2009-01-01

Geometric Tolerancing of Cylindricity Utilizing Support Vector Regression

Keun Joo Lee
University of Miami, lee.keun@gmail.com

UNIVERSITY OF MIAMI

GEOMETRIC TOLERANCING OF CYLINDRICITY UTILIZING SUPPORT VECTOR REGRESSION

By

Keun J. Lee

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science

Coral Gables, Florida

December 2009

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

GEOMETRIC TOLERANCING OF CYLINDRICITY UTILIZING SUPPORT VECTOR REGRESSION

Keun J. Lee

Approved:

_____	_____
Shihab Asfour, Ph.D. Professor and Associate Dean of College of Engineering	Terri A. Scandura, Ph.D. Dean of the Graduate School
_____	_____
Murat Erkoc, Ph.D. Assistant Professor of Industrial Engineering	Khaled A. Zakaria, Ph.D. Assistant Scientist of Industrial Engineering

_____
Sohyung Cho, Ph.D.
Assistant Professor of
Industrial and Manufacturing Engineering
Southern Illinois University Edwardsville

LEE, KEUN J. (M.S., Industrial Engineering)
Geometric Tolerancing of Cylindricity Utilizing Support Vector Regression (December 2009)

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Shihab Asfour.
No. of pages in text. (163)

In the age where quick turn around time and high speed manufacturing methods are becoming more important, quality assurance is a consistent bottleneck in production. With the development of cheap and fast computer hardware, it has become viable to use machine vision for the collection of data points from a machined part. The generation of these large sample points have necessitated a need for a comprehensive algorithm that will be able to provide accurate results while being computationally efficient. Current established methods are least-squares (LSQ) and non-linear programming (NLP). The LSQ method is often deemed too inaccurate and is prone to providing bad results, while the NLP method is computationally taxing. A novel method of using support vector regression (SVR) to solve the NP-hard problem of cylindricity of machined parts is proposed. This method was evaluated against LSQ and NLP in both accuracy and CPU processing time. An open-source, user-modifiable programming package was developed to test the model. Analysis of test results show the novel SVR algorithm to be a viable alternative in exploring different methods of cylindricity in real-world manufacturing.

Dedication

This thesis is dedicated to my parents and my sister.

Acknowledgement

I would like to thank Doctor Shihab Asfour for taking a chance on a random out-of-work engineer that happened to walk into his office uninvited. I thank him for his full support even in the worst of my moments, and always having the time to listen to problems despite the frivolity of many of them. I also thank him for the opportunity to work as a research assistant in the Industrial Assessment Center at the University of Miami. Everything I did there expanded my knowledge base to be the better person I am today.

I also thank everyone in the Industrial Assessment Center for being a part of my life over the past 3 years. Tony, you will always be a friend and a mentor to me, and I will always remember your words of wisdom, especially since I have begun a career path that you had taken before. Randy, I'm sorry you didn't get the job, but I'm sure if you tried again you'd be a shoo-in (although I'm not quite certain that you want the job, ask me about it in a year). Giro, I hope you can take backpack farther than I could ever dream of – unlike me, you have the imagination necessary to take it into uncharted territory. Dr. Zakaria, I will miss the times when we would compare phones and talk excitedly about the newest technology that would be coming out.

I would also like to extend a big thank you to Doctor Sohyung Cho for introducing me to the field of Support Vector Regression as a topic for my thesis, along with other topics he suggested in an effort to help me graduate. I thank him for always being patient with me as my deadlines to finish came, passed, and extended – I hope the results of this research will be enough to repay him for this debt. It was a stroke of divine intervention that I met him at the University of Miami, and his support never wavered when he left for Southern Illinois University.

I also thank my friends for the days when I wasn't buried in work. Aaron (both of you), Alexis, Deana, Delena, Eliu, Jade, Justin, Lloyd, Mauricio, Michael, Omine, Raul, Sergio ... (truncated for brevity), all of them helped keep a balance in my life. Special thanks goes to Jon and Stephanie, who fed me a steady diet of hamburgers and hot dogs during the late nights. Hookahs and grilled food at Bougainvillea's – Wednesdays, Fridays, and Saturdays! I also thank Scott for teaching me about many things, but most importantly, Minitab. Our politics are unreconcilable, but our common goal (which I reached first, but he has a harder job) and dislike for the current state of society will always give us something to talk about late into the night.

I also thank the workers of Starbucks across the street from the University, who I saw more than family and friends for the past year – Patrick, Phil, George, Dana, Bryan – they'll never read this but they've been a critical part of the process of completing my thesis and should be mentioned. More thanks goes to the colorful characters I've met at this Starbucks for providing me with much-needed distraction – John, Eric, Paul, David, Steve, Mike, and countless others.

This entire journey would not have been possible if it had not been for my longtime friend, Chase, who on a fateful trip to Orlando suggested that I go back to school. If it wasn't for his intervention at my time of need, I would still be at my old job, wondering what it was all for. I have not seen him much recently, but this is partly his fault for making me a student again.

Last but not least, I thank and love my parents for predicting that I would go back to school and saving accordingly. Despite my sister going to an expensive private school, somehow there was also money to send me to an expensive private school and pay for that crucial first semester. I thank them for their patience and understanding during the

past few years. Now that I am finally finished, I hope I will have many chances to show them my appreciation and pay them back in any way I can.

As for the person reading this thesis right now, thank you – either you decided it would help in your own research, or you were instrumental in the completion of my thesis and was provided a copy by me. For the people who are using this as a reference for their own work (which is an exciting thought, drop me a line at lee.keun@gmail.com if you want any advice) – I hope that it is helpful, reasonably well-written, and points you in the right direction. For the people who are reading out of curiosity over what I have been doing for the past few years – I don't ask you to understand what I did, but at least try to get to the end.

Table of Contents

List of Figures

List of Tables

CHAPTER 1 – INTRODUCTION

Recent advancements in the world economy, especially in the developing world, have caused an increased demand for manufactured goods. Technology has responded to this by advancing to fulfill this demand with techniques such as ultra high-speed machining. Lead times have decreased significantly due to the globalization and decentralization of the supply chain. Nonetheless, the majority of manufacturing engineers still point out that the bottleneck of manufacturing occurs at inspection. It has become imperative for any production entity to increase the speed and accuracy of quality inspection of a work piece. The challenge here is the balancing act during the trade-off between speed, cost, and accuracy of inspection [1]. When focus is shifted toward shorter time spend for inspection, time and cost typically goes down per unit, but has an effect of increasing the number of defective units. On the other hand, if higher levels of quality control are implemented, it invariably leads to more time spent per unit, raising the total cost of manufacture and also has the effect of decreasing the level of supply to meet demand. It has been a constant struggle in industry to have six sigma levels of quality while providing products in a timely, affordable manner.

1.1 TRADITIONAL METHODS

Traditional forms of quality control relied on blueprints that depicted the ideal forms of the parts being produced. The blueprints included dimensional tolerancing information to make sure that parts were in specification and interchangeable during assembly [2]. The quality inspector measures the dimensions listed in the blueprints with manual equipment, or a machine specifically configured for the task. Typically only one instance of the dimension is measured rather than multiple measurements along the edge.

That measurement is compared against the specified dimension in the blueprint, and the manufactured part is considered to be within specifications if it does not deviate beyond the tolerances.

While this method of quality inspection using dimensional tolerancing is easy and fast, it fails to address other issues that are necessary to determine the suitability of a manufactured item. For example, even if a circular cross section of a part is measured and is determined to have the required diameter to be within dimensional tolerances, the actual circularity might be insufficient due to the part being elliptical or it has unusually flat or bumpy spots on a small portion of its surface. These discrepancies occur due to tool wear on the production equipment, vibration, human error, and a variety of other factors. As the standard method of dimensional tolerancing only takes into account one or two measurements, it is not suitable for determining the overall fitness of a manufactured product.

1.2 GEOMETRIC TOLERANCING

As an alternative, the method of geometric tolerancing is promising to solve the problem of defining the overall fitness of a part. The approach of geometric tolerancing is to consider a part to be within specifications if the boundaries of multiple measurement points do not exceed defined “tolerance zones.” These tolerance zones are determined by the target ideal shape of the part being produced, combined with the amount of error that is allowed according to specifications [2]. With the two factors combined a minimum enclosing zone is created. If the data points fit inside this minimum enclosing zone, the sampled part is considered to be geometrically fit. This provides a fairly accurate assessment on the suitability of a part in both dimensions and overall shape.

Despite its advantages over dimensional tolerancing, geometric tolerancing is rarely used in a high-speed production environment. There are several factors that have prevented this method from being used outside of specialty CNC production facilities. Most significantly, it requires a larger amount of sampling data than dimensional tolerancing in order to provide an accurate assessment of the shape of the part. The typical method of obtaining measurement samples have been with contact-type methods such as calipers or some other manual measurement device or by coordinate measuring machines (CMMs). These methods are very time consuming when used to generate the desired amount of data for determining geometric fitness, as it requires a significant amount of sample points to determine the shape of a product rather than a single dimension.

The advent of powerful yet cheap computer hardware proposes to change this situation with machine vision technology [3]. A machine vision system is defined as the application of computer imaging in industrial and manufacturing situations. While computer imaging mainly focuses on image processing, machine vision also includes input/output devices and computer networks to control other manufacturing equipment such as robotic arms. As it offers non-contact sampling of parts, machine vision offers consistency, speed, and accuracy, making it suitable for a mass production environment.

Previously, procuring the equipment that was required to implement machine vision was infeasible. High speed digital cameras were only available in extremely special cases and computer hardware was expensive while not offering much in terms of processing power. In the past decade the price of technology has dropped drastically to the point where most of the components in a machine vision system can be purchased

over the counter for a reasonable price. Considering the cost-effective choices available in the consumer market, a machine vision system could even turn out to be cheaper than a traditional CMM system.

With machine vision becoming a viable solution to the problem of data acquisition, the bottleneck now shifts over to the analysis of the data that is being generated. In order to determine the geometric tolerance of a sampled part, the data must be processed by an algorithm that will convert it into a "best estimation" shape. Otherwise, it is impossible to compare it against the defined tolerance zone to determine overall fitness. For example, the data points of a cylinder are meaningless unless the central axis of the data set is known. Without a defined axis that the data points can reference against, the inner and outer bounds of the tolerance zone cannot be determined. A significant portion of the paper will be devoted on analytical algorithms that have been developed to do this, as well as propose a novel implementation of the support vector regression method for cylindricity analysis.

1.3 RESEARCH OBJECTIVES

Currently there is a lack of tools available to a researcher who is looking to investigate the field of geometric tolerancing. This has caused others who wish to explore the subject to create their own in-house applications using various means for further study on the subject. A framework that can be easily expanded upon needs to be developed to reduce the time it takes from conception of an algorithm to implementation into a form that allows experimentation. This research will explore the creation of a comprehensive and open system that can be used in this paper as well as subsequent research. Therefore, this paper will study the following:

- Development of an open-source, expandable, and easily modifiable software package that will allow repeated experiments with simulated cylinder sample data
- Integration of LSQ, NLP, and a novel SVR implementation to calculate the cylindricity of the sample data
- Comparison between LSQ, NLP, and SVR with regards to accuracy and computational performance

The software package could be translated into various different computer systems in a variety of settings. It could also be modified extensively according to the specifications of the user. With an open source framework that is already developed, researchers will not have to build an entirely new proprietary program to implement SVR algorithms to measure cylindricity. This will lower the effort required to bring new ideas into a field of study that has not been effectively studied in depth as of yet.

CHAPTER 2 – LITERATURE REVIEW AND METHODS

2.1 CYLINDRICITY

Current geometric tolerancing methods have focused on the measure of roundness, which is one of the most basic geometric elements of mechanical parts [4]. Taking this further, in real-world settings, parts with cylindrical shapes are crucial in applications that require tight tolerancing. The most obvious example in this case is a crank shaft in a motor where even a slight imbalance has the capability to cause catastrophic failure.

The cylindricity of the part in this instance affects directly on the efficiency, safety, and life of an assembly. As the measure of roundness, or cylindricity, is considered an NP-hard problem, there is a difficulty in solving any optimization problem designed to predict a suitable ideal cylinder out of measurement samples.

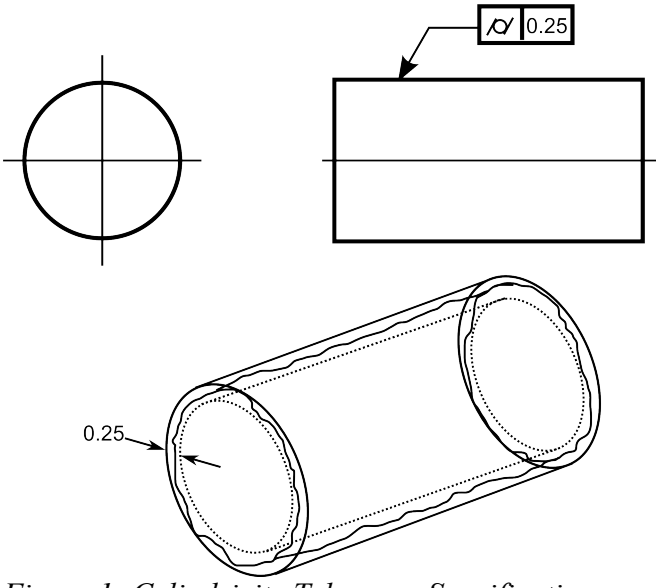


Figure 1: Cylindricity Tolerance Specification

In determining cylindricity, there are four tolerance definitions given in the Geometric Dimensioning & Tolerancing Y14.5 Standard [5]:

Cylindricity (**Figure 1**) – The condition of a surface of revolution in which all points of the surface as equidistant from a common axis. All points must be within a tolerance zone bounded by two coaxial cylinders.

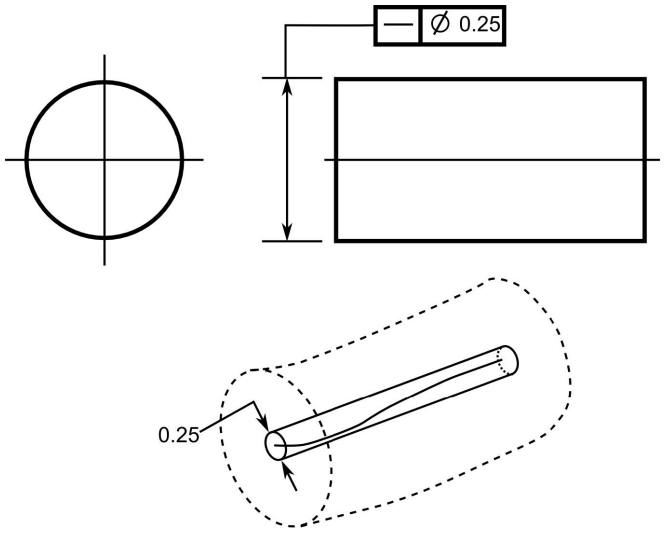


Figure 2: Straightness of a Median Line

Straightness of a median line (**Figure 2**) – Condition where an axis is a straight line. The derived median line must lie within some cylindrical zone whose diameter is the specified tolerance. This requires a second specification for straightness [6].

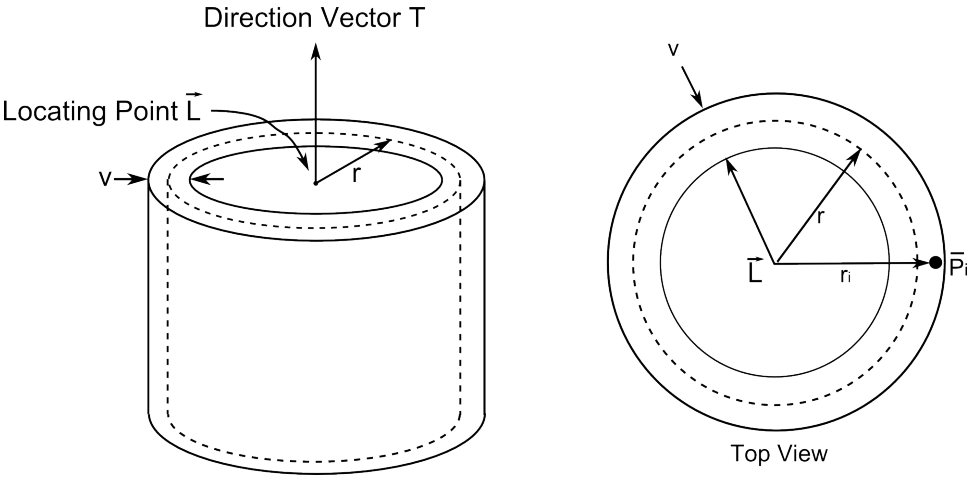


Figure 3: Two Coaxial Cylinders

Zone point-set definition: two coaxial cylinders (**Figure 3**) – *T* is the zone direction vector, *L* is the zone locating position vector, *t* is the zone size (radial distance

between the two cylinders), and r is the radial distance from the axis to the reference cylinder of the tolerance zone. The zone is defined by the set of points $\{P\}$ where

$$|mag[T\times(P_i-L)-r]|\leq\frac{t}{2}\quad\forall P_i\in\{P\}$$

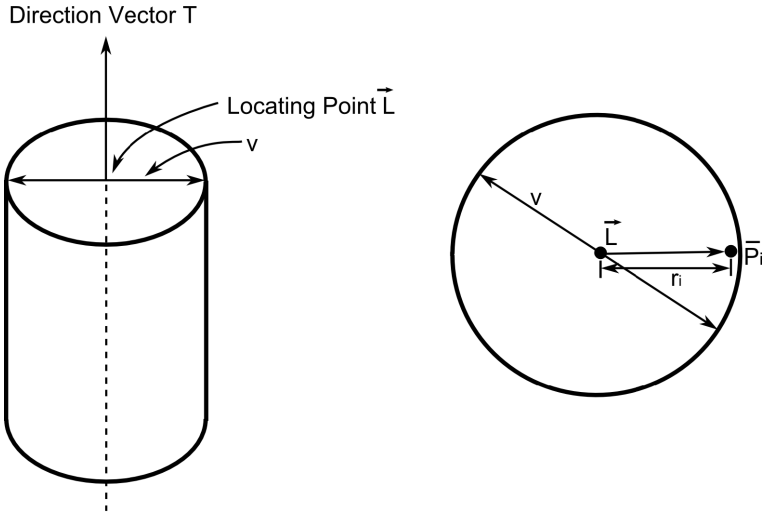


Figure 4: Cylinder Tolerance Zone

Zone point-set definition: cylindrical (**Figure 4**) – T is the zone direction vector, L is the zone locating position vector, and t is the zone size (diameter of the cylinder). The zone is defined by the set of points $\{P\}$ where

$$|mag[T\times(P_i-L)]|\leq\frac{t}{2}\quad\forall P_i\in\{P\}$$

These four guidelines will provide the general rules that will be used to determine cylindricity values. The first definition for overall cylindricity will be of particular focus as it will be assumed that a cylinder that conforms to this specification will generally be considered to be geometrically fit in most circumstances.

2.2 PREVIOUS METHODS

Considering ANSI and ISO standards, the non-linear program of obtaining the minimum zone solution is the most compliant way to measure geometric fitness [5, 7]. A common minimum enclosing zone method for a cylinder can be described with the following optimization problem:

$$\text{minimize} \quad r_{max} - r_{min} = v$$

$$\begin{aligned}
\text{subject to:} \quad & (mag[T \times (P_1 - L)]) \geq r_{min} \\
& \vdots \\
& (mag[T \times (P_n - L)]) \geq r_{min} \\
& (mag[T \times (P_1 - L)]) \leq r_{max} \\
& \vdots \\
& (mag[T \times (P_n - L)]) \leq r_{max} \\
& T_x^2 + T_y^2 + T_z^2 = 1 \\
& T_x L_x + T_y L_y + T_z L_z = 0 \\
& r_1, r_2 \geq 0
\end{aligned}$$

Where v is the tolerance zone margin, $T = (T_x, T_y, T_z)^T$ is the zone orientation direction vector, $L = (L_x, L_y, L_z)^T$ is the zone locating position vector, r_{min} is the distance from the axis to the inner cylinder, r_{max} is the radial distance from the axis to the outer cylinder, and n is the number of points in $\{P\}$. Despite being the preferred standard, this method has been difficult to implement in actual practice. As a result there is a plethora of alternate methods that aim to simplify the problem. These different methods will be discussed below.

One proposed method is to consider the cylinder as a collection of cross-sections normal to the ideal axis (**Figure 5**) rather than a continuous sequence of three dimensional data points. This has the effect of changing the three-dimensional problem into a two-dimensional one, greatly reducing the complexity involved. Using the set of center points of the cross-sections a unifying axis is determined. This allows for the

determination of the minimum zone circle (MZC) of the sampled data by using the inner circles of each cross-section to determine the inner cylinder, and the outer circles of each cross-section to determine the outer cylinder. (**Figure 6**) [8].

In this method, 2-d convex hulls and voronoi diagrams were used to determine the circular properties of each cross-section. Then the set of center points were analyzed using the least-squares straightness method to create the central unified axis. This method was developed specifically to provide a geometric tolerancing algorithm for CMMs.

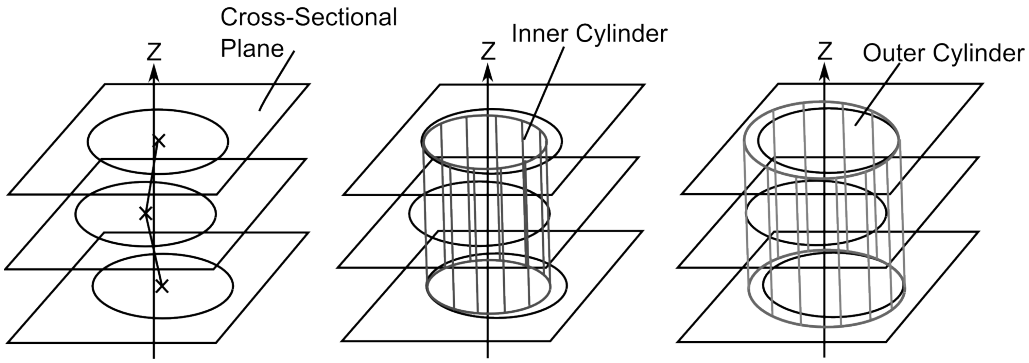


Figure 5: The Formulation of a Unified Axis Using Cross-Sectional Data

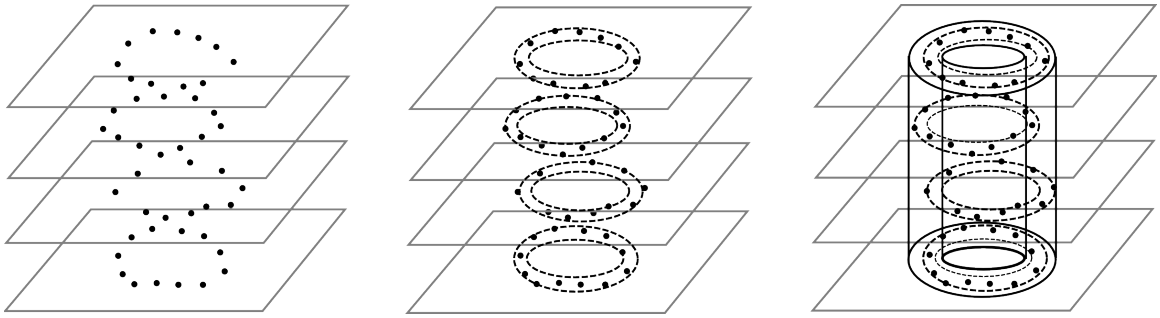


Figure 6: Generation of Tolerance Zones Using the Cross-Section Method

Using the cross-sectional method as described above, the typical cylindricity problem becomes a problem of solving multiple iterations of circularity for obtaining the minimum zone circle of a chosen cross-section. Once methods for circularity are considered, the number of algorithms that can be used for the problem of cylindricity becomes numerous.

A non-linear optimization algorithm is considered to be one of the most accurate methods to define the circularity of a set of data points. This involves the minimization of the distance between the inner and outer circle that envelopes all of the points. The optimization process assumes a 2-2 model as shown in **Figure 7** (2 points on the inner and outer circle) [9] as it offers the greatest accuracy. The 2-2 model can be expressed as a non-linear programming optimization equation:

minimize $r_{max} - r_{min}$

subject to: $r_{min} \leq \sqrt{((x_i - \bar{x})^2 + (y_i - \bar{y})^2)} \leq r_{max}$ for $i = 1, \ldots, n$

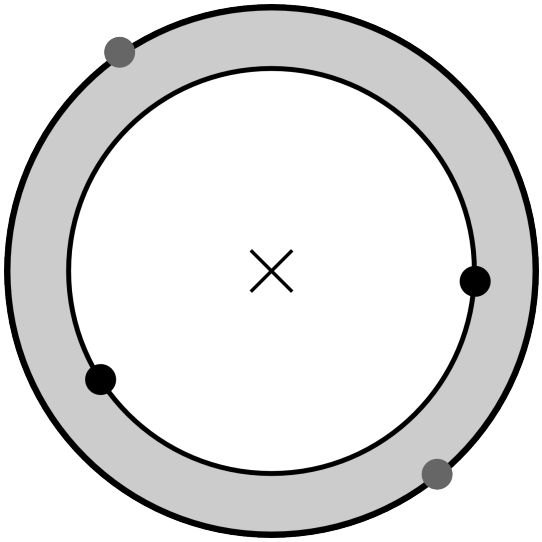


Figure 7: 2-2 NLP Circularity Model

Where r_{max} is the radius of the outer circle, r_{min} is the radius of the inner circle, n is the number of data points, and x_i and y_i are the x-coordinate and y-coordinate value of the data points, respectively. $\bar{x}$ and $\bar{y}$ are the center points of the circle. Although quite accurate and conforming to the standards of geometric tolerancing, the NLP method is slow due to the complexity of circular features making it an NP-hard problem. The NLP algorithm also has a problem with sensitivity where certain conditions such as slight manufacturing errors or asperities cause the algorithm to provide incorrect

results. Because of these problems with the NLP method, other algorithms have been developed to reduce the sensitivity to asperities and decrease the amount of computation time.

Current standard circularity verification algorithms used on CMMs are based on the least-square (LSQ) method. The LSQ circularity problem is mathematically well defined and widely accepted as being a good approximation method [10]. Another advantage of the LSQ method is that it is extremely fast in providing results due to its straightforward nature. It is also less sensitive to asperities. The LSQ method is simple to implement: an equation is derived to best fit the data by minimizing the sum of the squares of deviation from the equation.

There are different types of LSQ implementations within industry, with each CMM manufacturer using their own proprietary algorithm. In general, the derivation of least-squares circularity can be defined by starting with the following equation of the sum of the normal least squares deviation:

$$E_n = \sum \sqrt{((x_i-\bar{x})^2+(y_i-\bar{y})^2)}-R^2$$

$$= \sum (R_i-R)^2$$

Where x_i and y_i represent the data points on the circle being analyzed, $\bar{x}$ and $\bar{y}$ are the center points to be determined, and R and R_i are the radius of the approximated circle and the distance from the point of origin to the data being measured, respectively. These variables can be seen in a graphical form in **Figure 8**. This equation is partially differentiated with respect to the circular center coordinates and R.

$$N\bar{x} = \sum x_i - R\sum \frac{x_i - \bar{x}}{R_i}$$

$$N\bar{y} = \sum y_i - R\sum \frac{y_i - \bar{y}}{R_i}$$

$$NR = \sum R_i$$

To solve these series of equations, tedious mathematical calculations and trial and error procedures are required. Generally, using the various LSQ tools available that will automate the calculation process requires little computation time.

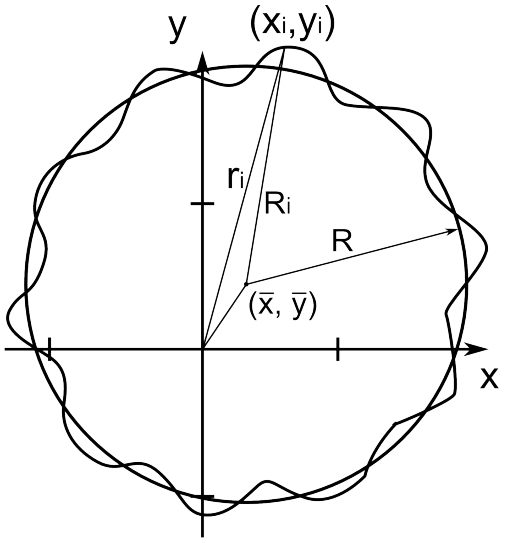


Figure 8: Least-Squares Circularity Fitting

However, there are disadvantages that are inherent to the LSQ method. It is well-known that the LSQ method is inferior with regards to results compared to the pure NLP algorithm, and gives circularity values that are larger than in reality. Its tendency to over-fit the data can get good parts rejected. Also, it has been found that not all LSQ solutions will necessarily give a result that is acceptable [11].

Other methods have been used to transform non-linear geometries to linear form, such as lines and planes, using transform matrices [12] and limacon approximation [13].

To transform non-linear geometries into a line or plane, a method called the control line rotation scheme (CLRS) is used, and then a straightness evaluation is applied. The CLRS method expresses the circle in polar coordinates:

$$u = \theta\,;\ 0 \le \theta \le 2\pi$$

$$v = r$$

Where u and v are variables of a new coordinate frame, r is the distance between the measurement point and the origin, and θ is the polar angle. Using these equations, a linear u-v coordinate plot can be derived by doing the following: First, the measurement points are fitted to a circle by LSQ. Then the fitted circle is transformed to match the center to the origin of the Cartesian coordinate:

$$\tilde{T}_p = \begin{bmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\tilde{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\tilde{R}_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\tilde{P}'_i = \tilde{H}\,\tilde{P}_i = \tilde{R}_y \tilde{R}_x \tilde{T}_p \tilde{P}_i$$

Where $\cos\alpha = \frac{n_z}{d}$, $\sin\alpha = \frac{n_y}{d}$, $\cos\beta = d$, and $\sin\beta = n_x$. P_i and P'_i are the coordinates of points of the original circle and the translated circle, respectively. The

parameters n_x, n_y, and n_z are direction cosines of the normal vector of the fitted circle and $d=(n_z^2+n_y^2)^{1/2}$. The translation matrix $\tilde{T}_p$ translates the center (x_c, y_c, z_c) to the origin $(0, 0, 0)$. R_x and R_y rotate the normal vector of the circle about the Y and X axes, respectively, so as to match the Z axis. The new coordinates (x_i, y_i) can then be converted into *u-v* coordinates:

$$u_i=\theta_i=\tan^{-1}(\frac{y_i'}{x_i'})$$

$$v_i=r_i-R_{sq}=(x_i'^2+y_i'^2)^{1/2}-R_{sq}$$

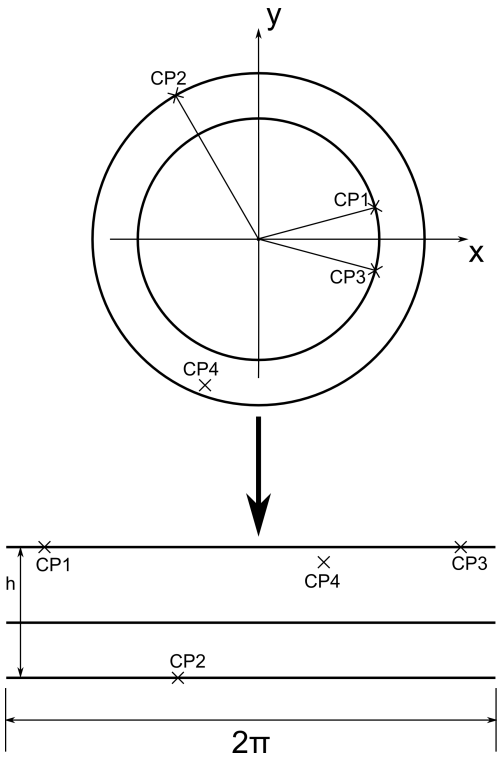


Figure 9: The CLRS Method

Where v_i represents the distance between the measurement point and the least-squares fitted circle, and R_{sq} is the radius estimated by the least-squares method. As a result of these calculations, the fitted circle becomes a line in the *u-v* plot as shown in

Figure 9. As it is shown, the process creates a linear set of lines that depict the minimum zone *h.* One thing to note is that the sequence of mapping the points to the *u*-*v* plot must be carefully controlled to have accurate results.

Limacon approximation to linearize the nonlinear coordinates of circular data is used because the shape difference between a circle and a limacon is similar to the error that occurs by manufacturing faults, and has the convenience of being linear in its parameters. As shown in **Figure 10**, the method of roundness suppression is used to create an approximate circle. This method is applied out of necessity stemming from the equipment that is used. A precision spindle is used coupled with a displacement transducer to measure the circle relative to the measurement area. The transducer output, ϵ, is used to indicate the separation of the component and a perfect, centered circle with radius equal to the suppression.

Because of the small size of the error, only the variation of the signal is precisely measured while the absolute value of the radius being lost or is able to be preserved only at a lower precision. To further explain this phenomenon, consider the display on a polar chart:

$$p(\theta) = M\epsilon + S$$

$$= ME\cos(\theta - \text{Ø}) + (R - L) + S - \frac{ME^2}{2\text{R}}\sin^2(\theta - \text{Ø})\ldots$$

The corresponding limacon would be:

$$r_L = M(E\cos(\theta - \text{Ø}) + R - L) + S$$

And a circle with the same parameters:

$$r_c = ME\cos(\theta - \text{Ø}) + M(R - L) + S - \frac{M^2E^2}{2(M(R - L) + S)}\sin^2(\theta - \text{Ø})\ldots$$

Where the approximated circle is magnified by M and plotted relative to an arbitrary radius, S. Since the equation for the circle is much larger than the true polar chart equation, ignoring second order terms altogether and using the limacon equation is a better representation providing that, approximately:

$$2(M(R-L)+S)<MR$$

Linear least squares analysis is then utilized on the derived limacon equation. It should be noted that in going through this process, the acceptability of limacon references depend on the quality of approximation.

Despite being novel ways to determine cylindricity, the CLRS and limacon methods rely on least-square analysis to determine the fitness of data. This is already on top of the approximations used to linearize the circle. Thus, it is determined that they are only a half-way measure and do not solve the problem of over-fitting.

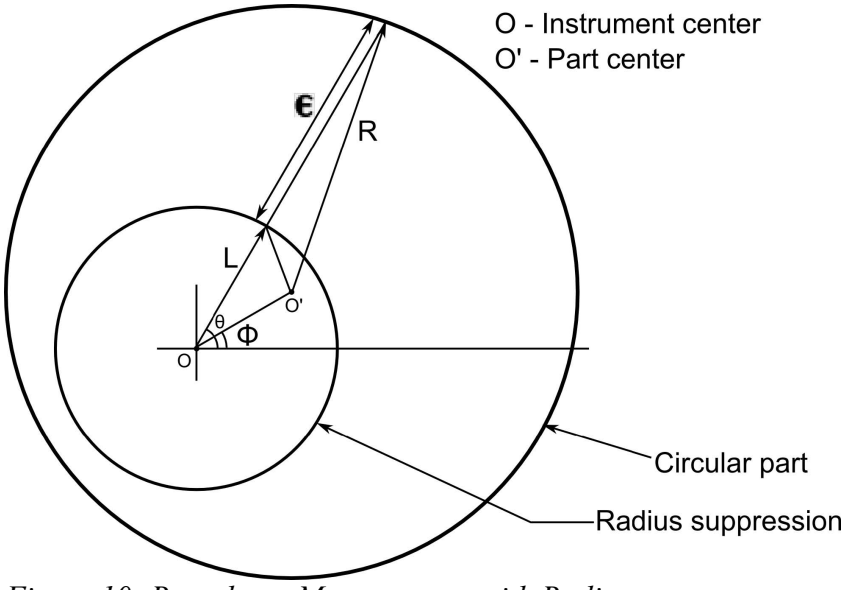


Figure 10: Roundness Measurement with Radius Suppression

Statistical learning algorithms have also been employed to solve the constrained optimization problem for MZC. The use of statistical learning theory reduces the

computational effort of conventional non-linear optimization methods. The simulated annealing algorithm is used for a variety of optimization problems [14]. Originally used for determining the distribution of atoms at a certain temperature, the simulated annealing algorithm uses a Hooke-Jeeves pattern search. This method requires an initial point with a specified step size. In the simulated annealing algorithm that was studied for circularity, an initial least-square circle solution was used to provide the initial point. Four basic components are used in the algorithm:

1. Configuration: A vector of decision variables (*S*) related to the circle-fitting models such as minimum zone circle.
2. Move generation mechanism: The moves in the optimization algorithm that are determined by the Hooke-Jeeves pattern search. The step sizes and direction are also determined.
3. Cost function: Using the nonlinear optimization problem of the minimum zone circle, a penalty technique is used to handle infeasible solution. This transforms the constrained problem into an unconstrained one, in which a penalty term is applied to the objective function for any violations of the constrains.
4. Cooling schedule: The process is considered completed when it is 'frozen,' which is determined by an arbitrary cooling schedule. In this case, the geometric cooling schedule is used.

Although it uses a novel method to determine circularity using machine vision, the simulated annealing method is not suitable for processing large amounts of data quickly. The conversion of a nonlinear constrained optimization problem into an unconstrained one is not desired for speed nor accuracy. Also, the dependence of least-

square analysis for the generation of an initial point for the search cannot be recommended due to the possibility of error involved.

In the genetic algorithm method [4], individual chromosomes are represented on a binary string. Each of the bits of the string are called genes and their varying values as alleles. Using basic genetic operators of reproduction, crossover, and mutation, a defined objective function is solved by using probabilities of reproduction of each individual within a population, generating a new population until a 'generation' is produced that meets the criteria for termination. This relationship is shown in **Figure 11**.

