scatter plot of prediction points over the design region for Experiment -3.

To assess the predictive accuracy of a particular QR model fit, the quantile regression models were used to predict the q-quantile of the cycle time distribution at each of the 50 prediction points corresponding to the appropriate simulation experiment. In this research, MATLAB was used to facilitate this process. Then, the quantile-regression generated estimates were compared to the ‘true’ cycle time quantile estimates obtained from the intensive simulation runs. The mean absolute percentage error (MAPE) value across all 50 of the prediction points was then calculated based on the difference between the predicted values and the ‘true’ values. The equation used to calculate the MAPE values is shown in the Equation 8. MAPEs are presented in terms of a percentage, and smaller MAPE values indicate better agreement between the quantile regression model’s prediction and the values obtained at the prediction points.

$$\frac{1}{\mathrm{n}} \sum \left[\frac{|Actual - Predicted|}{|Actual|} \right] * 100 \qquad (8)$$

Once MAPE values have been determined, comparisons on the predictive accuracy of various experimental design strategies could be made. The next chapter provides the results obtained using the methods presented in this chapter, while the final chapter discusses the implications of these findings.

Chapter 3 – Results and Analysis

1. Results

In this research, the objective was to compare the predictive accuracy of space filling, I-optimal, and each of the three previously described types of hybrid experimental design strategies. As described previously, for each experimental design strategy, the number of included design points, N, was also varied such that N = 6, 10, 15, and 25. Additionally, quantile regression models were fit for each of the 20 experimental designs at four different (k, λ) combinations: (2, 100), (2, 0.1), (3, 100), and (3, 0.1). Finally, for each (k, λ) combination at each experimental design, quantile regression fits were obtained to predict the 0.5, 0.8., 0.9, and 0.95 quantile. In all cases, calculations of mean absolute percentage error (MAPE) were used to evaluate predictive accuracy. Finally, as described previously, three simulation experiments were conducted using the Minifab model. In each experiment, two factors were varied as considered as predictor variables in the quantile regression model as shown in Table 2. The goal was to develop quantile regression models to predict quantiles of the cycle time distribution.

An initial objective in the experimentation was to determine whether there was a (k, λ) combination that was generally superior to the others. Such a determination would allow the comparison of results to focus on the differences between the experimental designs (design type and N) rather than the parameters involved in fitting the quantile regression model (i.e., k and λ). Figures 15-18 show box plots of the MAPE values obtained for each of the four (k, λ) combinations at each quantile. The box plots include MAPE values across all 60 quantile regression fits (20 experimental designs for each simulation experiment x three simulation experiments) obtained for each (k, λ) combination at the given quantile. Each plot shows the outliers as red colored crosshairs, the median as a black hollow circle, mean value is the black dot and the inner quartile range as the box. All the MAPE values shown in the figures are given as percentages (i.e., MAPE value 5 represents a 5% mean absolute prediction error).

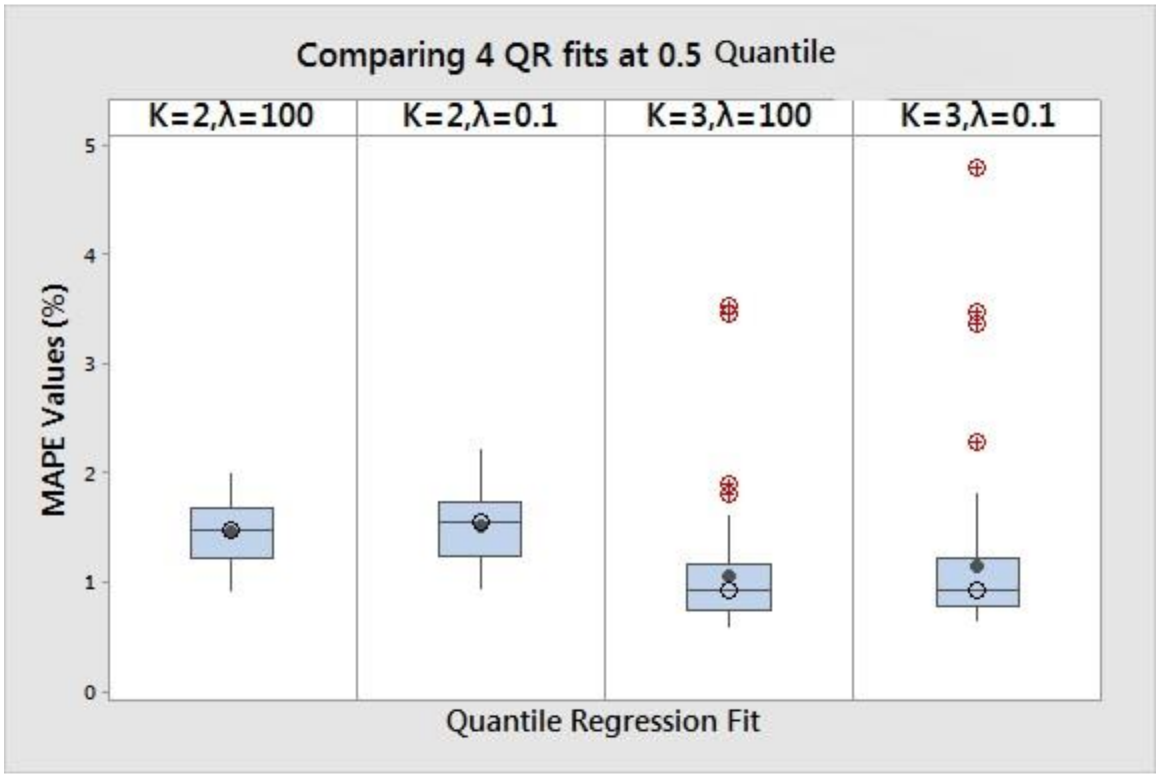


Figure 15. Box plot comparing (k, λ) combinations for all the QR fits at the 0.5 quantile.

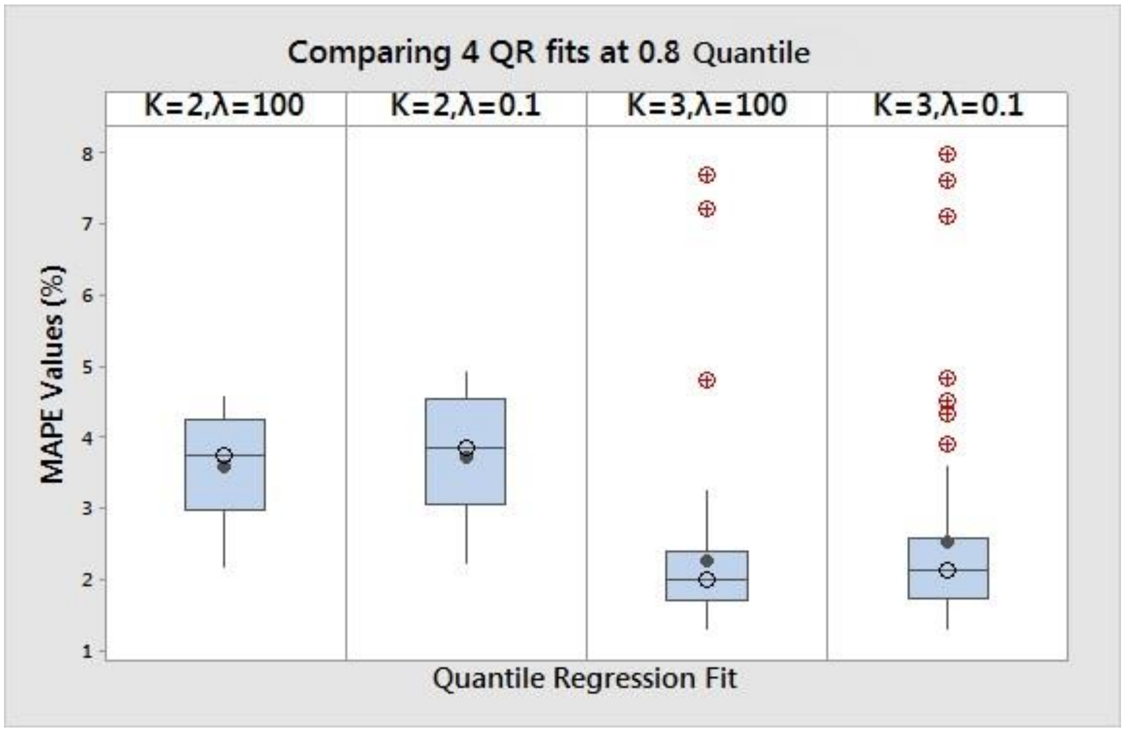


Figure 16. Box plot comparing (k, λ) combinations for all the QR fits at the 0.8 quantile.

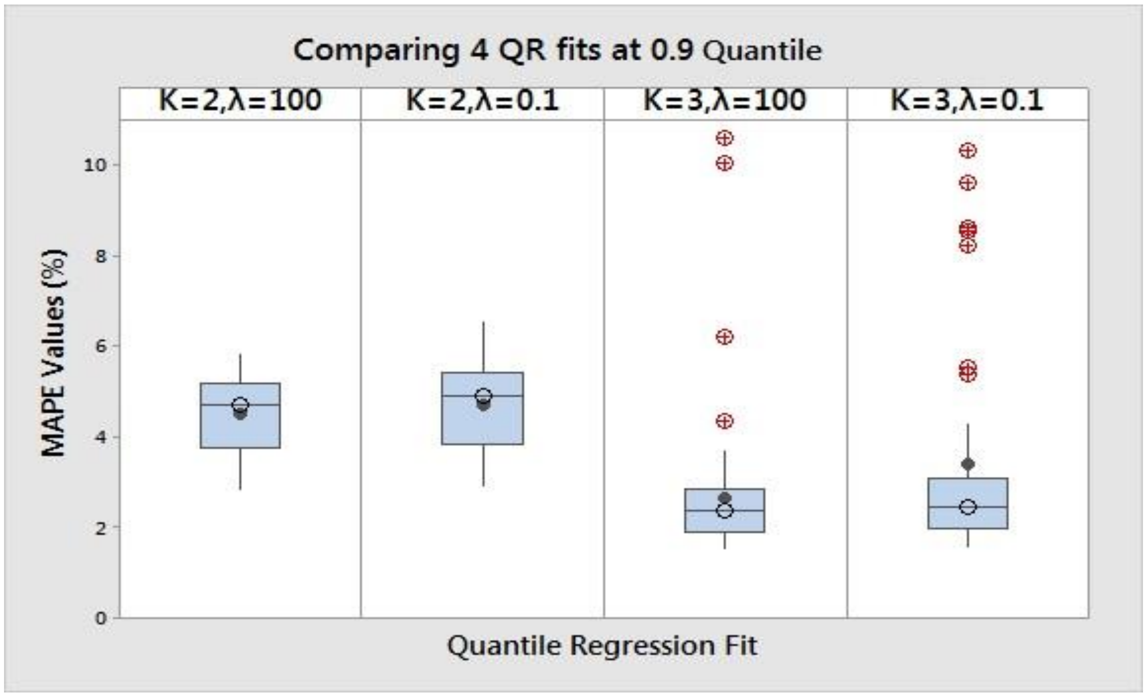


Figure 17. Box plot comparing (k, λ) combinations for all the QR fits at the 0.9 quantile.

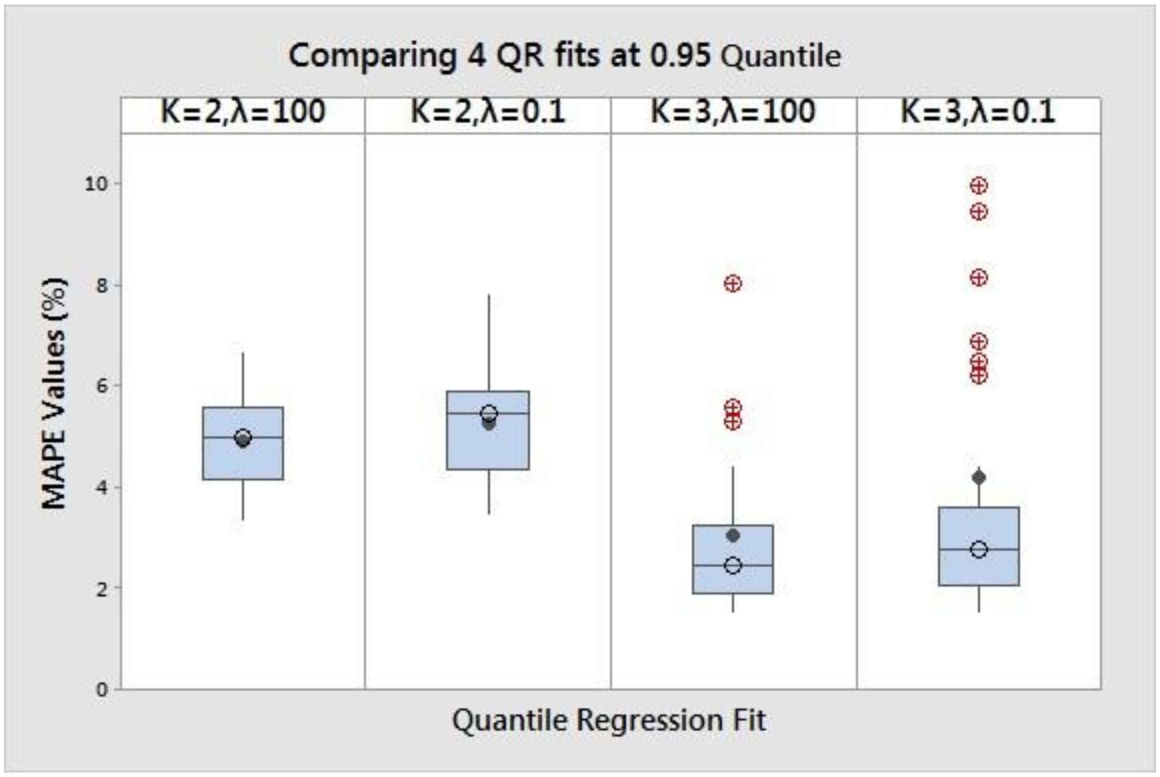


Figure 18. Box plot comparing (k, λ) combinations for all the QR fits at the 0.95 quantile.

Based on the results presented in Figures 15 - 18, which include all the examined quantiles, the case in which k = 3 and λ = 100 has least mean, median values and fewer outliers than other (k, λ) combinations. The inner quartile ranges of MAPE values are also less in the third order (k=3) model fits compared to the second order (k=2) model fits. Given these findings, the remainder of the results section focus only on the case in which k = 3 and λ = 100.

2. Analysis: Comparison of Experimental Designs

To analyze the comparative performance of the experimental designs, MAPE values between the validation estimates and estimates predicted by the quantile regression models were calculated for each simulation experiment. Results are given in Tables 9 – 11 for the

Sphere Packing design (SP), I-Optimal design (IOP), Arbitrary Hybrid design (AH), Clustering Hybrid design (CLH) and Centroid Clustering Hybrid designs (CCH) at N = 6, 10, 15 and 25. In all cases, k = 3 and λ = 100. Each cell in the final four columns represents the MAPE value obtained when predicting the given quantile based on the designated experimental design strategy made up of the designed number of design points (N).

Table 9. MAPE values for simulation experiment 1 with k = 3, λ = 100.

		MAPE (%)			
Experimental Design Strategy	Quantile	N=6	N=10	N=15	N=25
SP	0.5 Quantile	1.83	1.24	1.03	1.19
IOP		2.13	1.76	1.58	1.67
AH		2.22	1.62	1.29	1.20
CLH		1.95	1.67	1.52	1.25
CCH		1.66	1.62	1.14	1.24
SP	0.8 Quantile	4.56	3.13	2.53	2.80
IOP		4.60	4.35	4.12	4.23
AH		4.94	4.02	2.99	2.90
CLH		4.50	4.10	3.67	3.14
CCH		4.68	4.07	2.81	3.07
SP	0.9 Quantile	6.57	4.08	3.34	3.61
IOP		6.01	5.42	5.02	5.22
AH		6.51	5.33	3.80	3.60
CLH		5.63	5.46	4.61	3.94
CCH		6.36	5.33	3.70	3.76
SP	0.95 Quantile	6.57	4.47	3.92	4.05
IOP		7.37	6.19	5.73	5.83
AH		7.81	5.61	4.41	4.07
CLH		7.71	6.11	5.13	4.39
CCH		5.78	5.80	4.21	4.14

Table 10. MAPE values for simulation experiment 2 with k = 3, λ = 100.

		MAPE (%)			
Experimental Design Strategy	Quantile	N=6	N=10	N=15	N=25
SP	0.5 Quantile	1.66	1.64	1.19	1.13
IOP		2.00	1.84	1.74	1.83
AH		1.96	1.84	1.66	1.18
CLH		1.19	1.41	1.57	1.36
CCH		1.37	1.21	1.43	1.25
SP	0.8 Quantile	4.22	3.63	2.68	2.50
IOP		4.74	4.55	4.39	4.54
AH		4.71	4.71	3.90	2.57
CLH		3.12	3.49	3.81	3.07
CCH		3.01	2.90	3.35	3.06
SP	0.9 Quantile	5.04	4.34	3.48	3.31
IOP		5.67	5.31	5.11	5.32
AH		5.71	5.83	4.78	3.21
CLH		3.93	4.08	4.54	3.87
CCH		3.72	3.68	4.43	3.97
SP	0.95 Quantile	5.59	4.90	3.99	3.86
IOP		6.05	5.69	5.41	5.66
AH		5.99	6.40	5.37	3.73
CLH		4.51	5.49	4.96	4.44
CCH		4.15	4.17	4.97	4.70

Table 11. MAPE values for simulation experiment 3 with k = 3, λ = 100.

		MAPE (%)			
Experimental Design Strategy	Quantile	N=6	N=10	N=15	N=25
SP	0.5 Quantile	1.54	1.16	0.93	1.11
IOP		1.83	1.74	1.68	1.74
AH		1.79	1.66	1.33	0.99
CLH		1.79	1.36	1.52	0.99
CCH		1.58	1.62	1.45	1.36
SP	0.8 Quantile	4.02	3.09	2.21	2.67
IOP		4.66	4.64	4.58	4.63
AH		4.59	4.37	3.59	2.55
CLH		4.59	3.67	4.13	2.55
CCH		4.12	4.03	3.65	3.30
SP	0.9 Quantile	5.13	3.95	2.88	3.40
IOP		5.88	5.62	5.30	5.59
AH		5.81	5.17	4.60	3.29
CLH		5.81	4.99	5.03	3.29
CCH		5.24	5.20	4.54	4.16
SP	0.95 Quantile	5.74	4.33	3.44	3.89
IOP		6.70	6.29	5.83	6.21
AH		6.57	5.76	5.04	3.69
CLH		6.57	5.70	5.46	3.69
CCH		5.88	5.71	4.76	4.84

The MAPE values in Tables 9 – 11 highlight the fact that the prediction error, in general, is quite low (<7%) in all cases. This analysis, then, concentrates on the following goals:

- Understanding the impact of N on the predictive accuracy of quantile regression metamodels.
- Understanding the impact of experimental design strategy on the predictive accuracy of quantile regression metamodels.

- Examining the relative performance of the three hybrid experimental design strategies.
- Examining the relative performance of the hybrid and traditional (sphere packing and I-optimal) experimental design strategies.

2.1 *Impact of N on Predictive Accuracy of Quantile Regression Metamodels*

The predictive accuracy of a particular experimental design strategy depends on the number of design points at which the simulation runs are executed to generate the data used to develop the quantile regression metamodel. Increasing the number of design points within the experimental region produces improved predictive accuracy within the same region, though the related simulation effort is also greater. Figures 19 illustrates this point. In this figure, the horizontal axis gives the number of design points used to fit the meta-model, while the y-axis gives the resulting MAPE value for estimating the 0.5 quantile in simulation experiment 1. In Figure 19, different experimental design strategies are shown in different colors. Specifically, Sphere Packing (SP), I-Optimal (IOP), Arbitrary Hybrid (AH), Clustering Hybrid (CLH) and Centroid Clustering Hybrid (CCH) experimental design strategies are represented as orange, yellow, green, red and brown colors, respectively. Figure 19 illustrates that as more design points are included (i.e., N gets larger), the MAPE values tend to decrease, regardless of the experimental design strategy. It also illustrates that this trend is more pronounced for the CLH and AH experimental design strategies than for the IOP design strategy. Similar results were found across all three simulation experiments and for estimates of the 0.8, 0.9, and 0.95 quantiles. These results are shown in Figures 20 – 30.