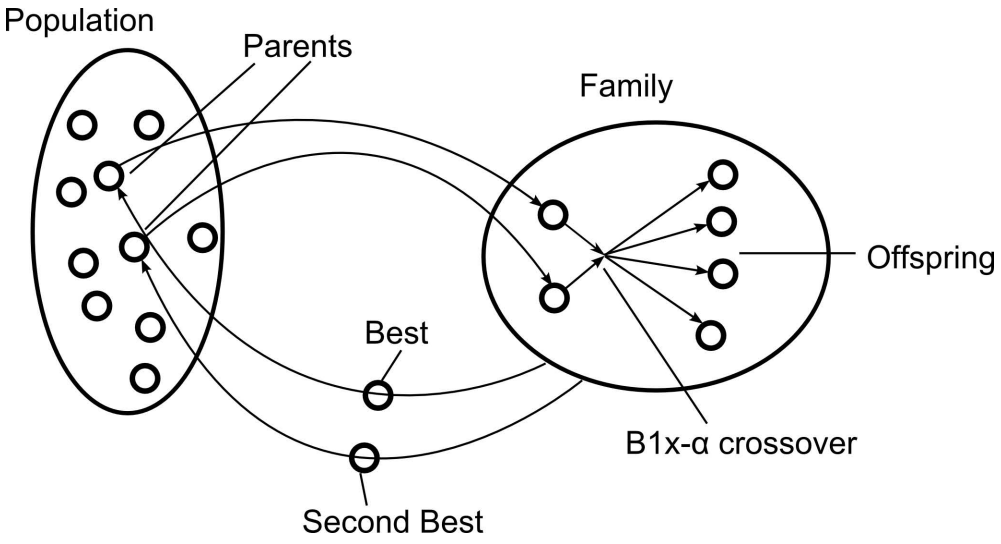


Figure 11: Flow Chart of Genetic Algorithm Circularity Evaluation

Mathematically, the genetic algorithm is applied as in the following:

1. Choose two parents v^1 and v^2 randomly from the population.
2. A value of each element v_i^c of the offspring vector v^c is chosen randomly from the interval $[V_i^1, V_i^2]$ following the uniform distribution, where

$$V_i^1 = min(v_i^1, v_i^2) - \alpha d,$$

$$V_i^2 = max(v_i^1, v_i^2) + \alpha d,$$

$$d = |v_i^1 - v_i^2|$$

Where v_i^1 and v_i^2 are the *i*th elements of v^1 and v^2, repectively, d_i is the distance between v_i^1 and v_i^2, and α is a positive parameter. The initial population that is used in the genetic algorithm is generated by using the least-square circularity algorithm. Once again, the reliance of the LSQ algorithm is a flaw that consequently affects the results that will be generated by this method.

Another use for statistical learning machines have been in the field of classification. Their advantage in this area of study comes in the form of generalization which allows for the solving of problems using a restricted amount of information. Traditional statistical learning methods include pattern recognition, density estimation, regression estimation, and parametric density estimation. By themselves, they have been proven insufficient in many situations because they are all based on the maximum likelihood method which rely solely on the Empirical Risk Minimization (ERM) principle. This method cannot estimate densities that are a mixture of different normal densities. This limits the type of data that can be analyzed into a very narrow criteria which is not acceptable. As a result, the theory of learning machines have evolved to use other methods in conjunction with the ERM principle.

To control the generalization ability of learning machines, a small sample of training instances much be used to construct an inductive principle for minimizing the risk functional. A small sample is defined by comparing the value derived by dividing the sample size with the Vapnik-Cheryonenkis (VC) dimension of functions. The Structural

Risk Minimization (SRM) inductive principle is used to minimize the risk-functional with respect to both empirical risk and the confidence interval. Using the SRM principle, a trade-off between the approximation of the given data and the complexity of the approximating function is discerned.

In constructing a learning machine, the generalizational ability must be carefully controlled for determining and appropriate approximation of the function the given sample data represents. From the SRM principle, to guarantee a high level of generalizational ability a structure must be constructed:

$$S_1 \subset S_2 \subset, \ldots, \subset S$$

Where S is the set of loss functions $S = Q(z, \alpha),\ \alpha \in \Lambda$. An appropriate element S_k and a function $Q(z, \alpha_l^k) \in S_k$ within the element is chosen that minimizes given bounds.

A simplified bound equation is written below:

$$R(\alpha_l^k) \leq R_{emp}(\alpha_l^k) + \Phi\left(\frac{l}{h_k}\right)$$

Where the first term, $R_{emp}(\alpha_l^k)$ is the empirical risk and $\Phi\left(\frac{l}{h_k}\right)$ is the confidence interval. l represents the sample size and h_k is the VC dimension of functions. If the machine is too complex, the confidence interval will be large. This produces an inordinate amount of errors on the test set, which is termed overfitting. To prevent this, a machine must be constructed with a small VC dimension h^*. However, if the VC dimension is too small, it is difficult to approximate the training data. In order to determine the balance between the confidence interval and approximation error, the learning machine has to be built to reflect a priori knowledge about the problem at hand.

In the learning process, the machine can use two methods to solve a problem. One is to minimize the empirical risk with a fixed confidence interval. The other is to fix a low empirical risk (close to zero) and minimize the confidence interval. The first strategy is used by neural networks while the second is utilized by support vector machines. Both use a set of linear indicator functions to generalize. The formulation of a neural network using this concept will be detailed below.

The following training set is used:

$$(x_{1,} y_1), ..., (x_l, y_l)$$

Where x_j is a vector, and $y_j \in 1, -1$, $j = 1, ..., l$. The set of linear indicator functions is denoted by the following:

$$f(x, w) = sign(w \cdot x), \; w \in R^n$$

Where $(w \cdot x)$ is the inner product between w and x. A vector of parameters w_0 (weights) needs to be found to minimize the empirical risk functional

$$R_{emp}(w) = \frac{1}{l} \sum (y_j - f(x_j, w))^2$$

If empirical risk can be reduced to zero, the vector w_0 can be found. However, if the training set cannot be separated without errors, the solution with the smallest number of errors is considered NP-complete. A gradient based procedure of the R_{emp} function cannot be performed because the gradient is either zero or undefined. Because of this, approximations of the indicator functions can be made with the use of sigmoid functions. The sigmoid function replaces the discontinuous function sign with a continuous sigmoid approximation. This allows the use of the sigmoid approximation to estimate the coefficients and use the threshold functions that are obtained.

Using these concepts, a neural network can be implemented, which is a combination of several layers of sigmoid elements in order to simulate a biological neural system acting under certain conditions. A back-propagation method, seen in **Figure 12,** has been proposed to calculate the gradient of the empirical risk for the sigmoid approximation of neural networks. To apply the neural network method for regression estimation, it is sufficient to use a linear function in the last layer in lieu of a sigmoidal one.

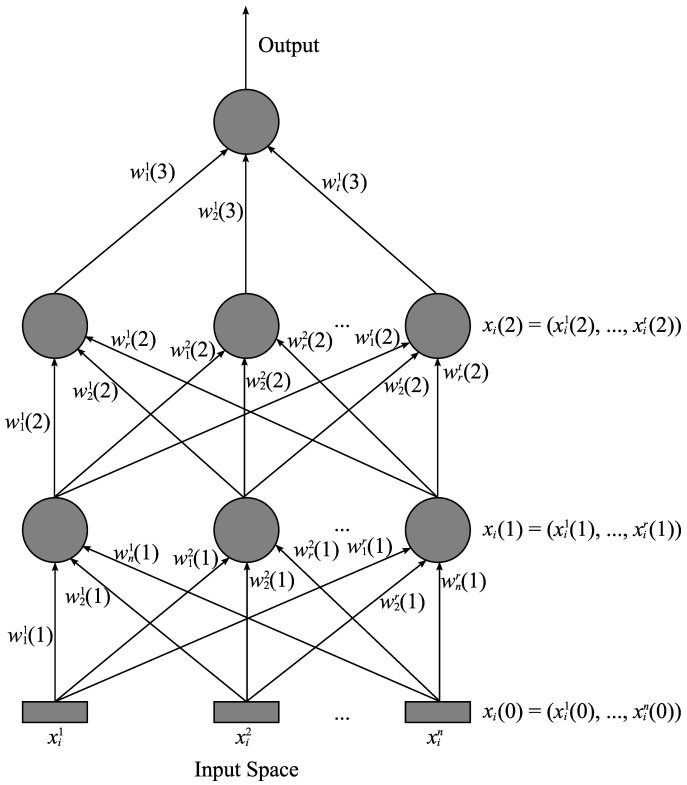


Figure 12: A Classic Back-propagation Neural Network

Even with its advantages over previous methods, the neural net has some deficiencies that prevent it from becoming a truly integrated approach. The empirical risk functional has many local minima which causes the quality of the obtained solution to depend on many factors, in particular the initial weight matrices. Heuristics can be

applied to control the size of the minimum. Computation time can be slow, which can also be remedied with heuristics. There is a marked trade-off between quality of approximation and rate of convergence. Because of its heavy reliance on heuristics in order to control the output, neural networks cannot be considered a well-controlled learning machine as it does not provide a truly general method in determining cylindricity that is not affected by the type of data that is provided.

From reviewing relevant literature, there is a lack of an integrated solution for circularity that provides more accuracy and stability with less computational effort. A novel approach that can improve in both areas will be a viable option in machine vision based inspection and is the motivation of this research.

2.3 SUPPORT VECTOR MACHINES

Support Vector Machines (SVM) [15] have recently gained traction as a new type of statistical learning algorithm, and will be discussed below. All of these examples estimate the model parameters for fitting based on the sample data, and their performance depends on the learning process that is affected by initial conditions.

The use of support vector machines has been shown to have good generalization performance, efficiency in computation, and robustness in higher dimensions. Unlike neural networks, SVM minimizes the confidence interval with a fixed empirical risk. The SVM method has been used in both classification and regression problems. Linear indicators are used again, but in a special format called optimal separating hyperplanes. To put the concept of using hyperplanes within the context of SVM in simple terms, it is described as follows. Linearly separable data is provided in the following form:

$$(x_{1,}\, y_1), \ldots, (x_i, y_i),\ x \in R,\ y \in +1, -1$$

This set of data is classified according to its values of *y* along the different values of *x*. They can be separated by a linear indicator, or commonly called a hyperplane as depicted in **Figure 13.**

The set of vectors are said to be separated by the optimal hyperplane if it is separated without error, while maximizing the distance between the closest vector and the hyperplane. The points x_i on the hyperplane satisfies $w \cdot x + b = 0$, where $w \in R^d$ is a vector normal to the hyperplane, $\|w\|$ is the Euclidian norm of *w*, and $b/\|w\|$ describes the perpendicular distance from the hyperplane to the origin. The margin denotes the shortest distance from the separating hyperplane to the closest positive and negative sample respectively. The margin of the hyperplane is defined as $d_+ + d_- = 0$. The distance of H_1 and H_2 from the origin can be calculated as $|1-b|/\|w\|$ and $|-1-b|/\|w\|$ respectively. The distance between the hyperplanes H_1 and H_2 equals to $2/\|w\|$.

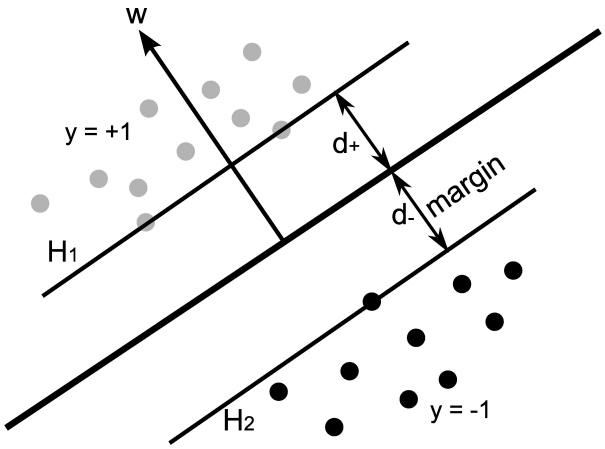


Figure 13: Data Separation Using a Hyperplane

The rules of the separating hyperplane can be defined by the following inequalities:

$$(w \cdot x_i) + b \geq 1 \text{ for } y_i = +1$$

Equation 1

$$(w \cdot x_i) + b \leq -1 \text{ for } y_i = -1$$

Equation 2

An optimal hyperplane separates the training data with the smallest norm of coefficients. To achieve this, the following optimization problem can be formulated:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to:} \quad y_i(x \cdot w + b) \geq 1, \ \forall i = 1, \ldots, l \end{aligned}$$

Equation 3

Although the original optimization function called for the maximization of $2/\|w\|$, Vapnik realized that it could be converted to a minimizing quadratic optimization problem with no effect to the final solution. In the case of tolerancing using the minimum enclosing zone, the objective function involves moving the two hyeprplanes as close as possible around the sample data. This converts the problem into a Support Vector Regression (SVR) problem, and the optimization problem is modified as follows:

$$\begin{aligned} &\text{maximize} \quad \frac{1}{2}\|w\|^2 \\ &\text{subject to:} \quad -1 \leq (x \cdot w) + b \leq +1, \ \forall i = 1, \ldots, l \end{aligned}$$

Equation 4

A visual representation of the enclosing hyperplanes can be seen in **Figure 14**. It is nearly identical to the separating hyperplanes as depicted in **Figure 13**, but acts in reverse to enclose the data points with the smallest margin rather than separate them with the biggest margin. As this ceases to be a classification algorithm, the classification variable y_i becomes unnecessary. Just like how Vapnik changed the original SVM optimization problem to better suit computational needs, it is possible to change the objective function of the SVR algorithm to a minimizing function:

$$\text{minimize} \quad \frac{2}{\|w\|^2}$$
$$\text{subject to:} \quad -1 \le w \cdot x + b \le +1 \;\; \forall i = 1, \ldots, l$$

Equation 5:

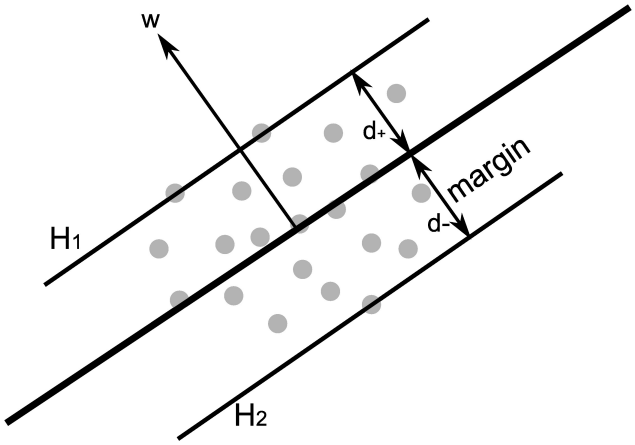


Figure 14: Formulation of Minimum Zone Using Hyperplanes

When this concept is introduced to a nonlinear surface in circularity and cylindricity analysis, it is necessary to revise **Equation 5** using kernel functions to transform the input space into the feature space where an optimal hyperplane can be formulated (**Figure 15**). Care must be taken to choose a kernel that provides good predictions while not being too computationally complex to cause a significant slowdown in processing time.

Three existing kernel machines are reviewed: The polynomial learning machine, the radial basis functions machine (RBF), and the two layer neural network. In order to use these kernel machines with an input space that is nonlinear, the convolution of the inner product is needed. The convolution function of the three can be expressed as follows:

$$K(x, x_i) = \left(\frac{(x \cdot x_i)}{n} \right)^d, \; d = 1, \ldots, 7$$

Equation 6

$$K(x, x_i) = \exp\left\{-\frac{(x - x_i)^2}{n\sigma^2}\right\}$$

Equation 7

$$K(x, x_i) = \tanh\left(\frac{b(x \cdot x_i)}{n} - C\right)$$

Equation 8

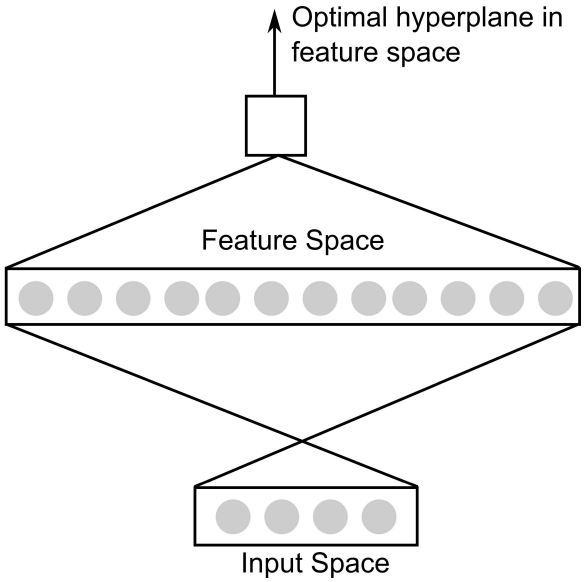


Figure 15: Mapping of Input Space into High Dimensional Feature Space

Where **Equation 6, Equation 7, and Equation 8** are the convolution functions of the polynomial learning machine, the RBF, and the two layer neural network respectively. In all three equations *n* represents the size of the input space. For example, for a camera with 16x16 resolution, the input space would be 256. As the decision rules are based off of the convoluted functions, it can be concluded that the size of the input space greatly affects both the results and the speed of the algorithms.

A new method where the decision variables are unaffected by the number of input space is needed for the analysis of a large amount of data points.

2.4 A NOVEL KERNEL FOR SVR

Taking account of the large input space that is generated from machine vision, a novel type of kernel for the formulation of circularity within the framework of Support Vector Regression is proposed. This novel SVR implementation is different from previous methods of nonlinear programming and conventional SVR algorithms. First, a function that maps the sampled points into a higher dimensional space is proposed:

$$x \to \phi(x): \phi(x) = [x^2 \; x \; y^2 \; y]$$

Equation 9

Where $x_i = [x_i, y_i] \in R^2$. As this problem deals with circularity analysis which will later be combined to create cylindricity results, only two dimensional data is considered. By putting non-linear data into higher dimensional space it is now possible to use a linear hyperplane to classify it.

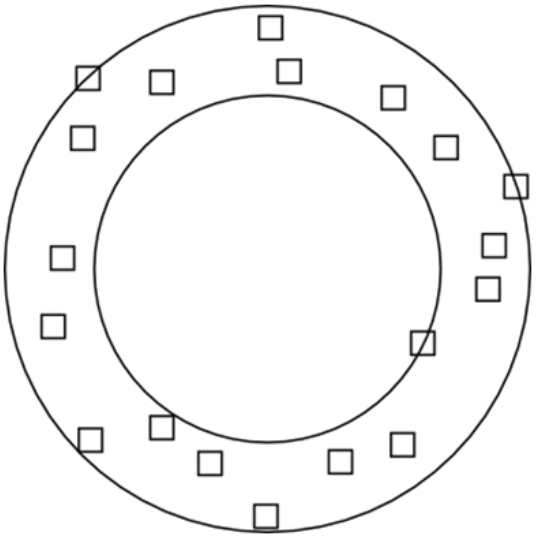


Figure 16: Circles for Computing Circularity in Higher Dimensional Space

The constraints in a hyperplane optimization model for tolerance analysis can be then modified as follows:

$$-1 \leq (w \cdot x) + b \leq +1 \rightarrow -1 \leq |w \cdot \phi(x)| + b \leq +1$$
Equation 10

The hyperplane is then represented with the following weight vector, which is now required to be four-dimensional due to the four dimensional nature of the feature space created by the new kernel machine:

$$w = [w_1 \; w_2 \; w_1 \; w_3]$$
Equation 11

As seen in **Figure 16**, the outer and inner circle can be represented by the following equations:

$$(w \cdot \phi(p)) + b = +1$$
Equation 12

$$(w \cdot \phi(p)) + b = -1$$
Equation 13

Where **Equation 12** represents the outer circle and **Equation 13** represents the inner circle, and *p* represents the sampled points of data. Note that following the rules of tolerancing, at least 3 points have to be intersected by the hyperplanes which requires the first and third element of the weight vector to be the same. This also ensures that the predicted shape is a circle rather than an ellipse. The resulting optimization problem with the new weight vector and kernel machine is

$$\text{minimize} \quad \frac{2}{\|w\|^2}$$
$$\text{subject to:} \quad -1 \leq [w_1 \; w_2 \; w_1 \; w_3]^T \cdot [x_i^2 \; x_i \; y_i^2 \; y_i] + b \leq +1 \; \forall \, i = 1, \ldots, l$$
Equation 14:

By expanding the equation, a sample point on the inner circle $p_1 = [x_1 \; y_1]$ can be represented with the following equation:

$$[w_1\ w_2\ w_1\ w_3]^T \cdot [x_1^2\ x_1\ y_1^2\ y_1] + b = -1$$
$$\downarrow$$
$$w_1 x_1^2 + w_2 x_1 + w_1 y_1^2 w_3 y_1 + b = -1$$
$$\downarrow$$
$$\left(x_1 + \frac{w_2}{2w_1}\right)^2 + \left(y_1 + \frac{w_3}{2w_1}\right)^2 = \frac{w_2^2 + w_3^2}{4w_1^2} + \frac{-1-b}{w_1}$$

Equation 15:

Likewise, the same treatment can be applied to a sample point on the outer circle $p_2 = [x_2\ y_2]$:

$$[w_1\ w_2\ w_1\ w_3]^T \cdot [x_2^2\ x_2\ y_2^2\ y_2] + b = 1$$
$$\downarrow$$
$$w_1 x_2^2 + w_2 x_2 + w_1 y_2^2 + w_2 y_2 + b = 1$$
$$\downarrow$$
$$\left(x_2 + \frac{w_2}{2w_1}\right)^2 + \left(y_2 + \frac{w_3}{2w_1}\right)^2 = \frac{w_2^2 + w_3^2}{4w_1^2} + \frac{1-b}{w_1}$$

Equation 16:

Using the equations above, it is simple to derive the formula to determine the shared center of the two circles:

$$\bar{x} = -\frac{w_2}{2w_1},\ \bar{y} = -\frac{w_3}{2w_1}$$

Equation 17

For the specific case of the circularity problem, it was determined that it was possible to simplify the objective function in the same vein as Vapnik's simplification for the original SVM. Due to the primary goal of circularity being the minimization of the distance between r_{max} and r_{min}, it is possible to define the objective function as thus:

$$\begin{aligned}
&\text{minimize} \quad r_{max}^2 - r_{min}^2 \\
&\text{subject to:} \quad -1 \le (w \cdot \phi(x_i)) + b \le +1 \;\; \forall i = 1, .., l \\
&\qquad\qquad\qquad \downarrow \\
&\text{minimize} \quad \left(\frac{w_2^2 + w_3^2}{4 w_1^2} + \frac{1-b}{w_1} \right) - \left(\frac{w_2^2 + w_3^2}{4 w_1^2} + \frac{-1-b}{w_1} \right) \\
&\text{subject to:} \quad -1 \le (w \cdot \phi(x_i)) + b \le +1 \;\; \forall i = 1, .., l \\
&\qquad\qquad\qquad \downarrow \\
&\text{minimize} \quad \frac{2}{w_1} \\
&\text{subject to:} \quad -1 \le (w \cdot \phi(x_i)) + b \le +1 \;\; \forall i = 1, .., l
\end{aligned}$$

Equation 18

The end result is an optimization problem that utilizes a nonlinear objective function with linear constraints, as opposed to the NLP algorithm with a linear objective function and nonlinear constraints. The optimization problem will take the sample points of a cylinder cross-section and dynamically change the inequality constraints depending on the sample points inputted. A defining feature of this kernel machine compared to the other established methods is that the input space has a drastically less of an effect on the solving of the problem.

CHAPTER 3 – BUILDING THE EXPERIMENT

3.1 FINDING THE CORRECT TOOLS

With the theory in place, it was needed to determine the correct way to realize it into a usable form. As defined in the beginning of this paper, the goal of this research was not only to determine the suitability of the proposed algorithm, but also to create an open framework which the algorithm could be tested against LSQ and NLP. In reviewing the available programming languages, it was determined that the Java Runtime Environment (JRE) [16] was suitable due to its portability and ease of use. What was also taken into account was the JRE's close integration with web applications as one of the future goals of the project is to create a program that can be accessed and used on a web page.

The biggest issue with making a mathematical application on Java is the inherent lack of functionality. As the language is designed to be completely modular, extra functions are added by either adding or creating new libraries for it to work off of. Searching the Internet, it was concluded that there was no specific general nonlinear solver available in the Java language at this current point in time. A few promising software packages such as JMathLib[17] and ojAlgo[18] are under development, but are too limited in scope and unfinished for the problem at hand. Other solvers mostly focus on unconstrained optimization problems, and it was decided that the loss of speed that is associated with converting a constrained nonlinear optimization problem to an unconstrained one was not desirable.

There was an initial attempt to write a nonlinear solver in the Java language with MATLAB's solvers as a reference. However, that proved to be too time-consuming and difficult of a task. A bridge between Java and a third party application was sought. The

first and most obvious choice was a library package that would allow communications with MATLAB and its included functions. However, it soon became apparent that the only possible way for Java to interact with MATLAB was to start within MATLAB itself – in short, MATLAB would call the Java program, and then the program will pass commands to MATLAB. As one of the project goals was to create a program that was open and provided the possibility of conversion into a standalone webpage application, MATLAB was determined to be unsuitable.

The possibilities of using GNU Octave [19] was explored. Octave is an open-source, freely available mathematical application that is designed to be syntactically compatible with MATLAB and is available in all commonly used operating systems. The Octave optimization package, freely available from the Octave package repository [20], included a sequential quadratic programming solver with the ability to compute solutions for nonlinear constrained optimization problems.

The search for a Java library that would provide a connection to Octave was a relatively simple one. Many programmers before me had the same issue, and there were a small number of libraries available for download. The most accessible and recently updated of those was the JavaOctave library [21]. By simply adding this library within the project file tree, it was possible to call up Octave as if it was another Java function.

3.2 PROGRAMMING THE GUI

With the proper tools and methods in place, it was possible to program an application to run the experiment. The Netbeans Integrated Development Environment [22] was chosen because of its recommendation by Sun, the creators of the Java

programming language, and its built-in graphical user interface (GUI) builder that conformed to all current standards of the constantly-changing JRE specifications.

The GUI that was to be built needed to have several options that can be changed by the user. Critical options that needed to be included were parameters that would change the generation of sample data to be analyzed by the algorithms. This included the radius, variance, number of points per cross-section, and the number of cross-sections. Non-critical options are parameters that would be set to change the display and output of data. This included the height of the ideal cylinder, which turned out to be irrelevant under analysis as the 3-D problem was collapsed into multiple 2-D problems, and iteration options to make the generation of multiple results easier. The end result of the GUI can be seen below in **Figure 17**.

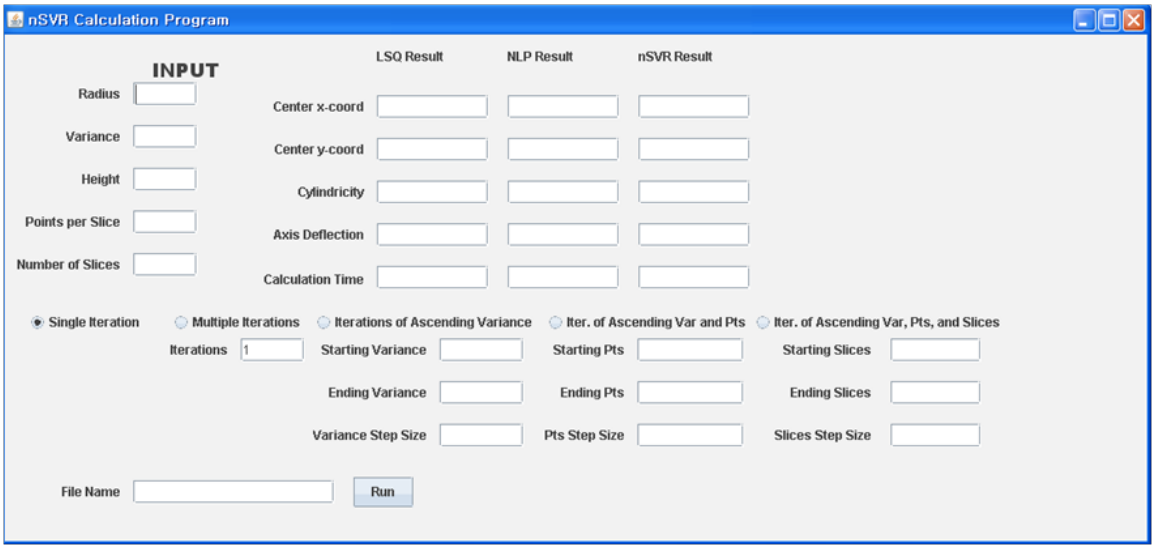


Figure 17: SVR Calculation Program GUI

In order to make it easier to generate results sequentially without having to input parameters one by one, a provision was made to provide for looping the analysis over a set number of iterations depending on the range of parameters and step-size. The results would then be outputted to user-defined comma-separated values (CSV) files. This GUI

was largely developed to suit the needs of the current experiment. Further modifications can be made as the source code of the program is a standard Netbeans project and can be opened from any computer and improved upon.

3.3 SETTING UP THE ALGORITHMS AND GENERATING RESULTS

The 3-dimensional method of determining the minimum enclosing zone of a cylinder shown previously can be difficult to calculate in a timely fashion. Because of this, several assumptions will be made:

- The unified axis of the cylinder is perfectly perpendicular to the inspection surface. This eliminates the need for determining the direction vector.
- It is unnecessary to determine the fitness of the zone position vector. This is due to the fact that the part being measured is an independent entity rather than attached to a larger assembly at the time of inspection.
- The analysis of axis deflection can be ignored – this is because the cylindricity measurements will largely eliminate the need for axis deflection measurements as it is assumed that excessive deflection will lead to the measured cylinder going outside the boundaries of the specified minimum enclosing zone.

With these assumptions, all aspects of the method of measuring cylindricity can be considered an extension of the measure of circularity, similar to what has been done in literature. To take it further, the unifying axis will be calculated using circularity analysis on the set of center points that will be derived from analyzing the cross-sectional data. The center point derived from circularity analysis of the set of center points of cross-sections will be the X and Y coordinate of the unifying axis of the cylinder.

The LSQ and NLP algorithms were chosen to compare against the novel implementation of SVR due to their common usage and mathematical clarity. Other algorithms tend to be a mixture of a separate concept with either the LSQ or NLP methods. The following steps will be taken by the program to provide results:

1. A user-defined number of cross-sections will be virtually generated by the program.
2. The LSQ, NLP, and SVR methods will analyze the cross-sections and provide the X and Y coordinates of the center points of each cross-section.
3. The generated center points will then be analyzed by the three circularity methods to create an overall center point to serve as the X and Y coordinates of the unifying axis.
4. The inner cylinder of the minimum enclosing cylinder will be found by calculating the closest data point from the unifying axis.
5. The outer cylinder of the minimum enclosing cylinder will be found by calculating the farthest data point from the unifying axis.
6. The cylindricity is then calculated by finding the distance between the inner and outer cylinder for each method.

The metrics that were used to compare these three methods were the calculated cylindricity resulting from the analysis, and the CPU time taken to come to an answer.

To calculate the LSQ results of the generated cylinder, a different solver than the SQP program in Octave was required. Since Octave was syntactically compatible with MATLAB, the Least Squares Geometric Elements Library package for MATLAB by the National Physical Laboratory Centre for Mathematics and Scientific Computing [23] was

used. This package had an easy to use least-square circularity analysis program, ls2dcircle.

To set up the non-linear programming algorithm, the following optimization model from literature that assumes a 2-2 model was programmed:

$$\begin{aligned} &\text{minimize} \quad r_{max} - r_{min} \\ &\text{subject to:} \quad r_{min} \le \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2} \le r_{max} \end{aligned}$$

Equation 19

Where r_{min} and r_{max} represent the inner and outer circles that enclose all sample points. This optimization problem is solved by the SQP solver available from Octave-Forge. The SVR algorithm as seen in **Equation 18** was also put through the SQP solver.

Initially, data was generated with the following parameters:

- Radius: 10
- Variance: 0.01 to 1.01 with a step size of 0.1
- Height: Not applicable
- Points per cross-section: 25 to 350 with a step size of 25
- Number of cross-sections: 6 to 15 with a step size of 1

With these parameters, 1,540 virtual cylinders were simulated and processed through the three methods, and the cylindricity and CPU time results were studied and compared. However, there was an issue that was immediately apparent in using this software package.

As seen in **Figure 18**, there is a failure during the solving of the SVR problem. This is only affected by the number of points, and was determined to be a failure in the SQP Solver within Octave itself, as the same algorithm generated favorable results with

much higher number of points within MATLAB. This was unsurprising as the SQP Solver is inherently deficient compared to the MATLAB general solver, as MATLAB's fmincon solver uses interior point code to solve an optimization problem in the most quickest and efficient way possible. Up until MATLAB version 5.3/R11, the QP optimizer within MATLAB had the same problem as the optimizer available for Octave when dealing with more than 100 data points at once.

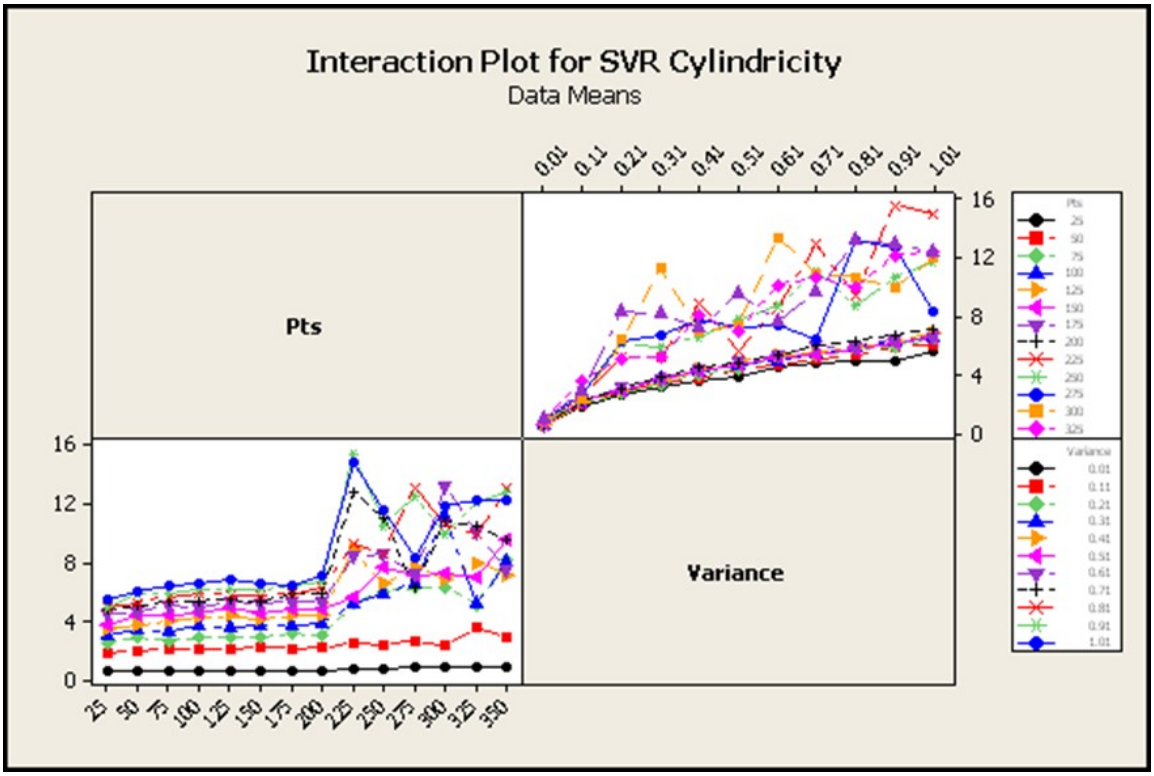


Figure 18: Example of Octave SQP Solver Failure

Taking note of this, a new set of data was generated:

- Radius: 10
- Variance: 0.01 to 1.01 with a step size of 0.1
- Height: Not applicable
- Points per cross-section: 25 to 200 with a step size of 25
- Number of cross-sections: 6 to 15 with a step size of 1

There were 2 sets of data generated at each intersection of these parameters, effectively creating 1,760 individual randomly generated cylinders. These cylinders were processed through the three methods, providing 5,280 results. It should be noted that in certain circumstances the LSQ algorithm would fail to converge and did not provide a valid solution (**Figure 19**). Unlike the failures seen with the SVR algorithm, the failure in convergence of the LSQ method was spread across all parameters, showing that it was the algorithm itself that had failed. In that case, all three results from that particular simulated cylinder were discarded to provide a more accurate statistical analysis of the results.

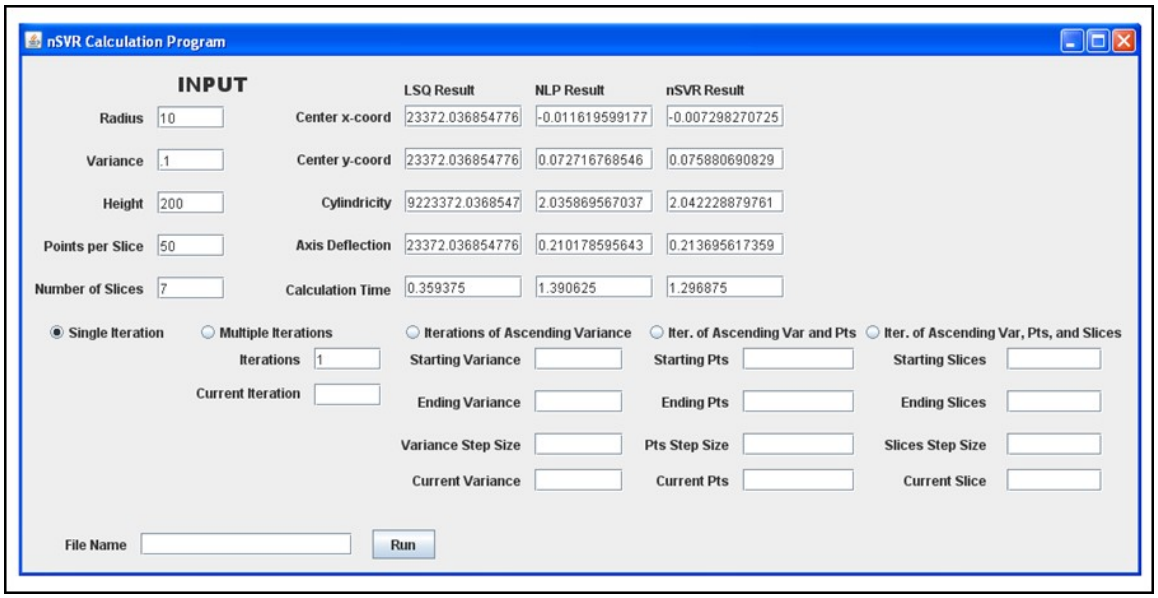


Figure 19: Example of LSQ Convergence Failure

After discarding these sets of data, 5,184 results were available for statistical analysis. The results were then processed through ANOVA analysis to measure the effect each parameter had on the results. Then, the results of each method were compared against each other, with the y-axis being the cylindricity/CPU Time measured, and the x-axis being the parameters that were considered to have significant effects.