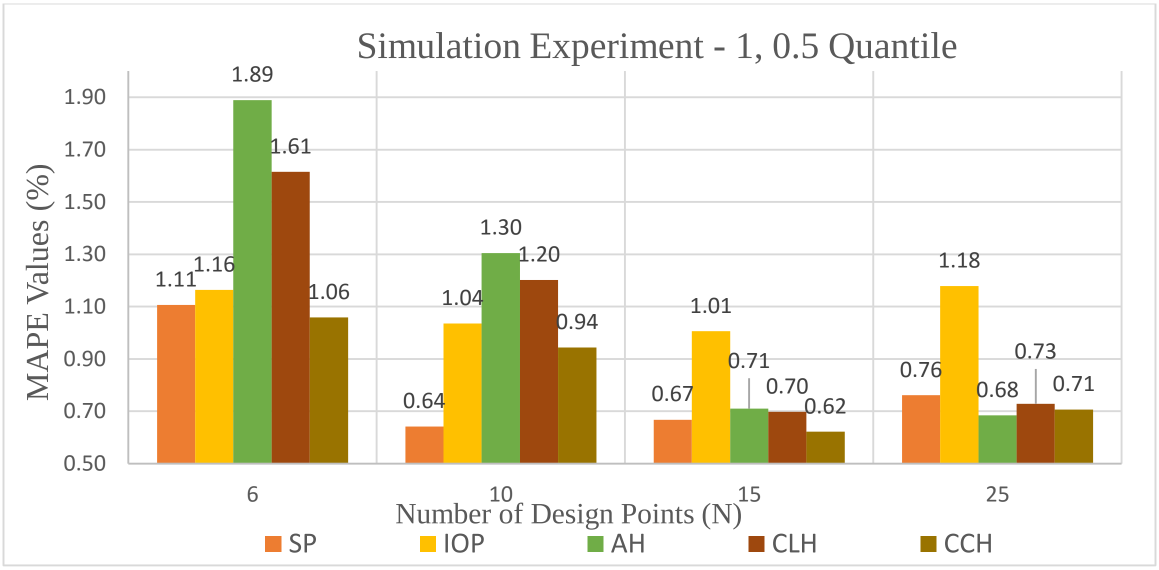


Figure 19. The impact of the number of design points on each of the experimental design strategies in simulation experiment 1, for the 0.5 quantile.

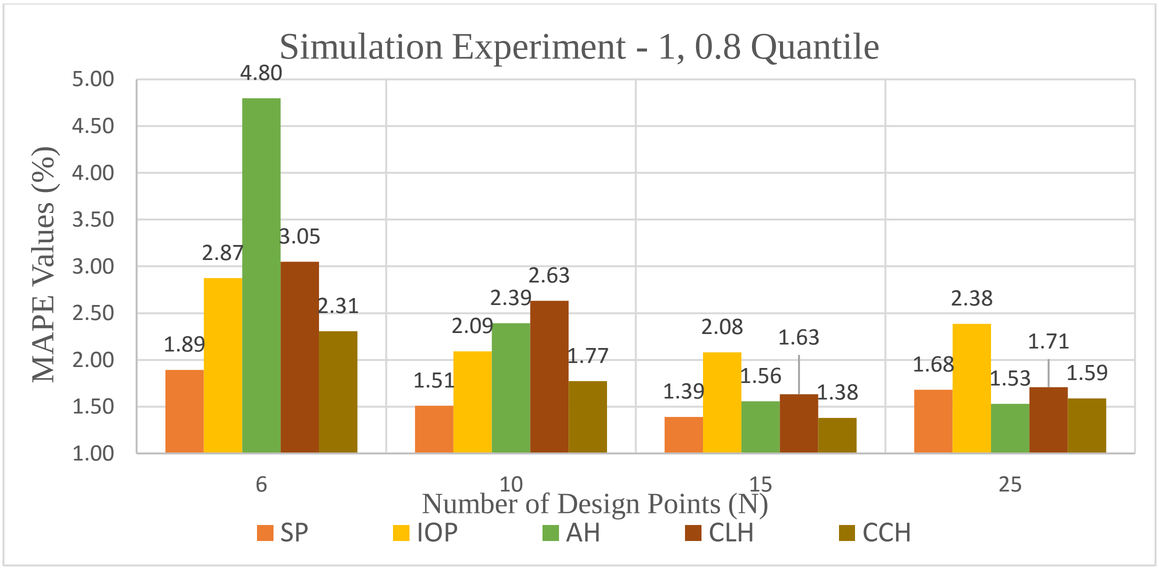


Figure 20. The impact of the number of design points on each of the experimental design strategies in simulation experiment 1, for the 0.8 quantile.

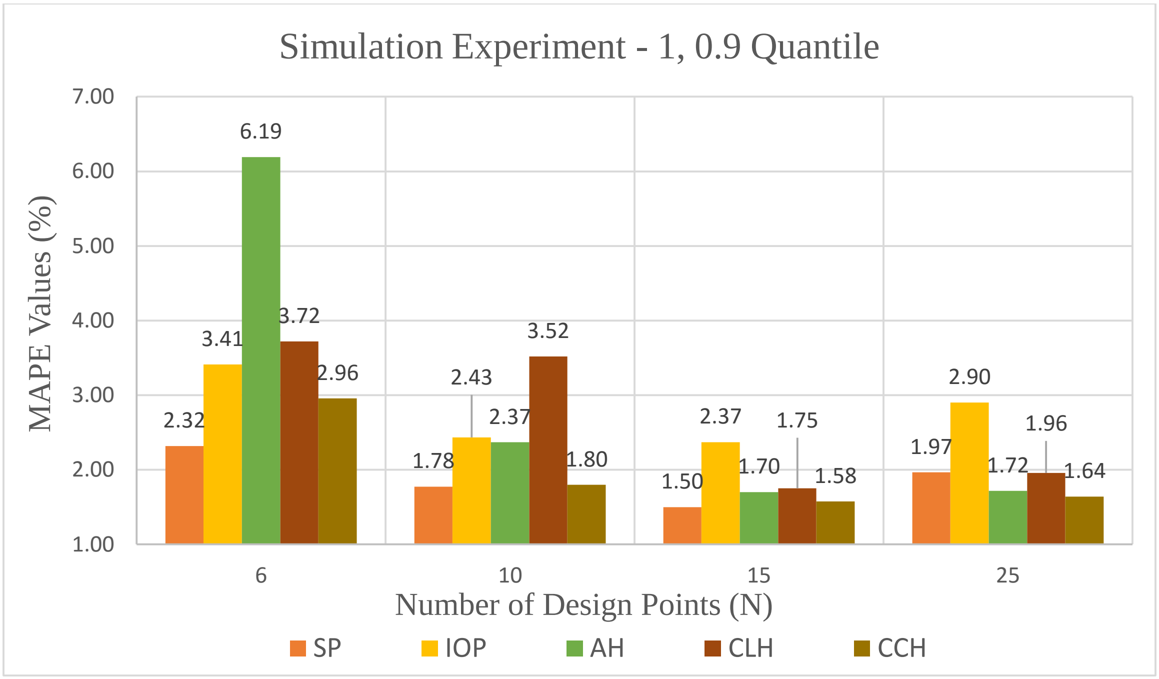


Figure 21. The impact of the number of design points on each of the experimental design strategies in simulation experiment 1, for the 0.9 quantile.

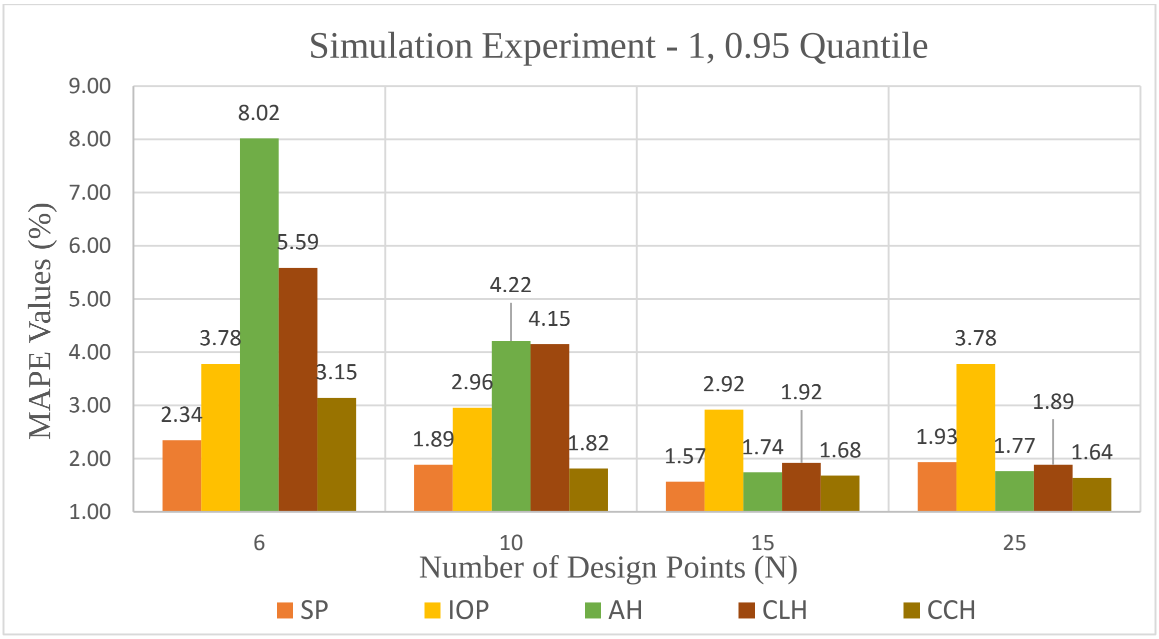


Figure 22. The impact of the number of design points on each of the experimental design strategies in simulation experiment 1, for the 0.95 quantile.

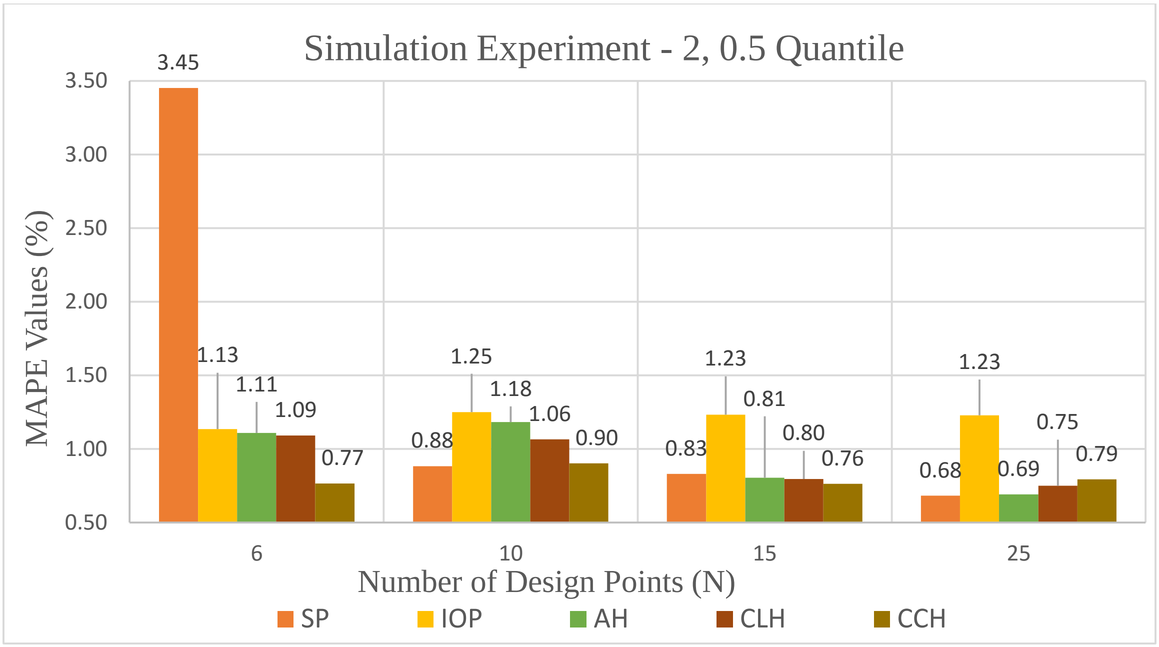


Figure 23. The impact of the number of design points on each of the experimental design strategies in simulation experiment 2, for the 0.5 quantile.

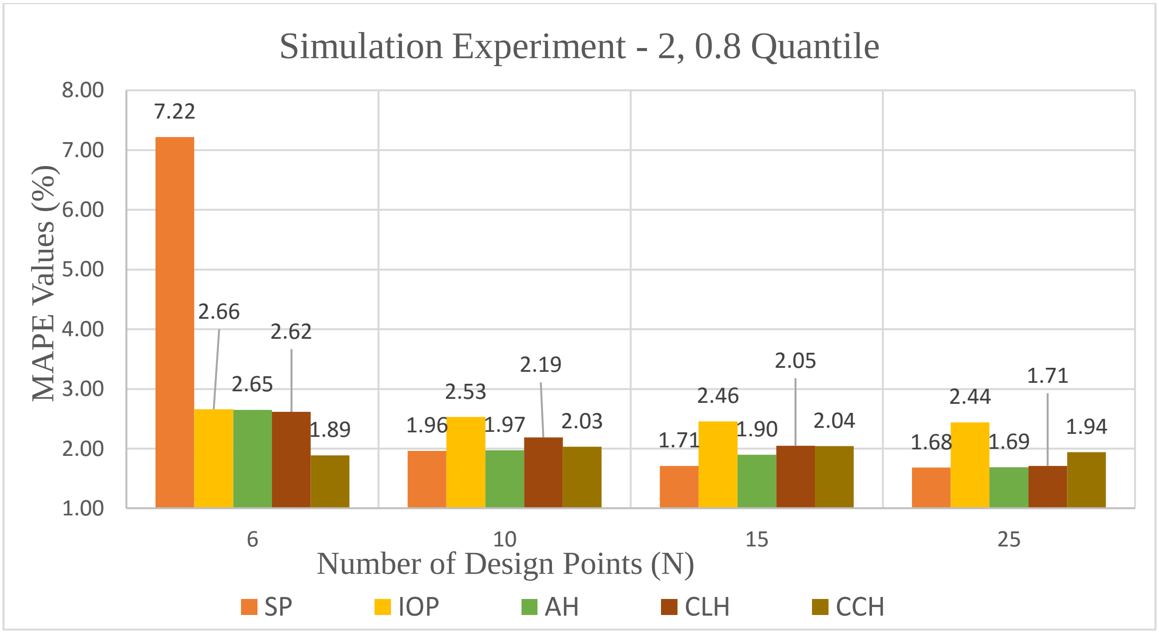


Figure 24. The impact of the number of design points on each of the experimental design strategies in simulation experiment 2, for the 0.8 quantile.

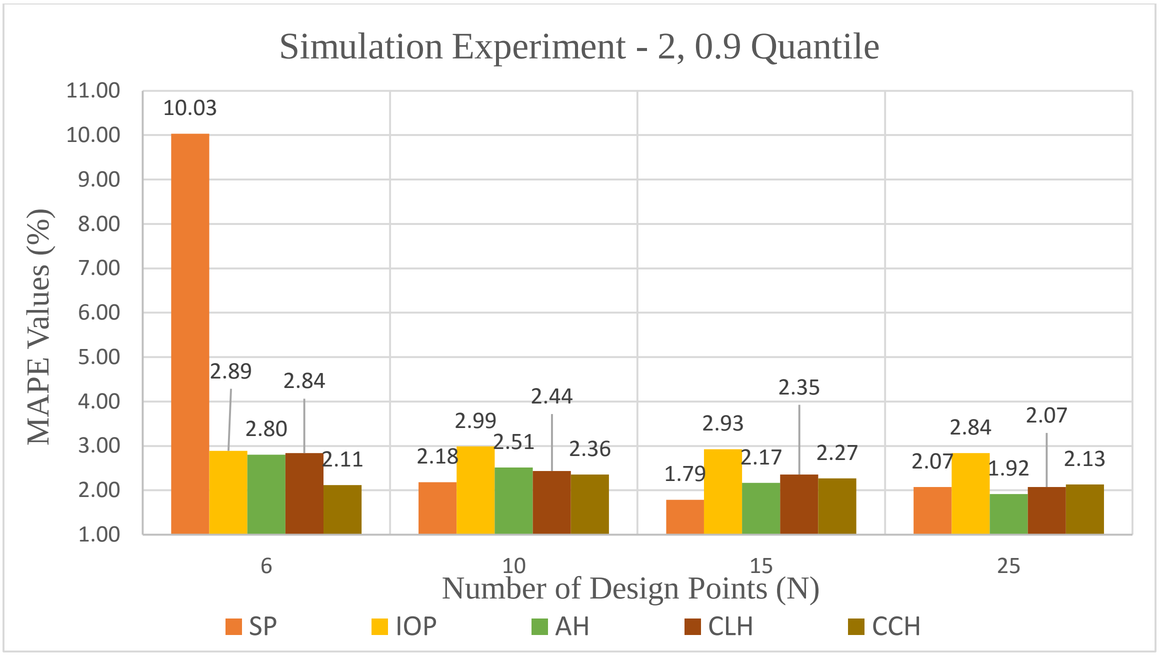


Figure 25. The impact of the number of design points on each of the experimental design strategies in simulation experiment 2, for the 0.9 quantile.

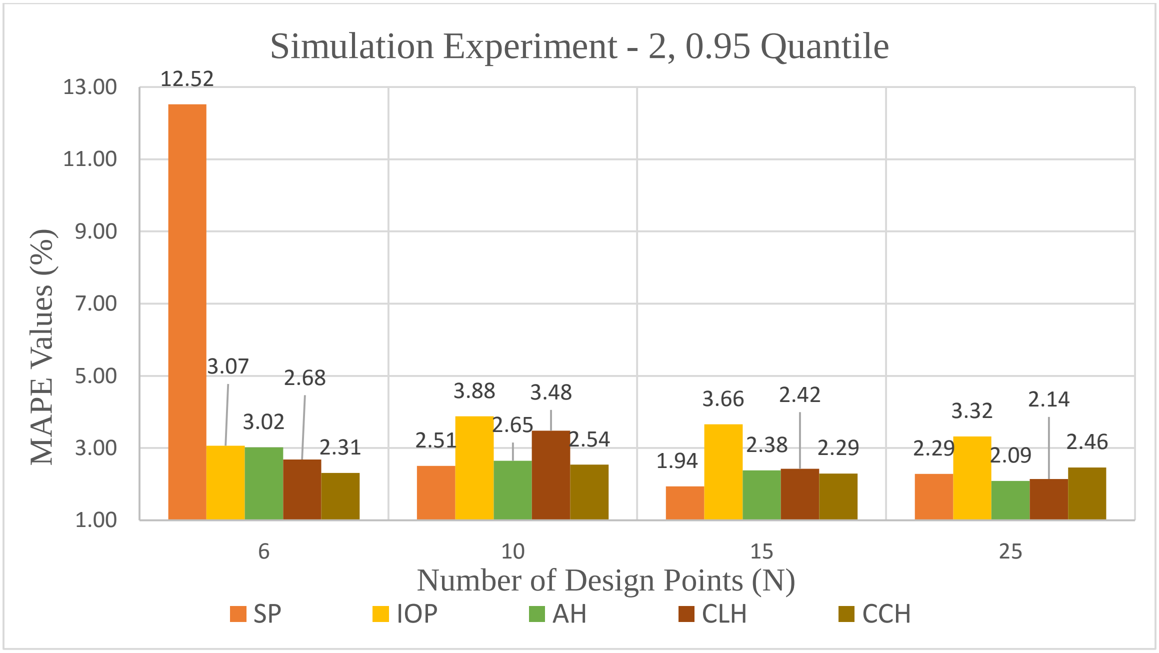


Figure 26. The impact of the number of design points on each of the experimental design strategies in simulation experiment 2, for the 0.95 quantile.

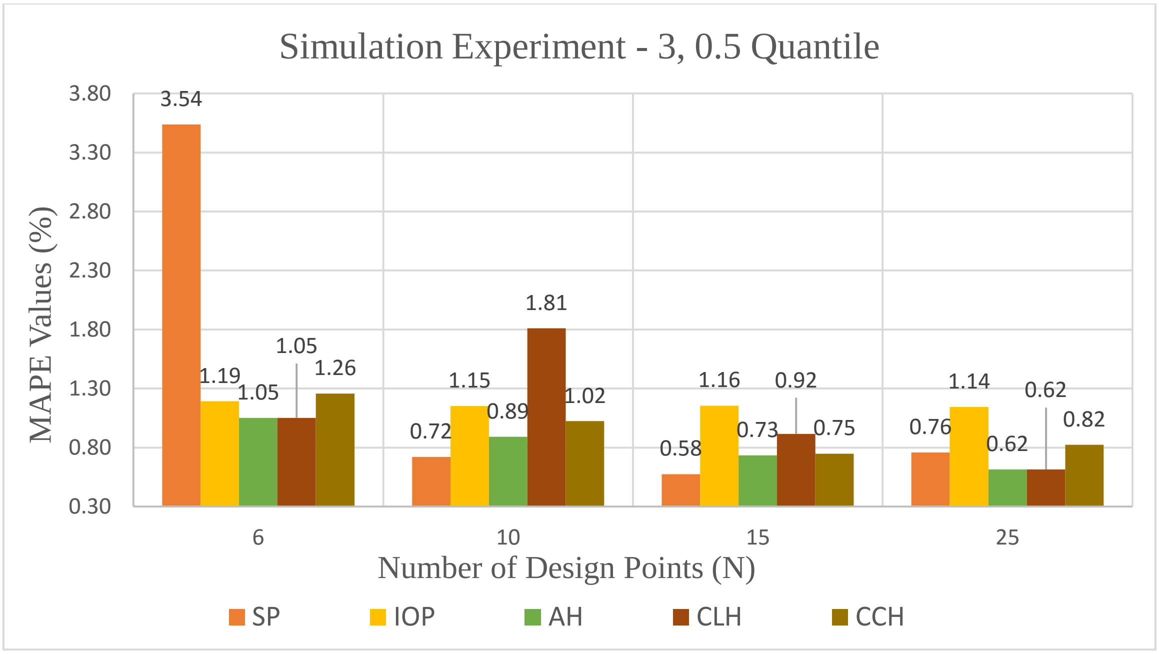


Figure 27. The impact of the number of design points on each of the experimental design strategies in simulation experiment 3, for the 0.5 quantile.

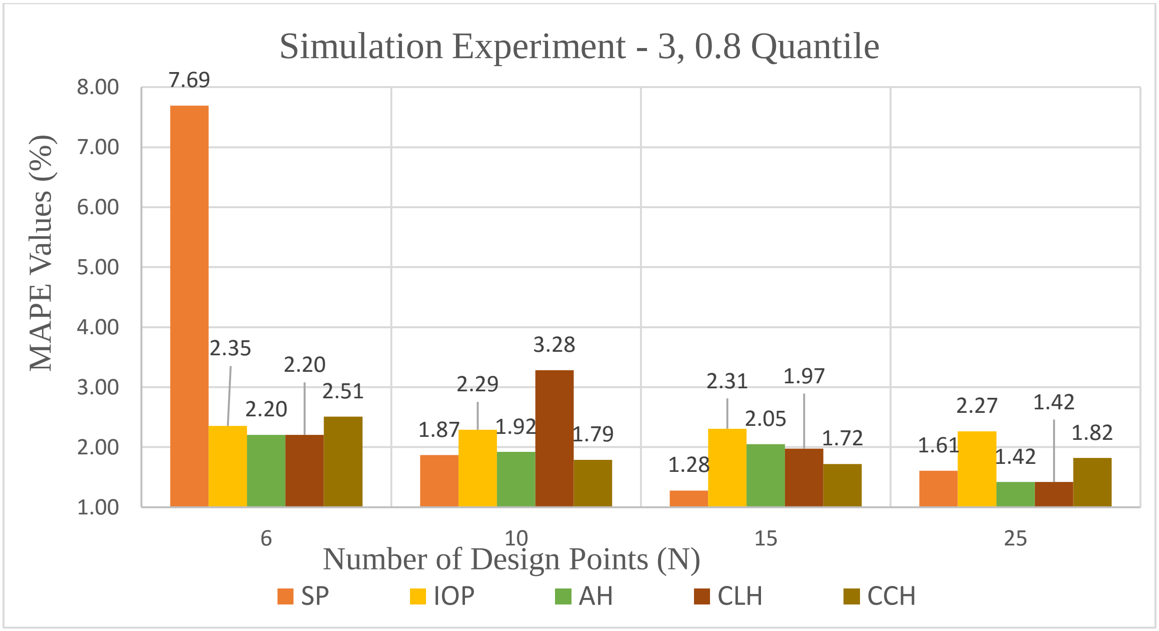


Figure 28. The impact of the number of design points on each of the experimental design strategies in simulation experiment 3, for the 0.8 quantile.

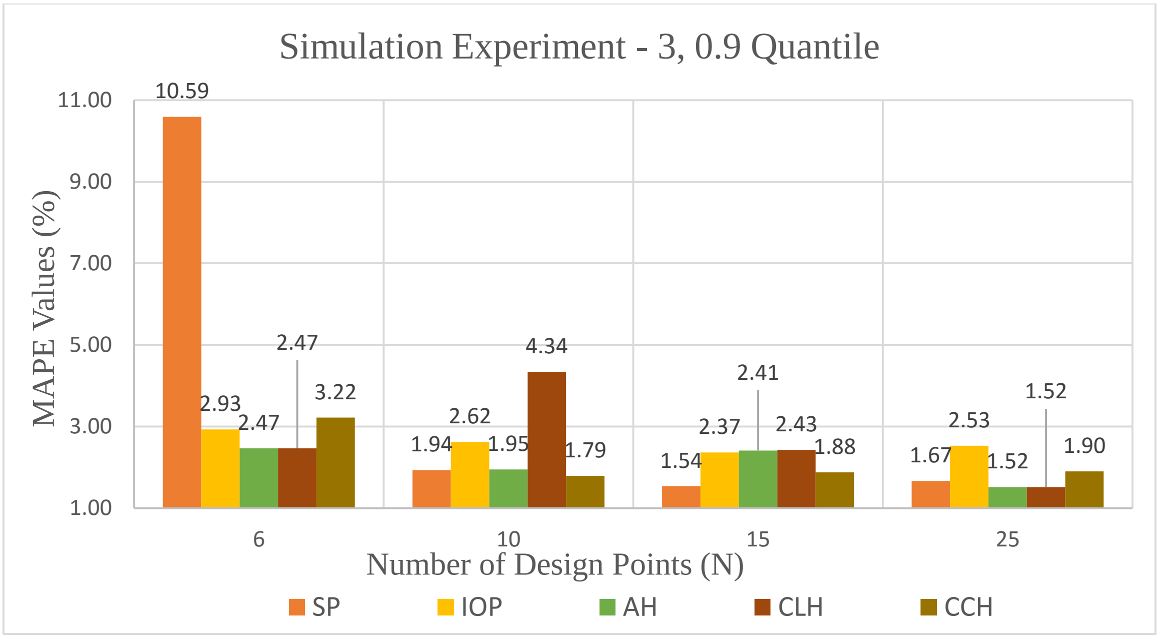


Figure 29. The impact of the number of design points on each of the experimental design strategies in simulation experiment 3, for the 0.9 quantile.

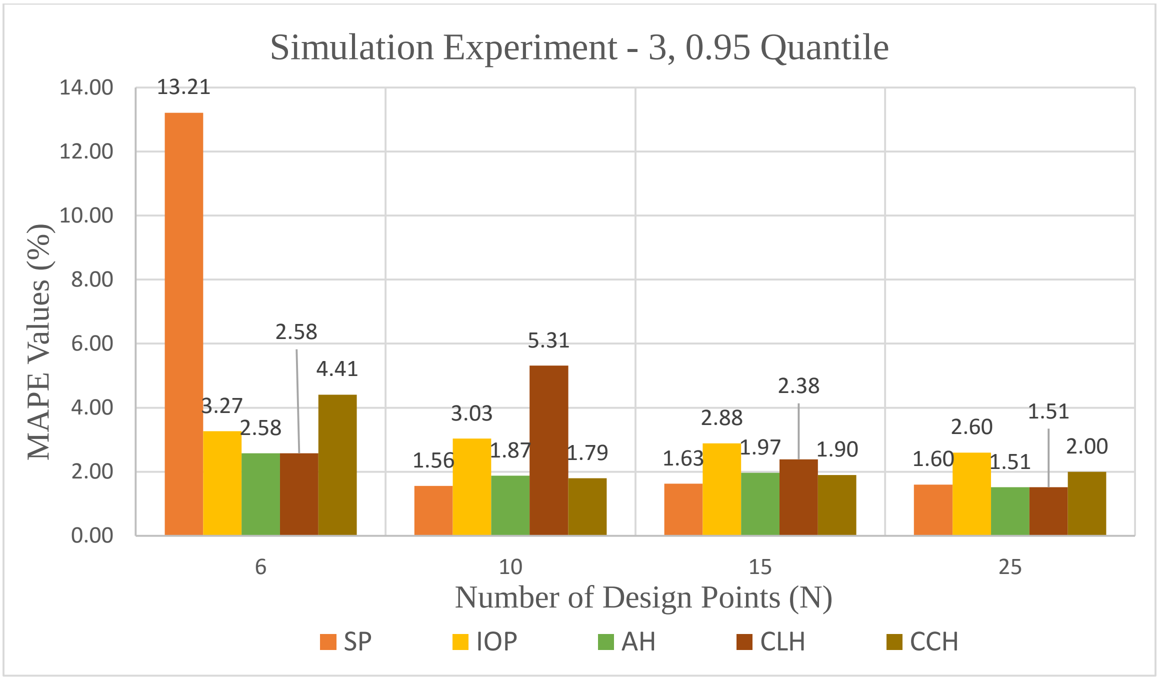


Figure 30. The impact of the number of design points on each of the experimental design strategies in simulation experiment 3, for the 0.95 quantile.

2.2 *Impact of Experimental Design Strategy on Predictive Accuracy of Quantile Regression Metamodels*

While the number of design points included in an experimental design strategy is clearly important to the resulting predictive accuracy of the fit metamodel, the location of the design points is also critical. Figure 31 illustrates the relative difference in performance across the experimental design strategies. In Figure 31, each of the experimental design strategies is represented by a different color, and the MAPE values for each of the four quantiles is given for each. In all cases, the experimental design strategy was created with 10 design points (i.e., N = 10). Comparisons at the 0.95 quantile show that the MAPE value obtained based a sphere packing design is 1.89%, while the MAPE value obtained based on a clustering hybrid design is more than twice as large, at 4.15%. This comparison highlights the impact of the location of design points on the resulting predictive accuracy. Comparative figures for all three simulation experiments and all values of N are given in Figures 32 – 42.

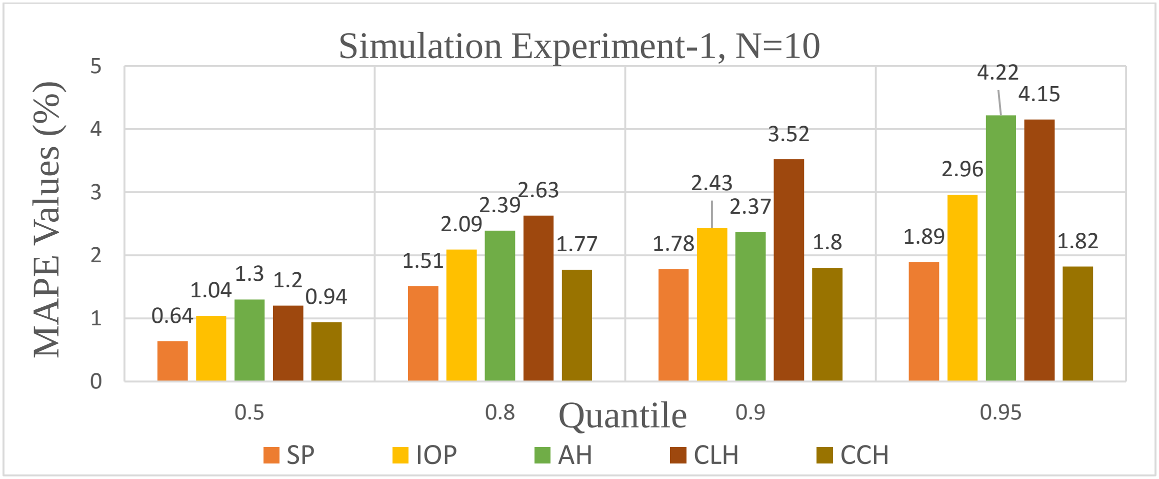


Figure 31. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 1 when N = 10.

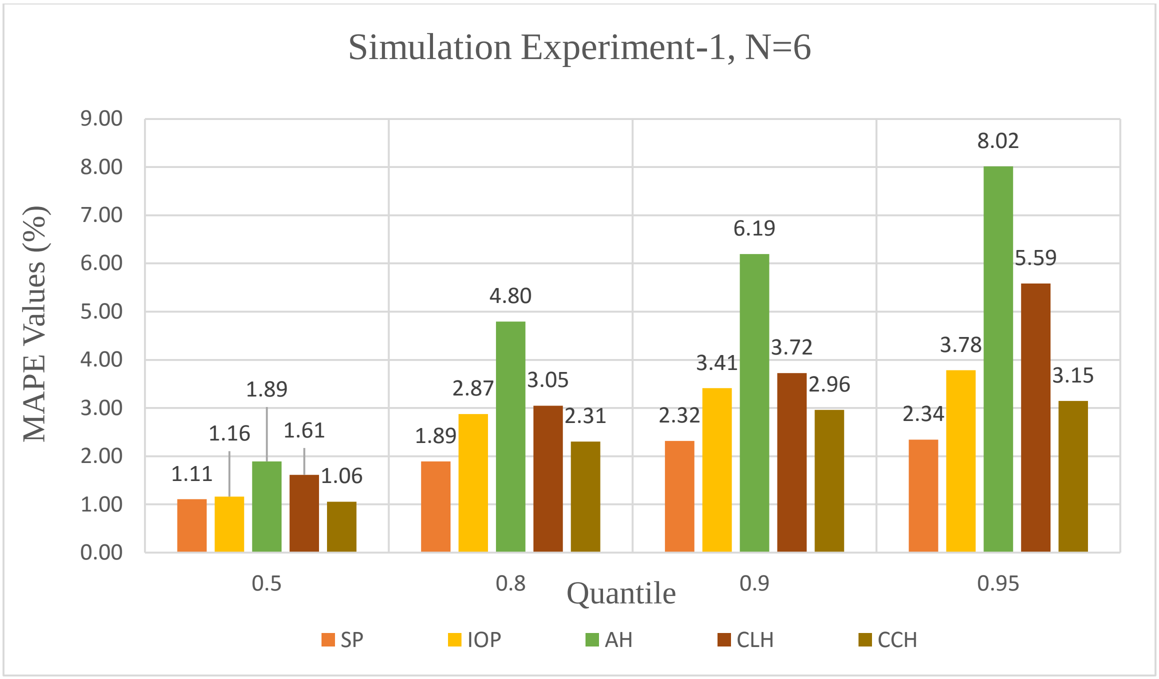


Figure 32. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 1 when N = 6.

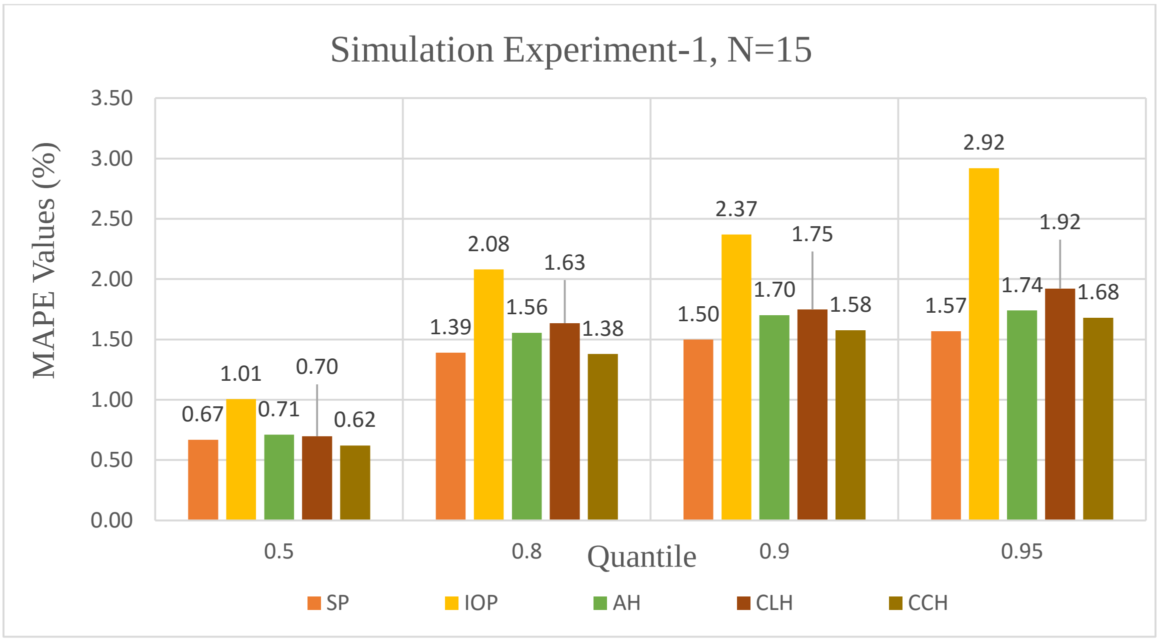


Figure 33. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 1 when N = 15.

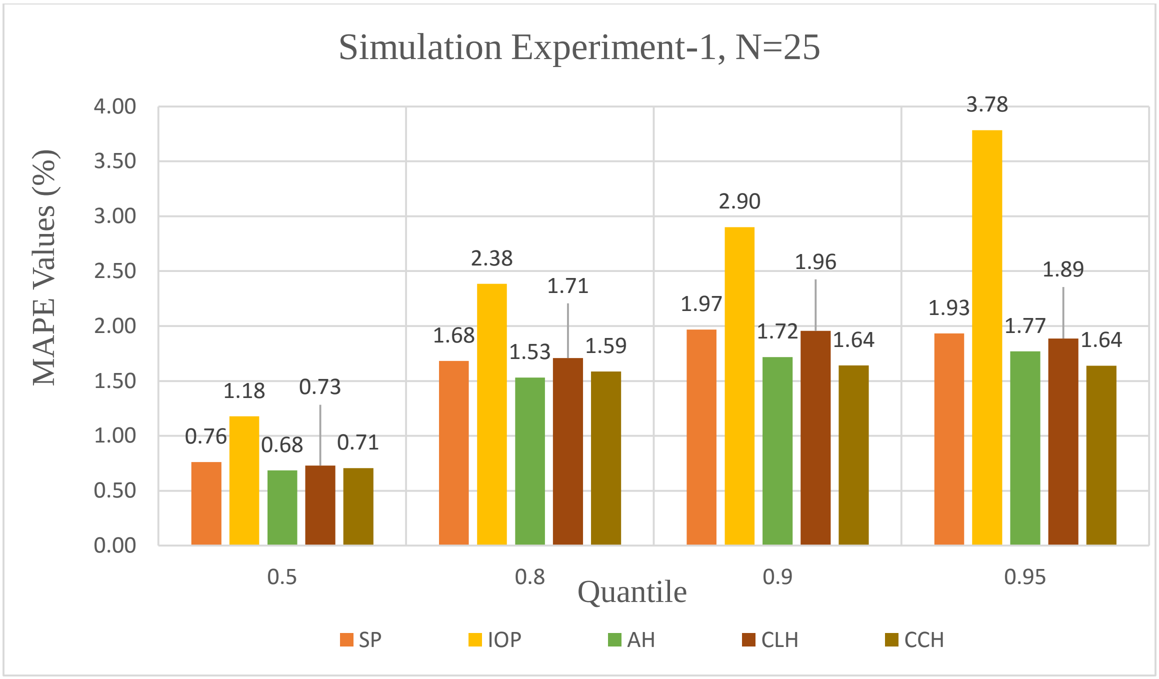


Figure 34. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 1 when N = 25.

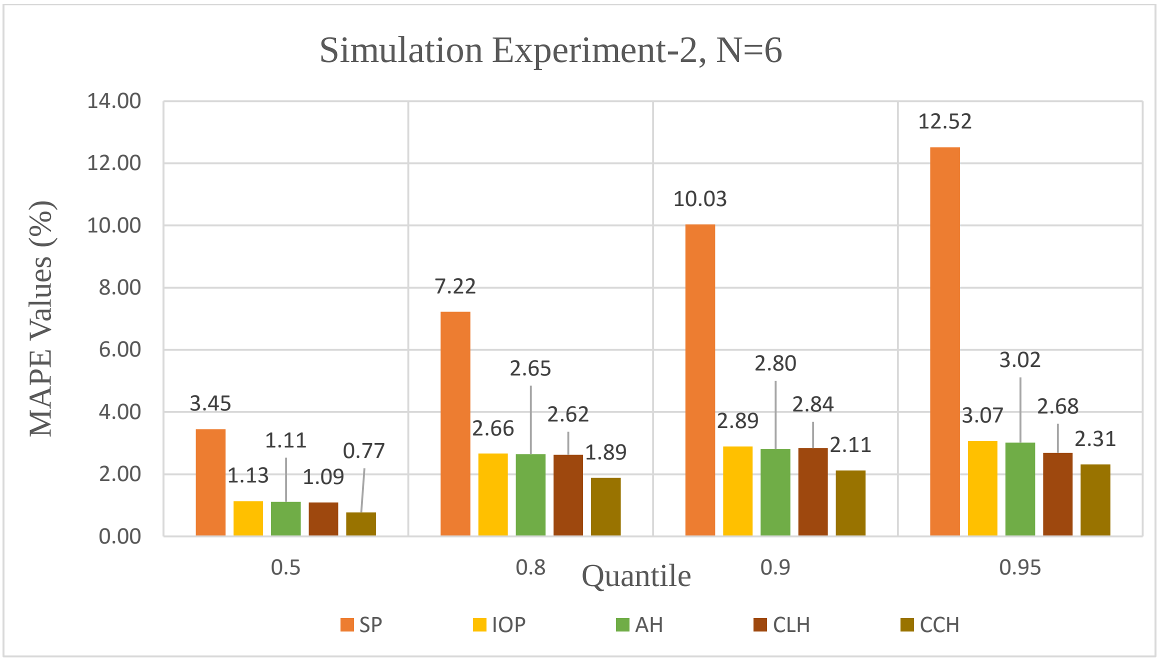


Figure 35. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 2 when N = 6.