3.4 AN EXAMPLE

The following example will illustrate the process this program will go through in order to give results. A virtual cylinder is generated with the following parameters:

- Radius: 10
- Variance: 0.5
- Height: 100
- Number of Points per Cross-Section: 100
- Number of Cross-Sections: 6

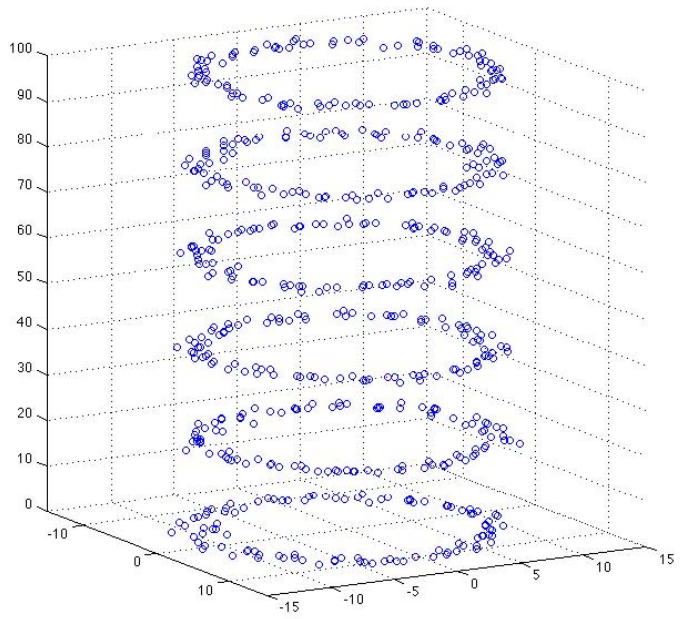


Figure 20: Graphical Scatterplot of Virtual Cylinder

The raw data points of the generated cylinder can be seen in **Appendix G**.

The next step involves the circularity analysis of each cross-section using least-squares circularity analysis, a non-linear program designed for circularity, and the SVR algorithm. The evaluation of the first cross section at z = 0 will be described below.

LSQ analysis is done first, using the ls2dcircle program from the National Physical Laboratory. This results in the output of an approximated location of the center point based on the data points at z=0.

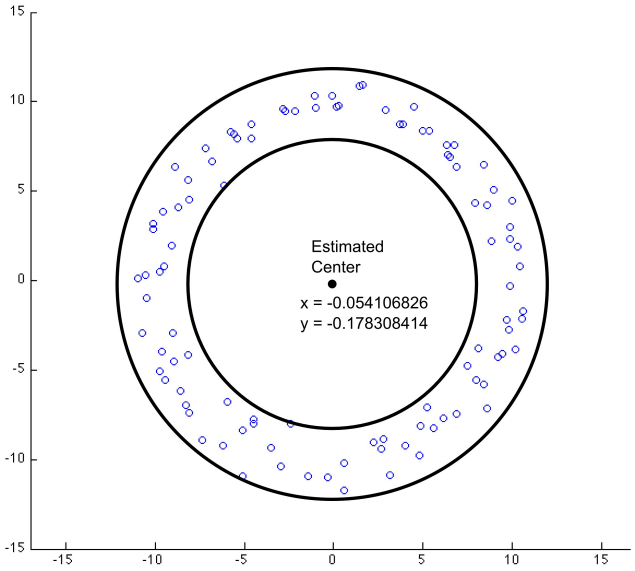


Figure 21: Circularity Estimation of Least-Squares Method

Figure 21 shows the results of the least-squares circularity analysis. The center point that is approximated by the sample data is marked in black, and the minimum enclosing zones are illustrated. With LSQ, the center x-y coordinates turned out to be x = -0.054106826 and y = -0.178308414. The closest sample point to the estimated center had a distance of r_{min} = 8.195834, and the furthest sample point had a distance of r_{max} = 11.86677, resulting in a circularity value of 3.670933.

Cross Section	LSQ x-coord	LSQ y-coord	NLP x-coord	NLP y-coord	SVR x-coord	SVR y-coord
Z = 0	-0.0541068	-0.1783084	-0.5265560	-0.3091609	-0.4483550	-0.3859560
Z = 20	0.0200256	0.1670919	0.5729632	0.0928081	0.5729632	0.0928081
Z = 40	0.0278199	-0.0310403	-0.0073595	0.0433884	-0.0073595	0.0433884
Z = 60	-0.0896733	-0.0639159	0.0794821	0.5145722	0.0794822	0.5145722
Z = 80	0.1454406	-0.1195368	0.0160528	-0.1885309	0.0160528	-0.1885309
Z = 100	0.0978473	0.1189759	0.1493220	-0.3095347	-0.0528080	0.0000000

Table 1: Center Point Results of Each Cross-Section

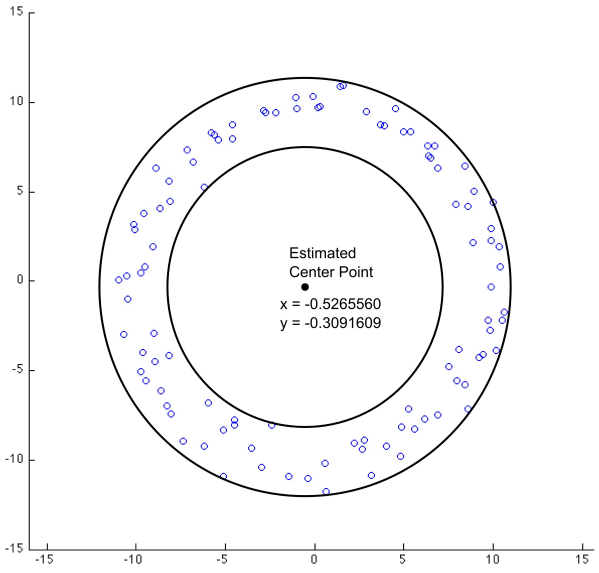


Figure 22: Circularity Estimation of Non-Linear Programming

The graphical results of the non-linear programming method can be seen in **Figure 22**. It can be observed that as specified in theory, the inner and outer circles of the minimum enclosing zone both intersect at least 2 sample points ith a x-y center coordinate of x = -0.526556002 and y = -0.309160915. The closest sample point to the estimated center had a distance of r_{min} = 7.945279 and the farthest sample point had a distance of r_{max} = 11.55348, resulting in a circularity value of 3.608203. This is an

improvement over the LSQ method as expected because the NLP method aims for a more stringent level of circularity fitting.

As with the LSQ and NLP methods, the novel SVR implementation is used for the sample points of the cross-section to obtain a center point estimation (**Figure 23)**. In the case of SVR, the x-y coordinates of the center point is x = -0.448355023 and y = -0.385955961. This gives us r_{min} = 7.889831 and r_{max} = 11.51436 for a circularity value of 3.62452. Although it is not as accurate as the NLP method in this case, the novel implementation of SVR shows to be an improvement over the LSQ method.

The remaining cross-sections are also analyzed using the three methods, and their respective center points are recorded. These results can be seen in **Table 1**. These center points are then analyzed themselves using the LSQ, NLP and SVR circularity methods.

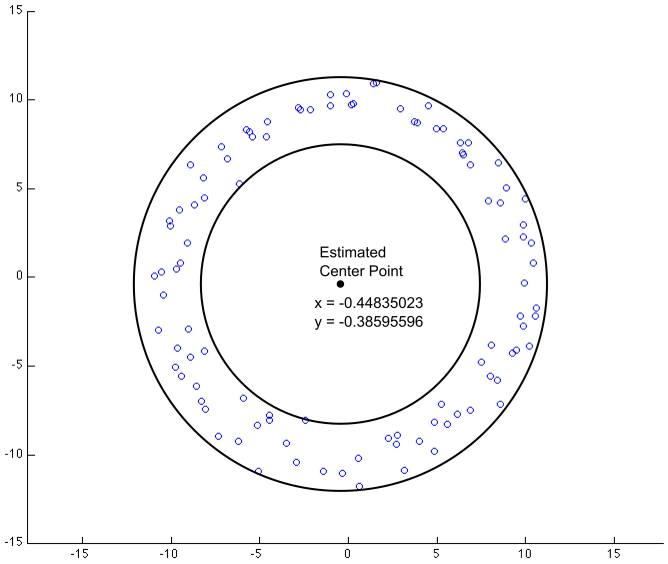


Figure 23: Circularity Estimation with SVR

Because of their relative simplicity in terms of the number of sample points, the optimization function that is used for NLP and SVR can be discerned. For the NLP method, the margin-reducing optimization problem transforms into the following:

$$\begin{aligned}
&\text{minimize} && r_{max} - r_{min} \\
&\text{subject to:} && r_{min} \le \sqrt{(-0.526556-\bar{x})^2+(-0.3091609-\bar{y})^2} \le r_{max} \\
& && r_{min} \le \sqrt{(0.5729632-\bar{x})^2+(0.0928081-\bar{y})^2} \le r_{max} \\
& && r_{min} \le \sqrt{(0.0073595-\bar{x})^2+(0.0433884-\bar{y})^2} \le r_{max} \\
& && r_{min} \le \sqrt{(0.0794821-\bar{x})^2+(0.5145722-\bar{y})^2} \le r_{max} \\
& && r_{min} \le \sqrt{(0.0160528-\bar{x})^2+(-0.1885309-\bar{y})^2} \le r_{max} \\
& && r_{min} \le \sqrt{(0.1493220-\bar{x})^2+(-0.3095342-\bar{y})^2} \le r_{max}
\end{aligned}$$

Likewise, the SVR optimization problem is solved in the following manner:

$$\begin{aligned}
&\min && \frac{2}{w_1} \\
&\text{s.t.:} && -1 \le [w_1\ w_2\ w_1\ w_3]\cdot[(-0.448355)^2\ -0.448355\ (-0.385956)^2\ -0.385956]+b \le 1 \\
& && -1 \le [w_1\ w_2\ w_1\ w_3]\cdot[(0.5729632)^2\ 0.5729632\ (0.0928081)^2\ 0.0928081]+b \le 1 \\
& && -1 \le [w_1\ w_2\ w_1\ w_3]\cdot[(-0.0073595)^2\ -0.0073595\ (0.0433884)^2\ 0.0433884]+b \le 1 \\
& && -1 \le [w_1\ w_2\ w_1\ w_3]\cdot[(0.0794822)^2\ 0.0794822\ (0.5145722)^2\ 0.5145722]+b \le 1 \\
& && -1 \le [w_1\ w_2\ w_1\ w_3]\cdot[(0.0160528)^2\ 0.0160528\ (-0.1885309)^2 - 0.1885309]+b \le 1 \\
& && -1 \le [w_1\ w_2\ w_1\ w_3]\cdot[(-0.052808)^2\ -0.052808\ (-0.1089133)^2\ -0.1089133]+b \le 1
\end{aligned}$$

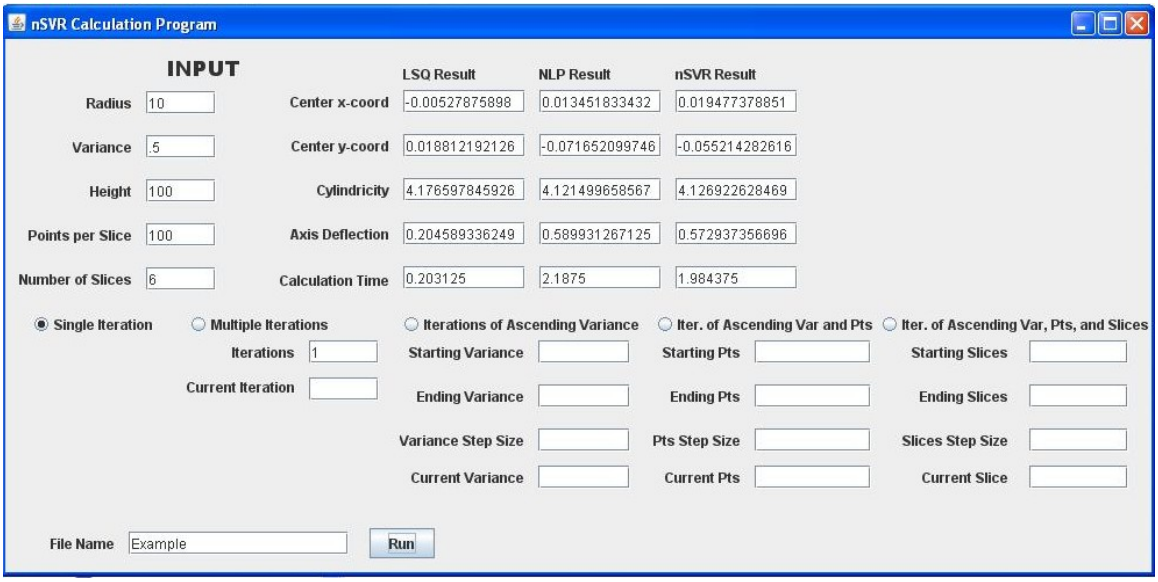


Figure 24: Results of the Example Calculation

The center point values that are calculated from the accumulated center points of each cross-section can be seen graphically in **Appendix H**. For LSQ, the values were x = -0.005279 and y = 0.01881. NLP resulted in x = 0.01343 and y = -0.07165. SVR gave

values of x = 0.01948 and y = -0.05521. With these values, as with the circularity values for each cross section, a cylindricity value can be derived by determining the closest point and the farthest point over all cross-sections from the calculated center points.

As seen in the results in **Figure 24**, the cylindricity values for LSQ was higher than either NLP and SVR method. As lower cylindricity values indicate a more accurate solution, this shows that the NLP and SVR method is superior to the LSQ method in terms of estimating the best-fit cylinder based on the sample data. What is also shown in the results is that SVR provided similar cylindricity results while being faster than the NLP method in terms of CPU processing time.

CHAPTER 4 – ANALYSIS AND CONCLUSION

4.1 ANALYSIS OF RESULTS

Analysis of Variance for Cylindricity, using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Method	2	24.17	24.09	12.05	13.92	0.000
Variance	10	16161.5	16161.5	1616.15	1867.53	0.000
Method* Variance	20	17.71	17.71	0.89	1.02	0.429
Error	5151	4457.65	4457.65	4457.65	0.87	
Total	5183	20611.04				

Table 2: Analysis of Variance for All Three Methods

As shown in **Table 2**, the circularity results are affected significantly by both method and variance, but the effects are independent of each other. Considering this, a boxplot graph with a confidence interval of 95% was created to see the differences between the three methods.

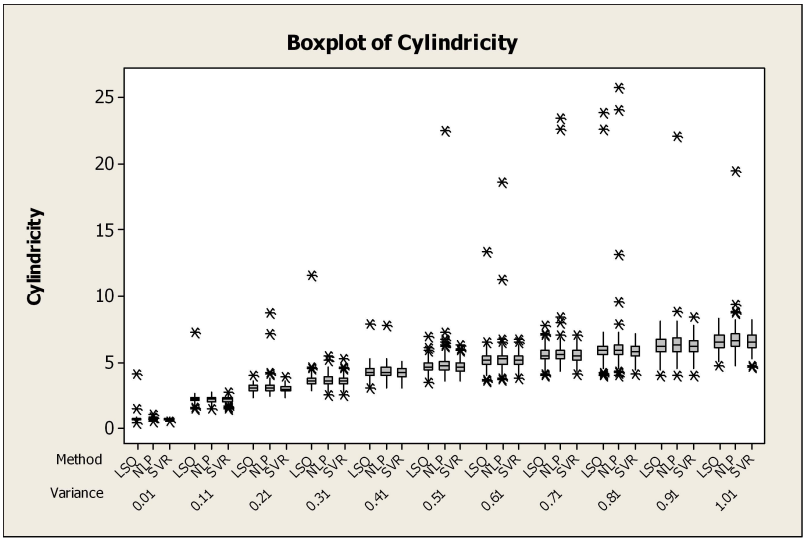


Figure 25: Boxplot of Cylindricity with Outliers

As seen in **Figure 25**, the values of cylindricity for LSQ and NLP show clear outliers that cannot be accepted as given as it is physically impossible to have these values. It is well known that the LSQ method can often give results that are unacceptable

due to the failure of convergence within the algorithm. In the figure above, however, these unconverged results were stricken from the graph in order to provide a clearer assessment between the three methods. Regardless, the LSQ method still showed to have immense outliers that skewed the data. Likewise, the NLP method is also sensitive to small defects such as dirt or asperities in the sample data and this is shown in outliers of its own. It should be noted that the SVR method does not show any immense outliers in its results. On the contrary, a good number of outliers for SVR are actually significantly lower than the 95% confidence interval threshold – in effect, they are considered outliers because the estimation was too good.

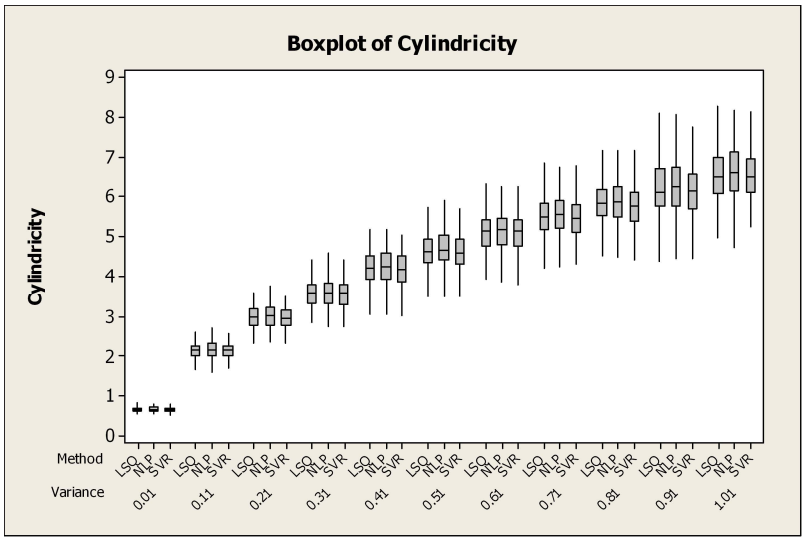


Figure 26: Boxplot of Cylindricity without Outliers

With the outliers taken out of the plot area (**Figure 26)**, the SVR method still shows its superiority in getting accurate cylindricity results compared to LSQ and NLP, and it becomes more apparent as the variance increases. The NLP method, as was expected, provided a larger confidence interval box as the experiment is providing noisy data. Due to the nature of the NLP algorithm, it overreacts to these simulated "asperities" and often provides cylindricity values that are higher than reality.

Analysis of Variance for Cylindricity using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Method	1	3.13	3.11	3.11	6.70	0.010
Variance	10	10422.15	10422.15	1042.22	2243.54	0.000
Method* Variance	10	4.54	4.54	0.45	0.98	0.000
Error	3434	1595.24	1594.24	0.46		
Total	3455	12025.06				

Table 3: Analysis of Variance LSQ vs SVR

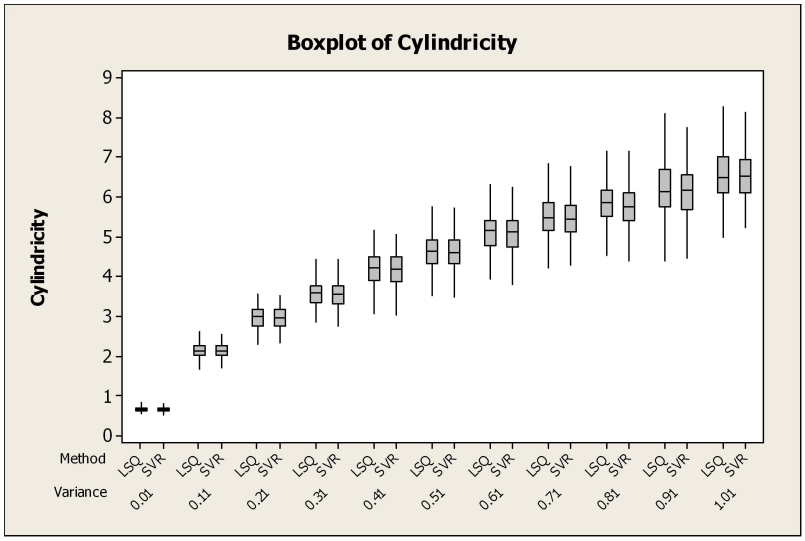


Figure 27: Boxplot of Cylindricity LSQ vs SVR

Analyzing the LSQ method against SVR individually, **Table 3** indicates that there is a statistical difference between the two methods. The same holds true across variance. Graphing this out on a boxplot as seen in **Figure 27** shows that SVR has an advantage over LSQ. Once again, it should be noted that this graph omits the impossible results that were generated by the LSQ algorithm, so the superiority of the SVR algorithm is actually diminished in the displayed plot.

Analysis of variance between NLP and SVR in **Table 4** demonstrates once again that there is a clear statistical difference between the two methods in terms of cylindricity. When plotted on a boxplot as shown in **Figure 28** there is a clear advantage by the SVR

method in estimating the cylindricity of a part. Furthermore, the range of the confidence interval for the SVR method is smaller than the NLP algorithm, which further validates SVR as a better way to measure cylindricity.

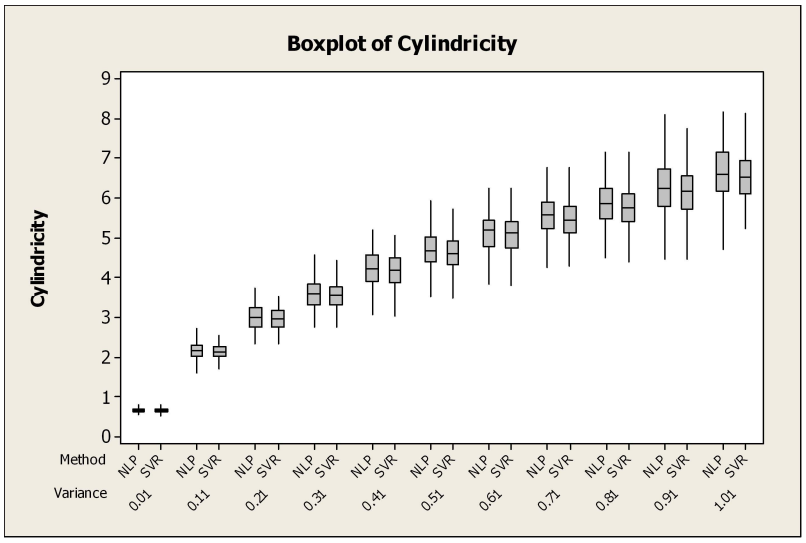


Figure 28: Boxplot of Cylindricity NLP vs SVR

Analysis of Variance for Cylindricity, using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Method	1	23.59	23.51	23.51	24.77	0.000
Variance	10	10882.61	10882.61	1088.26	1146.35	0.000
Method* Variance	10	13.39	13.39	1.34	1.41	0.169
Error	3434	3260.00	3260.00	0.95		
Total	3455	14179.59				

Table 4: Analysis of Variance NLP vs SVR

Further analysis of cylindricity data was done along number of points per cross-section and the number of cross-sections, and the resulting ANOVA tables can be seen in **Appendix I** (**Table 6 and Table 7**). The boxplots comparing the three methods are displayed in **Appendix J** (**Figure 31 and Figure 32**). In the case of the analysis along the number of cross-sections, only the results from sample points with a variance of 0.51 was

chosen for analysis to provide a clearer view of the differences. Both analysis results show that there is no discernible difference between the three methods with regards to cylindricity when taking the number of points or the number of cross-sections into account. This is because the variance in the distribution of the sample points is the main source of differing values of cylindricity.

Analysis of Variance for CPU Time, using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Method	1	58.57	61.65	61.65	11.11	0.001
Pts	7	38409.06	38321.05	5474.44	986.67	0.000
Variance	10	2724.08	2688.86	268.89	48.46	0.000
Method* Pts	7	145.58	156.70	22.39	4.03	0.000
Method* Variance	10	4337.94	4263.72	426.37	76.85	0.000
Pts* Variance	70	1180.27	1180.27	16.86	3.04	0.000
Method*Pts* Variance	70	1586.82	1586.82	22.67	4.09	0.000
Error	3280	18198.70	18198.70	5.55		
Total	3455	66641.03				

Table 5: Analysis of Variance of CPU Time NLP vs SVR

Analyzing the CPU processing time results show that the LSQ method is extremely fast compared to the NLP and SVR method. This is expected as the LSQ method is widely accepted as being mathematically simple and does not require much for a computer to calculate. Because of this inherent advantage in speed, the LSQ method was excluded from CPU time analysis.

A three-way ANOVA analysis (**Table 5)** shows that the variance and the number of points per cross-section greatly affect the speed of calculation. An analysis along the number of cross-sections can be seen in **Appendix I** (**Table 8**) which shows that the

number of cross-sections also affect CPU time. The data used for analysis of variance along the number of cross-sections had a fixed number of points per cross section at 100 to provide clarity. The resulting ANOVA tables and boxplot figures show that the CPU processing time of the SVR implementation is shorter in most cases across all parameters compared to the NLP algorithm.

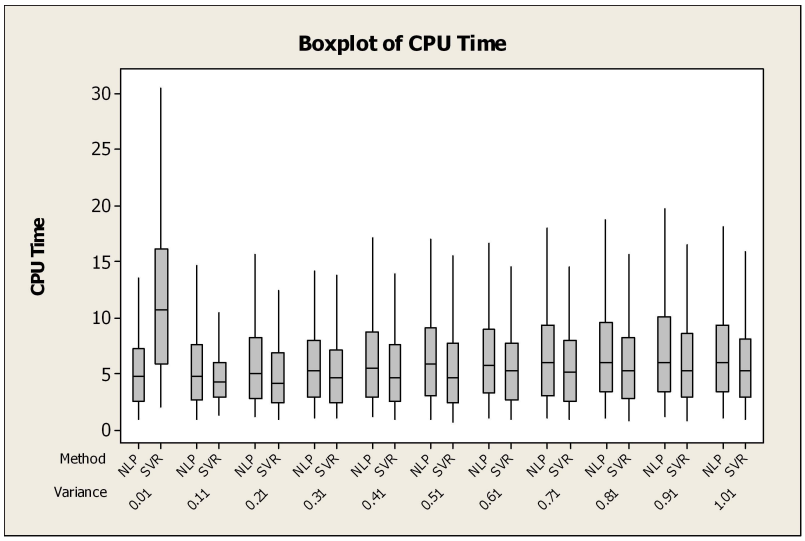


Figure 29: Boxplot of CPU Time NLP vs SVR Against Variance

Figure 29, Figure 30, and **Figure 33** in **Appendix J** shows the edge in computational efficiency the SVR method has over the NLP method along all parameters. The only area of deficiency is at low variance, which was to be expected due to the nature of the algorithm where the NLP method uses a linear objective function with nonlinear constraints, and the novel SVR implementation uses a nonlinear objective function with linear constraints.

Subsequently, in a situation with low variance, the NLP algorithm has an advantage as the constraints are not a significant factor. However, as the variance increases, the constraints take over the problem and the novel SVR implementation becomes more efficient than NLP. Because of this, in all other instances of the novel SVR

implementation computes faster than NLP. It is clear that the gap between the two methods gets wider as the constraint portion of the optimization problem becomes more complex due to the increase in the number of points per slice.

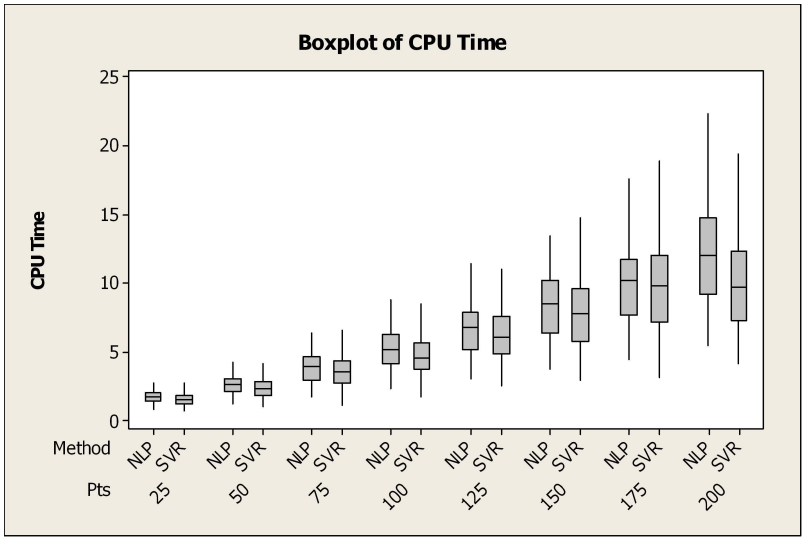


Figure 30: Boxplot of CPU Time NLP vs SVR Against Pts per Cross-Section

4.2 CONCLUSIONS

The results of the experiment show that the proposed SVR method is a viable alternative to the LSQ and NLP method in solving the problem of cylindricity. In particular, problems with a high number of sample points and a high variance are perfectly suited for the SVR algorithm. In addition, the proposed method is guaranteed to provide a reasonable result when the other two methods fail to do so. Using this new type of kernel within the framework of SVR provides results that are consistent in quality comparable to NLP with less processing time required. In a real-world setting, this would provide a manufacturer with an algorithm he or she can trust to give the correct answer rather than throwing out a good part because of bad computational results.

In addition, an open and accessible software package for the analysis of

cylindricity has been developed. This allows the ability of the user to easily generate data for analysis as well as modify the analytical algorithms that are employed. The results are either displayed or outputted into an easily understandable format that can be copied over to a spreadsheet program or a statistical program such as Minitab. This greatly reduces the time it takes for an idea to be implemented and studied in depth.

4.3 FUTURE WORK

With the implementation of a proven novel kernel within the SVR framework in an open source application, it is possible to make further improvements with little difficulty. The issue of Octave's SQP Solver failure will be worked on. Possible solutions to this problem will be a workaround the current solver, a new solver package within Octave, or a completely new mathematical library written in native Java code. It is hoped that, as is the nature of open-source programs, the existing code of the SQP Solver will be constantly improved and updated to be able to handle larger amounts of data. As the problem was well known in previous versions of MATLAB, it is expected that the open-source community will continuously improve the algorithm of the Octave SQP solver to the point where it is comparable to what is available in commercial programs.

The application will provide a launching point for further research into the study of the application of SVR for geometric tolerancing. With further work, a foolproof cylindricity analysis program could be developed by combining the LSQ algorithm for analyzing low variance data sets, and the SVR algorithm for high variance or in cases of obvious failure with other methods of analysis. In addition, a more well-designed GUI with graphing capability utilizing XML code is also planned to be included. The final product will be a web-based application that will accept external sample data from any

user for analysis. With the powerful nature of the Java programming language in this regard, this should be possible with no significant difficulties given the right hardware and bandwidth.

REFERENCES

[1] C. Prakasvudhisarn, T. B. Trafalis, and S. Raman, "Support vector regression for determination of minimum zone," *Journal of Manufacturing Science and Engineering*, vol. 125, pp. 736–739, 2003.

[2] A. A. G. Requicha, "Toward a theory of geometric tolerancing," *The International Journal of Robotics Research*, vol. 2, pp. 45–60, December 1983.

[3] M. C. Chen, "Roundness measurements for discontinuous perimeters via machine visions," *Computers in Industry*, vol. 47, pp. 185–197, 2002.

[4] X. Wen, Q. Xia, and Y. Zhao, "An effective genetic algorithm for circularity error unified evaluation," *International Journal of Machine Tools & Manufacture*, vol. 46, pp. 1770–1777, 2006.

[5] K. Carr and P. Ferreira, "Verification of form tolerances part ii: Cylindricity and straightness of a median line," *Precision Engineering*, vol. 17, pp. 144–156, 1995.

[6] K. Carr and P. Ferreira, "Verification of form tolerances part i: Basic issues, flatness, and straightness," *Precision Engineering*, vol. 17, pp. 131–143, 1995.

[7] H. T. Yau and C. H. Menq, "A unified least-squares approach to the evaluation of geometric errors using discrete measurement data," *International Journal of Machine Tools & Manufacture*, vol. 36, pp. 1269–1290, 1996.

[8] U. Roy and Y. Xu, "Form and orientation tolerance analysis for cylindrical surfaces in computer-aided inspection," *Computers in Industry*, vol. 26, pp. 127–134, 1995.

[9] W. Y. Jywe, C. H. Liu, and C. K. Chen, "The min-max problem of evaluating the form error of a circle," *Measurement*, vol. 26, pp. 273-282, 1999.

[10] M. Wang, S. H. Cheraghi, and A. S. Masud, "Circularity error evaluation theory and algorithm," *Precision Engineering*, vol. 23, pp. 164–176, 1999.

[11] T. Murthy and S. Abdin, "Minimum zone evaluation of surfaces," *International Journal of Machine Tool Design and Research*, vol. 20, pp. 123–136, 1980.

[12] J. Y. Lai and I. H. Chen, "Minimum zone evaluation of circles and cylinders," *International Journal of Machine Tools & Manufacture*, vol. 36, pp. 435–451, 1996.

[13] D. Chetwynd, "Roundness measurement using limacons," *Precision Engineering*, vol. 1, pp. 137–141, 1979.

[14] M. C. Chen, D. M. Tsai, and H. Y. Tseng, "A stochastic optimization approach for roundness measurements," *Pattern Recognition Letters*, vol. 20, pp. 707–719, 1999.

[15] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc, 1995.