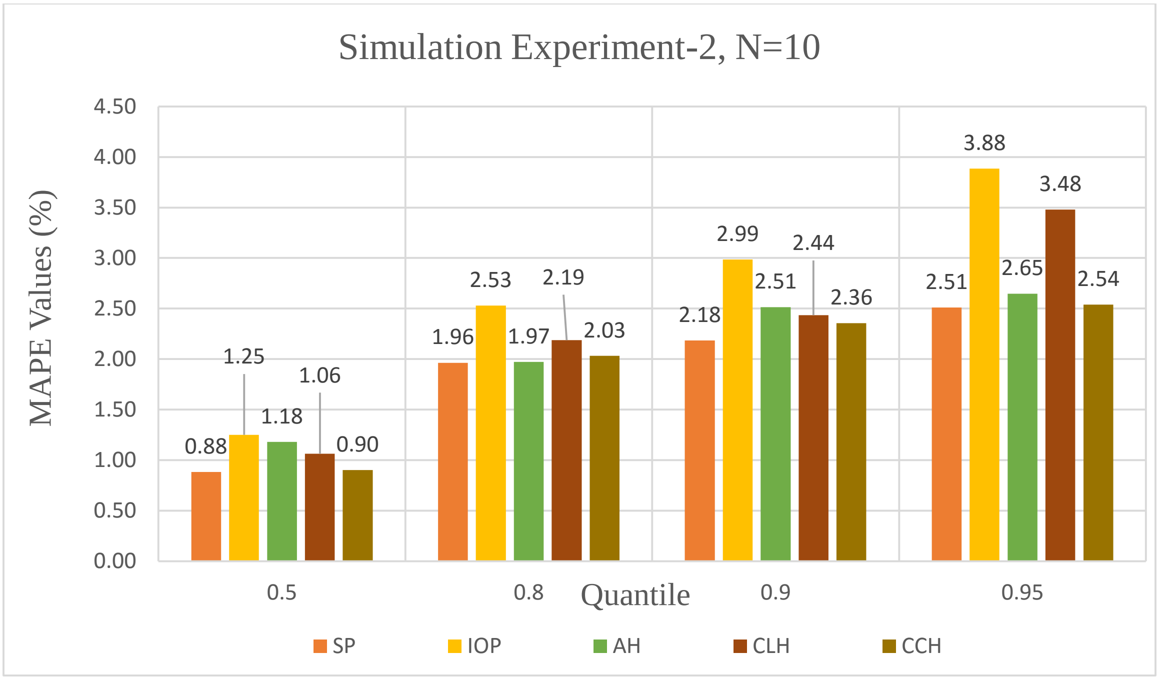


Figure 36. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 2 when N = 10.

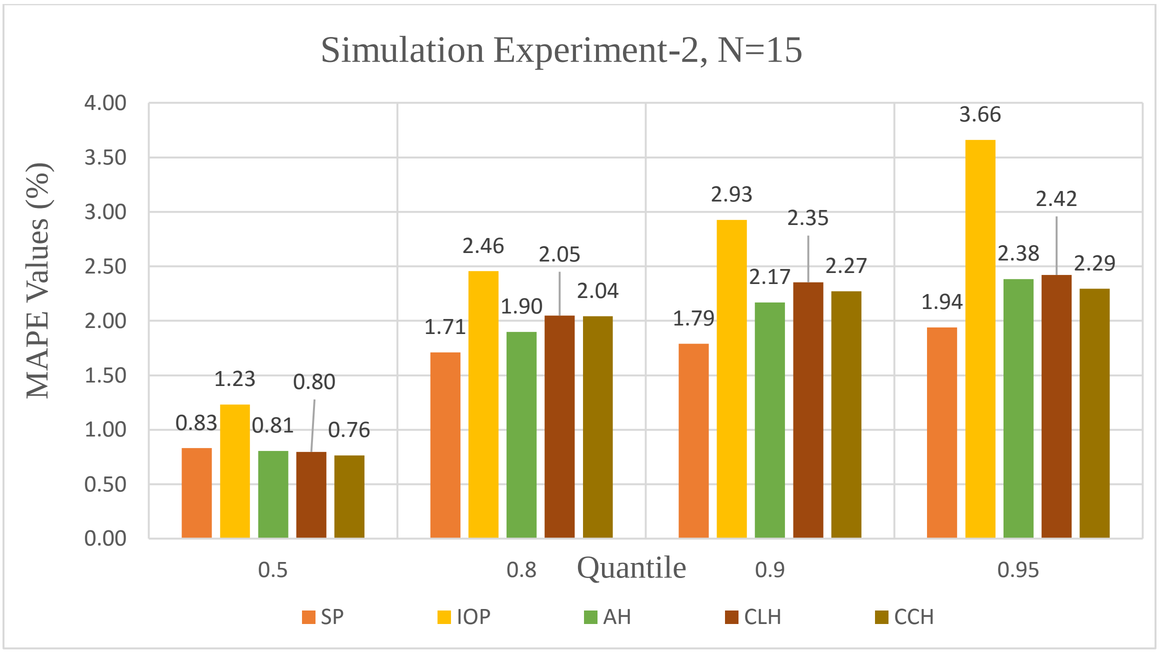


Figure 37. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 2 when N = 15.

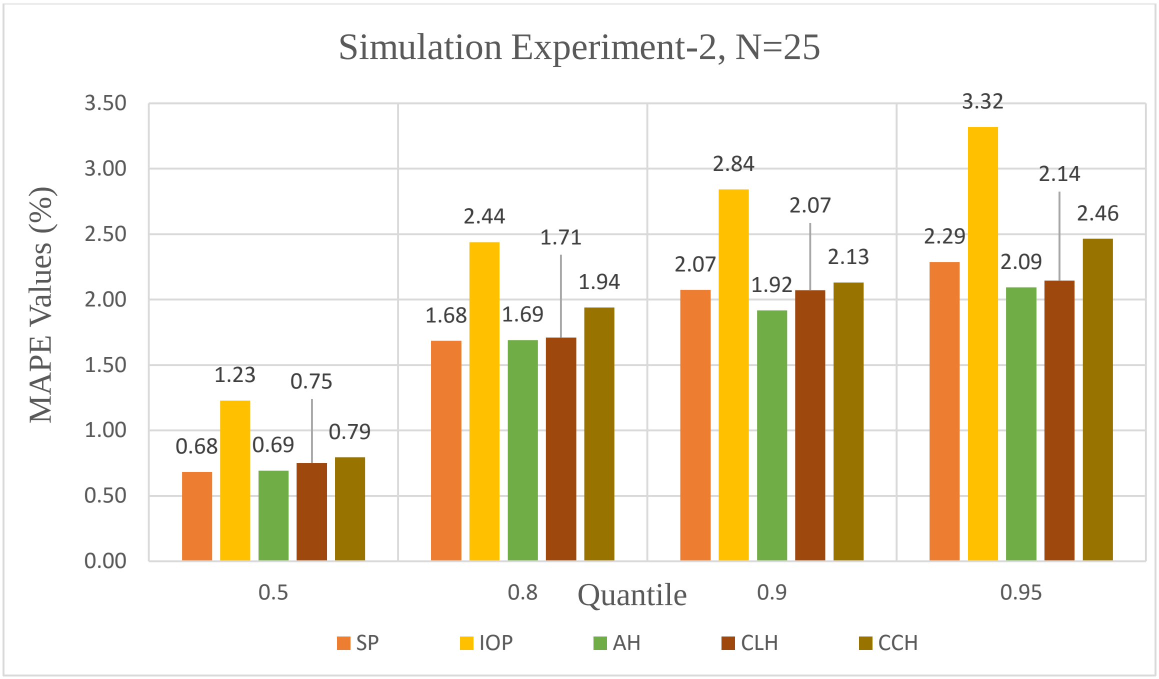


Figure 38. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 2 when N = 25.

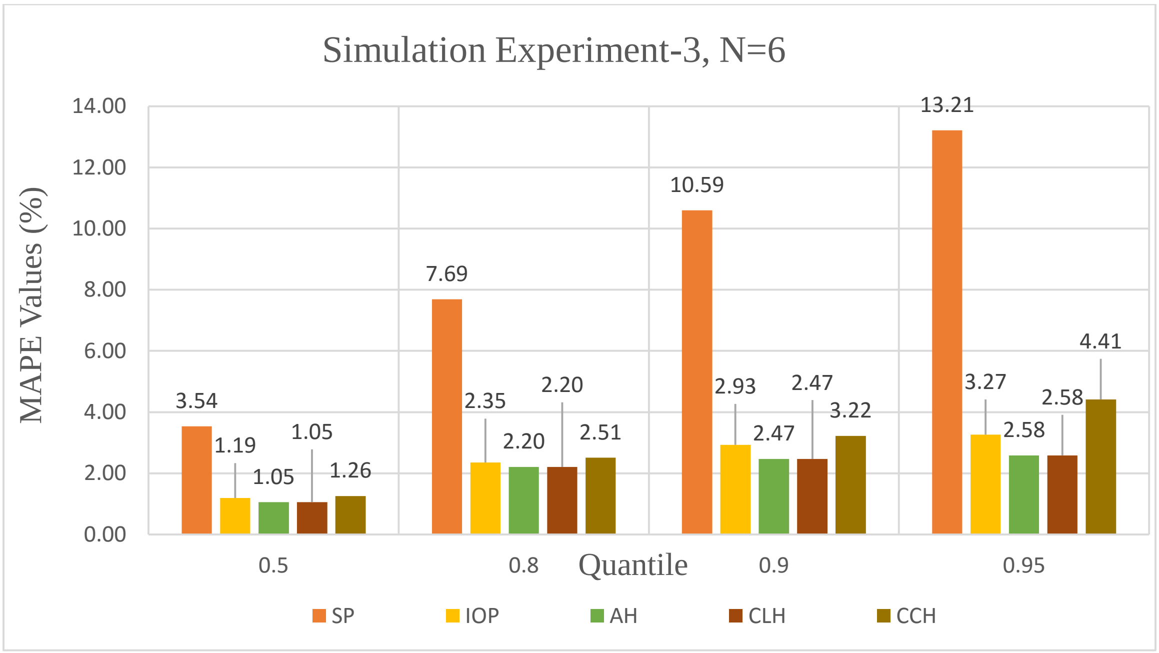


Figure 39. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 3 when N = 6.

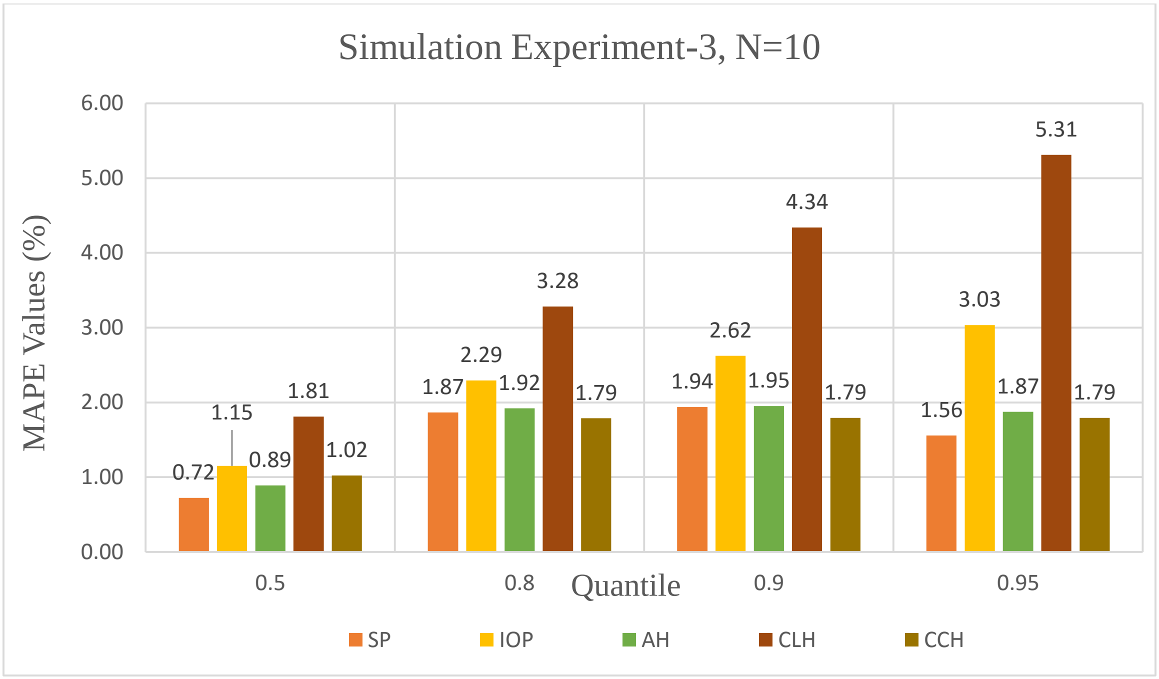


Figure 40. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 3 when N = 10.

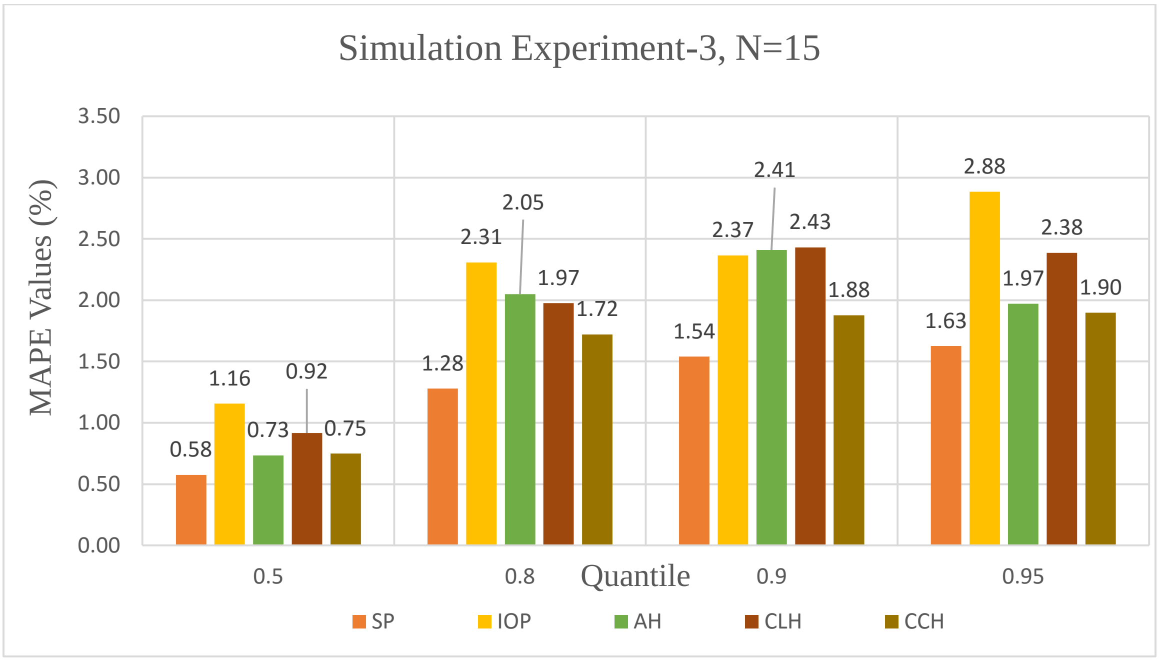


Figure 41. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 3 when N = 15.

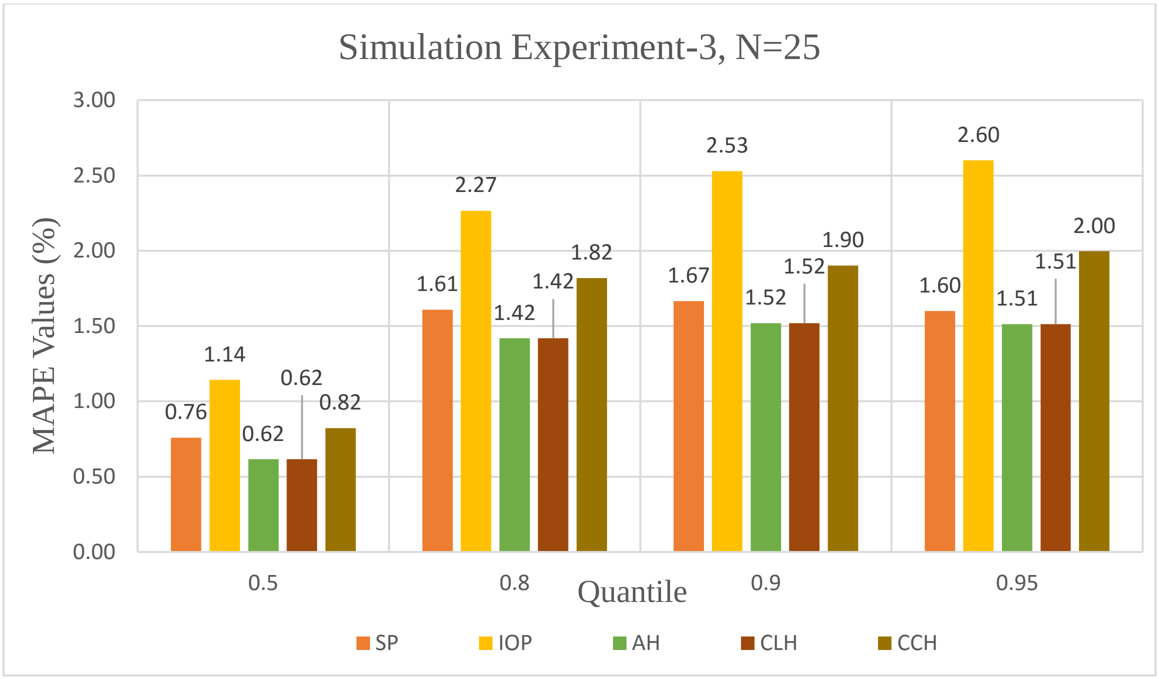


Figure 42. Comparison of MAPE values across experimental design strategies and quantiles for simulation experiment 3 when N = 25.

2.3 *Relative Performance of the Three Hybrid Experimental Design Strategies*

The previous subsections illustrated that both the number and location of design points within an experimental region have a clear impact on the predictive accuracy of quantile regression metamodels. A major contribution of this work is also in the development and examination of the three hybrid experimental design strategies, which attempt to harness the advantages of both the sphere packing and I-optimal designs, as discussed in Chapter 2. This subsection includes a comparative analysis of each of the three hybrid designs to determine which produces the best predictive accuracy for quantile regression metamodels.

One approach for comparing experimental design strategies is to examine the percentage of cases that one design has performed better (i.e., has lower MAPEs) than another design strategy. Here, a *case* is considered to be a particular simulation experiment, experimental design strategy, number of design point, and quantile combination. This research examined a total of 3 simulation experiments, 5 experimental design strategies, 4 different numbers of design points, and 4 quantiles.

Pairwise comparisons of the three hybrid design strategies were made based on this approach, and Table 11 provides such a comparison of the arbitrary hybrid and clustering hybrid experimental design strategies. Column AH gives the number of cases in which the arbitrary hybrid design performed better than the clustering hybrid design, while column CLH indicates the number of cases in which the clustering hybrid design performed better than the arbitrary hybrid design. The values are provided both by simulation experiment and overall. The percentage values in the final row of the table indicate the total percentage of cases in which one experimental design strategy has performed better than the other. Table 12 shows that in about 56% of the cases, the clustering hybrid experimental design strategy performed better than arbitrary hybrid design.

Table 12. Comparison of arbitrary and clustering hybrid designs.

	Arbitrary VS Clustering Hybrid Design		
	AH	CLH	Total cases
Simulation Experiment 1	8	8	16
Simulation Experiment 2	6	10	16
Simulation Experiment 3	7	9	16
Total	21	27	48
Percentage	43.7%	56.2%	

Table 13 provides a similar comparison between the arbitrary hybrid and centroid clustering hybrid experimental design strategies. In Table 13, the columns labeled CCH gives the number of cases in which the centroid clustering hybrid experimental design strategy performed better than the arbitrary hybrid experimental design strategy. Table 13 shows that in about 60% of the cases, the centroid clustering hybrid experimental design strategy performed better than arbitrary hybrid experimental design strategy.

Table 13. Comparison of Arbitrary and Centroid clustering hybrid designs.

	Centroid Clustering VS Arbitrary Hybrid Design		
	CCH	AH	Total cases
Simulation Experiment 1	14	2	16
Simulation Experiment 2	9	7	16
Simulation Experiment 3	6	10	16
Total	29	19	48
Percentage	60.4%	39.6%	

Finally, Table 14 provides a comparison between the clustering hybrid and centroid clustering hybrid experimental design strategies. In Table 14, the column labeled CCH indicates the number of cases where the centroid clustering hybrid experimental design strategy performed better than the clustering hybrid experimental design strategy. When comparing these two hybrid experimental design strategies, the centroid clustering hybrid experimental design strategy performed better in about 75% of the cases.