[16] *Developer Resources for Java Technology*. (n.d.). Retrieved October 25, 2009, from http://java.sun.com

[17] *JMathLib site*. (n.d.). Retrieved October 25, 2009, from http://www.jmathlib.de/

[18] *ojAlgo*. (n.d.). Retrieved October 25, 2009, from http://www.ojalgo.org

[19] *Octave*. (n.d.). Retrieved October 25, 2009, from http://www.gnu.org/software/octave

[20] *Octave-Forge*. (n.d.). Retrieved October 25, 2009, from http://octave.sourceforge.net/

[21] *JavaOctave: Wiki: Home – Project Kenai*. (n.d.). Retrieved October 25, 2009, from http://kenai.com/projects/javaoctave/pages/Home

[22] *Welcome to NetBeans* . (n.d.). Retrieved October 25, 2009, from http://www.netbeans.org/

[23] *LSGE: The Least Squares Geometric Elements library consits of MatLab functions to find ....* (n.d.). Retrieved October 26, 2009, from http://scicomp.npl.co.uk/eurometros/gen_report.php?category=distributions&pkey=14&subform=yes

APPENDIX A: SVR MAIN GUI AND EXECUTIBLE FILE

<filename: nSVR.java>

```
package my.nSVR;

import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * nSVR.java
 *
 * Created on 2009. 9. 14, 8:48:08 P.M.
 */
/**
 *
 * @author Administrator
 */
public class nSVR extends javax.swing.JFrame {

    public static void saveTextFile(String contents, File file) throws IOException {
        PrintWriter out = new PrintWriter(new FileWriter(file));
        out.print(contents);
        out.close();
    }

    /** Creates new form nSVR */
    public nSVR() {
        initComponents();
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {
```

```
buttonGroup1 = new javax.swing.ButtonGroup();
jLabel12 = new javax.swing.JLabel();
jLabel10 = new javax.swing.JLabel();
jLabel11 = new javax.swing.JLabel();
jLabel8 = new javax.swing.JLabel();
jLabel9 = new javax.swing.JLabel();
jTextField17 = new javax.swing.JTextField();
jTextField5 = new javax.swing.JTextField();
jTextField16 = new javax.swing.JTextField();
jButton1 = new javax.swing.JButton();
jTextField6 = new javax.swing.JTextField();
jTextField20 = new javax.swing.JTextField();
jTextField19 = new javax.swing.JTextField();
jTextField18 = new javax.swing.JTextField();
jTextField7 = new javax.swing.JTextField();
jTextField8 = new javax.swing.JTextField();
jTextField9 = new javax.swing.JTextField();
jTextField10 = new javax.swing.JTextField();
jTextField11 = new javax.swing.JTextField();
jTextField12 = new javax.swing.JTextField();
jTextField13 = new javax.swing.JTextField();
jTextField14 = new javax.swing.JTextField();
jLabel2 = new javax.swing.JLabel();
jLabel1 = new javax.swing.JLabel();
jTextField4 = new javax.swing.JTextField();
jTextField15 = new javax.swing.JTextField();
jTextField1 = new javax.swing.JTextField();
jLabel5 = new javax.swing.JLabel();
jLabel4 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jTextField3 = new javax.swing.JTextField();
jTextField2 = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
jLabel7 = new javax.swing.JLabel();
jLabel13 = new javax.swing.JLabel();
jLabel14 = new javax.swing.JLabel();
jTextField21 = new javax.swing.JTextField();
jLabel15 = new javax.swing.JLabel();
jTextField22 = new javax.swing.JTextField();
jLabel16 = new javax.swing.JLabel();
jLabel17 = new javax.swing.JLabel();
jLabel18 = new javax.swing.JLabel();
jRadioButton1 = new javax.swing.JRadioButton();
jRadioButton2 = new javax.swing.JRadioButton();
jRadioButton3 = new javax.swing.JRadioButton();
jTextField23 = new javax.swing.JTextField();
```

```
jTextField24 = new javax.swing.JTextField();
jLabel19 = new javax.swing.JLabel();
jTextField25 = new javax.swing.JTextField();
jRadioButton4 = new javax.swing.JRadioButton();
jRadioButton5 = new javax.swing.JRadioButton();
jLabel20 = new javax.swing.JLabel();
jLabel21 = new javax.swing.JLabel();
jLabel22 = new javax.swing.JLabel();
jTextField26 = new javax.swing.JTextField();
jTextField27 = new javax.swing.JTextField();
jTextField28 = new javax.swing.JTextField();
jLabel23 = new javax.swing.JLabel();
jLabel24 = new javax.swing.JLabel();
jLabel25 = new javax.swing.JLabel();
jTextField29 = new javax.swing.JTextField();
jTextField30 = new javax.swing.JTextField();
jTextField31 = new javax.swing.JTextField();
jLabel26 = new javax.swing.JLabel();
jTextField32 = new javax.swing.JTextField();
jLabel27 = new javax.swing.JLabel();
jTextField33 = new javax.swing.JTextField();
jLabel28 = new javax.swing.JLabel();
jTextField34 = new javax.swing.JTextField();
jLabel29 = new javax.swing.JLabel();
jTextField35 = new javax.swing.JTextField();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
setTitle("nSVR Calculation Program");

jLabel12.setText("Calculation Time");

jLabel10.setText("Cylindricity");

jLabel11.setText("Axis Deflection");

jLabel8.setText("Center x-coord");

jLabel9.setText("Center y-coord");

jTextField5.setNextFocusableComponent(jButton1);

jButton1.setLabel("Run");
jButton1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton1ActionPerformed(evt);
   }
```

```
});

jTextField6.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField6ActionPerformed(evt);
    }
});

jTextField20.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField20ActionPerformed(evt);
    }
});

jLabel2.setText("Variance");

jLabel1.setText("Radius");

jTextField4.setNextFocusableComponent(jTextField5);
jTextField4.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField4ActionPerformed(evt);
    }
});

jTextField15.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField15ActionPerformed(evt);
    }
});

jTextField1.setFocusCycleRoot(true);
jTextField1.setNextFocusableComponent(jTextField2);

jLabel5.setText("Number of Slices");

jLabel4.setText("Points per Slice");

jLabel3.setText("Height");

jTextField3.setNextFocusableComponent(jTextField4);

jTextField2.setNextFocusableComponent(jTextField3);

jLabel6.setFont(new java.awt.Font("Arial Black", 0, 18));
jLabel6.setText("INPUT");
```

```
jLabel7.setText("LSQ Result");

jLabel13.setText("NLP Result");

jLabel14.setText("nSVR Result");

jTextField21.setText("1");

jLabel15.setText("Iterations");

jLabel16.setText("File Name");

jLabel17.setText("Starting Variance");

jLabel18.setText("Ending Variance");

buttonGroup1.add(jRadioButton1);
jRadioButton1.setSelected(true);
jRadioButton1.setText("Single Iteration");

buttonGroup1.add(jRadioButton2);
jRadioButton2.setText("Multiple Iterations");
jRadioButton2.setActionCommand("jRadioButton2");
jRadioButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton2ActionPerformed(evt);
    }
});

buttonGroup1.add(jRadioButton3);
jRadioButton3.setText("Iterations of Ascending Variance");
jRadioButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jRadioButton3ActionPerformed(evt);
    }
});

jLabel19.setText("Variance Step Size");

buttonGroup1.add(jRadioButton4);
jRadioButton4.setText("Iter. of Ascending Var and Pts");

buttonGroup1.add(jRadioButton5);
jRadioButton5.setText("Iter. of Ascending Var, Pts, and Slices");
```

```
        jLabel20.setText("Starting Pts");

        jLabel21.setText("Ending Pts");

        jLabel22.setText("Pts Step Size");

        jLabel23.setText("Starting Slices");

        jLabel24.setText("Ending Slices");

        jLabel25.setText("Slices Step Size");

        jLabel26.setText("Current Variance");

        jLabel27.setText("Current Pts");

        jLabel28.setText("Current Slice");

        jLabel29.setText("Current Iteration");

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(21, 21, 21)
                        .addComponent(jRadioButton1)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jLabel6)
                                .addGap(34, 34, 34)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.TRAILING)
                                    .addComponent(jLabel12)
                                    .addComponent(jLabel8)
                                    .addComponent(jLabel9)
                                    .addComponent(jLabel10)
                                    .addComponent(jLabel11))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
```

```
LATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                            .addComponent(jTextField6,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField7,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField8,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField9,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField10,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel7))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                            .addComponent(jLabel13)
                            .addComponent(jTextField15,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField11,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField12,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField13,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField14,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                            .addComponent(jLabel14)
                            .addComponent(jTextField16,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextField17,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextField18,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextField19,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jTextField20,
javax.swing.GroupLayout.PREFERRED_SIZE, 105,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createSequentialGroup()
                    .addGap(17, 17, 17)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                        .addComponent(jRadioButton2)
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.TRAILING)
                                .addComponent(jLabel29)
                                .addComponent(jLabel15))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                                .addComponent(jTextField35, 0, 0, Short.MAX_VALUE)
                                .addComponent(jTextField21,
javax.swing.GroupLayout.DEFAULT_SIZE, 60, Short.MAX_VALUE))))
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                        .addComponent(jRadioButton3)
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.TRAILING)
                                .addComponent(jLabel17)
                                .addComponent(jLabel18)
                                .addComponent(jLabel19)
                                .addComponent(jLabel26))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                                .addComponent(jTextField32)
```

```
                                .addComponent(jTextField25)
                                .addComponent(jTextField24)
                                .addComponent(jTextField23,
javax.swing.GroupLayout.DEFAULT_SIZE, 79, Short.MAX_VALUE))))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RE
LATED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                            .addComponent(jRadioButton4)
                            .addGroup(layout.createSequentialGroup()
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING)
                                    .addComponent(jLabel22,
javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addComponent(jLabel21,
javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addComponent(jLabel20,
javax.swing.GroupLayout.Alignment.TRAILING)
                                    .addComponent(jLabel27,
javax.swing.GroupLayout.Alignment.TRAILING))
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlaceme
nt.RELATED)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayou
t.Alignment.LEADING, false)
                                    .addComponent(jTextField33)
                                    .addComponent(jTextField28)
                                    .addComponent(jTextField27)
                                    .addComponent(jTextField26,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE))))))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                        .addComponent(jRadioButton5)
                        .addGroup(layout.createSequentialGroup()
                            .addGap(21, 21, 21)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING)
                                .addComponent(jLabel25,
javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jLabel23,
javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jLabel24,
javax.swing.GroupLayout.Alignment.TRAILING)
                                .addComponent(jLabel28,
javax.swing.GroupLayout.Alignment.TRAILING))
```

```
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.LEADING, false)
                            .addComponent(jTextField34)
                            .addComponent(jTextField29,
javax.swing.GroupLayout.DEFAULT_SIZE, 85, Short.MAX_VALUE)
                            .addComponent(jTextField30)
                            .addComponent(jTextField31)))))
                .addGroup(layout.createSequentialGroup()
                    .addGap(39, 39, 39)
                    .addComponent(jLabel16)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addComponent(jTextField22,
javax.swing.GroupLayout.PREFERRED_SIZE, 189,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(18, 18, 18)
                    .addComponent(jButton1)))
                .addContainerGap(91, Short.MAX_VALUE))
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                .addGroup(layout.createSequentialGroup()
                    .addContainerGap()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
TRAILING)
                        .addComponent(jLabel2)
                        .addComponent(jLabel5)
                        .addComponent(jLabel4)
                        .addComponent(jLabel3)
                        .addComponent(jLabel1))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATE
D)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)
                        .addComponent(jTextField5, javax.swing.GroupLayout.DEFAULT_SIZE,
60, Short.MAX_VALUE)
                        .addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE,
60, Short.MAX_VALUE)
                        .addComponent(jTextField3, javax.swing.GroupLayout.DEFAULT_SIZE,
60, Short.MAX_VALUE)
                        .addComponent(jTextField4, javax.swing.GroupLayout.DEFAULT_SIZE,
60, Short.MAX_VALUE)
                        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 60,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addContainerGap(882, Short.MAX_VALUE)))
```

```
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel6))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(22, 22, 22)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.TRAILING)
                            .addComponent(jLabel7)
                            .addComponent(jLabel14)
                            .addComponent(jLabel13))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.TRAILING)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING)
                                .addGroup(layout.createSequentialGroup()
                                    .addComponent(jTextField16,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(jTextField17,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(jTextField18,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(jTextField19,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addGap(18, 18, 18)
                                    .addComponent(jTextField20,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jTextField11,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(18, 18, 18)
                            .addComponent(jTextField12,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(18, 18, 18)
                            .addComponent(jTextField13,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(18, 18, 18)
                            .addComponent(jTextField14,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                                .addComponent(jTextField6,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel8))
                            .addGap(18, 18, 18)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                                .addComponent(jTextField7,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel9))
                            .addGap(18, 18, 18)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                                .addComponent(jTextField8,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addComponent(jLabel10))
                            .addGap(18, 18, 18)
```

```
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                            .addComponent(jTextField9,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel11))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.
Alignment.BASELINE)
                            .addComponent(jTextField10,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField15,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))))
                    .addComponent(jLabel12))))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
EADING)
                .addGroup(layout.createSequentialGroup()
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                        .addComponent(jRadioButton2)
                        .addComponent(jRadioButton3)
                        .addComponent(jRadioButton4)
                        .addComponent(jRadioButton5))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELAT
ED)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                        .addGroup(layout.createSequentialGroup()
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
                                .addComponent(jLabel15)
                                .addComponent(jTextField21,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.U
NRELATED)
                            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
                                .addComponent(jLabel29)
```

```
                        .addComponent(jTextField35,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
                        .addComponent(jTextField23,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel17)
                        .addComponent(jLabel20)
                        .addComponent(jTextField26,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel23)
                        .addComponent(jTextField29,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
                        .addComponent(jTextField24,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel18)
                        .addComponent(jLabel21)
                        .addComponent(jTextField27,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addComponent(jLabel24)
                        .addComponent(jTextField30,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(18, 18, 18)
                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
                        .addComponent(jLabel19)
                        .addComponent(jTextField25,
javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel22)
                            .addComponent(jTextField28,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel25)
                            .addComponent(jTextField31,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))))
                .addComponent(jRadioButton1))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATE
D)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(jLabel26)
                .addComponent(jTextField32,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel27)
                .addComponent(jTextField33,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel28)
                .addComponent(jTextField34,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(34, 34, 34)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.B
ASELINE)
                .addComponent(jTextField22,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel16)
                .addComponent(jButton1))
            .addContainerGap())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
            .addGroup(layout.createSequentialGroup()
                .addGap(45, 45, 45)
```

```
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel1)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel2)
                    .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel3)
                    .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jTextField4,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(jTextField5,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(238, Short.MAX_VALUE)))
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
```

```
// TODO add your handling code here:
double[][] nSVR_Win_result = new double[3][5];
int num_iterations_var;
double max_var;
double min_var;
double var_step;
int num_iterations_pts;
int max_pts;
int min_pts;
int pts_step;
int num_iterations_slices;
int max_slices;
int min_slices;
int slices_step;
double current_var;
int current_pts;
int current_slices;

String Out = null;
File name_of_file = new File(jTextField22.getText() + ".csv");

// file output of input parameters - the x-y coordinates of the cylinder.
// this is only active when single iteration is used.
File name_of_file_input = new File(jTextField22.getText()+"_input.csv");

// file output of individual outputs of each cross-section. This is only
// active when single iteration is used.
File name_of_file_indiv = new File(jTextField22.getText() + "_indiv.csv");

jTextField6.setText(null);
jTextField7.setText(null);
jTextField8.setText(null);
jTextField9.setText(null);
jTextField10.setText(null);
jTextField11.setText(null);
jTextField12.setText(null);
jTextField13.setText(null);
jTextField14.setText(null);
jTextField15.setText(null);
jTextField16.setText(null);
jTextField17.setText(null);
jTextField18.setText(null);
jTextField19.setText(null);
jTextField20.setText(null);
jTextField32.setText(null);
jTextField33.setText(null);
```

```
        jTextField34.setText(null);
        jTextField35.setText(null);

        if (jRadioButton2.isSelected()) {
            Out = "Radius: " + jTextField1.getText() + ", Variance: " + jTextField2.getText()
+ ", Points: " + jTextField4.getText() + ", Slices: " + jTextField5.getText() + "\n";
            Out = Out + "Pts, Variance, Slices, LSQ Cylindricity, NLP Cylindricity, SVR
Cylindricity, LSQ Axis Def, NLP Axis Def, SVR Axis Def, LSQ CPU, NLP CPU, SVR
CPU \n";
            for (int i = 0; i < Integer.parseInt(jTextField21.getText()); i++) {
                jTextField35.setText(Integer.toString(i+1));
                nSVR_Win_result =
nSVR_Win.nSVR(Double.parseDouble(jTextField1.getText()),
Double.parseDouble(jTextField2.getText()), Double.parseDouble(jTextField3.getText()),
Integer.parseInt(jTextField4.getText()), Integer.parseInt(jTextField5.getText()), 0);
                Out = Out + jTextField4.getText() + "," + jTextField2.getText() + "," +
jTextField5.getText() + ",";
                for (int j = 2; j < 5; j++) {
                    for (int k = 0; k < 3; k++) {
                        Out = Out + Double.toString(nSVR_Win_result[k][j]) + ",";
                    }
                    if (j == 4) {
                        Out = Out + "\n";
                    }
                }
            }

            try {
                nSVR.saveTextFile(Out, name_of_file);
            } catch (IOException ex) {
                Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
            }
        } else if (jRadioButton3.isSelected()) {
            min_var = Double.parseDouble(jTextField23.getText());
            max_var = Double.parseDouble(jTextField24.getText());
            var_step = Double.parseDouble(jTextField25.getText());
            num_iterations_var = (int) Math.floor((max_var - min_var) / var_step) + 1;

            Out = "Radius: " + jTextField1.getText() + ", Variance: " + jTextField23.getText()
+ ":" + jTextField25.getText() + ":" + jTextField24.getText() + ", Points: " +
jTextField4.getText() + ", Slices: " + jTextField5.getText() + "\n";
            Out = Out + "Pts, Variance, Slices, LSQ Cylindricity, NLP Cylindricity, SVR
Cylindricity, LSQ Axis Def, NLP Axis Def, SVR Axis Def, LSQ CPU, NLP CPU, SVR
CPU \n";

            for (int i = 0; i < num_iterations_var; i++) {
```

```
            current_var = min_var + i * var_step;
            jTextField32.setText(Double.toString(current_var));
            nSVR_Win_result =
nSVR_Win.nSVR(Double.parseDouble(jTextField1.getText()), current_var,
Double.parseDouble(jTextField3.getText()), Integer.parseInt(jTextField4.getText()),
Integer.parseInt(jTextField5.getText()),0);
            Out = Out + jTextField4.getText() + "," + Double.toString(current_var) + "," +
jTextField5.getText() + ",";
            for (int j = 2; j < 5; j++) {
               for (int k = 0; k < 3; k++) {
                  Out = Out + Double.toString(nSVR_Win_result[k][j]) + ",";
               }
               if (j == 4) {
                  Out = Out + "\n";
               }
            }

         }
         try {
            nSVR.saveTextFile(Out, name_of_file);
         } catch (IOException ex) {
            Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
         }

      } else if (jRadioButton4.isSelected()) {
         min_var = Double.parseDouble(jTextField23.getText());
         max_var = Double.parseDouble(jTextField24.getText());
         var_step = Double.parseDouble(jTextField25.getText());
         min_pts = Integer.parseInt(jTextField26.getText());
         max_pts = Integer.parseInt(jTextField27.getText());
         pts_step = Integer.parseInt(jTextField28.getText());

         num_iterations_var = (int) Math.floor((max_var - min_var) / var_step) + 1;
         num_iterations_pts = (int) Math.floor((max_pts - max_pts) / pts_step) + 1;

         Out = "Radius: " + jTextField1.getText() + ", Variance: " + jTextField23.getText()
+ ":" + jTextField25.getText() + ":" + jTextField24.getText() + ", Points: " +
jTextField26.getText() + ":" + jTextField28.getText() + ":" + jTextField27.getText() + ",
Slices: " + jTextField5.getText() + "\n";
         Out = Out + "Pts, Variance, Slices, LSQ Cylindricity, NLP Cylindricity, SVR
Cylindricity, LSQ Axis Def, NLP Axis Def, SVR Axis Def, LSQ CPU, NLP CPU, SVR
CPU \n";

         for (int ptscount = 0; ptscount < num_iterations_pts; ptscount++) {
            current_pts = min_pts + ptscount * pts_step;
            jTextField33.setText(Integer.toString(current_pts));
```

```
            for (int i = 0; i < num_iterations_var; i++) {
                current_var = min_var + i * var_step;
                jTextField32.setText(Double.toString(current_var));

                nSVR_Win_result =
nSVR_Win.nSVR(Double.parseDouble(jTextField1.getText()), current_var,
Double.parseDouble(jTextField3.getText()), current_pts,
Integer.parseInt(jTextField5.getText()),0);
                Out = Out + Double.toString(current_pts) + "," +
Double.toString(current_var) + "," + jTextField5.getText() + ",";
                for (int j = 2; j < 5; j++) {
                    for (int k = 0; k < 3; k++) {
                        Out = Out + Double.toString(nSVR_Win_result[k][j]) + ",";
                    }
                    if (j == 4) {
                        Out = Out + "\n";
                    }
                }
            }
        }
        try {
            nSVR.saveTextFile(Out, name_of_file);
        } catch (IOException ex) {
            Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
        }

    } else if (jRadioButton5.isSelected()) {
        min_var = Double.parseDouble(jTextField23.getText());
        max_var = Double.parseDouble(jTextField24.getText());
        var_step = Double.parseDouble(jTextField25.getText());
        min_pts = Integer.parseInt(jTextField26.getText());
        max_pts = Integer.parseInt(jTextField27.getText());
        pts_step = Integer.parseInt(jTextField28.getText());
        min_slices = Integer.parseInt(jTextField29.getText());
        max_slices = Integer.parseInt(jTextField30.getText());
        slices_step = Integer.parseInt(jTextField31.getText());

        num_iterations_var = (int) Math.floor((max_var - min_var) / var_step) + 1;
        num_iterations_pts = (int) Math.floor((max_pts - min_pts) / pts_step) + 1;
        num_iterations_slices = (int) Math.floor((max_slices - min_slices) / slices_step) +
1;

        Out = "Radius: " + jTextField1.getText() + ", Variance: " + jTextField23.getText()
+ ":" + jTextField25.getText() + ":" + jTextField24.getText() + ", Points: " +
jTextField26.getText() + ":" + jTextField28.getText() + ":" + jTextField27.getText() + ",
Slices: " + jTextField29.getText() + ":" + jTextField31.getText() + ":" +
```

```
jTextField30.getText() + "\n";
        Out = Out + "Pts, Variance, Slices, LSQ Cylindricity, NLP Cylindricity, SVR
Cylindricity, LSQ Axis Def, NLP Axis Def, SVR Axis Def, LSQ CPU, NLP CPU, SVR
CPU \n";

        for (int slicecount = 0; slicecount < num_iterations_slices; slicecount++) {
           current_slices = min_slices + slicecount * slices_step;
           jTextField34.setText(Integer.toString(current_slices));

           for (int ptscount = 0; ptscount < num_iterations_pts; ptscount++) {
              current_pts = min_pts + ptscount * pts_step;
              jTextField33.setText(Integer.toString(current_pts));

              for (int i = 0; i < num_iterations_var; i++) {
                 current_var = min_var + i * var_step;
                 jTextField32.setText(Double.toString(current_var));

                 nSVR_Win_result =
nSVR_Win.nSVR(Double.parseDouble(jTextField1.getText()), current_var,
Double.parseDouble(jTextField3.getText()), current_pts, current_slices,0);
                 Out = Out + Double.toString(current_pts) + "," +
Double.toString(current_var) + "," + Double.toString(current_slices) + ",";
                 for (int j = 2; j < 5; j++) {
                    for (int k = 0; k < 3; k++) {
                       Out = Out + Double.toString(nSVR_Win_result[k][j]) + ",";
                    }
                    if (j == 4) {
                       Out = Out + "\n";
                    }
                 }
              }
           }
        }
        try {
           nSVR.saveTextFile(Out, name_of_file);
        } catch (IOException ex) {
           Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
        }

     } else {
        nSVR_Win_result =
nSVR_Win.nSVR(Double.parseDouble(jTextField1.getText()),
Double.parseDouble(jTextField2.getText()), Double.parseDouble(jTextField3.getText()),
Integer.parseInt(jTextField4.getText()), Integer.parseInt(jTextField5.getText()),1);
        jTextField6.setText(Double.toString(Math.round(nSVR_Win_result[0][0] * 1e12)
/ 1e12));
```

```
            jTextField7.setText(Double.toString(Math.round(nSVR_Win_result[0][1] * 1e12)
/ 1e12));
            jTextField8.setText(Double.toString(Math.round(nSVR_Win_result[0][2] * 1e12)
/ 1e12));
            jTextField9.setText(Double.toString(Math.round(nSVR_Win_result[0][3] * 1e12)
/ 1e12));
            jTextField10.setText(Double.toString(Math.round(nSVR_Win_result[0][4] *
1e12) / 1e12));
            jTextField11.setText(Double.toString(Math.round(nSVR_Win_result[1][0] *
1e12) / 1e12));
            jTextField12.setText(Double.toString(Math.round(nSVR_Win_result[1][1] *
1e12) / 1e12));
            jTextField13.setText(Double.toString(Math.round(nSVR_Win_result[1][2] *
1e12) / 1e12));
            jTextField14.setText(Double.toString(Math.round(nSVR_Win_result[1][3] *
1e12) / 1e12));
            jTextField15.setText(Double.toString(Math.round(nSVR_Win_result[1][4] *
1e12) / 1e12));
            jTextField16.setText(Double.toString(Math.round(nSVR_Win_result[2][0] *
1e12) / 1e12));
            jTextField17.setText(Double.toString(Math.round(nSVR_Win_result[2][1] *
1e12) / 1e12));
            jTextField18.setText(Double.toString(Math.round(nSVR_Win_result[2][2] *
1e12) / 1e12));
            jTextField19.setText(Double.toString(Math.round(nSVR_Win_result[2][3] *
1e12) / 1e12));
            jTextField20.setText(Double.toString(Math.round(nSVR_Win_result[2][4] *
1e12) / 1e12));
        }

}

    private void jTextField6ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

    private void jTextField20ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

    private void jTextField4ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
}

    private void jTextField15ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
```

```
}

    private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new nSVR().setVisible(true);
            }
        });
    }
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel10;
    private javax.swing.JLabel jLabel11;
    private javax.swing.JLabel jLabel12;
    private javax.swing.JLabel jLabel13;
    private javax.swing.JLabel jLabel14;
    private javax.swing.JLabel jLabel15;
    private javax.swing.JLabel jLabel16;
    private javax.swing.JLabel jLabel17;
    private javax.swing.JLabel jLabel18;
    private javax.swing.JLabel jLabel19;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel20;
    private javax.swing.JLabel jLabel21;
    private javax.swing.JLabel jLabel22;
    private javax.swing.JLabel jLabel23;
    private javax.swing.JLabel jLabel24;
    private javax.swing.JLabel jLabel25;
    private javax.swing.JLabel jLabel26;
    private javax.swing.JLabel jLabel27;
    private javax.swing.JLabel jLabel28;
    private javax.swing.JLabel jLabel29;
    private javax.swing.JLabel jLabel3;
```

```
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JRadioButton jRadioButton4;
    private javax.swing.JRadioButton jRadioButton5;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField10;
    private javax.swing.JTextField jTextField11;
    private javax.swing.JTextField jTextField12;
    private javax.swing.JTextField jTextField13;
    private javax.swing.JTextField jTextField14;
    private javax.swing.JTextField jTextField15;
    private javax.swing.JTextField jTextField16;
    private javax.swing.JTextField jTextField17;
    private javax.swing.JTextField jTextField18;
    private javax.swing.JTextField jTextField19;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField20;
    private javax.swing.JTextField jTextField21;
    private javax.swing.JTextField jTextField22;
    private javax.swing.JTextField jTextField23;
    private javax.swing.JTextField jTextField24;
    private javax.swing.JTextField jTextField25;
    private javax.swing.JTextField jTextField26;
    private javax.swing.JTextField jTextField27;
    private javax.swing.JTextField jTextField28;
    private javax.swing.JTextField jTextField29;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField30;
    private javax.swing.JTextField jTextField31;
    private javax.swing.JTextField jTextField32;
    private javax.swing.JTextField jTextField33;
    private javax.swing.JTextField jTextField34;
    private javax.swing.JTextField jTextField35;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField6;
    private javax.swing.JTextField jTextField7;
    private javax.swing.JTextField jTextField8;
    private javax.swing.JTextField jTextField9;        }
```

APPENDIX B: MAIN MODULE SOURCE CODE

Filename: <nSVR_Win.java>

```
package my.nSVR;

import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import dk.ange.octave.OctaveEngine;
import dk.ange.octave.OctaveEngineFactory;
import dk.ange.octave.type.*;
import java.util.Scanner;
import java.util.Random;

public class nSVR_Win {
	static Scanner sc = new Scanner(System.in);

	/**
	 * @param args
	 */

	public static double[][] nSVR(double radius, double variance, double height, int
no_points, int no_slices, int single) {
		// TODO Auto-generated method stub

		Random generator = new Random();
		double calc_time_lsq = 0;
		double calc_time_nlp = 0;
		double calc_time_svr = 0;
		double rmax_temp_lsq = 0;
		double rmax_temp_nlp = 0;
		double rmax_temp_svr = 0;
		double rmin_temp_lsq = 999999999;
		double rmin_temp_nlp = 999999999;
		double rmin_temp_svr = 999999999;
		double r_lsq = 0;
		double r_nlp = 0;
		double r_svr = 0;
	   double r_cen_lsq = 0;
	   double r_cen_nlp = 0;
	   double r_cen_svr = 0;
	   double r_cen_lsq_temp = 0;
	   double r_cen_nlp_temp = 0;
	   double r_cen_svr_temp = 0;
```

```
double[][] final_output = new double[3][5];
String Input = null;
String LSQ_Steps = null;
String NLP_Steps = null;
String SVR_Steps = null;

    // values to get from LSQ - estimated radius, circularity, axis
    // deflection, time taken
/*
    System.out.println("Enter the radius: ");
    radius = sc.nextDouble();
    System.out.println("Enter the height: ");
    height = sc.nextDouble();
    System.out.println("Enter the number of points per slice: ");
    no_points = sc.nextInt();
    System.out.println("Enter the variance: ");
    variance = sc.nextDouble();
    System.out.println("Enter the number of slices: ");
    no_slices = sc.nextInt(); */

    double[][] x_datapoints = new double[no_points][no_slices];
    double[][] y_datapoints = new double[no_points][no_slices];
    double[] theta = new double[no_points];
    double[] xx = new double[no_points];
    double[] yy = new double[no_points];
    double[][] lsq_results_overall = new double[no_slices][3];
    double[][] nlp_results_overall = new double[no_slices][3];
    double[][] svr_results_overall = new double[no_slices][3];
    double[][] xx_cen_lsq = new double[no_slices][1];
    double[][] yy_cen_lsq = new double[no_slices][1];
    double[][] xx_cen_nlp = new double[no_slices][1];
    double[][] yy_cen_nlp = new double[no_slices][1];
    double[][] xx_cen_svr = new double[no_slices][1];
    double[][] yy_cen_svr = new double[no_slices][1];

    // get the delta angle depending on the number of data points per slice
    // then create the matrix theta that holds all the angular increments
    double delta = 2 * Math.PI / no_points;
    for (int i = 0; i < no_points; i++) {
            theta[i] = i * delta;
    }

    // Generate the data points for x and y
    for (int j = 0; j < no_slices; j++) {
  if (single == 1){
    Input = Input + "\n Slice Number: " + Integer.toString(j+1) + "\n";
```

```
            }
                    for (int i = 0; i < no_points; i++) {
                        x_datapoints[i][j] = radius * Math.cos(theta[i])
                                + Math.sqrt(variance) *
generator.nextGaussian();
                        y_datapoints[i][j] = radius * Math.sin(theta[i])
                                + Math.sqrt(variance) *
generator.nextGaussian();
                        // if it is a single iteration, output input data into a file
                    if (single == 1){
                       Input = Input + Double.toString(x_datapoints[i][j]) + ", " +
Double.toString(y_datapoints[i][j]) + "\n";
                    }
                }
                }

        File Input_File = new File("radius" + Double.toString(radius) + "_var" +
Double.toString(variance) + "_pts"+ Integer.toString(no_points)
+"_slices"+Integer.toString(no_slices)+"_input.csv");
        try {
           nSVR.saveTextFile(Input, Input_File);
        } catch (IOException ex) {
           Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
        }

        // run code to determine least square circularity for cross-section
        OctaveMatrix LSQ_results = lsq_circle.lsq(x_datapoints, y_datapoints, radius,
no_slices);


        // run code to determine NLP circularity for cross-section using SQP.m in
Octave
            OctaveMatrix NLP_results = NLP_Circle.NLP(x_datapoints,
y_datapoints, radius, no_slices);

      // run code to determine nSVR circularity for cross-section using SQP.m in Octave
            OctaveMatrix SVR_results = novelSVR_circle.nSVR(x_datapoints,
y_datapoints, radius, no_slices);

            for (int j = 0; j < no_slices; j++) {
                // assign the data points of a particular cross section to
                // a temporary array for calculation
                for (int i = 0; i < no_points; i++) {
                    xx[i] = x_datapoints[i][j];
                    yy[i] = y_datapoints[i][j];
                }
```

```
                    xx_cen_lsq[j][0] = LSQ_results.get(1,j+1);
                    yy_cen_lsq[j][0] = LSQ_results.get(2,j+1);
                    calc_time_lsq = calc_time_lsq + LSQ_results.get(3,j+1);

                    xx_cen_nlp[j][0] = NLP_results.get(1,j+1);
                    yy_cen_nlp[j][0] = NLP_results.get(2,j+1);
                    calc_time_nlp = calc_time_nlp + NLP_results.get(3,j+1);

                    xx_cen_svr[j][0] = SVR_results.get(1,j+1);
                    yy_cen_svr[j][0] = SVR_results.get(2,j+1);
                    calc_time_svr = calc_time_svr + SVR_results.get(3,j+1);

                // output centerpoints of each cross-section for further analysis
                // if single iteration is selected
                if (single == 1){
                    LSQ_Steps = LSQ_Steps + "LSQ Slice Number: " +
Integer.toString(j+1) + "\n";
                    LSQ_Steps = LSQ_Steps + "x = " + Double.toString(xx_cen_lsq[j][0])
+ ", y = " + Double.toString(yy_cen_lsq[j][0]) + "\n";
                    NLP_Steps = NLP_Steps + "NLP Slice Number: " +
Integer.toString(j+1) + "\n";
                    NLP_Steps = NLP_Steps + "x = " + Double.toString(xx_cen_nlp[j][0])
+ ", y = " + Double.toString(yy_cen_nlp[j][0]) + "\n";
                    SVR_Steps = SVR_Steps + "SVR Slice Number: " +
Integer.toString(j+1) + "\n";
                    SVR_Steps = SVR_Steps + "x = " + Double.toString(xx_cen_svr[j][0])
+ ", y = " + Double.toString(yy_cen_svr[j][0]) + "\n";
                }
                }

            if (single == 1){
                File LSQ_Steps_File = new File("LSQ_Steps.txt");
                File NLP_Steps_File = new File("NLP_Steps.txt");
                File SVR_Steps_File = new File("SVR_Steps.txt");

                try {
                    nSVR.saveTextFile(LSQ_Steps, LSQ_Steps_File);
                } catch (IOException ex) {
                    Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
                }

                try {
                    nSVR.saveTextFile(NLP_Steps, NLP_Steps_File);
                } catch (IOException ex) {
                    Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
```

```
            }

            try {
                nSVR.saveTextFile(SVR_Steps, SVR_Steps_File);
            } catch (IOException ex) {
                Logger.getLogger(nSVR.class.getName()).log(Level.SEVERE, null, ex);
            }
        }


            // run these datapoints through their respective methods(because there are
only a few number of centerpoints)
            OctaveMatrix lsq_results_final = lsq_circle.lsq(xx_cen_lsq, yy_cen_lsq,
0.1,1);
            OctaveMatrix nlp_results_final = NLP_Circle.NLP(xx_cen_nlp,
yy_cen_nlp, 0.1,1);
            OctaveMatrix svr_results_final = novelSVR_circle.nSVR(xx_cen_svr,
yy_cen_svr, 0.1,1);

        // Calculate the axis deflection by getting the furthest center point from the axis
center
        for (int i = 0; i < no_slices; i++){
            r_cen_lsq_temp = Math.sqrt(Math.pow(lsq_results_final.get(1) -
xx_cen_lsq[i][0],2) +
                    Math.pow(lsq_results_final.get(2) - yy_cen_lsq[i][0],2));
            r_cen_nlp_temp = Math.sqrt(Math.pow(nlp_results_final.get(1) -
xx_cen_nlp[i][0],2) +
                    Math.pow(nlp_results_final.get(2) - yy_cen_nlp[i][0], 2));
            r_cen_svr_temp = Math.sqrt(Math.pow(svr_results_final.get(1) -
xx_cen_svr[i][0],2) +
                    Math.pow(svr_results_final.get(2) - yy_cen_svr[i][0], 2));
            if (r_cen_lsq_temp > r_cen_lsq){
                r_cen_lsq = r_cen_lsq_temp;
            }
            if (r_cen_nlp_temp > r_cen_nlp){
                r_cen_nlp = r_cen_nlp_temp;
            }
            if (r_cen_svr_temp > r_cen_svr){
                r_cen_svr = r_cen_svr_temp;
            }
        }

            // calculate min and max deviations for each method against data points to
determine
            // circularity
            for (int i = 0; i < no_slices; i++){
```

```
                    for (int j = 1; j < no_points; j++){
                        r_lsq = Math.sqrt(Math.pow(x_datapoints[j][i] -
lsq_results_final.get(1), 2) +
                                Math.pow(y_datapoints[j][i] -
lsq_results_final.get(2), 2));
                        r_nlp = Math.sqrt(Math.pow(x_datapoints[j][i] -
nlp_results_final.get(1), 2) +
                                Math.pow(y_datapoints[j][i] -
nlp_results_final.get(2), 2));
                        r_svr = Math.sqrt(Math.pow(x_datapoints[j][i] -
svr_results_final.get(1), 2) +
                                Math.pow(y_datapoints[j][i] -
svr_results_final.get(2), 2));

                        if (rmax_temp_lsq < r_lsq){
                            rmax_temp_lsq = r_lsq;
                        }
                        if (rmax_temp_nlp < r_nlp){
                            rmax_temp_nlp = r_nlp;
                        }
                        if (rmax_temp_svr < r_svr){
                            rmax_temp_svr = r_svr;
                        }
                        if (rmin_temp_lsq > r_lsq){
                            rmin_temp_lsq = r_lsq;
                        }
                        if (rmin_temp_nlp > r_nlp){
                            rmin_temp_nlp = r_nlp;
                        }
                        if (rmin_temp_svr > r_svr){
                            rmin_temp_svr = r_svr;
                        }
                    }
                }

                // print to screen calculated x-y coordinates
                final_output[0][0] = lsq_results_final.get(1);
                final_output[1][0] = nlp_results_final.get(1);
                final_output[2][0] = svr_results_final.get(1);
                final_output[0][1] = lsq_results_final.get(2);
                final_output[1][1] = nlp_results_final.get(2);
                final_output[2][1] = svr_results_final.get(2);

                // print to screen cylindricity
                final_output[0][2] = rmax_temp_lsq - rmin_temp_lsq;
                final_output[1][2] = rmax_temp_nlp - rmin_temp_nlp;
```

```
                final_output[2][2] = rmax_temp_svr - rmin_temp_svr;

                // print to screen axis deflection
                final_output[0][3] = r_cen_lsq;
                final_output[1][3] = r_cen_nlp;
            final_output[2][3] = r_cen_svr;

                // print to screen calculation time
                final_output[0][4] = lsq_results_final.get(3)+calc_time_lsq;
                final_output[1][4] = nlp_results_final.get(3)+calc_time_nlp;
                final_output[2][4] = svr_results_final.get(3)+calc_time_svr;

            return final_output;
        }

}
```

APPENDIX C: LSQ CIRCULARITY MODULE

<filename: lsq_circle.java>

```
package my.nSVR;

import dk.ange.octave.*;
import dk.ange.octave.type.*;

public class lsq_circle {
	public static OctaveMatrix lsq(double[][] x, double[][] y, double radius, int
no_slices){
		OctaveScalar o_radius = new OctaveScalar(radius);
		OctaveScalar no_points = new OctaveScalar(x.length);
	    OctaveScalar slices = new OctaveScalar(no_slices);
		final OctaveEngine lsq_calc = new
OctaveEngineFactory().getScriptEngine();

		// to prevent broken pipe errors due to how java streams interact with
Octave,
		// the following code is used to break down the arrays
		for (int i = 0; i < x.length; i++){
	      for (int j = 0; j < no_slices; j++){
			OctaveScalar x_ij = new OctaveScalar(x[i][j]);
			OctaveScalar y_ij = new OctaveScalar(y[i][j]);

			lsq_calc.put("x", x_ij);
			lsq_calc.put("y", y_ij);
			lsq_calc.eval("x_total(" + Integer.toString(i+1) + "," +
Integer.toString(j+1) + ") = x;\n");
			lsq_calc.eval("y_total(" + Integer.toString(i+1) + "," +
Integer.toString(j+1) + ") = y;\n");
		}
		}

		lsq_calc.put("radius", o_radius);
		lsq_calc.put("no_points", no_points);
	    lsq_calc.put("slices", slices);

		// Run least square circle analysis on the cross-sections
		final String LSQ_Command1 = ""
		+ "for i=1:slices\n"
			+ "Pts = [x_total(:,i) y_total(:,i)];\n"
			+ "t0 = time;\n"
			+ "[ex er]=ls2dcircle(Pts, [0 0]', radius, 0.0001, 0.0001);\n"
			+ "t1 = time;\n"
```

```
                    + "calc_time = t1 - t0;\n"
                    + "results(:,i) = [ex(1); ex(2); calc_time; er];\n"
                + "end\n"
                    + "";

                lsq_calc.eval(LSQ_Command1);
                OctaveMatrix o_output = lsq_calc.get("results");
                lsq_calc.destroy();
                return o_output;
        }
}
```

APPENDIX D: NLP CIRCULARITY MODULE SOURCE CODE

<filename: NLP_Circle.java>

```
package my.nSVR;

import dk.ange.octave.*;
import dk.ange.octave.type.*;

public class NLP_Circle {
	public static OctaveMatrix NLP(double[][] x, double[][] y, double radius, int
no_slices){
		OctaveScalar o_radius = new OctaveScalar(radius);
		OctaveScalar no_points = new OctaveScalar(x.length);
	    OctaveScalar slices = new OctaveScalar(no_slices);

		final OctaveEngine nlp_calc = new
OctaveEngineFactory().getScriptEngine();
		nlp_calc.put("radius", o_radius);
		nlp_calc.put("points", no_points);
	    nlp_calc.put("slices", slices);

		// to prevent broken pipe errors due to how java streams interact with
Octave,
		// the following code is used to break down the arrays
		for (int i = 0; i < x.length; i++){
	      for (int j = 0; j < no_slices; j++){
			OctaveScalar x_i = new OctaveScalar(x[i][j]);
			OctaveScalar y_i = new OctaveScalar(y[i][j]);

			nlp_calc.put("x", x_i);
			nlp_calc.put("y", y_i);
			nlp_calc.eval("x_total(" + Integer.toString(i+1) + "," +
Integer.toString(j+1) + ") = x;\n");
			nlp_calc.eval("y_total(" + Integer.toString(i+1) + "," +
Integer.toString(j+1) + ") = y;\n");
		}
		  }

		// establish initial guesses for the variables, initialize global variables
		nlp_calc.eval("x0 = [radius; radius; 0; 0];");
		nlp_calc.eval("global xx yy no_points;");

		// set objective function
		final String obj_func = ""
			+ "function f = phi(x)\n"
```

```
                + "f = x(1) - x(2);\n"
                + "endfunction\n"
                + "";

        // set inequality constraints
        final String ineq_constraints = ""
                + "function c = h(x)\n"
                + "global xx yy no_points;\n"
                + "coeff = [ones(no_points,1) ones(no_points,1) -2*xx -2*yy];\n"
                + "variables = ([x(3)^2 x(4)^2 x(3) x(4)]'*ones(1,no_points))' ;\n"
                + "x1 = x(1)^2*ones(no_points,1);\n"
                + "x2 = x(2)^2*ones(no_points,1);\n"
                + "c_1 = -x2 + sum(variables.*coeff,2) + xx.^2 + yy.^2;\n"
                + "c_2 = x1 - sum(variables.*coeff,2) - xx.^2 - yy.^2;\n"
                + "c = [c_1; c_2];\n"
                /* + "for i = 1:no_points\n"
                + "c(i, 1) = (xx(i) - x(3))^2 + (yy(i) - x(4))^2 - x(2)^2;\n"
                + "c(i + no_points, 1) = x(1)^2 - (xx(i) - x(3))^2 - (yy(i) - x(4))^2;\
n"
                + "end\n" */
                + "endfunction\n"
                + "";

        // String that runs the Sequential Quadratic Program solver
        // also measures computing time
        final String SQP_run1 = ""
        + "for i = 1:slices\n"
                + "xx = x_total(:,i);\n"
                + "yy = y_total(:,i);\n"
                + "no_points = points;\n"
                + "t0 = time;\n"
        + "[x_result, obj, info, iter, nf, lambda] = sqp(x0, @phi, [], @h);\n"
                + "t1 = time;\n"
                + "calc_time = t1 - t0;\n"
                + "sr = (x_result(1) + x_result(2)) / 2;\n"
                + "results(:,i) = [x_result(3); x_result(4); calc_time; sr];\n"
        + "end"
                + "";

        // Run the pre-set Strings in Octave
        nlp_calc.eval(obj_func);
        nlp_calc.eval(ineq_constraints);
        nlp_calc.eval(SQP_run1);

        // get results from Octave, close the engine, return the output
        OctaveMatrix output = nlp_calc.get("results");
```

```
                nlp_calc.destroy();
                return output;
        }
}
```

APPENDIX E: SVR CIRCULARITY MODULE

<filename: novelSVR_circle.java>

```
package my.nSVR;

import dk.ange.octave.*;
import dk.ange.octave.type.*;

public class novelSVR_circle {
	public static OctaveMatrix nSVR(double[][] x, double[][] y, double radius, int
no_slices){
		OctaveScalar o_radius = new OctaveScalar(radius);
		OctaveScalar no_points = new OctaveScalar(x.length);
	    OctaveScalar slices = new OctaveScalar(no_slices);

		final OctaveEngine nSVR_calc = new
OctaveEngineFactory().getScriptEngine();
		nSVR_calc.put("radius", o_radius);
		nSVR_calc.put("points", no_points);
	    nSVR_calc.put("slices", slices);

		// to prevent broken pipe errors due to how java streams interact with
Octave,
		// the following code is used to break down the arrays
		for (int i = 0; i < x.length; i++){
	      for (int j = 0; j < no_slices; j++){
			OctaveScalar x_i = new OctaveScalar(x[i][j]);
			OctaveScalar y_i = new OctaveScalar(y[i][j]);

			nSVR_calc.put("x", x_i);
			nSVR_calc.put("y", y_i);
			nSVR_calc.eval("x_total(" + Integer.toString(i+1) + "," +
Integer.toString(j+1) + ") = x;\n");
			nSVR_calc.eval("y_total(" + Integer.toString(i+1) + "," +
Integer.toString(j+1) + ") = y;\n");
		}
		  }

		// set objective function
		final String obj_func = ""
			+ "function f = phi(x)\n"
			+ "f = 2/x(1);\n"
			+ "f = 1/f;\n"
			+ "endfunction\n"
			+ "";
```

```
                // set inequality constraints
                final String ineq_constraints = ""
                        + "function c = h(x)\n"
                        + "global xx yy no_points;\n"
                        + "higher_dim = [(xx.^2) xx (yy.^2) (yy)];\n"
                        + "variables = ([x(1) x(2) x(1) x(3)]'*ones(1,no_points))';\n"
                        + "x4 = x(4)*ones(no_points,1);\n"
                        + "c_1 = sum(variables.*higher_dim,2) + x4 + ones(no_points,1);\
n"
                        + "c_2 = ones(no_points,1) - sum(variables.*higher_dim,2) - x4;\n"
                        + "c = [c_1; c_2];\n"
                        /* + "for i = 1:no_points\n"
                        + "c(i,1)=(x(1)*xx(i)*xx(i)+x(2)*xx(i)+x(1)*yy(i)*yy(i)
+x(3)*yy(i)) + x(4) + 1;\n"
                    + "c(i+no_points,1)=1 - (x(1)*xx(i)*xx(i)+x(2)*xx(i)
+x(1)*yy(i)^2+x(3)*yy(i))- x(4);\n"
                    + "end\n" */
                        + "endfunction\n"
                        + "";

                // String that runs the Sequential Quadratic Program solver
                // also measures computing time
                final String nSVR_run1 = ""
                + "for i = 1:slices\n"
                        + "global xx yy no_points;\n"
                        + "no_points = points;"
                        + "xx = x_total(:,i);\n"
                        + "yy = y_total(:,i);\n"
                        + "t0 = time;\n"
                        + "[sx, obj, info, iter, nf, lambda] = sqp([0.1; 0.1; 0.1; 8], @phi, [],
@h);\n"
                        + "t1 = time;\n"
                        + "calc_time = t1 - t0;\n"
                        + "xx_cen = -sx(2) / 2 / sx(1);\n"
                        + "yy_cen = -sx(3) / 2 / sx(1);\n"
                        + "sr = sqrt(-sx(4)/sx(1)+1/4*( sx(2)^2+sx(3)^2 )/sx(1));\n"
                        + "if (-sx(4)/sx(1)+1/4*( sx(2)^2+sx(3)^2 )/sx(1) < 0)\n"
                        + "sr = 0;\n"
                        + "end\n"
                        + "results(:,i) = [xx_cen; yy_cen; calc_time; sr];\n"
                + "end\n"
                        + "";


                // Run the pre-set Strings in Octave
```

```
                nSVR_calc.eval(obj_func);
                nSVR_calc.eval(ineq_constraints);
                nSVR_calc.eval(nSVR_run1);

                OctaveMatrix output = nSVR_calc.get("results");
                nSVR_calc.destroy();
                return output;
        }
}
```

APPENDIX F: GENERATED RESULTS – LSQ FAILURES OMITTED

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
25	0.01	6	0.58	0.61	0.61	0.97	1.89	2.16
25	0.01	6	0.68	0.64	0.65	0.28	0.88	2.09
25	0.01	7	0.43	0.45	0.48	0.42	1	2.63
25	0.01	7	0.67	0.65	0.64	0.28	1.09	2.53
25	0.01	8	0.56	0.57	0.57	0.3	1.08	2.41
25	0.01	9	0.54	0.54	0.54	0.33	1.19	3.3
25	0.01	9	0.59	0.59	0.59	0.39	1.27	3.08
25	0.01	10	0.54	0.64	0.64	0.41	1.44	3.56
25	0.01	10	0.59	0.59	0.57	0.73	1.86	3.23
25	0.01	11	0.57	0.57	0.57	0.36	1.48	4.11
25	0.01	11	0.59	0.64	0.63	0.34	1.23	3.27
25	0.01	12	0.56	0.58	0.58	0.63	1.77	4
25	0.01	12	0.56	0.57	0.56	0.5	1.52	4.19
25	0.01	13	0.53	0.68	0.62	0.39	1.67	4.2
25	0.01	13	0.69	0.64	0.66	0.5	1.8	4
25	0.01	14	0.56	0.6	0.6	0.39	1.56	4.84
25	0.01	14	0.6	0.58	0.61	0.42	1.84	4.48
25	0.01	15	0.53	0.54	0.54	0.69	1.92	4.36
25	0.01	15	0.64	0.62	0.62	0.47	1.84	4.81
25	0.11	6	1.43	1.42	1.42	0.27	0.98	2.02
25	0.11	6	1.55	1.59	1.59	0.31	1.02	1.39
25	0.11	7	2.07	1.84	1.89	0.31	1.28	1.52
25	0.11	8	2.08	2.13	2.12	0.34	1.34	1.91
25	0.11	8	1.72	1.97	1.73	0.33	1.23	1.63
25	0.11	9	1.56	1.46	1.49	0.41	1.61	2.45
25	0.11	9	1.85	2.02	1.89	0.39	1.41	2.16
25	0.11	10	1.67	1.58	1.59	0.41	1.63	2.3
25	0.11	10	1.98	1.89	1.89	0.36	1.63	2.44
25	0.11	11	1.95	1.96	1.87	0.48	1.86	2.59
25	0.11	11	1.67	1.66	1.64	0.34	1.67	2.27
25	0.11	12	2.01	2.15	2.1	0.39	1.63	2.58
25	0.11	12	1.94	1.9	1.9	0.42	1.75	2.88
25	0.11	13	1.66	1.68	1.73	0.42	1.95	3.03
25	0.11	13	1.99	1.99	2.01	0.41	1.89	2.91
25	0.11	14	2.14	1.93	1.93	0.44	2.06	2.91
25	0.11	14	1.8	1.94	1.94	0.42	1.94	3.41
25	0.11	15	2.34	2.23	2.25	0.5	1.88	3.31
25	0.11	15	2.24	2.17	2.19	0.44	2.17	3.77
25	0.21	6	2.88	2.81	2.89	0.45	1.75	1.02
25	0.21	6	2.53	2.46	2.55	0.3	1.23	1.17
25	0.21	7	3.03	2.7	2.8	0.31	1.36	1.42

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
25	0.21	7	2.44	2.36	2.36	0.31	1.59	0.92
25	0.21	8	2.58	2.64	2.64	0.38	1.31	1.31
25	0.21	8	2.78	2.48	2.48	0.3	1.23	1.16
25	0.21	9	2.44	2.53	2.54	0.41	1.69	1.34
25	0.21	9	2.43	2.34	2.33	0.44	1.41	1.23
25	0.21	10	2.54	2.66	2.59	0.39	1.67	1.42
25	0.21	10	2.6	2.65	2.61	0.41	1.75	1.42
25	0.21	11	2.73	2.76	2.78	0.38	1.8	1.66
25	0.21	11	2.83	2.78	2.8	0.34	1.61	1.48
25	0.21	12	3.18	3.07	3.07	0.66	1.7	1.27
25	0.21	12	2.46	2.43	2.44	0.39	1.69	1.31
25	0.21	13	3.01	3.1	3.01	0.42	2.08	1.78
25	0.21	13	3.18	3.08	3.03	0.41	1.92	1.58
25	0.21	14	2.47	2.59	2.63	0.47	1.97	1.38
25	0.21	14	2.78	2.74	2.77	0.52	2.09	1.89
25	0.21	15	2.45	2.63	2.67	0.36	2.06	1.98
25	0.21	15	2.76	2.73	2.8	0.44	2.25	1.73
25	0.31	6	2.92	2.53	2.49	0.3	1.22	2.61
25	0.31	6	2.85	2.79	2.75	0.33	1.06	1.11
25	0.31	7	3.6	3.94	3.5	0.39	1.52	1.31
25	0.31	7	3.26	3.04	3.04	0.36	1.2	1
25	0.31	8	2.88	2.82	2.74	0.38	1.3	1.14
25	0.31	8	3.29	3.3	3.28	0.31	1.3	1
25	0.31	9	3.51	5.4	3.49	0.39	1.5	1.33
25	0.31	9	3.11	3.37	3.48	0.42	1.45	1.17
25	0.31	10	3.17	3.25	3.26	0.42	1.72	1.59
25	0.31	10	3	3.25	3.04	0.34	1.5	1.33
25	0.31	11	2.89	3.13	3.07	0.41	1.94	1.75
25	0.31	11	3.05	3.04	2.98	0.34	1.61	1.36
25	0.31	12	2.86	2.74	2.87	0.36	1.73	1.53
25	0.31	12	3.06	3.25	3.16	0.39	1.89	1.45
25	0.31	13	3.69	3.49	3.55	0.44	2.03	1.48
25	0.31	13	3.01	3.13	2.93	0.41	1.78	1.52
25	0.31	14	3.53	3.49	3.56	0.45	2.05	1.45
25	0.31	14	3.33	3.25	3.25	0.42	1.92	1.44
25	0.31	15	3.24	3.44	3.44	0.38	1.91	1.44
25	0.31	15	3.56	3.64	3.61	0.47	2.06	2.05
25	0.41	6	3.62	3.7	3.4	0.42	1.5	1.09
25	0.41	6	3.34	3.75	3.34	0.28	1.27	0.98
25	0.41	7	3.05	3.11	3.02	0.33	1.16	1.17
25	0.41	7	3.07	3.06	3.05	0.33	1.39	0.95
25	0.41	8	4.72	4.69	4.69	0.41	1.36	1.17
25	0.41	8	4.22	4.06	4.06	0.33	1.39	1

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
25	0.41	9	3.02	3.15	3.15	0.38	1.45	1.14
25	0.41	9	3.57	3.42	3.42	0.38	1.42	1.16
25	0.41	10	3.38	3.38	3.39	0.39	1.72	1.34
25	0.41	10	3.71	3.88	3.88	0.38	1.39	1.16
25	0.41	11	3.64	3.47	3.47	0.41	1.83	2
25	0.41	11	3.49	3.4	3.4	0.38	1.52	1.13
25	0.41	12	3.26	3.74	3.29	0.44	2	1.53
25	0.41	13	3.94	4.25	4.14	0.45	1.83	1.8
25	0.41	13	3.57	3.51	3.58	0.42	2.05	1.66
25	0.41	14	3.5	3.56	3.51	0.56	2.3	1.61
25	0.41	14	4.09	3.92	3.96	0.42	1.97	1.66
25	0.41	15	3.95	4.12	4	0.41	2.11	1.48
25	0.41	15	3.58	3.48	3.6	0.5	2.14	1.92
25	0.51	6	3.39	3.61	4.04	0.28	1.45	1.31
25	0.51	6	3.74	3.59	3.5	0.28	0.97	0.73
25	0.51	7	3.86	4.51	3.87	0.34	1.36	1.06
25	0.51	7	3.62	3.61	3.61	0.48	1.25	1.03
25	0.51	8	3.64	3.68	3.76	0.38	1.44	1.08
25	0.51	8	4.02	3.65	3.68	0.34	1.5	1.08
25	0.51	9	3.93	3.79	3.66	0.36	1.48	1.2
25	0.51	9	3.51	3.51	3.55	0.38	1.39	1.03
25	0.51	10	3.95	6.7	4.39	0.39	2.16	1.78
25	0.51	10	3.63	3.62	3.7	0.38	2.06	1.39
25	0.51	11	4.56	4.42	4.21	0.56	1.86	1.7
25	0.51	11	3.79	3.66	3.66	0.39	1.69	1.27
25	0.51	12	4.14	4.78	4.23	0.42	2.09	1.66
25	0.51	12	6.02	4.49	4.47	0.53	1.97	1.45
25	0.51	13	4.53	4.53	4.53	0.44	1.98	1.72
25	0.51	13	4.54	4.48	4.52	0.52	2.06	1.77
25	0.51	14	4.54	4.24	4.9	0.47	2.3	2.08
25	0.51	14	4	3.7	3.69	0.44	2.05	1.5
25	0.51	15	3.99	4.22	4.2	0.42	2.02	1.69
25	0.51	15	3.86	22.52	3.88	0.69	2.7	1.67
25	0.61	6	5.07	5.25	5.32	0.3	1.16	0.95
25	0.61	6	3.94	4.06	4.16	0.34	1.03	0.89
25	0.61	7	4.05	4.47	4.1	0.39	1.58	1.38
25	0.61	7	3.92	3.77	3.96	0.36	1.27	1
25	0.61	8	3.66	3.85	3.85	0.34	1.31	1.02
25	0.61	8	4.01	3.61	3.79	0.39	1.53	1.44
25	0.61	9	4.78	4.88	4.82	0.38	1.75	1.34
25	0.61	9	5.19	6.7	5.75	0.34	1.53	1.34
25	0.61	10	4.86	4.48	4.49	0.56	1.8	1.39
25	0.61	10	4.24	4.28	4.25	0.39	1.92	1.89

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
25	0.61	11	5.28	5.57	5.37	0.42	1.86	1.22
25	0.61	11	3.58	3.64	3.7	0.39	1.66	1.33
25	0.61	12	4.52	4.26	4.47	0.52	2.05	1.48
25	0.61	12	5.42	5.22	5.21	0.44	2.02	1.38
25	0.61	13	4.55	4.57	4.53	0.42	2.2	2.2
25	0.61	13	4.51	5.36	5.31	0.48	2.22	1.78
25	0.61	14	4.73	4.78	4.78	0.47	2.48	1.64
25	0.61	14	4.45	4.44	4.28	0.42	2.06	1.8
25	0.61	15	5.27	5.33	5.26	0.42	2.03	1.69
25	0.61	15	5.85	5.42	5.53	0.52	2.7	2.72
25	0.71	6	3.99	4.58	4.58	0.3	1.11	0.95
25	0.71	6	5.04	4.85	4.71	0.3	1.19	1.05
25	0.71	7	4.01	4.25	4.05	0.39	1.19	1.08
25	0.71	7	4.04	4.79	4.69	0.39	1.3	1.17
25	0.71	8	4.41	22.55	4.72	0.34	2.11	1.16
25	0.71	8	5.58	5.81	5.82	0.42	1.69	1.72
25	0.71	9	4.42	4.75	4.31	0.36	1.81	1.55
25	0.71	9	4.21	4.51	4.51	0.39	1.5	1.17
25	0.71	10	5.47	5.47	5.46	0.47	1.88	1.67
25	0.71	10	4.56	4.35	4.38	0.39	1.67	1.33
25	0.71	11	6.96	4.8	5.01	0.47	1.91	1.58
25	0.71	11	5.21	4.54	4.49	0.48	1.69	1.36
25	0.71	12	5.62	5.56	5.7	2.03	2.55	1.97
25	0.71	12	4.35	4.53	4.78	0.52	1.94	1.52
25	0.71	13	4.94	4.78	4.88	0.48	2.02	1.8
25	0.71	13	5.52	5.26	5.22	0.41	2.16	1.59
25	0.71	14	5.51	5.3	4.94	0.47	2.22	2.08
25	0.71	14	5.38	5.57	5.2	0.45	2.61	1.7
25	0.71	15	5.64	6.23	6.19	0.44	2.31	1.69
25	0.71	15	4.53	4.69	4.69	0.81	2.97	2.13
25	0.81	6	3.99	3.96	4.07	0.3	1.08	1.08
25	0.81	6	5.49	5.5	5.5	0.3	1.22	0.86
25	0.81	7	4.11	4.3	4.41	0.38	1.45	1.23
25	0.81	7	4.13	4.15	4.02	0.33	1.13	0.83
25	0.81	8	4.69	4.57	4.66	0.33	1.48	1.08
25	0.81	8	4.92	4.92	4.9	0.34	1.83	1.36
25	0.81	9	5.93	5.53	5.53	0.41	1.64	1.28
25	0.81	9	5.59	5.79	5.45	0.34	1.48	1.19
25	0.81	10	4.69	4.61	4.6	0.42	1.86	1.64
25	0.81	10	4.71	4.62	4.5	0.41	1.63	1.52
25	0.81	11	5.12	4.71	4.89	0.45	1.91	1.64
25	0.81	11	5.62	6.07	5.73	0.39	2.08	1.48
25	0.81	12	5.12	4.49	4.82	0.45	2.09	2.52

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
25	0.81	12	5.75	5.75	5.5	0.45	2.23	1.89
25	0.81	13	4.81	4.7	5.03	0.42	2.3	1.92
25	0.81	13	4.52	4.55	4.49	0.45	2.05	1.67
25	0.81	14	4.86	4.97	4.8	0.47	2.36	1.64
25	0.81	14	5.06	5.06	4.93	0.45	2.33	1.95
25	0.81	15	5.75	5.55	5.5	0.47	2	1.88
25	0.81	15	4.52	4.96	4.61	0.83	3.34	2.33
25	0.91	6	4.39	4.45	4.45	0.27	1.33	0.81
25	0.91	6	5.48	5.27	5.38	0.28	1.19	0.98
25	0.91	7	5.23	5.48	5.2	0.33	1.95	1.13
25	0.91	7	3.98	3.95	3.92	0.33	1.55	1.22
25	0.91	8	5.07	22.09	5.31	0.36	1.5	1.28
25	0.91	8	4.82	4.77	4.73	0.34	1.55	1.22
25	0.91	9	5.89	5.74	5.81	0.41	1.66	1.19
25	0.91	9	5.16	5.59	5.33	0.38	1.55	1.25
25	0.91	10	5.32	5.09	5.08	0.41	1.95	1.72
25	0.91	10	4.63	4.7	4.65	0.41	1.8	1.2
25	0.91	11	5.31	5.26	5.44	0.41	2.42	1.73
25	0.91	11	5.15	6.64	5.23	0.39	1.86	1.55
25	0.91	12	4.74	5.11	4.74	0.59	2.45	1.53
25	0.91	12	5.39	5.43	5.41	0.41	1.8	1.36
25	0.91	13	4.82	5.02	4.95	0.5	2.13	1.72
25	0.91	13	4.46	4.85	4.57	0.5	2.11	1.59
25	0.91	14	6.68	7.26	7.07	0.53	2.42	1.8
25	0.91	14	5.12	4.93	4.93	0.5	2.03	1.64
25	0.91	15	5.55	5.64	6.54	0.48	2.17	1.63
25	0.91	15	5.82	6.22	5.71	0.47	2.77	2.09
25	1.01	6	6.3	5.39	5.62	0.28	1.25	1.2
25	1.01	6	6.11	5.42	5.42	0.31	1.08	0.92
25	1.01	7	5.37	5.16	5.59	0.31	1.7	1.31
25	1.01	7	4.73	4.71	4.72	0.31	1.39	1.33
25	1.01	8	4.99	5.89	5.4	0.39	1.53	1.22
25	1.01	8	6.17	6.55	6.28	0.33	1.58	1.34
25	1.01	9	5.99	5.77	6.37	0.39	1.77	1.38
25	1.01	9	4.73	7.19	4.62	0.45	1.61	1.36
25	1.01	10	6.1	6.67	6.73	0.38	1.69	1.56
25	1.01	10	5.88	5.54	5.54	0.36	2.06	1.39
25	1.01	11	5.59	5.52	5.75	0.48	1.94	1.75
25	1.01	11	6.9	6.74	6.72	0.66	2.02	1.52
25	1.01	12	5.68	6.2	5.56	0.42	1.94	1.73
25	1.01	12	5.28	5.96	5.31	0.42	2.13	1.88
25	1.01	13	6.61	6.42	6.51	0.45	2.05	1.44
25	1.01	13	6.24	6.45	6.37	0.47	2.19	1.88

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
25	1.01	14	6.02	5.74	5.93	0.48	2.39	1.94
25	1.01	14	6.1	6.7	5.7	0.47	2.31	1.94
25	1.01	15	5.71	5.82	5.85	0.47	2.16	1.81
25	1.01	15	5.11	5.13	5.41	0.56	2.55	1.59
50	0.01	6	0.59	0.58	0.58	0.25	1.66	3
50	0.01	6	0.62	0.6	0.6	0.27	1.25	2.86
50	0.01	7	0.56	0.55	0.55	0.31	1.52	3.67
50	0.01	7	0.59	0.61	0.59	0.33	1.59	3.42
50	0.01	8	0.53	0.53	0.52	0.34	1.73	4.16
50	0.01	8	0.67	0.67	0.67	0.36	1.92	3.88
50	0.01	9	0.53	0.57	0.53	0.33	1.95	4.47
50	0.01	9	0.53	0.61	0.58	0.34	1.86	4.5
50	0.01	10	0.56	0.61	0.61	0.36	2.03	5.25
50	0.01	11	0.58	0.61	0.6	0.41	2.53	5.47
50	0.01	11	0.64	0.63	0.63	0.34	2.02	4.98
50	0.01	12	0.69	0.67	0.67	0.39	2.23	5.63
50	0.01	12	0.55	0.55	0.55	0.38	2.22	5.61
50	0.01	13	0.58	0.6	0.6	0.41	2.7	6.34
50	0.01	13	0.68	0.75	0.7	0.38	2.39	5.84
50	0.01	14	0.58	0.62	0.57	0.44	2.78	6.69
50	0.01	14	0.61	0.62	0.61	0.38	3.44	5.88
50	0.01	15	0.6	0.63	0.64	0.38	2.64	6.38
50	0.01	15	0.65	0.63	0.63	0.61	3.89	8.28
50	0.11	6	1.7	1.7	1.7	0.5	1.31	1.42
50	0.11	6	1.86	2.03	2.13	0.25	1.61	1.77
50	0.11	7	2.04	2.03	2.04	0.28	1.8	1.78
50	0.11	7	1.98	2.03	1.92	0.36	1.67	1.97
50	0.11	8	2.1	2.11	2.13	0.39	1.92	2
50	0.11	8	1.87	1.79	1.81	0.25	1.84	1.3
50	0.11	9	2.13	2.13	2.13	0.34	2.36	2.45
50	0.11	9	1.82	1.97	1.97	0.34	1.91	2.06
50	0.11	10	2.03	2.06	2.06	0.5	2.22	2.39
50	0.11	10	2.18	2.44	2.04	0.36	2.16	2.08
50	0.11	11	1.91	2	1.89	0.38	2.75	2.81
50	0.11	11	1.91	2.06	2.07	0.34	2.06	2.22
50	0.11	12	2.09	2.04	2.06	0.38	2.55	2.48
50	0.11	12	1.99	1.83	1.87	0.42	2.55	2.2
50	0.11	13	1.98	2	2.05	0.41	3.39	3.59
50	0.11	13	2.15	2.15	2.15	0.42	2.56	2.27
50	0.11	14	2.18	2.22	2.22	0.45	3.11	3.2
50	0.11	14	2	1.89	1.89	0.44	2.83	2.98
50	0.11	15	2.26	2.2	2.17	0.39	2.72	3.52
50	0.11	15	2.19	2.43	2.21	0.56	3.16	3.41

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
50	0.21	6	2.62	2.45	2.43	0.33	1.53	1.45
50	0.21	6	3.34	3.34	3.35	0.47	1.56	1.17
50	0.21	7	3.01	2.83	2.85	0.36	1.56	1.63
50	0.21	8	2.45	2.62	2.56	0.36	1.83	1.66
50	0.21	8	3.06	2.9	2.92	0.3	1.86	1.56
50	0.21	9	2.69	2.83	2.83	0.45	2.39	1.92
50	0.21	9	3.09	3.63	3.03	0.36	2.45	1.84
50	0.21	10	2.6	2.61	2.69	0.39	2.64	2.56
50	0.21	10	2.84	2.78	2.82	0.39	2.3	1.8
50	0.21	11	2.83	2.84	2.85	0.36	2.61	2.28
50	0.21	11	2.31	2.48	2.48	0.34	2.34	1.94
50	0.21	12	3.1	2.94	2.94	0.34	2.67	2.36
50	0.21	12	2.57	2.48	2.54	0.55	2.63	2.2
50	0.21	13	2.54	2.63	2.63	0.45	3.28	2.97
50	0.21	13	3.27	3.14	3.29	0.45	3.16	2.67
50	0.21	14	2.97	3.38	2.96	0.42	3.97	3.27
50	0.21	14	2.95	3.06	3.06	0.39	2.89	2.36
50	0.21	15	3.44	3.37	3.45	0.41	3.19	2.36
50	0.21	15	2.71	2.78	2.73	0.42	3.02	2.33
50	0.31	6	3.09	2.97	2.94	0.38	1.36	1.39
50	0.31	6	11.56	3.4	3.25	0.3	1.33	1.23
50	0.31	7	3.38	3.54	3.53	0.38	1.95	1.83
50	0.31	7	2.86	2.82	3.02	0.34	2.05	1.63
50	0.31	8	3.53	3.57	3.56	0.33	2.03	1.64
50	0.31	8	2.98	3.06	2.89	0.36	2.08	1.58
50	0.31	9	3.11	3.78	3.54	0.36	2.38	1.78
50	0.31	9	3.29	3.24	3.27	0.38	2.13	2.42
50	0.31	10	3.32	3.32	3.21	0.39	2.94	2.13
50	0.31	10	4.23	3.98	3.98	0.36	2.3	2.38
50	0.31	11	3.04	3.08	3.01	0.5	2.67	2.56
50	0.31	11	3.33	3.25	3.25	0.33	2.38	1.91
50	0.31	12	3.35	3.26	3.32	0.41	2.42	1.58
50	0.31	12	3.76	3.69	4.02	0.42	3	2.48
50	0.31	13	3.36	3.24	3.31	0.47	2.97	2.45
50	0.31	13	4.05	4.14	4.14	0.44	3	2.69
50	0.31	14	3.78	4.2	3.85	0.48	3.38	2.89
50	0.31	14	3.37	3.33	3.31	0.44	3.31	2.2
50	0.31	15	3.42	3.72	3.46	0.39	2.88	2.34
50	0.31	15	3.34	4.16	3.39	0.41	3.5	2.67
50	0.41	6	3.92	3.91	3.93	0.31	1.5	1.7
50	0.41	6	4.02	4.09	3.71	0.27	1.45	1.77
50	0.41	7	3.48	3.48	3.42	0.28	1.81	1.59
50	0.41	7	3.19	3.23	3.15	0.34	1.91	1.55

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
50	0.41	8	4.18	3.93	3.8	0.34	2	1.61
50	0.41	8	3.75	3.71	3.73	0.36	2.03	1.84
50	0.41	9	3.31	3.17	3.19	0.39	2.8	1.95
50	0.41	9	3.6	3.25	3.25	0.33	2.28	1.86
50	0.41	10	3.83	3.72	3.75	0.42	2.7	2.23
50	0.41	10	4.07	3.69	3.7	0.36	2.81	2.69
50	0.41	11	3.62	3.81	3.71	0.53	2.98	2.23
50	0.41	11	3.98	4.21	4.03	0.34	2.81	2.08
50	0.41	12	4.5	4.61	4.51	0.42	2.81	2.52
50	0.41	12	3.59	3.67	3.61	0.48	2.95	2.25
50	0.41	13	4.01	4.06	3.87	0.41	3.34	3.31
50	0.41	13	3.44	3.89	3.86	0.42	2.63	2.33
50	0.41	14	3.81	3.82	3.81	0.44	3.09	3.13
50	0.41	14	4.16	4	4.1	0.42	3.05	2.52
50	0.41	15	3.55	3.77	3.59	0.41	2.98	2.28
50	0.41	15	4.29	4.16	4.21	0.41	3.55	2.98
50	0.51	6	3.88	3.99	3.9	0.27	1.55	1.42
50	0.51	7	4.28	4.46	4.38	0.33	2.06	1.81
50	0.51	7	3.97	4.12	4.01	0.44	2.33	1.97
50	0.51	8	3.67	3.78	3.67	0.33	2.53	1.98
50	0.51	8	4.43	4.46	4.48	0.36	2.2	1.86
50	0.51	9	4.51	4.1	4.14	0.41	2.41	2.09
50	0.51	9	4.89	4.84	4.77	0.31	2.06	1.92
50	0.51	10	4.33	4.4	4.33	0.47	2.3	1.94
50	0.51	10	4.39	4.3	4.39	0.34	2.45	1.78
50	0.51	11	4.15	3.87	3.89	0.41	2.61	2.06
50	0.51	11	4.19	5.43	4.23	0.47	2.47	2.05
50	0.51	12	4.28	4.34	4.3	0.38	3.34	2.31
50	0.51	12	4.38	4.33	4.33	0.41	2.94	2.39
50	0.51	13	4.41	4.79	4.57	0.45	2.97	2.28
50	0.51	13	3.88	4.08	3.95	0.44	3.55	2.97
50	0.51	14	5.26	5.06	5.12	0.45	3.38	2.61
50	0.51	14	5.62	5.42	5.33	0.41	3.31	2.95
50	0.51	15	4.59	4.54	4.71	0.41	3.17	3.39
50	0.51	15	4.21	4.36	4.18	0.48	3.41	2.67
50	0.61	6	4.2	4.01	4.21	0.3	2.3	1.66
50	0.61	6	4.18	4.18	4.2	0.27	1.56	1.23
50	0.61	7	4.75	4.68	5.09	0.41	2.42	2.11
50	0.61	8	5.28	5.24	5.24	0.38	2.2	2.02
50	0.61	8	3.93	4.09	4.09	0.31	1.77	1.64
50	0.61	9	4.25	4.54	4.48	0.38	2.55	2.28
50	0.61	9	4.3	4.54	4.53	0.36	2	1.78
50	0.61	10	5	4.99	4.99	0.41	2.64	1.86

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
50	0.61	10	4.93	4.89	4.89	0.36	2.36	1.83
50	0.61	11	4.22	4.49	4.25	0.48	4.2	2.44
50	0.61	11	5.3	5.19	5.17	0.34	2.63	2.13
50	0.61	12	4.98	4.85	4.82	0.44	3.06	1.94
50	0.61	12	4.42	4.32	4.35	0.44	2.63	2
50	0.61	13	5.23	5.22	5.19	0.48	3.36	2.8
50	0.61	13	4.92	4.92	5.02	0.41	3.39	2.69
50	0.61	14	5.45	5.01	5.04	0.67	3.61	2.73
50	0.61	14	4.46	4.37	4.32	0.45	2.98	2.73
50	0.61	15	5.43	5.2	5.27	0.42	3.17	3.14
50	0.61	15	5.31	5.22	5.12	0.42	4	3.09
50	0.71	6	4.91	5.47	5.4	0.3	1.64	1.67
50	0.71	6	4.38	5.7	4.39	0.28	1.47	1.09
50	0.71	7	5.42	5.39	5.44	0.33	2.23	1.89
50	0.71	7	4.42	4.42	4.43	0.36	1.78	1.41
50	0.71	8	4.87	4.88	4.88	0.36	2.33	1.58
50	0.71	8	5.39	6.19	5.35	0.36	2.13	1.7
50	0.71	9	5.04	5.37	4.92	0.38	2.5	2.14
50	0.71	9	4.96	5.01	5.3	0.36	2.67	1.98
50	0.71	10	5.11	8.36	5.48	0.41	3.06	2.61
50	0.71	10	4.89	5.29	5.1	0.33	2.48	2.09
50	0.71	11	6.19	5.84	6.08	0.44	2.98	2.47
50	0.71	11	6.08	6.19	5.49	0.38	2.89	2
50	0.71	12	6.57	6.52	6.6	0.39	2.83	2.31
50	0.71	12	5.77	5.82	5.79	0.44	3.02	2.45
50	0.71	13	5.09	4.74	4.75	0.5	3.11	2.34
50	0.71	13	5.19	5.18	5.2	0.45	3.33	2.33
50	0.71	14	5.06	5.03	5.03	0.48	3.98	2.77
50	0.71	14	5.37	5.43	5.23	0.45	4.3	3.05
50	0.71	15	4.95	4.99	4.92	0.42	3.44	3.13
50	0.71	15	4.68	5.53	4.47	0.48	3.81	3.09
50	0.81	6	5.14	5.22	5.19	0.33	1.77	1.48
50	0.81	6	5.41	4.94	4.94	0.3	1.48	1.23
50	0.81	7	5.67	5.98	5.28	0.34	2.77	1.97
50	0.81	7	5.25	4.94	4.92	0.34	2.25	1.89
50	0.81	8	5.32	5.31	5.38	0.33	1.8	1.45
50	0.81	8	23.89	5.94	5.54	0.38	2.33	1.91
50	0.81	9	6.23	5.9	5.83	0.41	2.52	2.31
50	0.81	9	22.6	5.43	5.33	0.47	2.09	1.8
50	0.81	10	5.61	5.34	5.54	0.48	2.77	2.58
50	0.81	10	5.41	5.42	5.25	0.36	2.92	2.7
50	0.81	11	5.59	5.09	5.07	0.45	3.02	2.73
50	0.81	11	6.21	6.23	5.6	0.38	2.44	2.13