Table 14. Comparison of Clustering and Centroid clustering hybrid designs.

	Centroid Clustering VS Clustering Hybrid Design		
	CCH	CLH	Total cases
Simulation Experiment 1	16	0	16
Simulation Experiment 2	12	4	16
Simulation Experiment 3	8	8	16
Total	36	12	48
Percentage	75%	25%	

2.4 *Relative Performance of the Hybrid and Traditional (Sphere Packing and I-optimal) Experimental Design Strategies*

In addition to comparing the hybrid design strategies amongst themselves, it is important to compare their performance to the more traditional sphere packing and I-optimal experimental design strategies. These comparisons were made in the same manner that the hybrid designs were compared to each other, and the findings are given in Tables 15 – 21 In all of these tables, SP represents the sphere packing, AH represents the arbitrary hybrid, IOP represents the I-optimal, CCH represents the centroid clustering hybrid, and CLH represents the clustering hybrid experimental design strategies.

Table 15. Comparison of Sphere packing and Centroid clustering hybrid designs.

	Sphere Packing VS Centroid Clustering Hybrid Design		
	SP	CCH	Total cases
Simulation Experiment 1	8	8	16
Simulation Experiment 2	9	7	16
Simulation Experiment 3	8	8	16
Total	25	23	48
Percentage	52%	48%	

Table 16. Comparison of Sphere packing and clustering hybrid designs.

	Sphere Packing VS Clustering Hybrid Design		
	SP	CLH	Total cases
Simulation Experiment 1	13	3	16
Simulation Experiment 2	9	6	16
Simulation Experiment 3	8	8	16
Total	30	17	48
Percentage	62.5%	35.4%	

Table 17. Comparison of Sphere packing and arbitrary hybrid designs.

	Sphere Packing VS Arbitrary Hybrid Design		
	SP	AH	Total cases
Simulation Experiment 1	12	4	16
Simulation Experiment 2	9	7	16
Simulation Experiment 3	8	8	16
Total	29	19	48
Percentage	60.4%	39.6%	

Table 18. Comparison of I-Optimal and arbitrary hybrid designs.

	I-Optimal VS Arbitrary Hybrid Design		
	IOP	AH	Total cases
Simulation Experiment 1	7	9	16
Simulation Experiment 2	0	16	16
Simulation Experiment 3	1	15	16
Total	8	40	48
Percentage	16.6%	83.3%	

Table 19. Comparison of I-Optimal and clustering hybrid designs.

	I-Optimal VS Clustering Hybrid Design		
	IOP	CLH	Total cases
Simulation Experiment 1	8	8	16
Simulation Experiment 2	0	16	16
Simulation Experiment 3	5	11	16
Total	13	35	48
Percentage	37%	73%	

Table 20. Comparison of I-Optimal and Centroid clustering hybrid designs.

	I-Optimal VS Centroid Clustering Hybrid Design		
	IOP	CCH	Total cases
Simulation Experiment 1	16	0	16
Simulation Experiment 2	16	0	16
Simulation Experiment 3	12	4	16
Total	44	4	48
Percentage	91.7%	8.3%	

Table 21. Comparison of Sphere packing and I-Optimal designs.

	Sphere Packing VS I-Optimal Design		
	SP	IOP	Total cases
Simulation Experiment 1	16	0	16
Simulation Experiment 2	12	4	16
Simulation Experiment 3	12	4	16
Total	40	8	48
Percentage	83.3%	16.6%	

These comparisons highlight the comparative performance of the various experimental design strategies. The next chapter provides more details on the associated implications of these results.

Chapter 4 – Discussion and Conclusions

1. Discussion:

Comparisons based on overall performance between the three hybrid experimental design strategies are provided in Tables 12 – 14. Although the clustering hybrid design strategy performed better than the arbitrary hybrid design strategy, the percentage difference was found to be very small (i.e., only six cases of CLH were better than AH), leading to the conclusion that the two design strategies performed very similarly. Tables 13 and 14 show that the centroid clustering hybrid design strategy performed better in 60% and 75% cases than the arbitrary and clustering hybrid designs respectively. Based on conclusions drawn from Tables 12 – 14, it is clear that the overall performance of the centroid clustering hybrid experimental design strategy is better than the performance of either of the other two hybrid design strategies.

Similarly, Tables 15 – 17 provide comparisons between the sphere packing and the three hybrid experimental design strategies. The results from Tables 16 and 17 show that the sphere packing design strategy performed notably better overall than the arbitrary and clustering hybrid design, by 10% and 12% respectively. However, when the comparison is made between the sphere packing and centroid clustering hybrid design strategies in Table 15, the sphere packing design strategy was found to perform better than centroid clustering hybrid design strategy in only in two cases, implying that the two approaches performed similarly overall. Furthermore, Tables 20 and 21 illustrate that the sphere packing and centroid clustering hybrid design strategies both outperformed the I-optimal design strategy, by 83% and 92% respectively. This finding is likely due to the distribution of design points in the design space for I-optimal designs (i.e., primarily in the corners and center points vs. throughout the experimental region). While it was described earlier that I-optimal designs have superior prediction *variance*, these results confirm that they have poor prediction *accuracy* when compared to sphere packing and hybrid designs.

The overall results from the comparisons of all the experimental design strategies are summarized in Table 22. In this table, comparisons are made between the row and column labels corresponding to a particular cell in the table. The internal cells of the table indicate which of the two compared designs had superior performance, based on the percentage of cases in which the design strategy outperformed its competitor. The final column of Table 22 gives the total number of comparisons in which the experimental design strategy identified in that row was found to be superior to other design strategies. So, for example, the sphere packing design strategy was comparatively better than all four of the other experimental design strategies, while the I-Optimal design strategy was not better than any of the other design strategies. Overall, Table 22 shows that the sphere packing design strategy had the best performance, followed by the centroid clustering hybrid design. These results are also demonstrated in Figures 19 – 30, which illustrate that both of these design strategies have good predictive accuracy across the varying numbers of design points investigated.

Table 22. Summarized results from the comparisons made among five experimental designs.

	SP	**IOP**	**AH**	**CLH**	**CCH**	**Total**
SP	-	SP	SP	SP	SP	4
CCH	SP	CCH	CCH	CCH	-	3
CLH	SP	CLH	CLH	-	CCH	2
AH	SP	AH	-	CLH	CCH	1
IOP	SP	-	AH	CLH	CCH	0

Along with investigating overall, relative performance of the various experimental design strategies, particular attention should also be paid to the design strategies that perform well in cases that require minimal simulation effort (i.e., the cases in which N = 6). In the semiconductor manufacturing industry, each simulation replication can take a substantial volume of resources to execute, so design strategies that perform comparatively better with smaller simulation efforts are preferable.

Figures 19-30 clearly illustrate that the variance of prediction accuracy between the five experimental design strategies is observed most strongly as N gets smaller. When N = 25, for example, prediction variation between the design strategies is very small, implying that when the density of design points performance is quite high, nearly saturating the experimental region, it matters less which experimental design strategy is used for the placement of those points.

When only simulation experiment 2 and 3 results are considered, the difference in MAPE values across sample size or method is small. Indicating, the conclusions we make appear to be dependent on simulation experiment as well. However, when overall performance of experimental design strategies between simulation experiments is made separately, sphere-packing and centroid clustering designs perform better among all design strategies.

However, further, analysis of the performance of the two most overall competitive design strategies (i.e., sphere packing and centroid clustering hybrid) when N = 6 demonstrates that the centroid clustering hybrid design outperforms the sphere packing design. Specifically, Table 23 illustrates the comparative performance of the two designs in each of the simulation experiments when N = 6. In 75% of these cases, the centroid clustering hybrid design strategy performed better than the sphere packing design strategy, highlighting the important finding that with fewer design points, the centroid hybrid experimental design strategy has superior predictive accuracy than the sphere packing design strategy, even though the sphere packing design strategy, across all N values, was superior.

Table 23. Comparing sphere packing and centroid clustering experimental design strategies when N = 6.

	Sphere Packing VS Centroid Clustering Hybrid Design		
	SP	CCH	Total cases
Experiment 1	3	1	4
Experiment 2	0	4	4
Experiment 3	0	4	4
Total	3	9	12
Percentage	25%	75%	

2. Conclusions:

For semiconductor manufacturing companies, cycle time is a key performance indicator. In concert, the ability to accurately predict cycle times is critical, as it influences the service level and associated customer satisfaction. Discrete event simulation models are often used for this purpose, though they can require a large time investment to execute. To address the resource-intensive nature of executing simulation models, a current, active area of research examines the development of a mathematical relationship between input vectors, **x**, (i.e., controllable factors in the simulation model) and output Y (e.g., cycle time). Such models are called metamodels, or models of the simulation models, and allow predictions of the cycle time to be made at points not simulated.

The research presented in this thesis examined the impact of several experimental design strategies on the resulting predictive accuracy of metamodels developed to predict *quantiles* of the cycle time distribution. Such metamodels allow decision makers to control the level of risk associated with lead-time quotations more effectively than could be done with estimates of mean cycle time alone. The five experimental design strategies investigated in this work were: sphere packing, I-optimal, arbitrary hybrid, clustering hybrid and centroid clustering designs. Additionally, experimentation was conducted for each design strategy in the case where N = 6, 10, 15 and 25.

Overall, results obtained through this research demonstrated that the predictive accuracy of quantile regression metamodels depends on both the location and density of the design points placed in the experimental region. Of the five experimental design strategies, the sphere-packing and centroid clustering hybrid strategies most accurately predicted the cycle time quantiles across all experiments overall. Additionally, when the density of the placement of design points was high (i.e., N =15, 25), results showed that the placement of those design points within the experimental region mattered less, implying that the experimental design strategy itself was less important. However, an important additional finding was that in experiments that included fewer numbers of design points (i.e., N = 6), the centroid clustering hybrid design was found to have the best performance. Findings also indicated that instead of considering only traditional space filling and optimal design strategies, practitioners should consider using a hybrid design strategy, particularly in cases, such as within the semiconductor manufacturing industry, where the resources required to execute simulation models can be substantial, in which a smaller number of simulation model executions is preferable.

The work presented here contributes to the field of experimental design by relating experimental design strategies for computer simulation metamodeling to the predictive performance of quantile regression metamodels. A major contribution is the development of the centroid hybrid clustering design strategy, which outperformed all others in sparse designs. This finding is particularly important for industries such as the semiconductor manufacturing industry, in which the associated savings in time and effort associated with simulation model execution is significant.

3. Future Work

Future work in this area will include several important topics. First, the use of other space filling and optimal designs strategies for the development of quantile regression metamodels will be investigated. Next steps will also include comparing the experimental design strategies using performance metrics other than MAPEs (i.e., mean-squared error values). Such analyses will give an even clearer picture of how experimental design strategies influence resulting predictive accuracy.

Future work will also address some limitations of the work presented here. First, the method utilized for determining which design strategies were superior simply counted the number of cases in which one design performed than another across all experiments. However, such an approach does not effectively allow for consideration of differences in these meta-results that are evident only when examining the results of an individual simulation experiment. Future work will include the use of different methods of comparing the performance of design strategies such that the conclusions are more generalizable across experiments based on different sets of predictor variables. In addition, the study here utilized as an experimental testbed a model of a semiconductor manufacturing facility. Future work will include an examination how generalizable those findings are to other manufacturing settings.

Finally, the work here demonstrated the impact of experimental design strategies on the predictive performance of quantile regression based metamodels for the case in which two predictor variables were included in metamodel. Theoretically, however, there is no limitation to the number of predictor variables that can be included in such metamodels. For example, a quantile regression metamodel could be developed to predict cycle time quantiles based on the predictor variables TBA, COV, and MTTF simultaneously (vs. only a subset of these two variables). The impact of experimental design strategies on the predictive performance of quantile regression metamodels developed based on more than two predictor variables represents another important aspect of future work.

REFERENCES

Atherton, L. F. and Atherton, R. W. (1995) *Wafer Fabrication: Factory Performance and Analysis*, Kluwer Academic Publishers, Boston, Dordrecht, London.

Ankenman, B., Nelson, B.L., Tongarlak, M., Fowler, J., Mackulak, G., Pabst, D. and Yang, F. (2007) Cycle time prediction for semiconductor manufacturing via simulation on demand, *Statistical tools for simulation practitioners*.

Babulak, E. and Wang, M. (2010) *Discrete Event Simulations Simulation: State of the Art, Discrete Event Simulations*, InTech, Croatia.

Boyaci, T. and Ray, S. (2006) The Impact of Capacity Costs on Product Differentiation in Delivery Time, Delivery Reliability, and Price, *Production and Operations Management*, **15**, 179–197.

Bekki, J.M., Chen, X. and Batur, D. (2014) Steady-state quantile parameter estimation: Empirical comparison of stochastic kriging and quantile regression, to appear in A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, & J. A. Miller (Eds.) *Proceedings of the 2014 Winter Simulation Conference*, IEEE Press, Savanah, GA, pp. 3880 – 3891.

Chung, S. and Huang, H. (2002) Cycle time estimation for wafer fab with engineering lots, *IIE Transactions*, **34**, 105-118.

Chen, X., Fernandez, E. and Kelton, D. (2012) Optimization model selection for simulation-based approximate dynamic programming approaches in semiconductor manufacturing operations, *Winter Simulation Conference Proceedings*, pp. 1977-1988.

Challenor, P. (2013) Experimental design for the validation of kriging metamodels in computer experiments, *Journal of Simulation*, **7**, 290-296.

Chen, T. and Wang, Y.C. (2009) A nonlinear scheduling rule incorporating fuzzy-neural remaining cycle time estimator for scheduling a semiconductor manufacturing factory—a simulation study, *The International Journal of Advanced Manufacturing Technology*, **45**(1), 110-121.

Choi, B.K. and Kang, D.H (2013) *Modeling and simulation of discrete- event systems*, Wiley, Hoboken, New Jersey.

Hunter, J.S. and Naylor, T.H. (1970) Experimental design for computer simulation experiments, in *Management Science (pre-1986)*, 422.

Hernández, Z. E., Niño, E., Lara, Y., Villarreal-Marroquín, M. G., Castro, J. M. and Ríos, M. C. (2010) Metamodeling-based optimization for true experiments. *IIE Annual Conference. Proceedings*, 1-6.

Johnson, R.T., Montgomery, D.C., Jones, B. and Parker P.A. (2010) Comparing Computer Experiments for Fitting High-Order Polynomial Metamodels, *Journal of Quality Technology*, **42**, 86-102.

Jin, R., Chen, W. and Simpson, T.W. (2001) Comparative studies of metamodeling techniques under multiple modelling criteria, *Journal of structural and multidisciplinary optimization*, **23**(1), pp. 1-13

Johnson R.T., Jones, B., Fowler, J.W. and Montgomery, D.C. (2008) Comparing designs for computer simulation experiments, *Proceedings of the 2008 Winter Simulation Conference*, Austin, TX, pp. 463-470.

Kleijnen, Jack P.C. (1987) *Design and Analysis of Simulation Experiments*, Springer, New York, NY.

Kerman, M.C., Jiang, W., Blumberg, A.F. and Buttrey, S.E. (2008) A comparison of robust metamodels for the uncertainty quantification of New York harbor oceanographic data, *Journal of operational oceanography*, **1**, 3-13.

Kennedy, K. (2013) Bridging the gap between space-filling and optimal designs design for computer experiments, Retrieved from Dissertations & Theses @ Arizona State University; ProQuest Dissertations & Theses Global.

Kerman, M.C., Jiang, W., Blumberg, A.F. and Buttrey S.E. (2009) The application of a quantile regression metamodel for salinity event detection confirmation within New York Harbour oceanographic data, *Journal of Operational Oceanography*, **2**(1), 49-70.

Katchova, Ani (2013, February 24) *Quantile Regression in R*, [Video files]. Retrieved from: https://www.youtube.com/watch?v=ucURUTVjBRo.

Katchova, Ani (2013, February 24) *Econometrics - Quantile Regression* [Video files]. Retrieved from: https://youtu.be/P9lMmEkXuBw?list=PLRW9kMvtNZOhHpZGCZ5lfQgDJ1yS27pVC.

Koenker, R. and Bassett, G. (1978) Regression quantiles, *Econometrica*, **46**, pp. 33-50.

Koenker, R. and Hallock, K.F. (2001) Quantile regression, *Journal of Economic Perspectives*, **15**, pp. 143-156.

Kempf, K. (1994) Detailed Description of Multiple Product Re-entrant Semiconductor Manufacturing System Example, *Prepared report*, Intel Corporation, Technology, and Manufacturing Group.

Montgomery, D.C. (2009) *Design and analysis of engineering experiments*, Wiley, New York, NY.

Meyersdorf, D. and Yang, T. (1997) Cycle time reduction for semiconductor wafer fabrication facilities, in *Advanced Semiconductor Manufacturing Conference and Workshop*, IEEE Press, Cambridge, MA, 418-423.

Myers, R.H., Montgomery, D.C. and Anderson –Cook, C.M. (2010), *Response Surface Methodology: Process and Product Optimization Using Designed Experiments, 3rd Edition*, Wiley, New York, NY.

Pfund, M.E., Mason, S.J. and Fowler, J.W. (2006) *Semiconductor manufacturing scheduling and dispatching*, Springer, New York.

Tibshirani, R. (2011) Regression shrinkage and selection via the lasso: A retrospective, *Journal of the Royal Statistical Society. Series B (Statistical Methodology*), **73**(3), 273-282.

Wang, C.N. and Wang, C.H. (2007) A simulated model for cycle time reduction by acquiring optimal lot size in semiconductor manufacturing, *The International Journal of Advanced Manufacturing Technology*, **34**, 1008-1015.

Yin, J., Ng, S.H. and Ng, K.M. (2010) A bayesian metamodeling approach for stochastic simulations, in *proceedings of 2010 winter conference*, IEEE Press, Los Alamitos, CA, pp. 1055-1066.