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
50	0.81	12	5.37	5.28	5	0.38	3.05	2.81
50	0.81	12	5.52	5.31	5.32	0.47	2.89	2.56
50	0.81	13	6.04	6.01	5.77	0.5	3.56	2.83
50	0.81	13	5.41	5.51	5.41	0.47	3.02	2.5
50	0.81	14	5.52	5.51	5.54	0.47	3.97	3.38
50	0.81	14	5.52	5.37	5.44	0.44	3.91	3.38
50	0.81	15	5.7	6.23	5.7	0.44	3.58	2.88
50	0.81	15	5.28	5.26	5.26	0.69	3.38	2.81
50	0.91	6	5.64	5.69	5.69	0.28	1.55	1.47
50	0.91	6	5.42	5.45	5.45	0.28	1.53	1.47
50	0.91	7	5.32	5.19	5.33	0.34	2.16	1.95
50	0.91	7	6.46	6.05	6.03	0.38	2.36	2.59
50	0.91	8	5.91	5.53	5.58	0.36	2.28	1.86
50	0.91	8	5.7	5.76	5.88	0.38	2.58	2.48
50	0.91	9	6.01	5.82	5.37	0.38	2.38	2.3
50	0.91	9	5.98	5.85	5.87	0.38	2.61	2.31
50	0.91	10	6.79	6.94	6.24	0.47	2.58	2.16
50	0.91	10	4.96	5.42	5.37	0.36	2.72	1.81
50	0.91	11	6.83	6.61	6.04	0.42	3.42	2.41
50	0.91	11	5.68	5.7	5.79	0.39	2.67	2.17
50	0.91	12	6.05	8.09	6.24	0.41	3.23	2.55
50	0.91	12	6.61	6.48	6.39	0.42	3.09	2.52
50	0.91	13	5.9	6.4	6.02	0.48	4.25	2.95
50	0.91	13	5.75	5.85	5.83	0.44	3.39	3.08
50	0.91	14	6.58	6.73	6.58	0.48	3.72	3.2
50	0.91	14	6.78	7.15	6.83	0.45	3.31	2.39
50	0.91	15	6.13	6.11	6.15	0.42	3.48	2.91
50	0.91	15	5.91	6.04	5.88	0.52	3.52	3.19
50	1.01	6	5.68	6.09	5.7	0.3	1.67	1.52
50	1.01	6	5.34	5.23	5.23	0.3	1.63	1.03
50	1.01	7	5.57	5.71	5.71	0.38	2.11	1.83
50	1.01	7	6	6.21	6.33	0.39	2.22	2.14
50	1.01	8	5.96	5.82	5.84	0.34	2.48	1.73
50	1.01	8	5.52	7.15	6.46	0.34	1.92	1.75
50	1.01	9	6.48	5.86	6	0.42	2.69	2.41
50	1.01	9	8.3	5.55	5.59	0.38	2.41	1.86
50	1.01	10	7.48	6.92	7.46	0.47	3.34	2.84
50	1.01	10	6.02	6.55	6.52	0.38	2.44	1.94
50	1.01	11	6.49	6.46	6.52	0.48	3.66	3.2
50	1.01	11	5.38	5.49	5.45	0.34	3	2.13
50	1.01	12	6.84	6.24	6.38	0.66	3.2	2.44
50	1.01	12	6.83	6.66	6.68	0.41	3.02	2.42
50	1.01	13	7.08	7.19	6.85	0.52	3.91	2.97

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
50	1.01	13	6.13	6.11	6.12	0.44	3.56	3.25
50	1.01	14	5.77	5.96	5.78	0.45	4.13	3.27
50	1.01	14	6.32	5.98	6.1	0.45	3.42	2.63
50	1.01	15	6.34	6.25	6.1	0.42	3.36	3.13
50	1.01	15	6.06	7.22	6.5	0.44	3.61	2.95
75	0.01	6	0.58	0.56	0.55	0.27	1.88	4.28
75	0.01	6	0.63	0.76	0.72	0.27	1.8	4.09
75	0.01	7	0.53	0.54	0.54	0.33	2.17	4.94
75	0.01	7	0.56	0.58	0.58	0.41	3.44	7.81
75	0.01	8	0.67	0.67	0.67	0.5	2.31	5.47
75	0.01	8	0.64	0.61	0.61	0.3	2.56	5.44
75	0.01	9	0.71	0.71	0.7	0.42	3.36	6.56
75	0.01	9	0.67	0.67	0.68	0.3	2.63	6.03
75	0.01	10	0.65	0.63	0.63	0.41	2.95	7.23
75	0.01	10	0.62	0.62	0.63	0.33	2.84	6.7
75	0.01	11	0.72	0.72	0.72	0.41	3.38	7.77
75	0.01	11	0.67	0.63	0.64	0.36	3.27	7.03
75	0.01	12	0.76	0.73	0.73	0.34	3.31	7.72
75	0.01	12	0.66	0.71	0.65	0.38	3.16	8.09
75	0.01	13	0.63	0.63	0.62	0.41	3.77	9.11
75	0.01	13	0.59	0.6	0.61	0.36	3.72	8.67
75	0.01	14	0.7	0.74	0.71	0.44	4.06	9.64
75	0.01	14	0.68	0.71	0.71	0.44	3.78	9.33
75	0.01	15	0.67	0.64	0.65	0.38	3.83	9.56
75	0.01	15	0.64	0.63	0.63	0.44	4.73	9.8
75	0.11	6	1.88	1.91	1.91	0.27	1.78	1.39
75	0.11	6	2.12	2.01	2.01	0.45	3.22	1.95
75	0.11	7	2.17	2.33	2.19	0.34	2.45	2.45
75	0.11	7	2.12	2.25	2.21	0.45	4.08	3.3
75	0.11	8	1.98	1.96	1.97	0.33	2.55	2.45
75	0.11	8	1.81	1.81	1.82	0.33	2.55	2.77
75	0.11	9	2.29	2.27	2.27	0.39	2.97	2.13
75	0.11	9	2.06	2.16	2.09	0.34	3	2.34
75	0.11	10	2.08	2.13	2.1	0.38	3.59	3.06
75	0.11	10	2.12	2.12	2.15	0.34	2.92	3.13
75	0.11	11	2.12	2.08	2.09	0.39	4.22	4.44
75	0.11	11	2.2	2.16	2.24	0.36	3.39	3.09
75	0.11	12	1.88	1.88	1.89	0.39	3.44	4.08
75	0.11	12	2.14	2.04	2.01	0.41	3.45	3.02
75	0.11	13	2.13	2.24	2.19	0.5	4.25	3.63
75	0.11	13	2.37	2.42	2.34	0.41	3.78	3.5
75	0.11	14	2.34	2.34	2.34	0.45	4.77	4.3
75	0.11	15	2.16	2.29	2.13	0.39	4.03	4.2

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
75	0.11	15	2.12	2.09	2.24	0.45	4.73	4.94
75	0.21	6	3.02	2.58	2.66	0.28	2.24	1.88
75	0.21	6	2.49	2.49	2.54	0.33	2.84	1.5
75	0.21	7	3.21	2.79	2.83	0.34	2.84	2.88
75	0.21	7	2.67	2.63	2.63	0.41	3.17	2.53
75	0.21	8	2.36	2.47	2.43	0.36	2.7	2.61
75	0.21	8	2.85	3.17	2.94	0.31	2.64	2.7
75	0.21	9	2.81	2.83	2.88	0.38	2.8	2.41
75	0.21	9	2.45	2.42	2.46	0.33	2.84	2.27
75	0.21	10	2.63	4.05	2.79	0.36	3.95	3.73
75	0.21	10	2.79	2.8	2.8	0.34	3.08	2.39
75	0.21	11	3.22	3.42	3.28	0.39	3.73	3.25
75	0.21	11	2.99	3	3	0.36	3.44	3.22
75	0.21	12	3.1	2.83	2.75	0.38	3.95	3.09
75	0.21	12	3.04	3.2	2.91	0.39	4.19	3.39
75	0.21	13	3.04	3.05	2.92	0.41	3.8	3.33
75	0.21	13	2.57	2.67	2.67	0.52	5	3.84
75	0.21	14	2.82	3.19	2.84	0.5	4.91	3.95
75	0.21	14	2.63	2.5	2.55	0.44	4.16	3.42
75	0.21	15	3.34	3.25	3.21	0.44	4.19	3.28
75	0.21	15	3.22	3.17	3.04	0.47	4.59	3.77
75	0.31	6	3.06	2.96	2.91	0.28	2	1.19
75	0.31	6	3.13	3.23	3.05	0.31	2	1.78
75	0.31	7	3.32	3.47	3.19	0.31	2.63	2.19
75	0.31	7	3.57	3.52	3.52	0.42	2.75	2.41
75	0.31	8	3.16	3.18	3.18	0.31	2.95	2.67
75	0.31	8	3.57	3.54	3.55	0.39	3.14	2.61
75	0.31	9	3.74	3.5	3.54	0.42	2.91	2.44
75	0.31	9	3.26	3.02	3.03	0.33	2.7	2.22
75	0.31	10	3.35	3.54	3.18	0.42	4.19	3.45
75	0.31	10	3.45	3.57	3.52	0.66	3.64	2.91
75	0.31	11	3.58	3.61	3.61	0.5	4.61	3
75	0.31	11	3.31	3.17	3.17	0.41	3.19	2.67
75	0.31	12	3.43	5.1	3.34	0.53	3.81	2.97
75	0.31	12	3.63	3.64	3.65	0.45	3.94	3.28
75	0.31	13	3.85	3.85	3.84	0.44	4.67	3.34
75	0.31	13	3.56	3.52	3.52	0.44	4.56	3.88
75	0.31	14	3.57	3.27	3.27	0.52	4.59	3.78
75	0.31	14	3.49	3.49	3.49	0.39	4.28	3.73
75	0.31	15	3.59	3.59	3.58	0.41	4.13	3.41
75	0.31	15	3.09	3.05	3.08	0.47	4.95	3.92
75	0.41	6	3.85	3.68	3.57	0.27	2.19	2.33
75	0.41	6	4.21	4	3.97	0.39	2.16	1.83

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
75	0.41	7	4.59	4.39	4.34	0.45	2.69	2.95
75	0.41	7	3.97	3.97	3.98	0.36	2.7	2.44
75	0.41	8	3.66	3.84	3.46	0.44	2.83	1.94
75	0.41	8	3.54	3.74	3.61	0.34	2.94	2.39
75	0.41	9	4.18	5.03	4.5	0.69	3.75	2.81
75	0.41	9	4.09	4.19	4.29	0.36	2.73	2.48
75	0.41	10	4.66	4.72	4.69	0.41	3.5	2.55
75	0.41	10	4.47	4.06	4.1	0.36	4.13	2.89
75	0.41	11	4.44	4.44	4.43	0.42	4.31	3.13
75	0.41	11	4.99	5.05	5.02	0.36	3.55	3.13
75	0.41	12	4.34	4.27	4.27	0.45	4.05	3.66
75	0.41	12	4	3.87	4.04	0.39	4.47	3.5
75	0.41	13	4.21	4.6	3.94	0.45	4.92	3.63
75	0.41	13	3.74	4.18	3.91	0.45	5	4
75	0.41	14	4.2	4.1	4	0.56	5.89	5.28
75	0.41	14	3.92	3.95	3.9	0.45	4.77	4.06
75	0.41	15	3.98	4.02	4.14	0.44	4.66	3.55
75	0.41	15	3.89	3.96	3.96	0.44	5.45	4.77
75	0.51	6	5.57	4.9	5.39	0.33	2.55	1.83
75	0.51	6	4.33	4.26	4.26	0.3	2.17	2
75	0.51	7	4.79	7.22	4.78	0.38	3.47	3.19
75	0.51	7	4.28	4.06	4.06	0.38	2.44	2.27
75	0.51	8	4.55	4.65	4.57	0.33	2.92	2.34
75	0.51	8	4.68	4.7	4.69	0.38	2.92	2.3
75	0.51	9	4.44	4.44	4.48	0.38	3.27	2.61
75	0.51	9	3.94	4.33	4.27	0.33	3.45	3.55
75	0.51	10	4.94	4.76	4.95	0.42	4.19	3.52
75	0.51	10	4.44	4.4	4.43	0.45	4.17	4.45
75	0.51	11	4.73	4.83	4.72	0.42	4	3.66
75	0.51	11	4.58	4.71	4.57	0.42	4	3.94
75	0.51	12	4.63	4.66	4.87	0.44	4.36	3.81
75	0.51	12	4.94	4.53	4.77	0.5	3.66	3.14
75	0.51	13	4.64	4.75	4.69	0.42	4.36	3.31
75	0.51	13	4.63	4.69	4.69	0.47	4.63	4.11
75	0.51	14	4.55	4.8	4.67	0.52	4.72	3.84
75	0.51	14	4.37	4.71	4.49	0.5	5.06	4.88
75	0.51	15	4.54	4.4	4.46	0.45	3.98	2.91
75	0.51	15	4.32	4.5	4.43	0.44	5.33	3.91
75	0.61	6	4.66	4.49	4.67	0.3	2.06	1.36
75	0.61	7	4.45	18.62	4.37	0.67	4.09	3.83
75	0.61	7	4.52	4.83	4.63	0.41	3.06	3.14
75	0.61	8	4.3	4.05	4.07	0.31	3.55	2.81
75	0.61	8	5.15	5.12	5.11	0.36	2.97	2.52

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
75	0.61	9	5.12	4.92	5.15	0.45	3.52	3.25
75	0.61	9	4.81	4.9	4.81	0.36	3.28	2.38
75	0.61	10	4.55	4.52	4.52	0.39	3.73	3.5
75	0.61	10	5.71	5.54	5.81	0.45	4.23	3.89
75	0.61	11	4.99	5.29	4.64	0.47	4.47	3.78
75	0.61	11	5.21	5.45	5.21	0.41	3.84	3.25
75	0.61	12	4.47	4.44	4.46	0.39	4.72	3.8
75	0.61	12	5.04	5.57	4.97	0.41	4.72	5.05
75	0.61	13	5.45	5.12	5.41	0.44	5.02	4.41
75	0.61	13	5.29	5.35	5.28	0.48	5.41	4.7
75	0.61	14	4.86	4.85	4.88	0.53	5.72	6.14
75	0.61	14	5.56	5.78	5.56	0.45	4.84	4.47
75	0.61	15	13.36	4.78	4.73	0.44	4.56	3.94
75	0.61	15	5.08	5.44	5.1	0.5	5.22	5.08
75	0.71	6	5.21	5.23	5.24	0.28	2.36	1.61
75	0.71	6	5.36	5.36	5	0.27	2.38	2.14
75	0.71	7	5.06	4.97	4.85	0.36	2.89	2.34
75	0.71	8	4.34	4.44	4.49	0.39	2.7	1.77
75	0.71	8	6.07	5.48	5.58	0.38	2.84	2.86
75	0.71	9	5.38	5.76	5.63	0.45	3.59	2.92
75	0.71	9	5.97	6.23	6.19	0.38	3	2.78
75	0.71	10	4.9	4.96	4.68	0.42	4.88	4.11
75	0.71	10	5.24	5.25	5.24	0.45	4.22	3.94
75	0.71	11	5.87	5.78	5.77	0.42	4.23	4.27
75	0.71	11	5.44	5.28	5.32	0.36	4.31	3.27
75	0.71	12	5.47	5.61	5.64	0.44	5.03	4.41
75	0.71	12	5.64	5.64	5.54	0.41	4.86	4.34
75	0.71	13	5.52	5.05	5.08	0.41	4.81	4.88
75	0.71	13	5.39	5.34	5.36	0.47	5.19	4.36
75	0.71	14	5.67	5.26	5.44	0.55	5.2	4.16
75	0.71	14	5.19	5.19	5.34	0.45	5.2	4.56
75	0.71	15	5.48	5.62	5.45	0.45	4.92	5.48
75	0.71	15	5.67	5.42	5.66	0.47	5.83	5.13
75	0.81	6	4.69	4.55	4.55	0.28	2.16	1.77
75	0.81	6	5.61	6.15	5.49	0.28	2.83	2.14
75	0.81	7	5.23	6.91	6.47	0.5	4.25	3.66
75	0.81	7	5.74	5.68	5.58	0.38	3.72	3.08
75	0.81	8	5.2	5.88	5.28	0.38	3.75	2.86
75	0.81	8	5.42	5.38	5.37	0.34	3.14	2.48
75	0.81	9	5.95	5.8	5.8	0.44	3.33	2.8
75	0.81	9	5.27	5.18	5.12	0.39	3.98	3.22
75	0.81	10	5.76	5.92	4.85	0.42	4.41	3.7
75	0.81	10	5.84	5.82	5.67	0.44	4.27	3.92

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
75	0.81	11	5	5.37	5.75	0.42	4.89	3.67
75	0.81	11	5.89	5.76	5.68	0.39	4.5	3.64
75	0.81	12	5.91	5.41	5.35	0.39	5.03	4.27
75	0.81	12	6.17	6.07	6.07	0.47	6.41	3.61
75	0.81	13	5.18	5.18	5.41	0.42	5.56	3.89
75	0.81	13	7.14	7.16	7	0.59	5.08	5.89
75	0.81	14	6.15	6.51	6.18	0.55	5.25	4.64
75	0.81	14	6.58	5.56	5.18	0.47	5.13	4.53
75	0.81	15	6.41	6.9	6.45	0.45	5.77	4.72
75	0.81	15	6.1	6.1	6.27	0.56	5.83	5.13
75	0.91	6	5.25	5.16	5.26	0.31	2.52	2.84
75	0.91	6	6.98	6.91	7.36	0.38	3.19	2.89
75	0.91	7	5.67	5.14	5.15	0.5	5.06	3.67
75	0.91	7	5.52	5.23	5.23	0.38	3.02	2.91
75	0.91	8	5.33	5.3	5.33	0.36	3.45	3.19
75	0.91	9	5.59	5.6	5.51	0.41	4.31	3.31
75	0.91	9	6.79	6.19	6.21	0.44	3.17	2.67
75	0.91	10	6.32	6.64	6.37	0.41	4.48	4.83
75	0.91	10	6.24	7.21	5.93	0.41	5.11	4.11
75	0.91	11	5.71	5.66	5.65	0.42	4.38	3.48
75	0.91	11	5.82	5.73	5.6	0.42	4.13	3.33
75	0.91	12	6.03	5.86	5.53	0.44	4.72	4.69
75	0.91	12	5.84	6.11	5.89	0.45	4.92	4.34
75	0.91	13	5.83	5.78	5.71	0.48	4.64	4.34
75	0.91	13	5.97	5.63	5.63	0.55	4.94	3.34
75	0.91	14	5.97	6.41	6.34	0.52	5.98	5.55
75	0.91	14	6.17	6.22	5.85	0.53	4.81	3.98
75	0.91	15	6.48	6.51	6.63	0.44	5.92	3.91
75	0.91	15	6.15	7.96	6.33	0.48	5.94	5.33
75	1.01	6	5.88	5.77	5.71	0.3	2.38	2.27
75	1.01	6	5.68	6.05	5.83	0.38	3.47	2
75	1.01	7	6.76	6.27	6.56	0.55	4.52	3.39
75	1.01	7	6.22	6.27	6.21	0.42	3.02	2.2
75	1.01	8	6.26	6.49	6.36	0.34	3.19	3.03
75	1.01	8	7.19	6.34	6.78	0.33	3.8	3.69
75	1.01	9	6.06	6.41	6.36	0.41	3.58	2.88
75	1.01	9	6.2	6.52	6.32	0.42	3.39	2.59
75	1.01	10	6.33	6.75	6.3	0.41	4.2	4.66
75	1.01	10	6.26	6.46	5.82	0.5	4.86	3.73
75	1.01	11	6.86	6.59	6.86	0.45	5.31	4.63
75	1.01	11	8.21	8.14	8.14	0.41	4.27	3.55
75	1.01	12	5.49	5.56	5.4	0.42	4.17	3.8
75	1.01	12	6.1	6.4	6.16	0.47	5.58	4.19

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
75	1.01	13	7.08	6.8	6.73	0.44	5.33	3.66
75	1.01	13	5.88	5.74	6.04	0.44	5.78	4.17
75	1.01	14	6.91	6.74	6.88	0.48	6	4.94
75	1.01	14	6.33	7.31	6.35	0.45	5.7	5.2
75	1.01	15	5.8	5.74	6.11	0.53	5.47	5.09
75	1.01	15	6.88	7.13	7.27	0.47	5.91	5.05
100	0.01	6	0.57	0.64	0.63	0.3	2.47	5.3
100	0.01	7	0.63	0.65	0.63	0.5	6.47	11.83
100	0.01	7	0.6	0.61	0.61	0.42	2.89	6.72
100	0.01	8	0.58	0.59	0.58	0.38	3.22	7.45
100	0.01	8	0.56	0.62	0.59	0.3	3.3	7.47
100	0.01	9	0.75	0.75	0.75	0.34	3.58	8.59
100	0.01	9	0.63	0.63	0.62	0.39	3.5	7.56
100	0.01	10	0.68	0.76	0.69	0.36	4.39	9.16
100	0.01	10	0.71	0.68	0.68	0.52	5.03	11.38
100	0.01	11	0.64	0.63	0.61	0.42	4.31	10.72
100	0.01	11	0.63	0.65	0.64	0.34	4.25	8.64
100	0.01	12	0.62	0.63	0.63	0.38	4.38	10.48
100	0.01	12	0.67	0.66	0.67	0.5	4.58	10.47
100	0.01	13	0.58	0.63	0.63	0.39	5	11.41
100	0.01	13	0.67	0.68	0.7	0.41	5.61	12.25
100	0.01	14	0.7	0.74	0.73	0.45	5.7	13.3
100	0.01	14	0.7	0.64	0.64	0.39	5	10.72
100	0.01	15	0.67	0.67	0.65	0.39	5.63	12.13
100	0.01	15	0.65	0.65	0.65	0.42	5.48	13.55
100	0.11	6	2.09	2.1	2.12	0.38	3.03	2.58
100	0.11	6	2.12	2.04	2.03	0.25	2.42	1.75
100	0.11	7	2.1	2.14	2.06	0.52	4.64	5.14
100	0.11	7	2.09	2.15	2.14	0.38	3.17	4.08
100	0.11	8	2.01	1.99	1.97	0.36	3.55	4.27
100	0.11	8	2.21	2.14	2.13	0.34	3.3	2.7
100	0.11	9	2.04	2.01	2	0.34	4.11	3.77
100	0.11	9	2.08	2.22	2.15	0.47	4.89	3
100	0.11	10	2.05	2.1	2.09	0.36	4.28	4.13
100	0.11	10	2.13	2.14	2.08	0.52	5.22	3.44
100	0.11	11	2.02	2.08	2.03	0.41	4.89	4.2
100	0.11	11	2.12	2.17	2.11	0.36	4.03	2.94
100	0.11	12	2.24	2.24	2.18	0.41	5.36	4.55
100	0.11	12	2.15	2.07	2.05	0.39	4.55	3.5
100	0.11	13	2.06	2.06	2.06	0.41	5.45	4.5
100	0.11	13	2.18	2.16	2.16	0.42	4.91	4.33
100	0.11	14	2.03	1.99	2.11	0.5	6.81	4.98
100	0.11	14	2.52	2.53	2.47	0.39	7.73	4.84

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
100	0.11	15	2.14	2.16	2.16	0.39	5.42	4.31
100	0.11	15	2.2	2.16	2.21	0.41	5.88	6.28
100	0.21	6	2.7	2.8	2.76	0.39	2.7	3.03
100	0.21	6	2.45	2.42	2.51	0.34	3.06	2.25
100	0.21	7	2.84	2.75	2.68	0.31	3.48	3.45
100	0.21	8	3.4	3.09	3.23	0.34	3.69	3.08
100	0.21	8	2.97	2.88	2.89	0.34	4	3.47
100	0.21	9	3.16	2.96	2.94	0.34	4.13	3.25
100	0.21	9	3.2	3.35	3.35	0.36	3.56	2.56
100	0.21	10	2.97	2.86	2.89	0.41	4.41	3.91
100	0.21	10	2.85	3.42	2.93	0.36	4.08	3.28
100	0.21	11	3.06	2.84	2.9	0.39	6.89	5.08
100	0.21	11	2.8	2.68	2.73	0.39	4.33	3.5
100	0.21	12	2.69	2.7	2.76	0.42	5.02	5.56
100	0.21	12	2.67	2.72	2.72	0.41	4.88	3.84
100	0.21	13	2.94	3.07	3	0.42	5.72	4.42
100	0.21	13	3.08	3.02	3.02	0.45	4.94	4.28
100	0.21	14	2.94	3.1	2.94	0.47	6.09	4.53
100	0.21	14	3.36	3.27	3.27	0.38	5.58	4.5
100	0.21	15	2.98	2.98	2.98	0.41	5.8	4.25
100	0.21	15	3.06	2.95	2.99	0.44	5.75	4.09
100	0.31	6	3.48	3.43	3.43	0.39	3.81	3.2
100	0.31	6	3.61	3.47	3.43	0.33	2.52	1.72
100	0.31	7	3.13	3.11	3.09	0.52	5.58	4.66
100	0.31	7	3.63	3.56	3.82	0.36	3.17	2.73
100	0.31	8	3.94	3.97	4.01	0.39	3.94	3.13
100	0.31	9	2.97	3.3	3.16	0.44	5.03	4.38
100	0.31	9	3.53	3.56	3.61	0.31	3.53	3.22
100	0.31	10	4.4	4.03	3.95	0.39	4.48	4.58
100	0.31	10	3.89	3.61	3.63	0.45	4.61	4.75
100	0.31	11	3.54	3.51	3.51	0.39	4.77	3.45
100	0.31	11	3.71	3.92	3.63	0.38	5.64	4.95
100	0.31	12	3.22	3.23	3.23	0.41	5.52	4.22
100	0.31	12	3.61	3.49	3.52	0.42	5.06	3.86
100	0.31	13	3.65	3.64	3.64	0.44	5.7	4.25
100	0.31	13	3.76	3.82	3.72	0.47	5.97	4.94
100	0.31	14	3.95	3.84	3.84	0.47	6.75	5.2
100	0.31	14	3.81	3.81	3.68	0.38	5.63	5.14
100	0.31	15	3.65	3.62	3.65	0.42	5.69	4.55
100	0.31	15	4.29	4.29	4.62	0.52	7.13	5.69
100	0.41	6	4.09	4.19	4.08	0.34	3.49	2.97
100	0.41	6	4.58	4.4	4.18	0.31	3.05	3.16
100	0.41	7	4.04	3.77	3.69	0.41	4.89	4.42

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
100	0.41	7	4.36	4.73	4.47	0.36	3.95	2.92
100	0.41	8	3.97	5.19	3.98	0.34	4.86	3.83
100	0.41	8	4.66	4.61	4.66	0.47	4.03	3.95
100	0.41	9	4.2	4.08	4.35	0.34	4.61	4.61
100	0.41	10	4.24	4.35	4.38	0.39	4.47	4.03
100	0.41	10	3.89	3.69	3.73	0.39	4.72	4.61
100	0.41	11	5.04	4.88	5.05	0.41	5.92	5.09
100	0.41	11	4.02	4.03	4.04	0.34	4.33	4.13
100	0.41	12	4.31	4.23	4.18	0.39	5.44	4.41
100	0.41	12	5.17	4.84	4.89	0.67	5.61	5.89
100	0.41	13	4.09	4.2	4.12	0.47	6.7	5.77
100	0.41	13	4.16	4.62	4.7	0.41	6	5.02
100	0.41	14	4.52	4.6	4.54	0.47	6.83	5.73
100	0.41	14	4.03	4.13	4.08	0.41	5.81	5.48
100	0.41	15	4.13	4.95	4.9	0.41	5.86	5.92
100	0.41	15	3.63	3.64	3.64	0.45	6.7	6.84
100	0.51	6	4.26	4.06	4.05	0.44	3.44	3.8
100	0.51	6	4.6	4.37	4.34	0.3	2.39	2.45
100	0.51	7	4.71	4.71	4.47	0.39	3.47	3.73
100	0.51	8	4.45	4.49	4.4	0.36	4.34	3.14
100	0.51	8	4.75	4.69	4.74	0.42	3.86	4.39
100	0.51	9	4.58	4.58	4.49	0.34	4.47	4.34
100	0.51	9	4.72	5.22	4.83	0.36	3.95	3.3
100	0.51	10	4.39	4.91	4.27	0.41	5.95	4.64
100	0.51	10	4.51	4.51	4.51	0.45	4.73	6.42
100	0.51	11	4.35	4.59	4.28	0.44	6.55	4.64
100	0.51	11	5.33	5.3	5.32	0.38	5.14	4.08
100	0.51	12	4.24	4.41	4.18	0.45	5.17	4.59
100	0.51	12	4.37	4.33	4.33	0.45	5.33	4.05
100	0.51	13	5.83	5.93	5.84	0.42	6.53	5.95
100	0.51	13	5.01	4.61	4.64	0.44	5.84	4.58
100	0.51	14	4.86	4.89	4.96	0.5	6.34	6.55
100	0.51	14	4.18	4.21	4.45	0.41	6.34	5.34
100	0.51	15	4.85	4.82	4.78	0.42	7.08	5.83
100	0.51	15	4.78	4.79	4.72	0.44	6.25	5.59
100	0.61	6	4.64	4.74	4.69	0.28	2.84	2.44
100	0.61	6	4.09	4.27	4.17	0.3	2.83	2.47
100	0.61	7	5.01	4.79	5.04	0.61	6.19	6.99
100	0.61	7	5.32	5.53	5.5	0.34	4.06	2.56
100	0.61	8	4.5	4.72	4.72	0.38	3.75	2.81
100	0.61	8	4.99	5.05	4.96	0.58	3.7	2.88
100	0.61	9	5.03	5.21	5.05	0.39	5.08	4.34
100	0.61	9	4.62	4.32	4.33	0.38	4.27	3.91

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
100	0.61	10	4.56	4.52	4.56	0.44	4.48	4.2
100	0.61	10	5.13	5.08	5.16	0.41	5.34	5.3
100	0.61	11	4.45	4.47	4.47	0.42	5.42	4.81
100	0.61	11	4.29	4.41	4.59	0.38	5.09	4.19
100	0.61	12	5.82	5.57	5.55	0.39	5.67	5.06
100	0.61	12	4.9	5.15	4.94	0.42	6.88	7.48
100	0.61	13	5.17	5.05	5.05	0.42	6.66	5.42
100	0.61	13	5.26	5.46	5.42	0.41	6.73	5.53
100	0.61	14	5.03	5.2	5.05	0.52	6.83	4.88
100	0.61	14	5.3	5.39	5.29	0.44	6.02	5.73
100	0.61	15	5.13	5.11	5.14	0.5	6.84	5.63
100	0.61	15	5.69	5.72	5.55	0.52	7.52	7.02
100	0.71	6	4.81	7.91	4.91	0.3	3.53	2.99
100	0.71	6	5.4	5.76	5.45	0.34	2.91	2.64
100	0.71	7	5.16	5.17	5.05	0.52	6.39	6.36
100	0.71	7	5.89	5.84	5.55	0.34	4.52	3.94
100	0.71	8	5.36	5.19	5.49	0.34	6.22	4.06
100	0.71	8	4.6	23.48	4.55	0.34	4.45	3.42
100	0.71	9	5.18	6.62	5.21	0.42	5.83	4.97
100	0.71	9	5.1	5.12	5.13	0.33	3.97	3.27
100	0.71	10	6.32	5.63	5.97	0.39	5.41	3.73
100	0.71	10	5.04	4.99	5.29	0.41	5.47	4.14
100	0.71	11	5.23	5.42	5.27	0.42	6.03	5.23
100	0.71	11	5.35	5.32	5.47	0.47	5.09	4.27
100	0.71	12	6.28	6.08	5.85	0.39	6.25	5.41
100	0.71	12	5.51	5.28	5.6	0.63	8.09	6.44
100	0.71	13	5.61	5.75	5.37	0.53	6.44	4.92
100	0.71	13	6.84	6.43	6.45	0.55	6.42	5.28
100	0.71	14	5.44	5.58	5.63	0.5	6.42	6.03
100	0.71	14	5.44	5.22	5.26	0.47	6.53	5.09
100	0.71	15	5.51	5.57	5.36	0.42	8.52	6.58
100	0.71	15	5.71	5.36	5.71	0.63	7.45	5.94
100	0.81	6	6.01	6	6	0.33	3.31	3.33
100	0.81	6	6.52	25.7	5.96	0.28	3.78	3.59
100	0.81	7	5.06	5.2	5.1	0.33	3.48	2.45
100	0.81	8	6.26	6.47	6.27	0.36	4.39	3.31
100	0.81	8	6.27	6.34	6.29	0.55	4.94	4.19
100	0.81	9	5.75	6.02	6	0.53	4.48	3.77
100	0.81	9	5.73	5.79	5.48	0.36	4.52	4.13
100	0.81	10	5.48	5.35	5.27	0.44	5.73	4.2
100	0.81	10	5.9	5.83	5.65	0.39	5.55	4.89
100	0.81	11	5.86	6.16	5.95	0.42	6.22	5.41
100	0.81	11	6.36	6.25	6.06	0.44	5.31	5.66

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
100	0.81	12	5.94	5.92	5.92	0.41	7.52	5.56
100	0.81	12	5.96	5.76	5.86	0.45	8.23	7.59
100	0.81	13	5.49	5.44	5.46	0.44	6.81	5.44
100	0.81	13	6.1	6.19	6.08	0.48	6.69	4.84
100	0.81	14	6.02	6.25	6.13	0.64	6.67	6.34
100	0.81	14	7.14	6.78	7.05	0.44	6.86	5.42
100	0.81	15	6.58	7.17	6.54	0.44	7.16	5.23
100	0.81	15	5.94	5.89	5.96	0.45	7.36	8.47
100	0.91	6	6.81	6.47	6.52	0.31	3.5	3.02
100	0.91	6	6.02	6.08	5.95	0.3	3.48	2.94
100	0.91	7	6.25	6.39	6.36	0.41	4.06	3.84
100	0.91	8	5.73	6.05	5.74	0.39	5.09	5.22
100	0.91	8	5.28	5.55	5.54	0.34	4.28	3.23
100	0.91	9	5.82	6.01	5.77	0.38	4.89	4.02
100	0.91	9	7.25	6.04	6.05	0.45	4.11	3.92
100	0.91	10	6.41	6.04	6.17	0.39	5.55	4.2
100	0.91	10	5.92	6.23	6.09	0.48	5.09	4.39
100	0.91	11	5.86	6.41	5.81	0.53	5.47	5.89
100	0.91	11	5.96	5.78	6.23	0.38	5.77	4.55
100	0.91	12	5.7	5.61	5.67	0.44	6.16	4.83
100	0.91	12	6.82	6.01	6.21	0.7	8.78	7.56
100	0.91	13	5.8	6.3	5.82	0.44	7.38	6.44
100	0.91	13	7.37	6.84	6.82	0.53	7.06	5.63
100	0.91	14	5.95	6.1	5.95	0.52	7.33	5.86
100	0.91	14	6.38	6.39	6.38	0.42	7.14	6.19
100	0.91	15	6.86	6.94	7.07	0.45	7.86	7.38
100	0.91	15	6.5	7.42	6.83	0.52	7.39	5.66
100	1.01	6	5.59	5.81	5.81	0.31	3.3	2.52
100	1.01	6	5.99	6.05	5.83	0.3	3.27	3.27
100	1.01	7	6.41	6.25	6.39	0.41	5.33	4.42
100	1.01	7	6.51	7.34	6.24	0.36	3.97	3.77
100	1.01	8	7	7.31	7.05	0.34	4.67	4.52
100	1.01	8	7.35	7.23	6.47	0.34	4.97	4.63
100	1.01	9	7.86	7.9	7.96	0.38	5.48	5.28
100	1.01	9	7.14	7.12	6.9	0.36	5.06	4.41
100	1.01	10	6.69	6.59	6.46	0.42	6.45	5.22
100	1.01	10	5.85	5.88	5.81	0.41	4.91	3.33
100	1.01	11	6.28	6.17	6.2	0.42	6.52	5.47
100	1.01	11	7.54	7.68	7.69	0.38	5.75	5.27
100	1.01	12	6.71	8.07	6.67	0.41	5.89	6.03
100	1.01	12	8.15	7.2	7.34	0.67	8.61	7.45
100	1.01	13	6.18	6.15	6.24	0.48	6.77	6.22
100	1.01	13	6.76	6.4	6.7	0.44	6.77	7.68

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
100	1.01	14	6.09	6.3	6.38	0.64	7.38	7.09
100	1.01	14	6.89	6.38	6.38	0.44	6.69	4.92
100	1.01	15	6.39	6.04	6.03	0.58	6.92	6.08
100	1.01	15	6.67	6.57	6.58	0.5	8.64	7
125	0.01	6	0.61	0.6	0.6	0.3	3.08	7.16
125	0.01	6	0.71	0.84	0.79	0.33	3.69	7.22
125	0.01	7	0.64	0.65	0.65	0.42	4.05	9.31
125	0.01	7	0.75	0.71	0.71	0.31	3.64	8.89
125	0.01	8	0.73	0.71	0.71	0.48	4.14	9.5
125	0.01	8	1.47	0.61	0.6	0.34	4.98	9.06
125	0.01	9	0.71	0.71	0.71	0.34	5.16	10.98
125	0.01	9	0.58	0.59	0.59	0.33	4.31	12.12
125	0.01	10	0.63	0.61	0.61	0.45	5.2	11.89
125	0.01	10	0.65	0.65	0.64	0.38	4.81	10.94
125	0.01	11	0.72	0.71	0.71	0.38	5.81	13.13
125	0.01	11	0.65	0.64	0.65	0.38	5.38	11.75
125	0.01	12	0.75	0.73	0.76	0.33	5.75	13.47
125	0.01	12	0.73	0.67	0.66	0.38	6.69	13.16
125	0.01	13	0.68	0.72	0.72	0.47	7.54	14.98
125	0.01	13	0.61	0.61	0.61	0.42	6.33	13.98
125	0.01	14	0.58	0.61	0.57	0.44	7.02	17.27
125	0.01	14	0.68	0.71	0.69	0.5	6.94	16.42
125	0.01	15	0.65	0.65	0.65	0.36	6.75	14.78
125	0.01	15	0.69	0.69	0.69	0.44	6.97	16.02
125	0.11	6	1.93	2.59	2.32	0.31	3.63	3
125	0.11	6	1.93	1.93	1.93	0.3	3.5	2.98
125	0.11	7	1.93	1.94	1.94	0.39	3.98	2.86
125	0.11	7	2.03	2.02	2.03	0.39	3.72	3.23
125	0.11	8	2.1	2.01	2.01	0.36	4.3	3.02
125	0.11	8	2.33	2.38	2.3	0.36	4.58	3.19
125	0.11	9	2.42	2.56	2.55	0.38	5.14	3.98
125	0.11	9	2.05	2.11	2.07	0.34	4.59	3.89
125	0.11	10	2.23	2.26	2.26	0.44	5.34	4.66
125	0.11	10	2.34	1.99	1.99	0.34	4.88	3.95
125	0.11	11	2.27	2.26	2.29	0.42	5.77	4.5
125	0.11	11	2.2	2.19	2.19	0.38	6.06	5.17
125	0.11	12	2.2	2.18	2.18	0.42	6.75	5.47
125	0.11	12	2.11	2.09	2.05	0.41	6.42	5.17
125	0.11	13	2.17	2.19	2.16	0.42	6.89	5.89
125	0.11	13	2.09	2.01	2.05	0.41	6.53	4.97
125	0.11	14	2.05	2.01	2.06	0.5	11.16	6.77
125	0.11	14	2.19	2.21	2.21	0.41	6.97	6.58
125	0.11	15	2.25	2.24	2.24	0.44	7.13	4.84

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
125	0.11	15	2.33	2.22	2.27	0.44	7.55	5.91
125	0.21	6	3.05	8.65	2.99	0.27	3.41	2.8
125	0.21	6	2.71	2.64	2.68	0.38	3.23	3.47
125	0.21	7	2.85	2.8	2.77	0.33	3.95	3.52
125	0.21	8	3.16	2.84	3.08	0.42	4.42	3.44
125	0.21	9	2.92	3.04	3.02	0.38	6.2	5.19
125	0.21	9	3.07	3.13	3.16	0.41	4.72	4.16
125	0.21	10	3.25	3.74	3.38	0.38	6.06	5.33
125	0.21	10	2.94	2.8	2.8	0.38	5.14	4.67
125	0.21	11	2.94	3.09	3.12	0.45	6.22	6
125	0.21	11	2.77	2.79	2.78	0.36	6.36	4.45
125	0.21	12	2.96	3.08	2.76	0.42	6.39	5.03
125	0.21	12	2.93	3.08	2.9	0.47	6.2	5.89
125	0.21	13	3.11	3.11	3.11	0.42	6.98	6.73
125	0.21	13	2.7	2.83	2.83	0.41	7.7	7.77
125	0.21	14	2.59	2.6	2.61	0.52	7.72	6.67
125	0.21	14	3.14	3.29	3.18	0.39	7.41	5.84
125	0.21	15	3.44	3.49	3.49	0.52	8.45	6.61
125	0.21	15	3.43	3.24	3.25	0.44	9.55	8.97
125	0.31	6	3.5	3.71	3.53	0.27	6.72	3.7
125	0.31	6	3.67	3.64	3.66	0.31	3.3	3.28
125	0.31	7	3.26	3.63	3.4	0.38	5.53	5.16
125	0.31	7	3.78	3.7	3.56	0.38	4.58	4.89
125	0.31	8	3.67	3.51	3.51	0.39	4.72	3.16
125	0.31	8	3.28	3.19	3.19	0.31	4.8	4.84
125	0.31	9	3.28	3.25	3.25	0.36	5.7	5.25
125	0.31	9	3.27	3.3	3.25	0.36	4.31	3.83
125	0.31	10	3.47	3.57	3.49	0.39	6.52	6.11
125	0.31	10	3.44	3.53	3.49	0.33	6.64	4.88
125	0.31	11	3.51	3.83	3.51	0.44	7.25	5.92
125	0.31	11	3.69	3.58	3.71	0.5	7.92	8.25
125	0.31	12	3.86	3.58	3.59	0.41	7.05	5.7
125	0.31	13	3.85	3.87	3.88	0.48	7.23	6.28
125	0.31	13	3.88	3.94	3.92	0.41	7.38	6.13
125	0.31	14	3.59	3.58	3.58	0.44	7.77	7.08
125	0.31	14	3.52	3.55	3.51	0.42	10.64	7.8
125	0.31	15	4.01	3.65	3.77	0.42	7.66	7.58
125	0.31	15	4.42	4.05	4.11	0.47	8.17	7.45
125	0.41	6	4.28	5.04	4.27	0.34	4.24	3.3
125	0.41	6	4.19	4.18	4.19	0.28	4.7	3.41
125	0.41	7	4.39	4.36	4.39	0.5	7.94	7.64
125	0.41	7	4.22	4.4	4.29	0.38	4.91	4.38
125	0.41	8	4.45	4.58	4.6	0.34	5.03	4.22

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
125	0.41	8	4.35	4.36	4.32	0.33	5.22	4.42
125	0.41	9	4.69	4.8	4.82	0.39	5.88	5.13
125	0.41	9	4.06	4.15	3.98	0.38	5.61	4.97
125	0.41	10	4.5	4.34	4.27	0.39	7.33	5.5
125	0.41	10	4.49	4.62	4.49	0.36	6	5.66
125	0.41	11	4.29	4.49	4.43	0.44	6.78	6.08
125	0.41	11	4.44	4.43	4.43	0.42	8.23	5.73
125	0.41	12	4.42	4.44	4.49	0.44	6.78	5.81
125	0.41	12	4.56	4.91	4.64	0.41	7.17	7.05
125	0.41	13	4.3	4.32	4.29	0.47	7.19	6.8
125	0.41	13	4.25	4.39	4.49	0.42	7.63	7.33
125	0.41	14	4.21	4.21	4.21	0.47	9.59	7.84
125	0.41	14	4.53	4.64	4.69	0.44	8.66	7.27
125	0.41	15	4.14	4.45	4.26	0.44	7.7	7.78
125	0.41	15	4.41	4.71	4.39	0.45	8.72	7.58
125	0.51	6	5.21	4.95	4.96	0.28	4.17	3.3
125	0.51	6	4.12	4.02	4.02	0.36	3.11	2.78
125	0.51	7	4.51	4.49	4.47	0.73	6.94	7.3
125	0.51	7	5.35	6.25	5.88	0.45	4.98	4.34
125	0.51	8	4.58	4.44	4.44	0.39	4.92	4.42
125	0.51	8	4.3	4.5	4.37	0.41	5.53	4.58
125	0.51	9	4.3	4.58	4.31	0.38	6	6.09
125	0.51	9	4.9	4.62	4.59	0.34	7.44	6.28
125	0.51	10	4.78	4.37	4.4	0.39	7.53	5.39
125	0.51	10	5.01	5.01	5.03	0.34	6.17	5.09
125	0.51	11	5.05	5.65	5	0.41	8.02	6.88
125	0.51	11	4.45	4.39	4.44	0.41	6.55	5.75
125	0.51	12	4.78	4.94	4.83	0.45	9.06	7.44
125	0.51	12	5.24	5.15	5.14	0.5	7.55	6.55
125	0.51	13	4.31	4.39	4.33	0.39	7.89	6.78
125	0.51	13	5.74	5.33	5.7	0.45	8.84	6.98
125	0.51	14	4.96	4.87	4.9	0.42	8.19	7.2
125	0.51	14	4.82	5.02	5.08	0.44	8.36	6.73
125	0.51	15	4.6	4.65	4.38	0.53	8.55	6.55
125	0.51	15	4.96	4.97	5.07	0.47	11.11	8.47
125	0.61	6	4.9	5.02	4.87	0.31	3.81	2.53
125	0.61	6	5.2	5.19	5.15	0.31	4.45	2.69
125	0.61	7	5.25	5.15	5.15	0.58	7.31	4.22
125	0.61	7	5.39	5.48	5.41	0.36	4.56	4.88
125	0.61	8	5.35	5.26	5.35	0.36	5.06	3.84
125	0.61	9	4.8	4.84	4.89	0.58	6.59	5.69
125	0.61	9	5.19	4.87	4.83	0.38	5.17	5.36
125	0.61	10	5.27	5.23	5.14	0.41	7.13	6.41

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
125	0.61	10	5.08	11.24	5.07	0.38	5.58	5.2
125	0.61	11	5.82	5.47	5.31	0.45	7.56	6.56
125	0.61	11	6.08	5.86	5.87	0.39	8.14	7.64
125	0.61	12	5.82	5.58	6	0.44	6.97	6.52
125	0.61	12	4.94	5.01	4.96	0.39	6.98	6.16
125	0.61	13	6.48	6.52	6.48	0.41	9.02	8.47
125	0.61	13	6.24	6.2	6.1	0.44	7.89	7.3
125	0.61	14	5.35	5.43	5.51	0.45	9.06	7.78
125	0.61	14	5.51	5.52	5.61	0.44	7.89	7.48
125	0.61	15	5.71	5.47	5.49	0.45	9.55	7.84
125	0.61	15	5.45	5.14	5.5	0.47	9.8	7.63
125	0.71	6	5.74	5.32	5.26	0.3	3.88	4.09
125	0.71	6	5.18	5.77	5.18	0.34	5.08	4.55
125	0.71	7	5.44	5.87	5.99	0.39	4.95	4.78
125	0.71	8	6.27	6.04	6.14	0.34	5.48	4.94
125	0.71	8	5.73	5.53	5.57	0.31	5.7	6.02
125	0.71	9	5.28	5.21	5.04	0.38	6.95	6.42
125	0.71	9	4.85	5.19	5.16	0.36	5.55	4.95
125	0.71	10	5.92	5.71	5.69	0.41	5.77	5.34
125	0.71	10	5.72	5.74	5.55	0.38	6.45	6.03
125	0.71	11	5.21	6.34	5.67	0.42	7.05	7.64
125	0.71	11	5.86	5.85	5.86	0.44	10.41	6.63
125	0.71	12	5.46	5.7	5.55	0.47	7.19	6.08
125	0.71	12	5.59	5.31	5.5	0.42	7.16	6.11
125	0.71	13	5.8	6.06	5.88	0.41	9.03	10.08
125	0.71	13	5.18	5.27	5.06	0.41	8.09	6.83
125	0.71	14	5.75	5.76	5.65	0.47	11.39	8.3
125	0.71	14	7.71	5.57	5.52	0.47	9.17	6.34
125	0.71	15	5.39	5.55	5.32	0.45	9.09	7.89
125	0.71	15	6.06	5.77	5.91	0.52	9.72	8.41
125	0.81	6	4.87	4.97	4.98	0.28	4.38	4.78
125	0.81	6	6.11	6.68	6.21	0.31	4.72	3.7
125	0.81	7	6.35	5.59	5.72	0.38	5.3	5.19
125	0.81	7	6.32	5.8	5.33	0.38	4.92	4.48
125	0.81	8	5.56	5.68	5.73	0.38	5.52	6.47
125	0.81	8	5.8	5.96	5.81	0.34	5.77	4.94
125	0.81	9	6.11	5.85	5.82	0.47	7.02	5.56
125	0.81	9	6.42	6.47	6.43	0.36	5.88	5.08
125	0.81	10	5.86	5.76	5.84	0.39	7.11	5.69
125	0.81	10	5.69	5.51	5.61	0.39	6.42	6.58
125	0.81	11	5.66	5.63	5.81	0.47	7.91	7.34
125	0.81	11	5.44	5.47	5.43	0.41	7.16	5.5
125	0.81	12	5.62	5.75	5.56	0.45	7.95	6.22

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
125	0.81	12	6.12	6.04	5.93	0.42	7.66	6.48
125	0.81	13	6.07	5.66	5.71	0.45	8.72	9.06
125	0.81	13	5.85	5.72	5.67	0.45	10.36	8.23
125	0.81	14	6.05	6.14	6.23	0.42	10.75	9.3
125	0.81	14	6.2	7.13	6.24	0.52	9.03	6.94
125	0.81	15	6	5.87	6.09	0.52	8.92	8.13
125	0.81	15	6.34	6.18	6.16	0.56	9.09	7.53
125	0.91	6	5.94	6.29	6.2	0.39	4.17	3.81
125	0.91	6	6.32	6.18	6.18	0.42	5.08	3.88
125	0.91	7	5.82	6.86	6.17	0.36	5.38	4.25
125	0.91	7	5.52	5.61	5.67	0.34	4.92	3.53
125	0.91	8	5.95	5.94	5.95	0.38	5.31	4.05
125	0.91	8	6.25	7.22	6.27	0.36	5.94	4.7
125	0.91	9	5.82	5.84	5.84	0.42	5.59	4.45
125	0.91	9	6.53	6.73	6.17	0.39	6.52	6.64
125	0.91	10	6.49	6.6	6.57	0.44	7.2	7.83
125	0.91	10	7	6.95	6.89	0.45	9	5.42
125	0.91	11	6.1	6.07	6.25	0.52	7.81	6.89
125	0.91	11	6.42	7.21	6.5	0.41	7.41	6.11
125	0.91	12	7.16	7	7.28	0.44	8.05	7.14
125	0.91	12	6.43	6.45	6.43	0.42	8.84	6.89
125	0.91	13	6.19	6.32	6.16	0.8	10.22	7.83
125	0.91	13	6.56	6.82	6.51	0.5	9.16	7.7
125	0.91	14	5.76	6.42	5.71	0.56	10.13	9.03
125	0.91	14	5.68	5.93	5.84	0.5	9	8.16
125	0.91	15	6.18	6.18	6	0.52	8.3	6.55
125	0.91	15	6.21	6.53	6	0.47	10.97	9.2
125	1.01	6	5.52	5.68	5.52	0.31	3.84	4.22
125	1.01	6	5.51	5.47	5.63	0.44	4.64	3.34
125	1.01	7	6.35	6.56	6.57	0.45	5.27	5.09
125	1.01	7	7.34	7.34	7.34	0.34	5.38	4.59
125	1.01	8	6.32	6.14	6.13	0.39	5.52	4.61
125	1.01	8	7.59	7.15	7.17	0.34	5.66	5.23
125	1.01	9	6.4	6.41	6.66	0.41	8.77	6.73
125	1.01	9	6.47	6.4	6.63	0.36	5.69	4.98
125	1.01	10	6.38	6.26	6.29	0.38	6.7	7.09
125	1.01	10	7.02	8.19	7.23	0.38	7.5	7.23
125	1.01	11	7.14	19.42	7.03	0.45	8.61	6.73
125	1.01	11	6.48	6.87	6.52	0.44	8	8.27
125	1.01	12	7.46	8.82	7.34	0.38	8.41	7.53
125	1.01	12	7.06	6.97	7	0.47	8.78	7.2
125	1.01	13	6.61	9.37	7.09	0.47	7.73	7.86
125	1.01	13	6.67	6.97	7.32	0.56	10.52	13.86

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
125	1.01	14	6.5	6.82	6.47	0.48	9.36	10.05
125	1.01	14	6.92	6.9	6.9	0.42	9.2	8.14
125	1.01	15	6.85	7.12	6.59	0.45	9.61	8.75
125	1.01	15	6.61	6.78	6.8	1.17	17.86	12.38
150	0.01	6	0.59	0.6	0.6	0.27	4.06	8.58
150	0.01	6	0.67	0.65	0.65	0.3	4.47	8.13
150	0.01	7	0.69	0.71	0.71	0.36	4.64	10.67
150	0.01	7	0.66	0.65	0.65	0.31	4.95	10.44
150	0.01	8	0.68	0.68	0.68	0.34	5.72	11.84
150	0.01	8	0.72	0.77	0.73	0.38	5.86	11.22
150	0.01	9	0.6	0.6	0.6	0.34	8.63	13.58
150	0.01	9	0.77	0.76	0.76	0.33	5.38	12.63
150	0.01	10	0.7	0.69	0.69	0.41	6.31	14.73
150	0.01	10	0.67	0.67	0.68	0.38	5.95	13.98
150	0.01	11	0.7	0.75	0.78	0.38	7.31	16.36
150	0.01	11	0.68	0.69	0.68	0.33	6.05	14.48
150	0.01	12	0.7	0.7	0.7	0.33	7.14	16.83
150	0.01	12	0.7	0.71	0.73	0.78	8.92	24.78
150	0.01	13	0.82	0.79	0.79	0.42	8.53	19.06
150	0.01	13	0.74	0.75	0.74	0.44	10	19.28
150	0.01	14	0.66	0.72	0.69	0.56	9.31	19.97
150	0.01	14	0.73	0.73	0.73	0.44	8.48	18.2
150	0.01	15	0.66	0.69	0.68	0.38	8.64	18.98
150	0.01	15	0.63	0.63	0.63	0.39	9.02	20.03
150	0.11	6	2.44	2.32	2.56	0.27	5.38	5.39
150	0.11	6	1.9	1.93	1.96	0.28	4.28	4.08
150	0.11	7	2.1	2.1	2.1	0.34	4.55	3.97
150	0.11	8	2.16	2.13	2.13	0.33	5.66	4.27
150	0.11	8	2.13	2.24	2.24	0.33	5.64	4.41
150	0.11	9	2.01	2.08	2.01	0.36	6.13	4.58
150	0.11	9	2.27	2.32	2.32	0.36	5.7	4.55
150	0.11	10	2.07	2.04	2	0.39	7.28	6.81
150	0.11	10	2.53	2.55	2.54	0.41	6.77	5.81
150	0.11	11	2.19	2.22	2.22	0.34	6.33	4.61
150	0.11	12	2.41	2.34	2.34	0.38	7.78	5.59
150	0.11	12	2.13	2.18	2.24	0.59	8.28	11.38
150	0.11	13	2.22	2.03	2.12	0.42	9.17	9.3
150	0.11	13	2.6	2.52	2.46	0.5	10.47	8.69
150	0.11	14	2.23	2.31	2.22	0.44	10.66	9.03
150	0.11	14	2.22	2.19	2.23	0.39	8.55	6.8
150	0.11	15	2.31	2.31	2.29	0.41	9.78	9.11
150	0.11	15	2.58	2.72	2.65	0.45	9.69	8.41
150	0.21	6	3.07	3.17	3.16	0.3	3.8	2.95

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
150	0.21	6	2.71	2.54	2.59	0.27	4.78	3.48
150	0.21	7	2.96	2.81	2.78	0.31	4.89	4.19
150	0.21	7	2.75	2.76	2.76	0.34	5.22	4.64
150	0.21	8	3.47	3.43	3.41	0.45	6.66	5.83
150	0.21	8	3.18	2.96	2.98	0.34	5.88	4.88
150	0.21	9	3.96	3.72	3.85	0.36	6.48	5.77
150	0.21	9	3.19	3.06	3.06	0.38	6.19	6.61
150	0.21	10	2.9	2.96	2.96	0.38	6.41	5.42
150	0.21	10	3.2	3.33	3.07	0.38	6.22	6.09
150	0.21	11	3.24	3.22	3.33	0.41	9.2	7.97
150	0.21	11	3.34	3.48	3.48	0.44	6.97	6.42
150	0.21	12	2.96	3.73	2.95	0.47	8.75	7.72
150	0.21	12	3.09	2.98	2.98	0.63	11.45	11.67
150	0.21	13	2.91	3.2	3.33	0.44	9.55	8.36
150	0.21	13	3.24	3.16	3.15	0.44	11.13	10.25
150	0.21	14	2.79	2.74	2.89	0.5	10.02	9.13
150	0.21	14	3.22	3.55	3.05	0.48	10.2	8.88
150	0.21	15	3.26	3.33	3.33	0.41	9.77	8.48
150	0.21	15	2.88	2.82	2.88	0.47	12.72	8.92
150	0.31	6	3.74	3.56	3.56	0.39	5.14	4.2
150	0.31	6	3.32	3.41	3.41	0.33	5.25	4.73
150	0.31	7	3.4	3.23	3.43	0.36	4.94	4.19
150	0.31	7	3.2	3.23	3.23	0.44	5.72	6.28
150	0.31	8	3.4	3.42	3.37	0.33	5.13	4.67
150	0.31	8	3.91	3.72	3.71	0.36	6.09	4.88
150	0.31	9	3.59	3.62	3.59	0.36	5.95	5.52
150	0.31	9	3.6	3.71	3.69	0.39	5.97	4.58
150	0.31	10	3.38	3.28	3.28	0.45	6.39	4.61
150	0.31	10	3.73	3.66	3.66	0.39	7.14	6.75
150	0.31	11	4.04	3.99	4.02	0.42	10.9	6.95
150	0.31	11	4.57	4.57	4.51	0.36	8.38	5.94
150	0.31	12	3.63	3.91	3.91	0.5	8.98	7.95
150	0.31	12	3.54	3.57	3.65	0.55	13.05	8
150	0.31	13	3.98	3.81	3.8	0.44	8.64	7.73
150	0.31	13	3.68	3.69	3.68	0.48	12.03	8.86
150	0.31	14	3.55	3.73	3.62	0.47	10.44	9.61
150	0.31	14	4.18	3.94	4.28	0.42	9.2	7.5
150	0.31	15	3.5	3.55	3.43	0.39	10.88	9.25
150	0.31	15	3.81	4.04	3.69	0.59	12.33	12.33
150	0.41	6	3.42	3.43	3.43	0.31	4.13	3.44
150	0.41	6	3.75	3.69	3.62	0.31	4.08	3.36
150	0.41	7	4.81	4.52	4.53	0.36	5.06	4.63
150	0.41	7	3.89	3.83	3.83	0.34	6.63	5.95

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
150	0.41	8	4.53	4.58	4.52	0.39	6.38	5.53
150	0.41	8	3.87	3.92	3.91	0.36	5.72	4.55
150	0.41	9	3.98	4.06	4.01	0.39	7.73	7.55
150	0.41	9	4.39	4.16	4.57	0.33	6.5	6.27
150	0.41	10	4.7	4.64	4.63	0.36	7.89	8.08
150	0.41	10	4.62	4.63	4.63	0.38	6.78	5.84
150	0.41	11	4.27	3.93	3.93	0.41	7.94	6.53
150	0.41	11	4.56	4.5	4.28	0.39	9.39	8.13
150	0.41	12	4.71	5.07	4.77	0.38	10.28	9.34
150	0.41	12	4.35	4.71	4.54	0.39	8.97	8.44
150	0.41	13	4.59	4.55	4.57	0.47	11.14	10.36
150	0.41	13	4.04	4.14	4.14	0.48	10.17	9.72
150	0.41	14	4.14	4.11	4.09	0.55	13.47	10.48
150	0.41	14	4.66	4.42	4.42	0.38	9.59	9.7
150	0.41	15	4.33	4.47	4.37	0.44	10.23	9.58
150	0.41	15	4.48	4.31	4.36	0.47	9.81	7.53
150	0.51	6	4.02	4.23	4.18	0.3	4.13	3.83
150	0.51	6	4.09	4.68	4.07	0.31	4.63	3.83
150	0.51	7	4.6	4.39	4.35	0.36	6.49	4.39
150	0.51	7	4.53	6.44	4.68	0.42	7.56	5.59
150	0.51	8	4.74	4.6	4.83	0.36	6.91	5.83
150	0.51	8	4.86	4.71	4.66	0.42	6.81	5.89
150	0.51	9	5.55	5.19	4.94	0.41	7.86	6.31
150	0.51	9	4.8	4.63	4.88	0.38	7.22	7.63
150	0.51	10	5.35	5.13	5.05	0.41	8.8	7.31
150	0.51	10	4.88	4.69	4.84	0.47	8.03	7.78
150	0.51	11	5.45	5.34	5.35	0.52	10.2	9.73
150	0.51	11	4.78	4.76	4.84	0.38	6.66	6.56
150	0.51	12	5.5	5.21	5.24	0.47	9.36	13.39
150	0.51	12	4.58	4.51	4.51	0.39	9.59	9.91
150	0.51	13	5.13	4.68	4.76	0.44	10.22	8.14
150	0.51	13	4.72	5.07	5.1	0.42	9.56	11.41
150	0.51	14	4.39	4.57	4.59	0.5	10.88	8.81
150	0.51	14	4.85	5.05	4.73	0.45	9.75	8.5
150	0.51	15	4.71	5.08	4.91	0.5	9.58	8.28
150	0.51	15	4.35	4.3	4.22	0.45	12.33	9.75
150	0.61	6	4.67	4.76	4.76	0.31	4.31	4.06
150	0.61	6	4.73	4.77	4.72	0.28	5.02	4.31
150	0.61	7	4.51	4.86	4.55	0.34	6.67	5.75
150	0.61	7	4.27	4.47	4.44	0.36	6.91	5.52
150	0.61	8	5.18	5.04	5.04	0.36	5.77	4.36
150	0.61	8	5.05	5.35	5.03	0.41	6.47	5.84
150	0.61	9	5.12	4.97	4.98	0.36	7.11	5.66

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
150	0.61	9	5.25	5.39	5.34	0.36	7.28	5.95
150	0.61	10	5.68	5.41	5.64	0.38	8.03	9.25
150	0.61	10	5.08	5.4	5.35	0.41	8.7	7.78
150	0.61	11	5.04	5.11	5.06	0.59	11.47	9.48
150	0.61	11	4.76	4.77	4.81	0.38	10.02	6.91
150	0.61	12	5.19	5.12	5.17	0.42	8.97	7.67
150	0.61	12	5.24	5.4	5.45	0.38	8.8	8.75
150	0.61	13	5.15	5.23	5.19	0.44	10.11	7.7
150	0.61	13	5.23	5.55	5.33	0.48	10.81	9.09
150	0.61	14	4.98	4.71	4.87	0.5	12.34	11.08
150	0.61	14	5.93	6.1	5.89	0.42	10.05	11.36
150	0.61	15	5.64	5.95	5.44	0.44	12.83	10.92
150	0.61	15	6.32	6.25	6.23	0.41	11.42	10.17
150	0.71	6	5.29	5.73	5.31	0.42	4.8	3.72
150	0.71	6	5.03	5.03	4.85	0.3	5.23	3.92
150	0.71	7	4.95	4.99	4.9	0.34	6.02	5.38
150	0.71	7	4.93	4.99	4.98	0.36	6.39	4.64
150	0.71	8	5.71	6.28	5.64	0.36	7.61	7.7
150	0.71	8	4.93	4.96	4.89	0.58	7.55	6.39
150	0.71	9	5.5	5.51	5.52	0.36	7.53	6.55
150	0.71	9	5.2	5.08	5.16	0.34	7.14	6.64
150	0.71	10	5.72	5.66	5.43	0.41	8.47	7.97
150	0.71	10	5.61	5.7	5.46	0.41	7.36	7.59
150	0.71	11	5.83	5.52	5.52	0.55	9.31	6.33
150	0.71	11	5.53	5.36	5.52	0.39	7.94	9.13
150	0.71	12	6.98	6.57	7.03	0.38	10.48	9.02
150	0.71	12	5.86	6.15	5.98	0.38	9.06	8.59
150	0.71	13	5.69	5.67	5.9	0.47	11.86	9.84
150	0.71	13	5.4	5.24	5.28	0.44	11.78	8.77
150	0.71	14	5.67	5.44	5.34	0.48	12.56	10.83
150	0.71	14	5.7	5.82	5.78	0.45	12.09	9.03
150	0.71	15	5.79	6.08	5.84	0.44	11.55	9.88
150	0.71	15	5.58	5.91	5.62	0.44	12.61	12.69
150	0.81	6	6.37	24.09	6.39	0.28	5.86	5.75
150	0.81	6	5.34	5.31	5.37	0.34	6.3	4.53
150	0.81	7	5.93	5.92	5.93	0.39	7.75	7.36
150	0.81	7	5.92	6.4	6.02	0.36	5.31	4.89
150	0.81	8	5.69	5.74	5.73	0.41	6.34	5.2
150	0.81	8	5.77	5.91	5.75	0.36	6.73	6.23
150	0.81	9	5.99	6.49	5.93	0.42	9.31	7.13
150	0.81	9	6.32	6.86	6.32	0.42	7.8	7.45
150	0.81	10	5.28	5.65	5.35	0.39	8.61	8.55
150	0.81	10	6.17	5.78	5.77	0.38	7.52	7.81

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
150	0.81	11	6.71	6.64	6.78	0.44	11.14	8.73
150	0.81	11	5.52	5.93	5.62	0.36	9.38	7.23
150	0.81	12	6.26	7.85	6.16	0.42	10.23	8.41
150	0.81	12	6.12	6.46	6.11	0.38	10.7	9.56
150	0.81	13	6.13	6.13	6.12	0.42	11.42	9.61
150	0.81	13	5.82	5.82	5.78	0.42	11.91	11.45
150	0.81	14	5.53	5.55	5.54	0.5	12.14	9.31
150	0.81	14	5.66	5.69	5.72	0.41	10.63	9.36
150	0.81	15	5.81	6.01	5.92	0.41	11.72	9.72
150	0.81	15	5.92	5.88	5.86	0.5	10.78	8.16
150	0.91	6	5.82	5.87	5.88	0.31	5.11	4.38
150	0.91	7	7.31	8.76	7.68	0.41	6.47	5.91
150	0.91	7	5.56	6.24	5.53	0.45	8.58	6.06
150	0.91	8	6.81	7.47	7.32	0.38	6.84	6.6
150	0.91	8	5.76	5.76	5.76	0.34	6.59	5.84
150	0.91	9	6.72	6.82	6.7	0.41	8.44	7.95
150	0.91	9	5.65	5.69	5.59	0.36	7.58	6.56
150	0.91	10	6.87	6.75	6.4	0.41	8.59	9
150	0.91	10	6.26	6.26	6.28	0.36	12.39	9.23
150	0.91	11	8.09	7.92	7.65	0.44	10.16	9.27
150	0.91	11	6.11	6.13	6.12	0.34	8.58	7.53
150	0.91	12	7.01	7	7.02	0.45	10.06	10.47
150	0.91	12	5.86	5.83	6.14	0.41	10.77	9.38
150	0.91	13	5.92	6.02	6.35	0.42	10.14	8.7
150	0.91	13	6.83	6.75	6.82	0.48	11.08	10.22
150	0.91	14	6.12	6.2	6.19	0.47	10.58	10.86
150	0.91	14	6.8	6.36	6.48	0.55	11.06	8.83
150	0.91	15	6.14	6.2	6.21	0.44	11.77	9.39
150	0.91	15	6.58	6.62	7.4	0.48	13.33	13.66
150	1.01	6	7.05	6.25	5.86	0.38	5.61	3.97
150	1.01	6	6.24	6.85	6.32	0.31	5.84	4.53
150	1.01	7	5.83	5.84	5.91	0.34	6.45	5.19
150	1.01	7	6.03	6.09	6.24	0.34	5.95	5.27
150	1.01	8	5.75	5.67	5.71	0.39	7.22	5.67
150	1.01	8	7.44	6.47	6.61	0.34	7.19	6.56
150	1.01	9	6.7	7.14	7.14	0.44	10.02	6.48
150	1.01	9	6.2	6.9	6.83	0.38	7.77	6.98
150	1.01	10	6.54	7.2	6.52	0.41	8.55	8.31
150	1.01	10	6.26	6.71	6.14	0.45	9.08	8
150	1.01	11	7.71	8.66	7.67	0.47	10.58	9.63
150	1.01	11	6.64	6.67	6.64	0.39	9.17	7.77
150	1.01	12	6.95	7.18	7.09	0.45	10.61	11.25
150	1.01	12	6.95	7.45	7.14	0.41	8.86	7.7

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
150	1.01	13	7.48	7.34	7.26	0.44	13.31	10.77
150	1.01	13	6.81	6.9	6.79	0.45	12.13	12.11
150	1.01	14	6.61	6.58	6.52	0.48	11.83	8.8
150	1.01	14	6.75	7.01	6.65	0.45	10.64	9.19
150	1.01	15	7.59	7.61	7.71	0.45	12.61	12.58
150	1.01	15	6.38	6.57	6.31	0.52	11.89	10.52
175	0.01	6	0.7	0.66	0.67	0.28	4.66	9.86
175	0.01	6	0.69	0.66	0.65	0.27	5.2	11.14
175	0.01	7	0.61	0.65	0.66	0.3	5.67	12
175	0.01	7	4.12	0.58	0.58	0.31	5.58	11.69
175	0.01	8	0.59	0.6	0.61	0.34	6.08	14.05
175	0.01	8	0.65	0.66	0.66	0.3	5.5	12.53
175	0.01	9	0.63	0.65	0.62	0.38	7.22	16.3
175	0.01	9	0.7	0.68	0.68	0.33	6.58	14.28
175	0.01	10	0.76	0.76	0.76	0.36	7.8	18.42
175	0.01	10	0.65	0.64	0.64	0.39	7.63	17.77
175	0.01	11	0.65	0.63	0.63	0.38	8.48	20.31
175	0.01	11	0.69	0.69	0.68	0.39	8.52	18.91
175	0.01	12	0.67	0.68	0.68	0.39	8.89	20.23
175	0.01	12	0.63	0.67	0.64	0.38	8.88	20.22
175	0.01	13	0.66	0.67	0.67	0.42	10.28	21.52
175	0.01	13	0.65	0.67	0.67	0.45	10.16	23.14
175	0.01	14	0.71	0.7	0.7	0.42	11.17	25.08
175	0.01	14	0.68	0.67	0.67	0.42	10.27	23.05
175	0.01	15	0.74	0.76	0.73	0.41	10.3	22.97
175	0.01	15	0.66	0.65	0.63	0.39	10.81	24.17
175	0.11	6	2.23	2.24	2.25	0.28	4.45	3.88
175	0.11	6	1.81	1.83	1.88	0.28	5.36	5.03
175	0.11	7	2.55	1.95	1.99	0.45	6.11	5.34
175	0.11	7	1.95	2	1.92	0.38	5.52	3.77
175	0.11	8	1.88	2.22	1.9	0.33	6.33	5.89
175	0.11	8	2.16	2.37	2.26	0.36	6.09	5.89
175	0.11	9	2.13	2.4	2.13	0.38	7.27	5.88
175	0.11	9	1.94	2.08	1.98	0.39	7.09	5.23
175	0.11	10	2.24	2.41	2.24	0.41	8.36	8.13
175	0.11	10	2.36	2.35	2.36	0.39	7.86	5.97
175	0.11	11	2.37	2.46	2.46	0.39	8.98	7.16
175	0.11	11	2.18	2.19	2.18	0.36	8.53	8.47
175	0.11	12	2.19	2.49	2.05	0.38	10.27	8.88
175	0.11	12	2.42	2.48	2.37	0.39	9.58	9.09
175	0.11	13	2.43	2.38	2.38	0.36	10.72	9.42
175	0.11	13	2.25	2.3	2.31	0.45	11.16	9.05
175	0.11	14	2.29	2.25	2.24	0.45	11.61	10.78

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
175	0.11	14	2.23	2.36	2.36	0.5	10.39	7.59
175	0.11	15	2.51	2.41	2.5	0.39	10.8	9.25
175	0.11	15	2.37	2.46	2.33	0.42	14.61	10.48
175	0.21	6	3.04	3.22	2.94	0.33	5.39	4.91
175	0.21	6	3.56	7.15	3.36	0.31	8.18	6.3
175	0.21	7	2.87	2.83	2.86	0.36	6.31	5.34
175	0.21	7	3.19	3.11	3.19	0.31	6.8	5.34
175	0.21	8	3.42	3.5	3.49	0.34	7.69	5.41
175	0.21	8	2.71	2.79	2.8	0.36	6.89	5.5
175	0.21	9	3.54	3.32	3.33	0.39	8.59	8.63
175	0.21	9	3.12	3.27	3.18	0.34	7.53	5.89
175	0.21	10	3.06	3.1	3.16	0.39	9.11	9.14
175	0.21	10	3.34	3.21	3.12	0.36	8	7.92
175	0.21	11	2.92	3.07	2.96	0.39	10.38	9.28
175	0.21	11	3.12	3.36	3.17	0.45	10.67	10.7
175	0.21	12	2.69	2.74	2.66	0.42	10.53	8.92
175	0.21	12	3.55	3.52	3.52	0.47	10.47	10.03
175	0.21	13	3.53	3.59	3.45	0.5	10.19	12.2
175	0.21	13	3.06	3.14	3.12	0.39	10.81	11.78
175	0.21	14	3.27	3.28	3.27	0.45	11.5	11.17
175	0.21	14	2.89	3.06	3.07	0.44	11.67	12
175	0.21	15	3.1	3.15	3.1	0.41	11.84	11.08
175	0.21	15	3.38	3.38	3.38	0.45	12.66	10.11
175	0.31	6	4.44	4.21	4.26	0.27	5.41	6.28
175	0.31	6	3.72	3.59	3.59	0.28	5.03	4.97
175	0.31	7	3.48	3.38	3.48	0.33	6.19	4.89
175	0.31	7	4.27	3.48	3.52	0.36	5.72	5.58
175	0.31	8	4.35	4.18	4.17	0.36	7.44	6.27
175	0.31	8	4.1	3.9	3.83	0.41	6.34	6.17
175	0.31	9	3.76	3.83	3.83	0.39	7.72	7.06
175	0.31	9	3.99	4.27	3.98	0.34	7.75	7.97
175	0.31	10	3.82	3.79	3.9	0.44	8.93	6.48
175	0.31	10	3.76	3.68	3.82	0.36	8.2	8.14
175	0.31	11	3.84	3.7	3.77	0.42	9.89	9.97
175	0.31	11	3.86	3.81	3.73	0.42	10.28	10.16
175	0.31	12	3.57	3.52	3.66	0.45	12.36	10.16
175	0.31	12	3.77	4.01	3.78	0.41	10.03	9.38
175	0.31	13	3.68	4.45	3.87	0.41	11.28	10.34
175	0.31	13	3.72	3.73	3.73	0.47	9.78	10.14
175	0.31	14	3.71	3.72	3.9	0.45	12.13	15.87
175	0.31	14	3.75	4.25	4.21	0.5	11.47	11.5
175	0.31	15	3.99	4.09	4.07	0.41	11.52	10.53
175	0.31	15	3.67	4.18	3.74	0.53	13.03	12.94

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
175	0.41	6	7.82	3.9	3.66	0.31	4.8	4.11
175	0.41	6	3.98	4	3.97	0.31	5.02	4.67
175	0.41	7	3.91	3.87	3.75	0.3	6.89	6.2
175	0.41	7	4.57	5.02	4.31	0.31	6.61	5.95
175	0.41	8	4.34	7.75	4.38	0.36	7.13	6.13
175	0.41	8	4.22	4.22	4.46	0.36	7.31	6.42
175	0.41	9	4.27	4.3	4.05	0.39	10.56	10.17
175	0.41	9	4.32	4.18	4.11	0.34	8.67	9.34
175	0.41	10	4.47	4.46	4.49	0.44	10.11	10.03
175	0.41	10	4.76	4.66	4.78	0.42	10.91	11.05
175	0.41	11	4.14	4.26	4.01	0.41	9.09	8.5
175	0.41	11	4.65	4.71	4.5	0.39	9.8	9.7
175	0.41	12	4.69	4.62	4.65	0.45	11.27	11.47
175	0.41	13	4.2	4.09	3.97	0.42	10.42	8.69
175	0.41	13	4.79	4.71	4.67	0.44	11.42	11.25
175	0.41	14	4.52	4.26	4.42	0.48	12.75	11.45
175	0.41	14	4.77	4.8	4.79	0.47	11.75	13.34
175	0.41	15	4.57	4.54	4.51	0.44	12.69	13.89
175	0.41	15	4.09	4.01	4.02	0.48	15.34	13.64
175	0.51	6	6.89	4.18	4.11	0.31	4.95	3.19
175	0.51	6	4.41	4.61	4.58	0.31	5.88	5.31
175	0.51	7	4.62	4.59	4.59	0.36	6.3	5.23
175	0.51	7	4.69	4.57	4.69	0.31	7.5	7.09
175	0.51	8	4.55	6.49	4.66	0.34	8	8.7
175	0.51	8	4.42	4.38	4.24	0.47	7.81	8.56
175	0.51	9	4.67	4.67	5.15	0.39	11.95	10.05
175	0.51	9	5.24	5.22	5.08	0.44	9.38	7.67
175	0.51	10	4.91	4.89	4.91	0.39	10.45	10.34
175	0.51	10	4.67	4.71	4.65	0.39	8.88	7.63
175	0.51	11	5.14	5.88	4.95	0.56	11.48	10.3
175	0.51	11	5.05	5.03	5	0.39	11.63	10
175	0.51	12	4.91	5.04	4.86	0.42	12.23	12.08
175	0.51	12	4.92	5.06	4.78	0.42	11.2	9.66
175	0.51	13	5.25	5.25	5.23	0.44	11.27	11.33
175	0.51	13	5.14	5.14	5.14	0.5	11.34	10.55
175	0.51	14	4.63	4.5	4.5	0.52	13.61	16.3
175	0.51	14	5.01	5.18	5.15	0.45	12.41	10.03
175	0.51	15	5.63	5.82	5.66	0.45	13	10.69
175	0.51	15	5.7	5.76	5.71	0.42	13.58	15.5
175	0.61	6	5.38	5.07	5.23	0.27	7.72	5.83
175	0.61	6	4.8	4.92	4.76	0.33	5.89	6.34
175	0.61	7	5.34	5.31	5.42	0.38	7.2	8.38
175	0.61	7	5.5	5.38	5.61	0.31	5.95	6.27