APPENDIX A

SPHERE PACKING DESIGN POINTS FOR ALL SIMULATION EXPERIMENTS

A. Sphere packing design points for all simulation experiments.

Table 1. Design points of sphere packing design for simulation experiment-1 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00297	0.69176
2	0.00395	0.9
3	0.005	0.7137
4	0.00403	0.1
5	0.002	0.1
6	0.002	0.9
7	0.005	0.30847
8	0.002	0.48352
9	0.00401	0.50828
10	0.00302	0.29394

Table 2. Design points of sphere packing design for simulation experiment-1 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00306	0.66376
2	0.00404	0.6481
3	0.00428	0.9
4	0.00484	0.49997
5	0.00426	0.30637
6	0.002	0.67689
7	0.00374	0.1
8	0.0025	0.46666
9	0.00335	0.9
10	0.005	0.1
11	0.002	0.25747
12	0.00334	0.35463
13	0.00282	0.14291
14	0.00241	0.9
15	0.005	0.74415

B. Sphere packing design points for all simulation experiments.

Table 3. Design points of sphere packing design for simulation experiment-1 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	0.1
2	0.002	0.292401
3	0.002	0.714683
4	0.002066	0.523901
5	0.002182	0.9
6	0.002603	0.396662
7	0.002621	0.196197
8	0.00275	0.703383
9	0.002901	0.9
10	0.003121	0.529266
11	0.003224	0.300456
12	0.003243	0.1
13	0.003518	0.802161
14	0.003752	0.620995
15	0.003775	0.423364
16	0.003845	0.204225
17	0.004136	0.9
18	0.00437	0.718861
19	0.004379	0.527294
20	0.004397	0.326939
21	0.004448	0.1
22	0.005	0.222726
23	0.005	0.625091
24	0.005	0.431126
25	0.005	0.878039

Table 4. Design points of sphere packing design for simulation experiment-2 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	24.99998
2	0.004	25.00003
3	0.003	10
4	0.003	40
5	0.005	40
6	0.005	10

A. Sphere packing design points for all simulation experiments.

Table 5. Design points of sphere packing design for simulation experiment-2 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0050	22.508
2	0.0020	10.000
3	0.0029	19.664
4	0.0039	39.998
5	0.0049	10.000
6	0.0020	28.938
7	0.0050	35.055
8	0.0038	27.405
9	0.0037	10.000
10	0.0026	40.0000

Table 6. Design points of sphere packing design for simulation experiment-2 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00417	24.80161
2	0.00344	31.33343
3	0.005	19.43216
4	0.00296	12.37855
5	0.002	10
6	0.002	38.41348
7	0.00396	39.71065
8	0.005	30.17318
9	0.00297	40
10	0.00494	40
11	0.00472	10
12	0.00208	20.21398
13	0.00309	22.13211
14	0.00389	15.36945
15	0.00246	29.71288

A. Sphere packing design points for all simulation experiments.

Table 7. Design points of sphere packing design for simulation experiment-2 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	10
2	0.002	17.21504
3	0.002	33.0506
4	0.002066	25.8963
5	0.002182	40
6	0.002603	21.12484
7	0.002621	13.60739
8	0.00275	32.62684
9	0.002901	40
10	0.003121	26.09747
11	0.003224	17.5171
12	0.003243	10
13	0.003518	36.33105
14	0.003752	29.53733
15	0.003775	22.12617
16	0.003845	13.90843
17	0.004136	40
18	0.00437	33.2073
19	0.004379	26.02353
20	0.004397	18.5102
21	0.004448	10
22	0.005	14.60221
23	0.005	29.69093
24	0.005	22.41722
25	0.005	39.17646

Table 8. Design points of sphere packing design for simulation experiment-3 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	215.9999
2	0.004	216.0002
3	0.003	108
4	0.003	324
5	0.005	324
6	0.005	108

A. Sphere packing design points for all simulation experiments.

Table 9. Design points of sphere packing design for simulation experiment-3 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	108
2	0.002	249.7403
3	0.00378	126.9187
4	0.002	108
5	0.00271	324
6	0.005	276.872
7	0.00465	192.436
8	0.00275	177.4822
9	0.00362	239.5642
10	0.00397	324

Table 10. Design points of sphere packing design for simulation experiment-3 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002567	175.76
2	0.002558	240.6486
3	0.004301	179.8666
4	0.003324	208.7261
5	0.004172	324
6	0.005	140.822
7	0.00335	277.3105
8	0.003543	147.0445
9	0.002	126.8763
10	0.002748	324
11	0.004242	108
12	0.002	290.0418
13	0.005	277.7912
14	0.004081	241.548
15	0.002845	108

A. Sphere packing design points for all simulation experiments.

Table 11. Design points of sphere packing design for simulation experiment-3 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	108
2	0.002	159.9483
3	0.002	273.9643
4	0.002066	222.4534
5	0.002182	324
6	0.002603	188.0989
7	0.002621	133.9732
8	0.00275	270.9133
9	0.002901	324
10	0.003121	223.9017
11	0.003224	162.1231
12	0.003243	108
13	0.003518	297.5836
14	0.003752	248.6688
15	0.003775	195.3084
16	0.003845	136.1407
17	0.004136	324
18	0.00437	275.0925
19	0.004379	223.3694
20	0.004397	169.2734
21	0.004448	108
22	0.005	141.1359
23	0.005	249.7747
24	0.005	197.404
25	0.005	318.0705

APPENDIX B

I - OPTIMAL DESIGN POINTS FOR ALL SIMULATION EXPERIMENTS

B. I-Optimal design points for all simulation experiments

Table 12. Design points of I-optimal design for simulation experiment-1 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	0.9
2	0.005	0.9
3	0.002	0.5
4	0.002	0.1
5	0.005	0.1
6	0.0035	0.1
7	0.0035	0.5
8	0.005	0.5
9	0.0035	0.9
10	0.0035	0.5

Table 13. Design points of I-optimal design for simulation experiment-1 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	0.5
2	0.002	0.5
3	0.0035	0.9
4	0.0035	0.1
5	0.002	0.1
6	0.005	0.5
7	0.005	0.9
8	0.005	0.1
9	0.0035	0.5
10	0.005	0.5
11	0.0035	0.1
12	0.002	0.9
13	0.0035	0.5
14	0.0035	0.9
15	0.0035	0.5

B. I-Optimal design points for all simulation experiments

Table 14. Design points of I-optimal design for simulation experiment-1 when N=25.

Design Point	**Time Between Arrivals**	**Coefficient of variance**
1	0.002	0.5
2	0.003395	0.468
3	0.002	0.9
4	0.005	0.1
5	0.005	0.5
6	0.0035	0.5
7	0.0035	0.5
8	0.0035	0.5
9	0.002	0.9
10	0.0035	0.5
11	0.0035	0.9
12	0.0035	0.1
13	0.002	0.1
14	0.005	0.5
15	0.0035	0.9
16	0.0035	0.9
17	0.002	0.5
18	0.005	0.1
19	0.005	0.5
20	0.005	0.9
21	0.0035	0.5
22	0.002	0.1
23	0.0035	0.1
24	0.0035	0.5
25	0.005	0.9

Table 15. Design points of I-optimal design for simulation experiment-2 when N=6.

Design Point	**Time Between Arrivals**	**Coefficient of variance**
1	0.002	40
2	0.005	10
3	0.005	40
4	0.0035	25
5	0.00263	10
6	0.002	21.25

B. I-Optimal design points for all simulation experiments

Table 16. Design points of I-optimal design for simulation experiment-2 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	40
2	0.002	25
3	0.0035	40
4	0.0035	25
5	0.005	25
6	0.005	10
7	0.002	10
8	0.0035	10
9	0.005	40
10	0.0035	25

Table 17. Design points of I-optimal design for simulation experiment-2 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	40
2	0.0035	25
3	0.002	10
4	0.005	10
5	0.0035	25
6	0.005	40
7	0.002	25
8	0.005	25
9	0.0035	10
10	0.0035	10
11	0.005	25
12	0.0035	40
13	0.0035	40
14	0.0035	25
15	0.002	25

B. I-Optimal design points for all simulation experiments

Table 18. Design points of I-optimal design for simulation experiment-2 when N=25.

Design Point	**Time Between Arrivals**	**Coefficient of variance**
1	0.005	25
2	0.0035	25
3	0.002	10
4	0.0035	25
5	0.005	25
6	0.0035	10
7	0.002	25
8	0.002	25
9	0.002	25
10	0.002	40
11	0.0035	25
12	0.0035	25
13	0.002	40
14	0.005	40
15	0.0035	40
16	0.005	40
17	0.005	10
18	0.003635	26.65
19	0.002	10
20	0.005	10
21	0.0035	25
22	0.0035	40
23	0.0035	25
24	0.0035	10
25	0.0035	10

Table 19. Design points of I-optimal design for simulation experiment-3 when N=6.

Design Point	**Time Between Arrivals**	**Coefficient of variance**
1	0.005	108
2	0.005	324
3	0.0035	216
4	0.003365	324
5	0.002	108
6	0.002	275.4

B. I-Optimal design points for all simulation experiments

Table 20. Design points of I-optimal design for simulation experiment-3 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0035	216
2	0.005	324
3	0.005	108
4	0.0035	108
5	0.002	108
6	0.002	216
7	0.002	324
8	0.0035	216
9	0.005	216
10	0.0035	324

Table 21. Design points of I-optimal design for simulation experiment-3 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	108
2	0.005	108
3	0.0035	216
4	0.005	324
5	0.002	216
6	0.005	216
7	0.0035	216
8	0.0035	216
9	0.0035	324
10	0.002	216
11	0.0035	108
12	0.0035	108
13	0.002	324
14	0.005	216
15	0.0035	324

B. I-Optimal design points for all simulation experiments

Table 22. Design points of I-optimal design for simulation experiment-3 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	324
2	0.0035	216
3	0.002	216
4	0.005	216
5	0.005	324
6	0.0035	216
7	0.005	108
8	0.0035	108
9	0.005	108
10	0.0035	324
11	0.0035	324
12	0.0035	216
13	0.0035	216
14	0.0035	216
15	0.002	108
16	0.00365	207.36
17	0.0035	108
18	0.002	108
19	0.002	324
20	0.005	216
21	0.002	216
22	0.005	324
23	0.0035	324
24	0.002	216
25	0.0035	216

APPENDIX C

ARBITRARY HYBRID DESIGN POINTS FOR ALL SIMULATION EXPERIMENTS

C. Arbitrary hybrid design points for all simulation experiments.

Table 23. Design points of arbitrary hybrid design for simulation experiment-1 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00297	0.69176
2	0.00395	0.9
3	0.005	0.7137
4	0.00403	0.1
5	0.005	0.30847
6	0.00302	0.29394
7	0.002	0.9
8	0.005	0.9
9	0.002	0.1
10	0.0035	0.5

Table 24. Design points of arbitrary hybrid design for simulation experiment-1 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.00306	0.66376
2	0.00404	0.6481
3	0.00428	0.9
4	0.00426	0.30637
5	0.002	0.67689
6	0.00374	0.1
7	0.0025	0.46666
8	0.00282	0.14291
9	0.0035	0.9
10	0.002	0.1
11	0.005	0.5
12	0.005	0.9
13	0.005	0.1
14	0.0035	0.5
15	0.002	0.9

C. Arbitrary hybrid design points for all simulation experiments

Table 25. Design points of arbitrary hybrid design for simulation experiment-1 when N=25

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	0.5
2	0.002	0.9
3	0.005	0.1
4	0.005	0.5
5	0.0035	0.5
6	0.0035	0.9
7	0.0035	0.1
8	0.005	0.9
9	0.002	0.1
10	0.002	0.292401
11	0.002	0.714683
12	0.002603	0.396662
13	0.002621	0.196197
14	0.00275	0.703383
15	0.002901	0.9
16	0.003224	0.300456
17	0.003752	0.620995
18	0.003845	0.204225
19	0.004136	0.9
20	0.00437	0.718861
21	0.004379	0.527294
22	0.004397	0.326939
23	0.004448	0.1
24	0.005	0.222726
25	0.005	0.625091

Table 26. Design points of arbitrary hybrid design for simulation experiment-2 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	40
2	0.0035	25
3	0.00263	10
4	0.002	24.99998
5	0.005	40
6	0.005	10

C. Arbitrary hybrid design points for all simulation experiments

Table 27. Design points of arbitrary hybrid design for simulation experiment-2 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	40
2	0.002	25
3	0.005	10
4	0.0035	10
5	0.005	40
6	0.005	22.50811
7	0.002	10
8	0.00285	19.66358
9	0.00385	39.9979
10	0.00384	27.40471

Table 28. Design points of arbitrary hybrid design for simulation experiment-2 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	40
2	0.005	10
3	0.0035	25
4	0.005	40
5	0.0035	10
6	0.0035	40
7	0.002	25
8	0.00417	24.80161
9	0.00344	31.33343
10	0.005	19.43216
11	0.00296	12.37855
12	0.002	10
13	0.00396	39.71065
14	0.005	30.17318
15	0.00246	29.71288

C. Arbitrary hybrid design points for all simulation experiments

Table 29. Design points of arbitrary hybrid design for simulation experiment-2 when N=25

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	10
2	0.002	17.21504
3	0.002	33.0506
4	0.002603	21.12484
5	0.002621	13.60739
6	0.00275	32.62684
7	0.002901	40
8	0.003121	26.09747
9	0.003224	17.5171
10	0.003752	29.53733
11	0.003845	13.90843
12	0.004136	40
13	0.00437	33.2073
14	0.004379	26.02353
15	0.004397	18.5102
16	0.004448	10
17	0.005	29.69093
18	0.005	25
19	0.0035	25
20	0.0035	10
21	0.002	25
22	0.002	40
23	0.005	40
24	0.0035	40
25	0.005	10

Table 30. Design points of arbitrary hybrid design for simulation experiment-3 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0035	216
2	0.002	108
3	0.002	275.4
4	0.003	324
5	0.005	324
6	0.005	108

C. Arbitrary hybrid design points for all simulation experiments

Table 31. Design points of arbitrary hybrid design for simulation experiment-3 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	324
2	0.0035	108
3	0.002	324
4	0.005	216
5	0.0035	324
6	0.005	108
7	0.002	249.7403
8	0.002	108
9	0.00275	177.4822
10	0.00362	239.5642

Table 32. Design points of arbitrary hybrid design for simulation experiment-3 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	108
2	0.005	108
3	0.0035	216
4	0.005	324
5	0.005	216
6	0.002	216
7	0.0035	108
8	0.002	324
9	0.002567	175.76
10	0.004301	179.8666
11	0.004172	324
12	0.00335	277.3105
13	0.002748	324
14	0.004242	108
15	0.005	277.7912

C. Arbitrary hybrid design points for all simulation experiments

Table 33. Design points of arbitrary hybrid design for simulation experiment-3 when N=25

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	324
2	0.002	216
3	0.0035	108
4	0.0035	324
5	0.00365	207.36
6	0.005	216
7	0.005	324
8	0.005	108
9	0.002	108
10	0.002	159.9483
11	0.002	273.9643
12	0.002603	188.0989
13	0.002621	133.9732
14	0.00275	270.9133
15	0.002901	324
16	0.003121	223.9017
17	0.003224	162.1231
18	0.003752	248.6688
19	0.003845	136.1407
20	0.004136	324
21	0.00437	275.0925
22	0.004379	223.3694
23	0.004397	169.2734
24	0.004448	108
25	0.005	141.1359

APPENDIX D

CLUSTERING HYBRID DESIGN POINTS FOR ALL SIMULATION EXPERIMENTS

D. Clustering hybrid design points for all simulation experiments

Table 34. Design points of clustering hybrid design for simulation experiment-1 when N=10

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0035	0.1
2	0.002	0.9
3	0.0035	0.5
4	0.00401	0.50828
5	0.005	0.9
6	0.002	0.48352
7	0.005	0.7137
8	0.005	0.30847
9	0.0035	0.9
10	0.002	0.1

Table 35. Design points of clustering hybrid design for simulation experiment-1 when N=15

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	0.1
2	0.005	0.1
3	0.00426	0.30637
4	0.00282	0.14291
5	0.00306	0.66376
6	0.005	0.5
7	0.005	0.5
8	0.00484	0.49997
9	0.00428	0.9
10	0.00241	0.9
11	0.002	0.67689
12	0.002	0.9
13	0.005	0.9
14	0.005	0.74415
15	0.0035	0.1

D. Clustering hybrid design points for all simulation experiments

Table 36. Design points of clustering hybrid design for simulation experiment-1 when N=25

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	0.5
2	0.005	0.5
3	0.005	0.5
4	0.0035	0.1
5	0.0035	0.1
6	0.003243	0.1
7	0.003845	0.204225
8	0.004448	0.1
9	0.005	0.222726
10	0.00275	0.703383
11	0.003121	0.529266
12	0.005	0.1
13	0.005	0.1
14	0.003395	0.468
15	0.0035	0.5
16	0.0035	0.5
17	0.0035	0.5
18	0.0035	0.5
19	0.0035	0.5
20	0.0035	0.5
21	0.004397	0.326939
22	0.005	0.625091
23	0.003518	0.802161
24	0.003775	0.423364
25	0.002603	0.396662

Table 36. Design points of clustering hybrid design for simulation experiment-2 when N=6

Design Point	Time Between Arrivals	Coefficient of variance
1	0.003	40
2	0.0035	25
3	0.004	25.00003
4	0.005	40
5	0.005	40
6	0.002	40

D. Clustering hybrid design points for all simulation experiments

Table 37. Design points of clustering hybrid design for simulation experiment-2 when N=10

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0035	40
2	0.0039	39.998
3	0.002	40
4	0.002	28.938
5	0.0035	25
6	0.0035	25
7	0.0038	27.405
8	0.005	10
9	0.0035	10
10	0.0049	10

Table 38. Design points of clustering hybrid design for simulation experiment-2 when N=15

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	10
2	0.00472	10
3	0.005	25
4	0.005	25
5	0.005	19.43216
6	0.0035	10
7	0.0035	10
8	0.00296	12.37855
9	0.00309	22.13211
10	0.0035	25
11	0.0035	25
12	0.0035	25
13	0.002	25
14	0.002	25
15	0.00208	20.21398

D. Clustering hybrid design points for all simulation experiments

Table 39. Design points of clustering hybrid design for simulation experiment-2 when N=25

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	40
2	0.005	40
3	0.005	39.176465
4	0.002621	13.607386
5	0.004448	10
6	0.005	10
7	0.005	10
8	0.005	14.602209
9	0.00275	32.626844
10	0.0035	25
11	0.0035	25
12	0.0035	25
13	0.0035	25
14	0.0035	25
15	0.0035	25
16	0.003121	26.097465
17	0.003635	26.65
18	0.002	25
19	0.002	25
20	0.002	25
21	0.002	33.050598
22	0.004397	18.510201
23	0.005	29.690929
24	0.003518	36.331051
25	0.003775	22.126166

Table 40. Design points of clustering hybrid design for simulation experiment-3 when N=6

Design Point	Time Between Arrivals	Coefficient of variance
1	0.003	324
2	0.0035	216
3	0.004	216.0002
4	0.005	324
5	0.005	324
6	0.005	108

D. Clustering hybrid design points for all simulation experiments

Table 41. Design points of clustering hybrid design for simulation experiment-3 when N=10

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	108
2	0.002	108
3	0.0035	216
4	0.0035	216
5	0.00275	177.4822
6	0.005	276.872
7	0.0035	324
8	0.00271	324
9	0.00362	239.5642
10	0.005	324

Table 42. Design points of clustering hybrid design for simulation experiment-3 when N=15

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	324
2	0.0035	216
3	0.0035	216
4	0.0035	216
5	0.003324	208.7261
6	0.002	216
7	0.002	216
8	0.005	277.7912
9	0.005	216
10	0.005	216
11	0.004301	179.8666
12	0.004081	241.548
13	0.003543	147.0445
14	0.005	108
15	0.005	140.822

D. Clustering hybrid design points for all simulation experiments

Table 43. Design points of clustering hybrid design for simulation experiment-3 when N=25

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0035	216
2	0.0035	216
3	0.0035	216
4	0.003121	223.9017
5	0.002621	133.9732
6	0.003243	108
7	0.004448	108
8	0.005	141.1359
9	0.00275	270.9133
10	0.002	108
11	0.002	108
12	0.002	108
13	0.005	108
14	0.005	108
15	0.0035	324
16	0.0035	324
17	0.0035	324
18	0.002901	324
19	0.004136	324
20	0.004397	169.2734
21	0.005	216
22	0.005	216
23	0.005	249.7747
24	0.003518	297.5836
25	0.0035	108

APPENDIX E

CENTROID CLUSTERING HYBRID DESIGN POINTS FOR ALL SIMULATION EXPERIMENTS

E. Centroid Clustering Design points for all simulation experiments.

Table 44. Design points of centroid clustering hybrid design for simulation experiment-1 when N=10.

Design Point	**Time Between Arrivals**	**Coefficient of variance**
1	0.0038	0.1
2	0.002	0.9
3	0.0033	0.5639
4	0.004	0.5083
5	0.005	0.9
6	0.0023	0.4258
7	0.005	0.7137
8	0.005	0.3028
9	0.0037	0.9
10	0.002	0.1

Table 45. Design points of centroid clustering hybrid design for simulation experiment-1 when N=15.

Design Point	**Time Between Arrivals**	**Coefficient of variance**
1	0.005	0.1
2	0.0043	0.3064
3	0.0028	0.1429
4	0.0031	0.6638
5	0.0049	0.5
6	0.0043	0.9
7	0.0021	0.8256
8	0.005	0.8221
9	0.0036	0.1
10	0.004	0.6481
11	0.0035	0.9
12	0.0035	0.4637
13	0.002	0.1787
14	0.0034	0.9
15	0.0022	0.4889

D. Clustering hybrid design points for all simulation experiments

Table 46. Design points of centroid clustering hybrid design for simulation experiment-1 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	0.5
2	0.0035	0.1261
3	0.0044	0.1
4	0.005	0.2227
5	0.0027	0.7034
6	0.0031	0.5293
7	0.005	0.1
8	0.0035	0.4954
9	0.0044	0.3269
10	0.005	0.6251
11	0.0035	0.8022
12	0.0038	0.4234
13	0.0026	0.3967
14	0.0035	0.9
15	0.0021	0.9
16	0.0022	0.124
17	0.002	0.2924
18	0.0032	0.3005
19	0.0044	0.5273
20	0.002	0.508
21	0.005	0.8927
22	0.005	0.4311
23	0.0038	0.621
24	0.002	0.7147
25	0.0044	0.7189

Table 47. Design points of centroid clustering hybrid design for simulation experiment-2 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.003	40
2	0.0038	25
3	0.005	40
4	0.002	40
5	0.0039	10
6	0.002	23.125

D. Clustering hybrid design points for all simulation experiments

Table 48. Design points of centroid clustering hybrid design for simulation experiment-2 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0037	39.999
2	0.002	40
3	0.002	28.938
4	0.0036	25.8017
5	0.0043	10
6	0.002	25
7	0.0026	40
8	0.0029	19.664
9	0.005	30.6407
10	0.002	10

Table 49. Design points of centroid clustering hybrid design for simulation experiment-2 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0049	10
2	0.005	23.1441
3	0.0033	10.7929
4	0.0031	22.1321
5	0.0035	25
6	0.002	23.4047
7	0.0039	15.3695
8	0.005	30.1732
9	0.0025	29.7129
10	0.0042	24.8016
11	0.005	40
12	0.0035	39.9277
13	0.002	39.2067
14	0.0034	31.3334
15	0.002	10

D. Clustering hybrid design points for all simulation experiments

Table 50. Design points of centroid clustering hybrid design for simulation experiment-2 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	39.7255
2	0.0026	13.6074
3	0.0044	10
4	0.005	11.5341
5	0.0027	32.6268
6	0.0034	25.1568
7	0.0036	26.65
8	0.002	25
9	0.002	33.0506
10	0.0044	18.5102
11	0.005	29.6909
12	0.0035	36.3311
13	0.0038	22.1262
14	0.0026	21.1248
15	0.005	25
16	0.0021	40
17	0.0035	10.7817
18	0.002	11.8038
19	0.0032	17.5171
20	0.0044	26.0235
21	0.0021	25.8963
22	0.0035	40
23	0.005	22.4172
24	0.0038	29.5373
25	0.0044	33.2073

Table 51. Design points of centroid clustering hybrid design for simulation experiment-3 when N=6.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0032	324
2	0.0038	216.0001
3	0.005	324
4	0.005	108
5	0.0025	108
6	0.002	245.6999

D. Clustering hybrid design points for all simulation experiments

Table 52. Design points of centroid clustering hybrid design for simulation experiment-3 when N=10.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.002	108
2	0.0033	203.1607
3	0.005	276.872
4	0.0031	324
5	0.0036	239.5642
6	0.005	324
7	0.004	324
8	0.0036	117.4593
9	0.0049	156.109
10	0.002	263.2468

Table 53. Design points of centroid clustering hybrid design for simulation experiment-3 when N=15.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.005	324
2	0.0035	214.1815
3	0.002	216
4	0.005	277.7912
5	0.0048	203.9555
6	0.0041	241.548
7	0.0035	147.0445
8	0.005	124.4
9	0.0028	108
10	0.0036	312.3276
11	0.0026	175.76
12	0.0022	312.6806
13	0.0037	108
14	0.002	117.4382
15	0.0026	240.6486

D. Clustering hybrid design points for all simulation experiments

Table 54. Design points of centroid clustering hybrid design for simulation experiment-3 when N=25.

Design Point	Time Between Arrivals	Coefficient of variance
1	0.0034	217.1288
2	0.0029	120.9866
3	0.0044	108
4	0.005	141.1359
5	0.0027	270.9133
6	0.002	108
7	0.005	108
8	0.0035	324
9	0.0044	169.2734
10	0.005	216
11	0.005	249.7747
12	0.0035	297.5836
13	0.0036	117.3802
14	0.0037	201.3342
15	0.0026	188.0989
16	0.005	322.0235
17	0.002	311.4911
18	0.002	216
19	0.002	159.9483
20	0.0032	162.1231
21	0.0044	223.3694
22	0.0021	222.4534
23	0.005	197.404
24	0.0038	248.6688
25	0.0044	275.0925

APPENDIX F

“.MOD” AND “.EXP” FILES OF MINIFAB SIMULATION MODEL

F. “.mod” and “.exp” files of Minifab simulation model.

.mod file :

```
;     Model statements for module:  BasicProcess.Create 1
(Part X)
;

66$           CREATE,
1,MinutesToBaseTime(0.01),Entity Type
X:MinutesToBaseTime(EXPO(1/TBA)):NEXT(67$);

67$           ASSIGN:        Part X.NumberOut=Part
X.NumberOut + 1:NEXT(14$);


;
;
;     Model statements for module:  BasicProcess.Assign 1
(Assign X)
;
14$           ASSIGN:        Arrival Time=TNOW:
                             Entity.Sequence=Sequence:
                             Entity.Type=1:NEXT(10$);


;
;
;     Model statements for module:  AdvancedTransfer.Route
4 (Route1)
;
10$           ROUTE:         0.000000000000000,SEQ;


;
;
;     Model statements for module:  BasicProcess.Create 2
(Part Y)
;

70$           CREATE,
1,MinutesToBaseTime(0.01),Entity Type
Y:MinutesToBaseTime(EXPO(333.33)):NEXT(71$);
```

```
71$            ASSIGN:        Part Y.NumberOut=Part
Y.NumberOut + 1:NEXT(18$);


;
;
;     Model statements for module:  BasicProcess.Assign 2
(Assign Y)
;
18$            ASSIGN:        Entity.Sequence=Sequence:
                              Entity.Type=2:NEXT(10$);


;
;
;     Model statements for module:
AdvancedTransfer.Station 1 (Station2)
;

4$             STATION,       Station 2;
76$            DELAY:         0.0,,VA:NEXT(17$);


;

;     Model statements for module:  BasicProcess.Decide 3
(Job Step?)
;
17$            BRANCH,        1:
                              If,Entity.JobStep==2,77$,Yes:
                              Else,78$,Yes;
77$            ASSIGN:        Job Step?.NumberOut True=Job
Step?.NumberOut True + 1:NEXT(27$);

78$            ASSIGN:        Job Step?.NumberOut False=Job
Step?.NumberOut False + 1:NEXT(32$);


;
;
;     Model statements for module:  BasicProcess.Process 10
(Load 1 CD)
;
27$            ASSIGN:        Load 1 CD.NumberIn=Load 1
CD.NumberIn + 1:
                              Load 1 CD.WIP=Load 1 CD.WIP+1;
```

F. ".mod" and ".exp" files of Minifab simulation model

```
82$            QUEUE,         Load 1 CD.Queue;
81$            SEIZE,         2,NVA:
                              operator 2,1:
                              Machine C D,1:NEXT(80$);

80$            DELAY:         Normal(15,1.5),,NVA;
127$           ASSIGN:        Load 1 CD.NumberOut=Load 1
CD.NumberOut + 1:
                              Load 1 CD.WIP=Load 1 CD.WIP-
1:NEXT(28$);


;
;
;     Model statements for module:  AdvancedProcess.Release
2 (Release 1)
;
28$            RELEASE:       operator 2,1:NEXT(1$);


;
;
;     Model statements for module:  BasicProcess.Process 2
(Tool Group 2a)
;
1$             ASSIGN:        Tool Group 2a.NumberIn=Tool
Group 2a.NumberIn + 1:
                              Tool Group 2a.WIP=Tool Group
2a.WIP+1;
131$           DELAY:         Processing Time,,VA;
178$           ASSIGN:        Tool Group 2a.NumberOut=Tool
Group 2a.NumberOut + 1:
                              Tool Group 2a.WIP=Tool Group
2a.WIP-1:NEXT(29$);


;
;
;     Model statements for module:  AdvancedProcess.Seize 2
(Seize 1)
;
29$            QUEUE,         Seize 1.Queue;
               SEIZE,         2,Other:
                              operator 2,1:NEXT(182$);
```

```
182$           DELAY:         0.0,,VA:NEXT(31$);


;
;     Model statements for module:  BasicProcess.Process 11
(Unload 1 CD)
;
31$            ASSIGN:        Unload 1 CD.NumberIn=Unload 1
CD.NumberIn + 1:
                              Unload 1 CD.WIP=Unload 1
CD.WIP+1;
184$           DELAY:         Normal(15,1.5*COV),,NVA;
183$           RELEASE:       operator 2,1:
                              Machine C D,1;
231$           ASSIGN:        Unload 1 CD.NumberOut=Unload 1
CD.NumberOut + 1:
                              Unload 1 CD.WIP=Unload 1
CD.WIP-1:NEXT(12$);


;
;
;     Model statements for module:  BasicProcess.Decide 1
(Rework?)
;
12$            BRANCH,        1:
                              With,(2)/100,234$,Yes:
                              Else,235$,Yes;
234$           ASSIGN:        Rework?.NumberOut
True=Rework?.NumberOut True + 1:NEXT(13$);

235$           ASSIGN:        Rework?.NumberOut
False=Rework?.NumberOut False + 1:NEXT(7$);



;
;
;     Model statements for module:  BasicProcess.Process 4
(Rework Process)
;
13$            ASSIGN:        Rework Process.NumberIn=Rework
Process.NumberIn + 1:
                              Rework Process.WIP=Rework
Process.WIP+1;
239$           QUEUE,         Rework Process.Queue;
```

F. “.mod” and “.exp” files of Minifab simulation model

```
238$           SEIZE,         2,VA:
                              Rework operator,1:
                              Rework,1:NEXT(237$);

237$           DELAY:         (0.5)*(Processing Time),,VA;
236$           RELEASE:       Rework operator,1:
                              Rework,1;
284$           ASSIGN:        Rework
Process.NumberOut=Rework Process.NumberOut + 1:
                              Rework Process.WIP=Rework
Process.WIP-1:NEXT(17$);


;
;
;     Model statements for module:  AdvancedTransfer.Route
2 (Route3)
;
7$             ROUTE:         0.000000000000000,SEQ;


;
;
;     Model statements for module:  BasicProcess.Process 13
(Load 2 CD)
;
32$            ASSIGN:        Load 2 CD.NumberIn=Load 2
CD.NumberIn + 1:
                            Load 2 CD.WIP=Load 2 CD.WIP+1;
290$           QUEUE,         Load 2 CD.Queue;
289$           SEIZE,         2,NVA:
                              operator 2,1:
                              Machine C D,1:NEXT(288$);

288$           DELAY:         Normal(15,1.5),,NVA;
335$           ASSIGN:        Load 2 CD.NumberOut=Load 2
CD.NumberOut + 1:
                              Load 2 CD.WIP=Load 2 CD.WIP-
1:NEXT(33$);


;
;
```

F. “.mod” and “.exp” files of Minifab simulation model

```
;     Model statements for module:  AdvancedProcess.Release
3 (Release 2)
;
33$            RELEASE:       operator 2,1:NEXT(16$);



;
;
;     Model statements for module:  BasicProcess.Process 6
(Tool Group 2b)
;
16$            ASSIGN:        Tool Group 2b.NumberIn=Tool
Group 2b.NumberIn + 1:
                              Tool Group 2b.WIP=Tool Group
2b.WIP+1;
339$           DELAY:         Processing Time,,VA;
386$           ASSIGN:        Tool Group 2b.NumberOut=Tool
Group 2b.NumberOut + 1:
                              Tool Group 2b.WIP=Tool Group
2b.WIP-1:NEXT(34$);



;
;
;     Model statements for module:  AdvancedProcess.Seize 3
(Seize 2)
;
34$            QUEUE,         Seize 2.Queue;
               SEIZE,         2,Other:
                              operator 2,1:NEXT(390$);

390$           DELAY:         0.0,,VA:NEXT(36$);


;
;
;     Model statements for module:  BasicProcess.Process 14
(Unload 2 CD)
;
36$            ASSIGN:        Unload 2 CD.NumberIn=Unload 2
CD.NumberIn + 1:
                              Unload 2 CD.WIP=Unload 2
CD.WIP+1;
392$           DELAY:         Normal(15,1.5*COV),,NVA;
391$           RELEASE:       operator 2,1:
                              Machine C D,1;
```

```
439$          ASSIGN:        Unload 2 CD.NumberOut=Unload 2
CD.NumberOut + 1:
                             Unload 2 CD.WIP=Unload 2
CD.WIP-1:NEXT(12$);

;
;
;     Model statements for module:
AdvancedTransfer.Station 2 (Station3)
;

5$            STATION,       Station 3;
444$          DELAY:         0.0,,VA:NEXT(44$);

;
;
;     Model statements for module:  BasicProcess.Decide 6
(Setup)
;
44$           BRANCH,        1:
                             If,Parts Setup == Entity.Type
&& Step == Entity.JobStep,43$,Yes:
                             If,Parts
Setup==Entity.Type,45$,Yes:

If,Step==Entity.JobStep,46$,Yes:
                             Else,47$,Yes;

;
;
;     Model statements for module:  BasicProcess.Assign 7
(Diff Step Diff Type)
;
47$           ASSIGN:        Step=Entity.JobStep:
                             Parts Setup=Entity.Type:
                             SetUpTime=NORM (12,
6):NEXT(37$);

;
;
;     Model statements for module:  BasicProcess.Process 16
(Load E)
;
37$           ASSIGN:        Load E.NumberIn=Load
E.NumberIn + 1:
                             Load E.WIP=Load E.WIP+1;
```

F. ".mod" and ".exp" files of Minifab simulation model

```
450$           QUEUE,         Load E.Queue;
449$           SEIZE,         2,VA:
                              operator 3,1:
                              Machine E,1:NEXT(448$);

448$           DELAY:         SetUpTime+ NORM(10,1),,VA;
495$           ASSIGN:        Load E.NumberOut=Load
E.NumberOut + 1:
                              Load E.WIP=Load E.WIP-
1:NEXT(38$);


;
;
;     Model statements for module:  AdvancedProcess.Release
4 (Release)
;
38$            RELEASE:       operator 3,1:NEXT(2$);



;
;
;     Model statements for module:  BasicProcess.Process 3
(Tool Group 3)
;
2$             ASSIGN:        Tool Group 3.NumberIn=Tool
Group 3.NumberIn + 1:
                              Tool Group 3.WIP=Tool Group
3.WIP+1;
499$           DELAY:         Processing Time,,VA;
546$           ASSIGN:        Tool Group 3.NumberOut=Tool
Group 3.NumberOut + 1:
                              Tool Group 3.WIP=Tool Group
3.WIP-1:NEXT(39$);

;
;     Model statements for module:  AdvancedProcess.Seize 4
(Seize)
;
39$            QUEUE,         Seize.Queue;
               SEIZE,         2,Other:
                              operator 3,1:NEXT(550$);

550$           DELAY:         0.0,,VA:NEXT(41$);
```

F. ".mod" and ".exp" files of Minifab simulation model

```
;
;
;     Model statements for module:  BasicProcess.Process 17
(Unload E)
;
41$           ASSIGN:        Unload E.NumberIn=Unload
E.NumberIn + 1:
                             Unload E.WIP=Unload E.WIP+1;
552$          DELAY:         Normal(10,10*COV),,NVA;
551$          RELEASE:       operator 3,1:
                             Machine E,1;
599$          ASSIGN:        Unload E.NumberOut=Unload
E.NumberOut + 1:
                             Unload E.WIP=Unload E.WIP-
1:NEXT(8$);


;
;
;     Model statements for module:  AdvancedTransfer.Route
3 (Route4)
;
8$            ROUTE:         0.000000000000000,SEQ;


;
;
;     Model statements for module:  BasicProcess.Assign 4
(Same Step Same Type)
;
43$           ASSIGN:        Step=Entity.JobStep:
                             Parts Setup=Entity.Type:
                             SetUpTime=0:NEXT(37$);


;
;
;     Model statements for module:  BasicProcess.Assign 5
(Same Type Diff Steps)
;
45$           ASSIGN:        Step=Entity.JobStep:
                             Parts Setup=Entity.Type:
                             SetUpTime=NORM(10,
5):NEXT(37$);
```

F. “.mod” and “.exp” files of Minifab simulation model

```
;
;
;     Model statements for module:  BasicProcess.Assign 6
(Same Step Diff Type)
;
46$            ASSIGN:        Step=Entity.JobStep:
                              Parts Setup=Entity.Type:
                              SetUpTime=NORM (5,
2.5):NEXT(37$);

;
;
;     Model statements for module:
AdvancedTransfer.Station 3 (Station4)
;

9$             STATION,       Station 4;
604$           DELAY:         0.0,,VA:NEXT(53$);


;
;
;     Model statements for module:  BasicProcess.Decide 7
(Decide 7)
;
53$            BRANCH,        1:
                              If,Entity.Type==Entity Type
X,605$,Yes:
                              Else,606$,Yes;
605$           ASSIGN:        Decide 7.NumberOut True=Decide
7.NumberOut True + 1:NEXT(55$);

606$           ASSIGN:        Decide 7.NumberOut
False=Decide 7.NumberOut False + 1:NEXT(54$);



;
;
;     Model statements for module:  BasicProcess.Decide 8
(Reached Truncation Pt?)
;
55$            BRANCH,        1:
                              If,Truncation<TNOW,607$,Yes:
                              Else,608$,Yes;
```

F. “.mod” and “.exp” files of Minifab simulation model

```
607$          ASSIGN:       Reached Truncation
Pt?.NumberOut True=Reached Truncation Pt?.NumberOut True +
1:NEXT(52$);

608$          ASSIGN:       Reached Truncation
Pt?.NumberOut False=Reached Truncation Pt?.NumberOut False
+ 1:NEXT(56$);


;
;
;     Model statements for module:  BasicProcess.Assign 9
(output)
;
52$           ASSIGN:       CycleTime=TNOW - Arrival
Time:NEXT(61$);


;
;
;     Model statements for module:  BasicProcess.Assign 17
(batch CT Size)
;
61$           ASSIGN:       Batch_Size=Batch_Size+1:

Batch_Cycletime=Batch_Cycletime + CycleTime:NEXT(63$);


;
;
;     Model statements for module:  BasicProcess.Decide 12
(Batch is complete?)
;

63$           BRANCH,       1:
                            If,Batch_Size==3,609$,Yes:
                            Else,610$,Yes;
609$          ASSIGN:       Batch is complete?.NumberOut
True=Batch is complete?.NumberOut True + 1:NEXT(64$);

610$          ASSIGN:       Batch is complete?.NumberOut
False=Batch is complete?.NumberOut False + 1:NEXT(65$);
```

F. “.mod” and “.exp” files of Minifab simulation model

```
;
;     Model statements for module:  BasicProcess.Assign 24
(Average batch CT)
;
64$           ASSIGN:
Batch_Cycletime=Batch_Cycletime/3:NEXT(62$);


;
;
;     Model statements for module:  BasicProcess.Assign 20
(Assign Batch CT)
;
62$           ASSIGN:
BatchedCT=Batch_Cycletime:NEXT(58$);


;
;
;     Model statements for module:  BasicProcess.Assign 15
(Reset lag Batch_CT Batch_Size)
;
58$           ASSIGN:        lag_Tally=lag_Tally+1:
                             Batch_Cycletime=0:
                             Batch_Size=0:NEXT(57$);


;
;
;     Model statements for module:  BasicProcess.Decide 9
(Reached Lag_Value)
;
57$           BRANCH,        1:

If,lag_Tally>Lag_Value,611$,Yes:
                             Else,612$,Yes;
611$          ASSIGN:        Reached Lag_Value.NumberOut
True=Reached Lag_Value.NumberOut True + 1:NEXT(60$);

612$          ASSIGN:        Reached Lag_Value.NumberOut
False=Reached Lag_Value.NumberOut False + 1:NEXT(59$);


;
;
```

F. “.mod” and “.exp” files of Minifab simulation model

```
;     Model statements for module:  BasicProcess.Assign 16
(Reset_Lag_Tally)
;
60$           ASSIGN:        lag_Tally=0:NEXT(42$);



;
;
;     Model statements for module:  BasicProcess.Record 1
(Count Entities)

;
42$           COUNT:         Exits,1:NEXT(51$);



;
;
;     Model statements for module:
AdvancedProcess.ReadWrite 2 (Write Output)
;
51$           WRITE,         Minifab Output:
                             TBA,
                             MTBF,
                             MTTR,
                             COV,
                             BatchedCT:NEXT(3$);

;
;
;     Model statements for module:  BasicProcess.Dispose 1
(Dispose 1)
;
3$            ASSIGN:        Dispose 1.NumberOut=Dispose
1.NumberOut + 1;
613$          DISPOSE:       Yes;

;
;
;     Model statements for module:  BasicProcess.Dispose 5
(Dispose 5)
;
59$           ASSIGN:        Dispose 5.NumberOut=Dispose
5.NumberOut + 1;
614$          DISPOSE:       Yes;
```

F. “.mod” and “.exp” files of Minifab simulation model

```
;
;
;     Model statements for module:  BasicProcess.Dispose 8
(Dispose)
;
65$           ASSIGN:
Dispose.NumberOut=Dispose.NumberOut + 1;
615$          DISPOSE:       Yes;


;
;
;     Model statements for module:  BasicProcess.Dispose 4
(Dispose 4)
;
56$           ASSIGN:        Dispose 4.NumberOut=Dispose
4.NumberOut + 1;
616$          DISPOSE:       Yes;


;
;
;     Model statements for module:  BasicProcess.Dispose 3
(Dispose 3)
;
54$           ASSIGN:        Dispose 3.NumberOut=Dispose
3.NumberOut + 1;
617$          DISPOSE:       Yes;


;
;

;     Model statements for module:
AdvancedTransfer.Station 4 (Station1)
;

11$           STATION,       Station 1;
620$          DELAY:         0.0,,VA:NEXT(20$);


;
;
;     Model statements for module:  BasicProcess.Decide 4
(JobStep?)
```

F. ".mod" and ".exp" files of Minifab simulation model

```
20$           BRANCH,        1:
                             If,Entity.JobStep==1,621$,Yes:
                             Else,622$,Yes;
621$          ASSIGN:        JobStep?.NumberOut
True=JobStep?.NumberOut True + 1:NEXT(15$);

622$          ASSIGN:        JobStep?.NumberOut
False=JobStep?.NumberOut False + 1:NEXT(21$);


;
;
;     Model statements for module:  BasicProcess.Batch 1
(Batch A B)
;
15$           QUEUE,         Batch A B.Queue;
623$          GROUP,         ,Temporary:3,Last,Entity Type
X:NEXT(624$);

624$          ASSIGN:        Batch A B.NumberOut=Batch A
B.NumberOut + 1:NEXT(22$);


;
;
;     Model statements for module:  BasicProcess.Process 7
(Load AB)
;
22$           ASSIGN:        Load AB.NumberIn=Load
AB.NumberIn + 1:
                             Load AB.WIP=Load AB.WIP+1;
628$          QUEUE,         Load AB.Queue;
627$          SEIZE,         2,NVA:
                             operator 1,1:
                             Machine A B,1:NEXT(626$);

626$          DELAY:         Normal(20,2),,NVA;
673$          ASSIGN:        Load AB.NumberOut=Load
AB.NumberOut + 1:
                             Load AB.WIP=Load AB.WIP-
1:NEXT(23$);


;
```

F. “.mod” and “.exp” files of Minifab simulation model

```
;     Model statements for module:  AdvancedProcess.Release
1 (Release Op)
;
23$            RELEASE:       operator 1,1:NEXT(0$);


;
;
;     Model statements for module:  BasicProcess.Process 1
(Tool Group 1 Process)
;
0$             ASSIGN:        Tool Group 1
Process.NumberIn=Tool Group 1 Process.NumberIn + 1:
                              Tool Group 1 Process.WIP=Tool
Group 1 Process.WIP+1;
677$           DELAY:         Processing Time,,VA;
724$           ASSIGN:        Tool Group 1
Process.NumberOut=Tool Group 1 Process.NumberOut + 1:
                              Tool Group 1 Process.WIP=Tool
Group 1 Process.WIP-1:NEXT(24$);


;
;
;     Model statements for module:  AdvancedProcess.Seize 1
(Seize Op)
;
24$            QUEUE,         Seize Op.Queue;
               SEIZE,         2,Other:
                              operator 1,1:NEXT(728$);

728$           DELAY:         0.0,,VA:NEXT(26$);


;
;
;     Model statements for module:  BasicProcess.Process 8
(Unload AB)
;
26$            ASSIGN:        Unload AB.NumberIn=Unload
AB.NumberIn + 1:
                              Unload AB.WIP=Unload AB.WIP+1;
730$           DELAY:         Normal(40,40*COV),,NVA;
729$           RELEASE:       operator 1,1:
```

F. ".mod" and ".exp" files of Minifab simulation model

```
                             Machine A B,1;
777$          ASSIGN:        Unload AB.NumberOut=Unload
AB.NumberOut + 1:
                             Unload AB.WIP=Unload AB.WIP-
1:NEXT(19$);


;
;
;     Model statements for module:  BasicProcess.Separate 1
(Separate A B)
;
19$           SPLIT::NEXT(780$);

780$          ASSIGN:        Separate A B.NumberOut
Orig=Separate A B.NumberOut Orig + 1:NEXT(6$);



;
;
;     Model statements for module:  AdvancedTransfer.Route
1 (Route2)
;
6$            ROUTE:         0.000000000000000,SEQ;



;
;
;     Model statements for module:  BasicProcess.Batch 5
(Batch 2 A B)

;
21$           QUEUE,         Batch 2 A B.Queue;
783$          GROUP,
Entity.Type,Temporary:3,Last,Entity Type X:NEXT(784$);

784$          ASSIGN:        Batch 2 A B.NumberOut=Batch 2
A B.NumberOut + 1:NEXT(22$);



;
;
;     Model statements for module:  BasicProcess.Create 3
(Create 3)
```

F. “.mod” and “.exp” files of Minifab simulation model

```
785$          CREATE,        1,HoursToBaseTime(0.0),Entity
1:HoursToBaseTime(EXPO(1)),1:NEXT(786$);

786$          ASSIGN:        Create 3.NumberOut=Create
3.NumberOut + 1:NEXT(49$);


;
;
;     Model statements for module:
AdvancedProcess.ReadWrite 1 (Read Input data 1)
;
49$           READ,          Minifab Input 1:
                             TBA,
                             MTBF,
                             MTTR,
                             COV:NEXT(50$);


;
;
;     Model statements for module:  BasicProcess.Assign 8
(Assign 8)
;
50$           ASSIGN:        MeanTBFailure=EXPO(MTBF):
                             MeanTTRepair=GAMM(MTTR, 0.25
):
                             Truncation=200000:
                             Batch_Cycletime=0:
                             lag_value=300:
                             Batch_Size=0:
                             lag_Tally=lag_value:NEXT(48$);


;
;
;     Model statements for module:  BasicProcess.Dispose 2
(Dispose 2)
;
48$           ASSIGN:        Dispose 2.NumberOut=Dispose
2.NumberOut + 1;
789$          DISPOSE:       Yes;
```

F. “.mod” and “.exp” files of Minifab simulation model

.exp file :

```
PROJECT,
"MINIFAB","nimmarishikesh@live.com",,,No,Yes,Yes,Yes,No,No,
No,No,No,No;

ATTRIBUTES:    Arrival Time,DATATYPE(Real):
               SetUpTime,DATATYPE(Real):
               Processing Time,DATATYPE(Real):
               BatchedCT,DATATYPE(Real):
               CycleTime,DATATYPE(Real);

FILES:         Minifab Output,"E:\ARENA\Model\Model Input
fies\Minifab Output.txt",Sequential,Free
Format,Dispose,,Rewind:
               Minifab Input 1,"E:\ARENA\Model\Model Input
fies\Minifab Input 1.txt",Sequential,Free
Format,Dispose,,Rewind;

VARIABLES:     Dispose
2.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Dispose
5.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Unload 2
CD.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
               Unload 2
CD.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Batch is complete?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
               Tool Group
2b.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Unload 1
CD.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
               Job Step?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
               MeanTTRepair,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
               Load 1
CD.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
               Reached Truncation Pt?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
```

F. “.mod” and “.exp” files of Minifab simulation model

```
              Reached Truncation Pt?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Load 2
CD.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload
E.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group
2a.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              MTBF,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Decide 7.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Decide 7.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload
AB.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Load 1
CD.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):

Dispose.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload
AB.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload 1
CD.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group
2b.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload 2
CD.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group
3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload
E.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Dispose
4.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload
AB.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Tool Group 1
Process.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):

              Load E.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
```

F. “.mod” and “.exp” files of Minifab simulation model

```
Batch A B.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):

              Batch 2 A
B.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload E.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              MTTR,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Dispose
1.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Parts Setup,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real),0:
              Part
Y.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Load AB.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Tool Group
3.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Rework?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Batch is complete?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group
2a.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Step,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real),0:
              Reached Lag_Value.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              Rework
Process.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              JobStep?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Load
E.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Unload 1
CD.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              COV,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Load 1
CD.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Load 2
CD.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
```

F. “.mod” and “.exp” files of Minifab simulation model

```
Load 2 CD.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):

Truncation,CLEAR(System),CATEGORY("User Specified-User

Specified"),DATATYPE(Real):

              Reached Lag_Value.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group 1
Process.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              lag_value,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Part
X.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Dispose
3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group
2a.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):
              Job Step?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Rework
Process.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Load
E.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Tool Group
2b.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Separate A B.NumberOut
Orig,CLEAR(Statistics),CATEGORY("Exclude"):
              TBA,CLEAR(System),CATEGORY("User Specified-
User Specified"),DATATYPE(Real):
              Rework?.NumberOut
False,CLEAR(Statistics),CATEGORY("Exclude"):
              Batch_Cycletime,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Load
AB.NumberIn,CLEAR(Statistics),CATEGORY("Exclude"):

              Create
3.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Rework
Process.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
```

F. “.mod” and “.exp” files of Minifab simulation model

```
Tool Group 3.WIP,CLEAR(System),CATEGORY("Exclude-
Exclude"),DATATYPE(Real):
              Tool Group 1
Process.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              Load
AB.NumberOut,CLEAR(Statistics),CATEGORY("Exclude"):
              JobStep?.NumberOut
True,CLEAR(Statistics),CATEGORY("Exclude"):
              lag_Tally,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              Batch_Size,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real):
              MeanTBFailure,CLEAR(System),CATEGORY("User
Specified-User Specified"),DATATYPE(Real);

QUEUES:       Seize Op.Queue,FIFO,,AUTOSTATS(Yes,,):
              Batch A B.Queue,FIFO,,AUTOSTATS(Yes,,):
              Seize 1.Queue,FIFO,,AUTOSTATS(Yes,,):
              Load 2 CD.Queue,FIFO,,AUTOSTATS(Yes,,):
              Seize 2.Queue,FIFO,,AUTOSTATS(Yes,,):
              Load AB.Queue,FIFO,,AUTOSTATS(Yes,,):
              Load E.Queue,FIFO,,AUTOSTATS(Yes,,):
              Load 1 CD.Queue,FIFO,,AUTOSTATS(Yes,,):
              Batch 2 A B.Queue,FIFO,,AUTOSTATS(Yes,,):
              Rework Process.Queue,FIFO,,AUTOSTATS(Yes,,):
              Seize.Queue,FIFO,,AUTOSTATS(Yes,,);

PICTURES:     Picture.Airplane:
              Picture.Green Ball:
              Picture.Blue Page:
              Picture.Telephone:
              Picture.Blue Ball:
              Picture.Yellow Page:
              Picture.EMail:
              Picture.Yellow Ball:
              Picture.Bike:
              Picture.Report:
              Picture.Van:
              Picture.Widgets:
              Picture.Envelope:
              Picture.Fax:
              Picture.Truck:
              Picture.Person:
              Picture.Letter:
```

F. “.mod” and “.exp” files of Minifab simulation model

```
              Picture.Box:
              Picture.Woman:
              Picture.Package:
              Picture.Man:
              Picture.Diskette:
              Picture.Boat:
              Picture.Red Page:
              Picture.Ball:
              Picture.Green Page:
              Picture.Red Ball;

FAILURES:     Prev
Maint,Time(10080.000000000000000,60.000000000000000,):
              Int
Check,Time(43200.000000000000000,360.000000000000000,):

              Eme
Maint,Time(DaysToBaseTime(MeanTBFailure),MeanTTRepair,);

RESOURCES:    Machine C
D,Capacity(2),,,COST(0.0,0.0,0.0),CATEGORY(Resources),FAILU
RE(Prev Maint,Wait),FAILURE(Int Check,Wait),
              AUTOSTATS(Yes,,):

Rework,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),
FAILURE(Int Check,Wait),FAILURE(Prev Maint,Wait),
              AUTOSTATS(Yes,,):
              Rework
operator,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources
),,AUTOSTATS(Yes,,):
              operator
1,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTO
STATS(Yes,,):
              operator
2,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTO
STATS(Yes,,):
              operator
3,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),,AUTO
STATS(Yes,,):
              Machine A
B,Capacity(2),,,COST(0.0,0.0,0.0),CATEGORY(Resources),FAILU
RE(Int Check,Wait),FAILURE(Prev Maint,Wait),
              AUTOSTATS(Yes,,):
```

F. “.mod” and “.exp” files of Minifab simulation model

```
              Machine
E,Capacity(1),,,COST(0.0,0.0,0.0),CATEGORY(Resources),FAILU
RE(Eme Maint,Preempt),FAILURE(Int Check,Wait),
              FAILURE(Prev Maint,Wait),AUTOSTATS(Yes,,);

STATIONS:     Station 1,,,Station 1,AUTOSTATS(Yes,,):
              Station 2,,,Station 2,AUTOSTATS(Yes,,):
              Station 3,,,Station 3,AUTOSTATS(Yes,,):
              Station 4,,,Station 4,AUTOSTATS(Yes,,);

SEQUENCES:    Sequence,Station 1,STEPNAME=Step
1,,,Processing Time=NORM(225,11.25)&Station 2,STEPNAME=Step
2,,,Processing Time=
              NORM(30,1.5)&Station 3,STEPNAME=Step
3,,,Processing Time=NORM(55,2.75)&Station 2,STEPNAME=Step
4,,,
              Processing Time=NORM(50,2.5)&Station
1,STEPNAME=Step 5,,,Processing Time=NORM(255,12.75)&Station
3,STEPNAME=
              Step 6,,,Processing Time=NORM(10,0.5)&Station
4,STEPNAME=Step 7;

COUNTERS:     Exits,,,,DATABASE(,"Count","User
Specified","Exits");

REPLICATE,
1,,,Yes,Yes,,NC(Exits)>4999,,24,Minutes,No,No,,,Yes,No;

ENTITIES:     Entity Type
X,Picture.Report,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,):
              Entity Type
Y,Picture.Report,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,):
              Entity
1,Picture.Report,0.0,0.0,0.0,0.0,0.0,0.0,AUTOSTATS(Yes,,);

ACTIVITYAREAS: Station 1,0,,AUTOSTATS(Yes,,):
              Station 2,0,,AUTOSTATS(Yes,,):
              Station 3,0,,AUTOSTATS(Yes,,):
              Station 4,0,,AUTOSTATS(Yes,,);
```

APPENDIX G

“R” PROGRAMMING CODE

G. “R” Programming Code:

```
#store current directory
#
initial.dir<-getwd()
#
#loading relevant libraries
#note that to access the packages in the library, they need
to already be installed
#if they are not installed, prior to running the script, use
the following commands:
#install.packages("quantreg")
#install.packages("MatrixModels")
library(quantreg)
library(MatrixModels)
#setting working directory (set to location in which data
file is stored); update this for different computers
setwd("D:\\ARENA\\R")
#setting up an output file (this is where results get written
to); update this to change the output file name.
#if you do not update it, subsequent executions of the same
script will write over previous output files.
file.create("quantreg_out_exp1.csv")
sink("quantreg_out_exp1.csv", append=TRUE)
```

G. “R” Programming Code

```
#reading input data
#
#here factor A is TBA and factor B is the one relevant to the
experiment on hand (changes for different expeirments)
#factors can be adjusted -- "a" and "b" are just text
placholders to keep things generic
#note that the input data file name should be updated and
should be in the same folder with the script file.

#change file name here for each fold

data=read.table("Exp1_Rep1_50obs.txt", header=FALSE)
ct=data[,5]
a_nonstandard=data[,1]
b_nonstandard=data[,3]
b_nonstandard=b_nonstandard*4

#
#standardizing variables
#
mean_a = mean(a_nonstandard)
mean_b = mean(b_nonstandard)
```

G. “R” Programming Code

```
sd_a = sd(a_nonstandard)

sd_b = sd(b_nonstandard)

a = (a_nonstandard-mean_a)/sd_a

b = (b_nonstandard-mean_b)/sd_b

#creating new variables for quantreg models based on
standardized variables

a2=a*a

a3=a*a*a

a4=a*a*a*a

a5=a*a*a*a*a

b2=b*b

b3=b*b*b

b4=b*b*b*b

b5=b*b*b*b*b

ab=a*b

a2b=a*a*b

a3b=a*a*a*b

a4b=a*a*a*a*b

a5b=a*a*a*a*a*b

ab2=a*b*b

ab3=a*b*b*b

ab4=a*b*b*b*b
```

G. “R” Programming Code

```
ab5=a*b*b*b*b*b

a2b2=a*a*b*b

a2b3=a*a*b*b*b

a2b4=a*a*b*b*b*b

a3b2=a*a*a*b*b

a4b2=a*a*a*a*b*b

a3b3=a*a*a*b*b*b

a6=a*a*a*a*a*a

b6=b*b*b*b*b*b

cat("Mean of TBA: ", mean_a)

cat("\nMean of COV: ", mean_b)

cat("\nStdev of TBA: ", sd_a)

cat("\nStdev of COV: ", sd_b)


#

#quantile regression + file write-out

#'cat' command writes out to the output file; '\n' puts a new
line

#print also writes out to the output file, but supports
different output formats

qr_lambda=100

#adjust range of 'i' to get fits with additional lambda values
```

G. “R” Programming Code

```
#for each iteration through the loop, i is reduced by 1/10
for(i in 1:4)
{
	#
	#second order model
	#
	cat("\nOrder=2,","\nlambda=", qr_lambda,"\n")
	qr_fit_second_order=rq(ct~a+b+ab+a2+b2,
tau=c(0.5,0.8,0.9,0.95),   method   =   "lasso",   lambda   =
qr_lambda)
	print(coef(qr_fit_second_order))
	#write.table(coef(qr_fit_second_order),"quantreg_out-
1000",sep=",",row.names=FALSE)
	#
	#third order model
	#
	cat("Order=3,","\nlambda=", qr_lambda,"\n")
	qr_fit_third_order=rq(ct~a+b+a2+b2+ab+ab2+a2b+a3+b3,
tau=c(0.5,0.8,0.9,0.95),   method   =   "lasso",   lambda   =
qr_lambda)
	print(coef(qr_fit_third_order))
```

G. “R” Programming Code

```
    #write.table(coef(qr_fit_third_order),"quantreg_out-
1000",sep=",",row.names=FALSE)
    #
    #fourth order model
    #
    cat("Order=4,","\nlambda=", qr_lambda,"\n")
    qr_fit_fourth_order=rq(ct~a+b+a2+b2+a3+b3+ab+ab2+a2b+a
3b+ab3+a2b2+a4+b4,  tau=c(0.5,  0.8,  0.9,  0.95),  method  =
"lasso", lambda = qr_lambda)
    print(coef(qr_fit_fourth_order))
    #write.table(coef(qr_fit_fourth_order),"quantreg_out-
1000",sep=",",row.names=FALSE)
    qr_lambda=qr_lambda/10
}
#close output file
sink()
#change back to original directory
setwd(initial.dir)
```

APPENDIX H

“MATLAB” PROGRAMMING CODE

H. MATLAB Programming Code

```
clear all; clc;
exp1 = [PP1; PP2;   ...];

exp2 = [PP1; PP2;   ...];

 filename = 'quantreg_out.xlsx';
Sheet = 1;
xlRange = 'B1:E124';
VarName = xlsread(filename);

syms c  a   b   a2  b2  a3  b3  a4  b4  ab  ab2 a2b a3b
ab3...
    a2b2 real

tt2 = [c    a   b   ab  a2  b2  c   a   b   a2  b2  ab  ab2
a2b a3  b3...
    c   a   b   a2  b2  a3  b3  ab  ab2 a2b a3b ab3 a2b2
a4  b4...
    c   a   b   ab  a2  b2  c   a   b   a2  b2  ab  ab2 a2b
a3  b3...
    c   a   b   a2  b2  a3  b3  ab  ab2 a2b a3b ab3 a2b2
a4  b4...
    c   a   b   ab  a2  b2  c   a   b   a2  b2  ab  ab2 a2b
a3  b3...
    c   a   b   a2  b2  a3  b3  ab  ab2 a2b a3b ab3 a2b2
a4  b4...
    c   a   b   ab  a2  b2  c   a   b   a2  b2  ab  ab2 a2b
a3  b3...
    c   a   b   a2  b2  a3  b3  ab  ab2 a2b a3b ab3 a2b2
a4  b4];
for i = 1:length(exp1)
    x(i) = exp1(i);       y(i) = exp2(i);
    x2(i) = exp1(i)^2;    y2(i) = exp2(i)^2;
    x3(i) = exp1(i)^3;    y3(i) = exp2(i)^3;
    x4(i) = exp1(i)^4;    y4(i) = exp2(i)^4;

    xy(i) = (exp1(i)) * (exp2(i));
    x2y(i) = (exp1(i)^2) * (exp2(i));    xy2(i) = (exp1(i))
* (exp2(i)^2);
    x3y(i) = (exp1(i)^3) * (exp2(i));    xy3(i) = (exp1(i))
* (exp2(i)^3);

    x2y2(i) = (exp1(i)^2) * (exp2(i)^2);
end
```

H. MATLAB Programming Code

```
for j = 1:length(exp1)
    for k = 1:length(tt2)
            if(tt2(k) == a)
               temp23(k,j,:) = x(j)*VarName(k,:);
            elseif(tt2(k) == b)
                temp23(k,j,:) = y(j)*VarName(k,:);
            elseif(tt2(k) == c)
                temp23(k,j,:) = VarName(k,:);
            elseif(tt2(k) == a2)
                temp23(k,j,:) = x2(j)*VarName(k,:);
            elseif(tt2(k) == b2)
                temp23(k,j,:) = y2(j)*VarName(k,:);
            elseif(tt2(k) == a3)
                temp23(k,j,:) = x3(j)*VarName(k,:);
            elseif(tt2(k) == b3)
                temp23(k,j,:) = y3(j)*VarName(k,:);
            elseif(tt2(k) == a4)
                temp23(k,j,:) = x4(j)*VarName(k,:);
            elseif(tt2(k) == b4)
                temp23(k,j,:) = y4(j)*VarName(k,:);
            elseif(tt2(k) == ab)
                temp23(k,j,:) = xy(j)*VarName(k,:);
            elseif(tt2(k) == ab2)
                temp23(k,j,:) = xy2(j)*VarName(k,:);
            elseif(tt2(k) == a2b)
                temp23(k,j,:) = x2y(j)*VarName(k,:);
            elseif(tt2(k) == ab3)
                temp23(k,j,:) = xy3(j)*VarName(k,:);
            elseif(tt2(k) == a3b)
                temp23(k,j,:) = x3y(j)*VarName(k,:);
            elseif(tt2(k) == a2b2)
                temp23(k,j,:) = x2y2(j)*VarName(k,:);
            end
    end

end

Quantile5  = [temp23(:,:,1)];
Quantile8  = [temp23(:,:,2)];
Quantile9  = [temp23(:,:,3)];
Quantile95 = [temp23(:,:,4)];
```