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
175	0.61	8	5.31	5.39	5.43	0.36	8.53	7.2
175	0.61	8	5.16	5.03	5.13	0.36	8.61	7.7
175	0.61	9	5.44	5.45	5.46	0.38	9.66	8.81
175	0.61	9	5.46	5.55	5.95	0.33	9.02	9.59
175	0.61	10	5.7	5.2	5.31	0.41	9.48	8.08
175	0.61	10	5.5	5.29	5.17	0.38	8.92	8.11
175	0.61	11	5.44	5.55	5.34	0.42	12.06	12.95
175	0.61	11	6.34	6.18	6.26	0.42	10.94	10.11
175	0.61	12	5.76	5.71	5.77	0.48	11.66	10.44
175	0.61	12	4.92	5.1	4.77	0.41	11.05	10.41
175	0.61	13	5.31	5.26	5.35	0.48	12.2	8.83
175	0.61	13	5.74	5.63	5.61	0.5	14.36	11.72
175	0.61	14	5.39	5.39	5.38	0.45	13.13	11.02
175	0.61	14	5.09	5.37	5.05	0.42	13.91	12.56
175	0.61	15	5.36	5.39	5.39	0.42	13.09	10.92
175	0.61	15	5.94	5.82	5.91	0.47	14.66	14.27
175	0.71	6	5.89	6.1	5.61	0.3	6.49	5.28
175	0.71	6	5.46	5.5	5.45	0.38	5.7	5.14
175	0.71	7	5.44	5.78	5.45	0.31	7.81	6.3
175	0.71	7	6.03	5.96	5.82	0.34	7.11	7.02
175	0.71	8	5.21	5.31	5.34	0.39	8.55	6
175	0.71	8	6.04	5.97	6.03	0.36	8.34	8.33
175	0.71	9	5.73	6.09	5.89	0.39	10.36	7.91
175	0.71	9	5.69	5.6	5.55	0.34	11.36	9.38
175	0.71	10	6.04	5.95	6.04	0.42	9.42	9.7
175	0.71	10	6.63	6.77	6.77	0.44	11.13	12.02
175	0.71	11	5.46	5.36	5.38	0.47	11.7	10.25
175	0.71	11	7.16	7.02	7.06	0.44	10.39	9.38
175	0.71	12	5.35	5.36	5.36	0.42	11.45	9.92
175	0.71	12	5.97	5.9	5.92	0.42	11.31	10.58
175	0.71	13	5.87	5.88	5.88	0.47	11.69	11.28
175	0.71	13	6.12	5.01	5.15	0.55	16.16	13.59
175	0.71	14	5.9	5.87	5.82	0.44	14.55	13.78
175	0.71	14	6.02	6.25	5.92	0.45	14.48	14.52
175	0.71	15	5.93	6.08	6.04	0.47	13.98	13.64
175	0.71	15	6.34	6.12	6.13	0.92	13.73	11
175	0.81	6	6.17	6.1	6.02	0.3	6.88	6.05
175	0.81	6	5.74	5.63	5.8	0.28	5.53	5.48
175	0.81	7	5.96	5.74	5.77	0.36	7.78	5.95
175	0.81	7	6.38	6.06	6.33	0.41	6.97	5.98
175	0.81	8	6.12	6.5	6.24	0.38	9.55	7.48
175	0.81	8	5.47	5.9	5.6	0.39	9.58	5.59
175	0.81	9	5.79	5.79	5.79	0.38	9.45	9.27

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
175	0.81	9	5.81	13.08	5.79	0.38	8.98	7.41
175	0.81	10	6	5.97	5.74	0.39	10.61	9.63
175	0.81	10	6.81	6.19	6.6	0.38	11.45	9.38
175	0.81	11	6.44	6.94	6.27	0.45	12.73	11.31
175	0.81	11	5.38	5.41	5.41	0.45	11.23	11.47
175	0.81	12	5.61	5.65	5.61	0.44	13.25	10.34
175	0.81	12	6.04	6.69	6	0.38	14.25	10.14
175	0.81	13	6.69	6.61	6.63	0.44	12.05	10.8
175	0.81	13	5.63	5.85	5.9	0.47	17.61	15.67
175	0.81	14	6.48	6.32	6.41	0.45	14.39	15.67
175	0.81	14	5.61	5.89	5.81	0.45	13.23	12.31
175	0.81	15	6.49	6.64	6.45	0.44	14.27	13.53
175	0.81	15	6.34	6.62	6.33	0.48	14.86	13.25
175	0.91	6	5.4	5.44	5.45	0.59	5.41	4.64
175	0.91	6	6.54	6.11	6.05	0.3	5.7	5.23
175	0.91	7	6.18	6.34	6.09	0.31	7.09	5.92
175	0.91	7	7.24	7.19	7.16	0.33	6.66	7.03
175	0.91	8	7.17	6.77	6.78	0.38	8.02	7.34
175	0.91	8	6.07	6.49	6.29	0.38	9.03	10.58
175	0.91	9	6.04	6.57	5.92	0.39	10.08	12.17
175	0.91	9	6.92	6.83	6.65	0.38	10.58	10.3
175	0.91	10	6.71	6.56	6.57	0.41	11.09	8.28
175	0.91	10	6.48	6.43	6.36	0.42	11.06	9.22
175	0.91	11	6.86	7.04	6.65	0.42	12.75	11.86
175	0.91	11	6.85	6.99	6.91	0.42	11.52	8.5
175	0.91	12	6.42	7.67	6.36	0.42	11.75	8.59
175	0.91	12	7.18	8.04	7.21	0.42	13.45	13.53
175	0.91	13	7.71	7.59	7.77	0.45	15.53	13.41
175	0.91	13	6.15	5.91	5.86	0.53	15.39	14.84
175	0.91	14	6.49	6.75	6.8	0.44	16.44	14.75
175	0.91	14	6.28	6.39	6.27	0.45	14.22	14.86
175	0.91	15	7.25	8.82	7.04	0.48	15.33	16.45
175	0.91	15	5.86	5.82	5.83	0.47	16.52	13.47
175	1.01	6	6.54	7.46	6.43	0.3	6.34	6.67
175	1.01	6	7.27	7.2	7.07	0.33	7.02	4.61
175	1.01	7	6.22	6.28	6.18	0.33	7.44	7.11
175	1.01	7	5.58	5.57	5.62	0.33	8.25	7.39
175	1.01	8	6.82	7.42	7.43	0.41	9.2	8.52
175	1.01	8	6.37	6.46	6.19	0.36	8.25	7.48
175	1.01	9	7.74	7.52	7.29	0.47	9.02	7.69
175	1.01	9	7.07	6.63	6.83	0.39	8.78	7.59
175	1.01	10	6.81	6.49	6.64	0.44	10.52	9.31
175	1.01	10	7.18	7.17	7.2	0.41	10.19	10.33

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
175	1.01	11	7.55	7.25	7.42	0.42	12.33	11.75
175	1.01	11	7.24	7.4	6.88	0.41	12.56	10.77
175	1.01	12	6.76	6.53	6.7	0.44	17.95	14.33
175	1.01	12	6.99	7.05	6.53	0.39	12.5	10.52
175	1.01	13	7.24	7.14	7.16	0.5	13.33	12.42
175	1.01	13	6.33	6.4	6.14	0.52	15.53	13.97
175	1.01	14	8	8.07	7.75	0.47	16.2	14.14
175	1.01	14	6.78	6.76	6.77	0.47	14.77	13.89
175	1.01	15	7.11	6.91	6.98	0.53	18.11	14.83
175	1.01	15	6.28	6.36	6.31	0.48	15.06	12.73
200	0.01	6	0.6	0.58	0.59	0.28	7.2	14.07
200	0.01	6	0.72	1.02	0.7	0.27	5.52	11.59
200	0.01	7	0.69	0.66	0.7	0.28	7.02	14.56
200	0.01	7	0.75	0.71	0.72	0.31	6.83	14.56
200	0.01	8	0.65	0.62	0.62	0.34	7.78	17.08
200	0.01	8	0.71	0.71	0.71	0.33	6.64	14.22
200	0.01	9	0.68	0.7	0.69	0.33	9.86	19.06
200	0.01	9	0.67	0.67	0.67	0.36	7.78	18.22
200	0.01	10	0.67	0.69	0.69	0.36	9.5	21.23
200	0.01	10	0.68	0.7	0.67	0.33	9	19.42
200	0.01	11	0.69	0.69	0.7	0.38	10.59	23.98
200	0.01	11	0.72	0.69	0.68	0.39	10.44	21.94
200	0.01	12	0.71	0.73	0.72	0.64	13.25	78.2
200	0.01	12	0.72	0.71	0.71	0.38	10.53	22.67
200	0.01	13	0.78	0.77	0.77	0.39	11.56	25.84
200	0.01	13	0.64	0.64	0.64	0.56	17.33	44.81
200	0.01	14	0.71	0.69	0.7	0.44	13.55	30.39
200	0.01	14	0.74	0.75	0.73	0.41	12.94	27.55
200	0.01	15	0.71	0.7	0.69	0.39	12.27	28.22
200	0.01	15	0.74	0.78	0.76	0.44	14.17	29.56
200	0.11	6	7.23	2.14	2.13	0.33	5.7	4.23
200	0.11	7	2.27	2.28	2.29	0.3	7.36	6.5
200	0.11	7	2.05	2.06	2.06	0.33	6.44	4.39
200	0.11	8	2.35	2.35	2.33	0.34	8.44	6.81
200	0.11	8	2.09	2.12	2.07	0.31	7.72	5.34
200	0.11	9	2.21	2.22	2.21	0.41	8.61	5.97
200	0.11	9	2.38	2.31	2.33	0.39	9.23	7.05
200	0.11	10	2.13	2.15	2.15	0.41	9.78	6.47
200	0.11	10	2.34	2.32	2.29	0.33	9.02	7.33
200	0.11	11	2.27	2.37	2.3	0.44	10.73	9.77
200	0.11	11	2.36	2.3	2.33	0.36	9.72	8.31
200	0.11	12	2.31	2.38	2.29	0.95	11.98	13.06
200	0.11	12	2.39	2.39	2.4	0.41	10.94	7.94

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
200	0.11	13	2.33	2.33	2.33	0.42	11.95	9.56
200	0.11	13	2.31	2.37	2.36	0.47	20.05	12.81
200	0.11	14	2.26	2.27	2.36	0.48	17.45	9.7
200	0.11	14	2.29	2.22	2.22	0.42	13.98	9.7
200	0.11	15	2.37	2.47	2.45	0.39	14.53	11.08
200	0.11	15	2.27	2.26	2.25	0.42	12.94	9.23
200	0.21	6	3.08	2.89	2.89	0.31	5.61	4.19
200	0.21	6	2.75	2.77	2.74	0.3	6.08	4.14
200	0.21	7	3.23	3.05	3.04	0.36	7.08	5.63
200	0.21	7	2.87	2.89	2.9	0.34	6.8	5.39
200	0.21	8	3.04	3.01	3.09	0.36	9.91	6.88
200	0.21	8	3.15	3.05	3.12	0.3	8.27	6
200	0.21	9	3.08	3.31	3.17	0.34	9.91	9.41
200	0.21	9	3.14	3.5	3.31	0.34	9.34	7.38
200	0.21	10	3.09	3.18	3.1	0.47	9.77	8.75
200	0.21	10	3.02	3.09	3.17	0.38	13.44	12.25
200	0.21	11	3.5	4.19	3.52	0.41	10.69	8.69
200	0.21	11	3.06	3	3.18	0.36	11.11	9.3
200	0.21	12	3.37	3.33	3.37	0.44	11.17	9.94
200	0.21	12	3.26	3.26	3.04	0.42	12.3	9.95
200	0.21	13	3.36	3.35	3.38	0.44	14.78	10.66
200	0.21	13	2.93	3	2.97	0.48	14.06	11.64
200	0.21	14	3.26	3.29	3.34	0.48	13.59	12.28
200	0.21	14	3.17	3.2	3.18	0.42	13.28	10.44
200	0.21	15	3.05	3	3	0.45	12.09	11.59
200	0.21	15	3.33	3.31	3.27	0.45	15.67	12.36
200	0.31	6	3.71	3.67	3.62	0.3	6.85	5.38
200	0.31	6	3.38	3.31	3.28	0.34	6.42	5.75
200	0.31	7	3.7	3.81	3.65	0.38	7.92	5.28
200	0.31	7	3.6	3.85	5.18	0.36	7.67	5.56
200	0.31	8	4	4	4	0.34	7.56	7.45
200	0.31	8	3.48	4.19	4.05	0.34	6.75	5.5
200	0.31	9	3.47	3.6	3.5	0.34	9.64	8.61
200	0.31	9	3.89	3.97	3.72	0.34	9.81	8.39
200	0.31	10	3.94	4.02	3.86	0.42	10.14	9.53
200	0.31	10	3.94	4.12	4.08	0.97	10.91	6.94
200	0.31	11	3.58	3.5	3.69	0.42	11.03	9.06
200	0.31	11	3.78	3.48	3.75	0.38	12.08	9.23
200	0.31	12	3.64	3.71	3.61	0.41	14.09	10.19
200	0.31	12	3.72	3.67	3.72	0.41	11.02	7.56
200	0.31	13	3.61	3.67	4.42	0.45	13.45	10.28
200	0.31	13	3.9	4.04	3.89	0.73	19.67	13.31
200	0.31	14	4.14	4.2	4.18	0.53	19.59	11.19

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
200	0.31	14	3.43	3.56	3.54	0.44	13.78	10.67
200	0.31	15	3.7	3.7	3.69	0.45	13.7	13.31
200	0.31	15	4	3.99	4.04	0.47	16.58	13.75
200	0.41	6	4.34	4.41	4.93	0.38	6.75	5.61
200	0.41	6	4.46	4.5	4.88	0.42	6.19	4.89
200	0.41	7	3.87	3.94	3.9	0.39	10.27	6.2
200	0.41	7	4.29	4.2	4.15	0.33	7.47	7.61
200	0.41	8	4.17	4.22	4.12	0.41	9.03	6.66
200	0.41	8	4.39	4.56	4.3	0.38	7.64	5.06
200	0.41	9	4.79	4.38	4.59	0.36	11.13	7.73
200	0.41	9	4.48	4.71	4.67	0.38	10.09	9.19
200	0.41	10	4.34	4.22	4.67	0.42	11.14	8.61
200	0.41	10	4.59	4.44	4.55	0.38	12.42	6.67
200	0.41	11	4.23	4.82	4.23	0.47	12.88	9.59
200	0.41	11	4.15	4.36	4.12	0.34	12.59	9.81
200	0.41	12	4.56	4.6	4.5	0.41	12.59	9.59
200	0.41	12	4.55	4.66	4.62	0.44	12	9.3
200	0.41	13	4.55	4.66	4.77	0.52	15.83	11.77
200	0.41	13	4.75	4.56	4.58	1.06	21.89	18.48
200	0.41	14	4.56	4.52	4.55	0.53	17.16	13.55
200	0.41	14	4.73	4.85	4.83	0.41	14.09	13.72
200	0.41	15	4.39	4.43	4.42	0.41	14.34	12.17
200	0.41	15	4.53	4.38	4.26	0.48	16.41	10.41
200	0.51	6	4.93	5.05	5.07	0.36	10.56	7.02
200	0.51	6	4.11	4.17	4.55	0.39	6.52	5.86
200	0.51	7	4.75	5.19	4.79	0.97	7.53	5.89
200	0.51	7	4.3	4.41	4.3	0.34	7.63	6.05
200	0.51	8	4.82	4.65	5.09	0.39	9.17	6.92
200	0.51	8	4.66	4.79	4.59	0.42	9.05	7.73
200	0.51	9	4.8	4.56	4.61	0.38	10.28	6.39
200	0.51	9	4.94	4.94	5.03	0.45	9.61	8.83
200	0.51	10	5.22	6.13	5.67	0.42	11.5	9.8
200	0.51	10	4.87	5.03	4.68	0.64	11.53	10.36
200	0.51	11	5.26	5.39	5.91	0.55	12.44	9.13
200	0.51	11	4.96	5.06	5.09	0.45	13.05	9.63
200	0.51	12	5.27	5.29	5.18	0.47	13.92	9.66
200	0.51	12	4.52	4.6	5.04	0.45	12.22	8.98
200	0.51	13	4.69	4.79	4.83	0.53	16.11	12.34
200	0.51	13	4.91	5.02	4.82	0.56	16.06	11.75
200	0.51	14	4.88	4.89	4.9	0.55	14.5	12.14
200	0.51	14	5.03	5.03	5.64	0.44	16.81	12.75
200	0.51	15	5.69	5.54	6.25	0.47	21.45	12
200	0.51	15	4.85	4.68	4.65	0.47	16.98	13

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
200	0.61	6	5.3	5.26	5.54	0.42	8.85	5.84
200	0.61	6	5.17	4.96	4.96	0.3	6.19	4.98
200	0.61	7	4.85	5.31	4.92	0.3	7.27	7.95
200	0.61	7	4.82	4.82	5.08	0.34	8.08	7.22
200	0.61	8	5.47	5.54	5.66	0.36	10.19	6.73
200	0.61	8	5.21	5.1	4.89	0.34	8.34	6.39
200	0.61	9	4.8	4.87	4.98	0.41	9.91	6.64
200	0.61	9	4.84	4.74	4.75	0.38	11.3	7.64
200	0.61	10	5.39	5.5	5.75	0.39	12.52	8.64
200	0.61	10	5.26	5.29	5.47	0.48	11.33	8.52
200	0.61	11	5.36	5.41	5.32	0.41	14.5	12.91
200	0.61	11	5.43	5.38	5.4	0.41	13.31	10.44
200	0.61	12	5.97	5.47	5.77	0.47	15.41	11.08
200	0.61	12	5.71	5.79	5.84	0.41	14.73	10.45
200	0.61	13	5.55	5.58	5.55	0.52	14.42	11.56
200	0.61	13	5.41	5.67	5.4	0.53	20.44	11.92
200	0.61	14	5.04	5.09	5.13	0.52	19.69	19.09
200	0.61	14	6.45	6.5	6.7	0.44	16.59	13.95
200	0.61	15	5.81	5.09	5.42	0.47	15.16	13.75
200	0.61	15	5.51	5.49	5.51	0.5	16.41	14.52
200	0.71	6	4.91	5.03	4.95	0.38	8.58	6.85
200	0.71	6	5.33	5.74	5.41	0.31	6.77	4.41
200	0.71	7	5.41	5.42	5.49	0.48	9.17	6.75
200	0.71	7	6.26	6.1	6.44	0.47	9.3	7.48
200	0.71	8	5.84	5.97	5.89	0.41	10.25	7.05
200	0.71	8	6.22	6.43	6.38	0.38	10.72	9.23
200	0.71	9	6.04	6.12	6.03	0.38	9.97	6.36
200	0.71	9	5.79	5.86	5.63	0.45	11.45	9.89
200	0.71	10	5.11	5.14	5.06	0.42	12.16	10.55
200	0.71	10	5.73	5.79	5.91	0.39	11.23	8.28
200	0.71	11	5.44	5.46	5.53	0.42	17.2	10.7
200	0.71	11	5.98	5.84	5.72	0.39	12.44	7.88
200	0.71	12	5.53	5.6	5.52	0.42	14.52	13.48
200	0.71	12	6.36	6.18	6.15	0.38	16.33	11.92
200	0.71	13	5.75	5.91	6.13	0.44	15.03	11.05
200	0.71	13	6.08	6.23	6.11	0.55	16.73	20.95
200	0.71	14	5.5	5.52	5.63	0.47	17.95	14.17
200	0.71	15	6.03	6.1	6.02	0.45	17.94	12.03
200	0.71	15	6.31	6.34	6.44	0.48	21.78	14.14
200	0.81	6	5.6	5.57	5.67	0.56	9.08	6.14
200	0.81	6	6.36	6.11	6.7	0.31	9.19	4.83
200	0.81	7	6.21	5.87	6.01	0.31	9.13	6.09
200	0.81	7	6.26	6.3	6.75	0.34	9.48	7.95

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
200	0.81	8	5.83	5.87	6.95	0.41	12.02	8.3
200	0.81	8	6.04	6.43	6.42	0.48	9.84	7.41
200	0.81	9	6	5.89	6.06	0.41	12.23	9.83
200	0.81	10	6.52	6.5	6.52	0.42	12.45	9.2
200	0.81	10	6.76	6.45	6.23	0.41	15.77	9.98
200	0.81	11	6.88	6.76	6.75	0.42	14.5	9.72
200	0.81	11	5.98	6.1	5.92	0.45	12.48	9.88
200	0.81	12	7.17	7.03	6.78	0.41	14.47	11.5
200	0.81	12	6.16	6.3	6.12	0.38	14.3	11.92
200	0.81	13	6.23	9.56	6.11	0.52	15.08	11.5
200	0.81	13	6.04	6.36	6.31	0.56	27.25	20.98
200	0.81	14	6.06	6.17	6.11	0.52	18.69	13.91
200	0.81	14	5.86	6.29	5.94	0.45	16.2	13.72
200	0.81	15	6.97	6.64	7.16	0.55	16.55	10.61
200	0.81	15	6.56	6.47	6.49	0.45	19.8	12.88
200	0.91	6	6.84	6.57	6.55	0.28	8.44	6.88
200	0.91	7	6.01	6.19	6.06	0.34	8.61	7.31
200	0.91	7	6.54	6.44	8.42	0.36	8.2	5.63
200	0.91	8	6.62	6.5	6.7	0.39	9.61	7.42
200	0.91	8	6.01	5.98	6	0.34	9.02	6.19
200	0.91	9	7.13	7.66	6.89	0.41	10.56	10
200	0.91	9	6.6	6.41	6.27	0.34	10.98	7.33
200	0.91	10	7.27	7.05	7.01	0.41	16.81	8.98
200	0.91	10	7.06	7.19	7.09	0.42	11.73	8.95
200	0.91	11	6.42	6.23	6.42	0.45	13.77	11.41
200	0.91	11	6.13	6.26	6.21	0.42	15.39	10.63
200	0.91	12	6.69	6.67	6.78	0.42	14.42	10.09
200	0.91	12	7.12	7.16	6.93	0.42	13.98	9.39
200	0.91	13	6.86	6.55	6.4	0.48	16.06	12.91
200	0.91	13	6.39	6.4	6.76	0.73	23.23	15.64
200	0.91	14	6.8	6.76	7.18	0.48	19.7	15.31
200	0.91	14	6.72	7.01	6.47	0.42	19.36	12.78
200	0.91	15	7.01	7.27	6.77	0.53	18.17	13.13
200	0.91	15	6.8	6.62	6.66	0.48	18.52	13.78
200	1.01	6	6.96	6.91	6.85	0.34	8.88	5.78
200	1.01	6	6.84	7.45	7.12	0.3	6.97	6.8
200	1.01	7	7.14	7.01	6.92	0.36	8.25	7.84
200	1.01	7	6.76	6.74	6.69	0.36	9.3	6.88
200	1.01	8	7.63	7.56	7.72	0.33	10.91	7.7
200	1.01	9	6.27	6.32	6.96	0.41	12.38	7.77
200	1.01	9	6.47	6.68	7.26	0.44	11.3	7.66
200	1.01	10	7.6	7.25	7.3	0.44	12.06	9.52
200	1.01	10	6.23	6.43	6.25	0.56	17.08	12.22

			Cylindricity			CPU Time		
Pts	**Variance**	**Slices**	**LSQ**	**NLP**	***n*SVR**	**LSQ**	**NLP**	**SVR**
200	1.01	11	6.4	7.43	6.28	0.44	14.16	9.56
200	1.01	11	7.53	7.4	7.45	0.44	13	10.58
200	1.01	12	6.7	6.68	6.58	0.48	16.59	8.89
200	1.01	12	6.62	6.76	7.51	0.52	12.67	11.31
200	1.01	13	7.12	7.01	6.97	0.45	15.75	12.95
200	1.01	13	6.89	6.85	6.83	0.53	18.06	21.38
200	1.01	14	7.1	6.75	7.41	0.47	22.3	15.91
200	1.01	14	7.08	7.18	7.24	0.58	17.97	12.23
200	1.01	15	7.4	7.26	7.3	0.44	17.44	12.19
200	1.01	15	7	6.74	7.57	0.52	18.7	13.08

APPENDIX G: EXAMPLE PROBLEM SAMPLE POINTS

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
0	10.41	0.8	10.51	11	10.93
0	9.91	-0.31	9.96	10.43	10.36
0	10.33	1.92	10.6	11.09	11.02
0	8.85	2.18	9.21	9.7	9.65
0	9.88	2.3	10.24	10.73	10.67
0	9.89	2.97	10.43	10.92	10.87
0	8.6	4.2	9.7	10.18	10.14
0	10	4.45	11.07	11.55	11.51
0	7.9	4.31	9.13	9.61	9.58
0	8.93	5.07	10.41	10.88	10.85
0	8.44	6.47	10.78	11.24	11.22
0	6.89	6.34	9.53	9.96	9.96
0	6.4	7.03	9.68	10.09	10.09
0	6.5	6.93	9.67	10.09	10.09
0	6.74	7.57	10.3	10.72	10.72
0	6.33	7.57	10.04	10.45	10.45
0	5.36	8.38	10.13	10.5	10.52
0	4.51	9.7	10.88	11.2	11.24
0	4.97	8.4	9.94	10.3	10.32
0	3.89	8.72	9.73	10.05	10.09
0	3.7	8.75	9.69	10	10.04
0	2.92	9.5	10.13	10.4	10.45
0	1.6	10.95	11.25	11.45	11.51
0	0.17	9.74	9.92	10.08	10.15
0	0.29	9.78	9.96	10.12	10.19
0	1.42	10.9	11.17	11.37	11.44
0	-1.02	9.66	9.88	9.98	10.06
0	-0.09	10.36	10.53	10.67	10.75
0	-1.03	10.3	10.53	10.62	10.71
0	-2.15	9.47	9.87	9.91	10

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
0	-2.73	9.46	10	10.02	10.11
0	-2.83	9.57	10.14	10.14	10.24
0	-4.62	7.95	9.33	9.22	9.33
0	-4.59	8.76	10.02	9.93	10.04
0	-5.78	8.32	10.24	10.1	10.21
0	-5.59	8.19	10.04	9.9	10
0	-5.41	7.92	9.71	9.57	9.68
0	-7.16	7.37	10.36	10.14	10.25
0	-6.16	5.3	8.2	7.95	8.05
0	-6.82	6.69	9.64	9.41	9.52
0	-8.9	6.34	10.99	10.69	10.8
0	-8.16	5.61	9.96	9.66	9.77
0	-8.1	4.51	9.31	8.98	9.08
0	-8.68	4.11	9.63	9.27	9.38
0	-9.56	3.82	10.32	9.94	10.04
0	-10.07	2.88	10.48	10.07	10.17
0	-10.11	3.17	10.6	10.2	10.3
0	-9.08	1.94	9.27	8.85	8.94
0	-9.5	0.82	9.5	9.05	9.13
0	-10.54	0.29	10.49	10.03	10.11
0	-9.73	0.51	9.7	9.24	9.33
0	-10.94	0.11	10.89	10.43	10.51
0	-10.45	-0.96	10.43	9.95	10.02
0	-9.02	-2.91	9.37	8.88	8.93
0	-10.71	-2.96	11.02	10.53	10.58
0	-8.94	-4.49	9.88	9.4	9.43
0	-9.75	-5.05	10.85	10.37	10.4
0	-9.44	-5.57	10.82	10.35	10.38
0	-9.64	-3.99	10.32	9.83	9.88
0	-8.13	-4.13	8.99	8.51	8.54
0	-8.27	-6.97	10.66	10.22	10.22

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
0	-8.6	-6.14	10.42	9.96	9.98
0	-8.06	-7.39	10.77	10.34	10.34
0	-5.95	-6.81	8.87	8.46	8.45
0	-7.36	-8.92	11.39	10.99	10.98
0	-6.19	-9.23	10.93	10.56	10.54
0	-4.48	-7.74	8.76	8.42	8.39
0	-4.48	-8.02	9	8.66	8.63
0	-5.12	-8.35	9.61	9.26	9.23
0	-5.1	-10.92	11.87	11.55	11.51
0	-3.5	-9.35	9.8	9.52	9.47
0	-2.95	-10.39	10.61	10.37	10.31
0	-2.41	-8.03	8.2	7.95	7.89
0	-1.43	-10.93	10.84	10.66	10.59
0	0.62	-11.75	11.59	11.5	11.41
0	-0.35	-11.02	10.85	10.72	10.64
0	0.57	-10.19	10.03	9.94	9.86
0	2.23	-9.06	9.17	9.17	9.08
0	2.69	-9.39	9.62	9.64	9.54
0	4.01	-9.21	9.9	9.99	9.89
0	3.17	-10.85	11.15	11.17	11.08
0	4.83	-9.8	10.78	10.89	10.79
0	2.77	-8.87	9.14	9.18	9.08
0	4.86	-8.14	9.36	9.5	9.4
0	5.25	-7.12	8.74	8.93	8.82
0	5.59	-8.24	9.84	10.01	9.9
0	6.89	-7.48	10.08	10.32	10.21
0	6.15	-7.69	9.74	9.95	9.84
0	8.58	-7.15	11.09	11.38	11.28
0	7.98	-5.55	9.67	9.99	9.89
0	7.52	-4.76	8.85	9.2	9.09
0	8.42	-5.8	10.17	10.5	10.39

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
0	9.47	-4.09	10.29	10.69	10.59
0	9.23	-4.26	10.14	10.53	10.42
0	8.08	-3.78	8.9	9.28	9.18
0	10.18	-3.86	10.88	11.28	11.18
0	9.84	-2.74	10.23	10.65	10.56
0	10.52	-2.17	10.76	11.21	11.12
0	9.72	-2.19	9.98	10.42	10.33
0	10.6	-1.7	10.76	11.21	11.12
20	10.74	-0.09	10.72	10.17	10.17
20	10.1	0.9	10.11	9.56	9.56
20	10.84	1.47	10.9	10.36	10.36
20	9.98	2.26	10.18	9.65	9.65
20	10.32	1.64	10.4	9.87	9.87
20	8.93	1.34	8.99	8.45	8.45
20	8.87	3.78	9.55	9.07	9.07
20	9.02	4.79	10.12	9.67	9.67
20	9.85	3.71	10.45	9.96	9.96
20	8.29	3.91	9.08	8.61	8.61
20	8.49	5.25	9.87	9.45	9.45
20	7.44	5.88	9.36	8.98	8.98
20	7.34	6.64	9.77	9.42	9.42
20	6.8	6.99	9.62	9.29	9.29
20	7.39	9.34	11.77	11.49	11.49
20	6.17	9.33	11.04	10.8	10.8
20	5.88	9.06	10.65	10.42	10.42
20	4.74	9.59	10.54	10.38	10.38
20	4.6	8.74	9.72	9.54	9.54
20	2.7	9.14	9.36	9.29	9.29
20	2.23	10.51	10.58	10.55	10.55
20	2.55	9.34	9.51	9.45	9.45
20	2.59	10.85	10.99	10.95	10.95

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
20	0.79	10.68	10.54	10.59	10.59
20	0.41	9.13	8.97	9.03	9.03
20	0.37	9.7	9.54	9.61	9.61
20	0.39	10.71	10.55	10.62	10.62
20	-1.13	10.89	10.78	10.93	10.93
20	-1.68	9.37	9.36	9.55	9.55
20	-4.01	9	9.71	10.02	10.02
20	-3.21	8.47	8.9	9.19	9.19
20	-4.72	9.59	10.55	10.87	10.87
20	-4.98	9.79	10.85	11.18	11.18
20	-5.7	9.21	10.7	11.07	11.07
20	-4.56	9.69	10.57	10.89	10.89
20	-6.38	7.86	10.01	10.43	10.43
20	-7.55	8.67	11.39	11.82	11.82
20	-7.34	6.86	9.95	10.41	10.41
20	-7.28	6.63	9.75	10.22	10.22
20	-7.16	6.45	9.54	10.01	10.01
20	-7.08	6.62	9.6	10.06	10.06
20	-9.66	4.96	10.81	11.34	11.34
20	-8.77	4.58	9.83	10.36	10.36
20	-9.16	5.26	10.5	11.02	11.02
20	-8.95	4.05	9.77	10.31	10.31
20	-10.5	3.39	11	11.55	11.55
20	-9.26	2.26	9.51	10.07	10.07
20	-9.48	1.44	9.59	10.15	10.15
20	-9.4	1.47	9.51	10.07	10.07
20	-9.78	1.51	9.89	10.45	10.45
20	-9.68	-0.43	9.71	10.26	10.26
20	-9.17	0.1	9.19	9.75	9.75
20	-11.16	-0.83	11.22	11.77	11.77
20	-10.11	-2.37	10.44	10.96	10.96

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
20	-9.03	-3.17	9.64	10.14	10.14
20	-9.23	-3.04	9.79	10.3	10.3
20	-9.37	-4.51	10.49	10.95	10.95
20	-7.22	-3.17	7.97	8.45	8.45
20	-9.72	-4.26	10.7	11.18	11.18
20	-8.57	-5.46	10.27	10.7	10.7
20	-8.89	-6.61	11.19	11.6	11.6
20	-6.04	-6.29	8.85	9.19	9.19
20	-7.53	-6.82	10.28	10.65	10.65
20	-6.6	-6.52	9.41	9.76	9.76
20	-6.87	-7.93	10.63	10.94	10.94
20	-6.06	-8.47	10.57	10.83	10.83
20	-4.77	-8.83	10.19	10.4	10.4
20	-5.26	-8.78	10.39	10.62	10.62
20	-4.3	-9.66	10.73	10.9	10.9
20	-4.23	-9.37	10.44	10.62	10.62
20	-4.23	-9.1	10.19	10.37	10.37
20	-3.74	-9.38	10.26	10.41	10.41
20	-0.59	-8.9	9.09	9.07	9.07
20	-2.3	-11.37	11.77	11.82	11.82
20	-0.48	-10.58	10.76	10.73	10.73
20	-0.12	-9.85	10.02	9.97	9.97
20	0.83	-9.99	10.19	10.09	10.09
20	1.44	-10.15	10.41	10.28	10.28
20	1.85	-9.7	10.03	9.87	9.87
20	3.47	-9.92	10.66	10.42	10.42
20	1.97	-10.21	10.56	10.4	10.4
20	4.35	-8.5	9.69	9.39	9.39
20	5.26	-8.38	10.03	9.69	9.69
20	2.61	-8.55	9.09	8.88	8.88
20	4.11	-8.33	9.43	9.14	9.14

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
20	5.55	-7.28	9.27	8.89	8.89
20	5.66	-6.86	9.01	8.62	8.62
20	6.8	-7.14	9.96	9.54	9.54
20	7.8	-6.8	10.44	9.99	9.99
20	8.1	-6.2	10.28	9.81	9.81
20	8.45	-5.54	10.18	9.68	9.68
20	8.22	-5.62	10.04	9.55	9.55
20	9.23	-4.75	10.44	9.92	9.92
20	8.7	-5	10.1	9.59	9.59
20	9.03	-4.73	10.25	9.73	9.73
20	9.18	-3.64	9.92	9.38	9.38
20	10.07	-3.38	10.65	10.11	10.11
20	10.58	-1.74	10.73	10.17	10.17
20	9.9	-1.65	10.04	9.49	9.49
20	9.26	-0.97	9.31	8.75	8.75
40	9.89	-0.46	9.88	9.91	9.91
40	9.83	0.48	9.82	9.85	9.85
40	10.75	0.9	10.76	10.79	10.79
40	10.66	2.78	11	11.01	11.01
40	9.79	2.79	10.16	10.17	10.17
40	9.72	3	10.16	10.17	10.17
40	10.08	3.7	10.72	10.73	10.73
40	9.47	4.85	10.63	10.62	10.62
40	8.55	5.31	10.06	10.05	10.05
40	8.37	5.37	9.94	9.93	9.93
40	8.09	6.85	10.6	10.57	10.57
40	7.36	6.99	10.15	10.12	10.12
40	7.2	6.7	9.83	9.81	9.81
40	6.66	7.62	10.12	10.09	10.09
40	6.82	7	9.78	9.75	9.75
40	6.62	8.87	11.07	11.04	11.04

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
40	5.43	7.68	9.41	9.37	9.37
40	5.42	9.44	10.9	10.85	10.85
40	5.14	9.15	10.51	10.46	10.46
40	2.81	8.37	8.85	8.79	8.79
40	2.76	8.92	9.36	9.3	9.3
40	0.74	8.85	8.91	8.84	8.84
40	2.39	8.94	9.28	9.22	9.22
40	1.94	10.36	10.57	10.5	10.5
40	1.43	9.78	9.91	9.85	9.85
40	0.11	10.16	10.19	10.11	10.11
40	-0.83	10.05	10.12	10.04	10.04
40	-1.92	8.92	9.16	9.08	9.08
40	-0.94	9.59	9.67	9.59	9.59
40	-1.99	9.95	10.19	10.11	10.11
40	-1.45	10.28	10.41	10.33	10.33
40	-4.47	10.12	11.1	11.02	11.02
40	-4.9	8.66	9.99	9.91	9.91
40	-4.17	8.28	9.31	9.23	9.23
40	-4.77	8.97	10.2	10.12	10.12
40	-6.76	8.39	10.81	10.73	10.73
40	-6.7	7.34	9.98	9.9	9.9
40	-7.03	7.24	10.13	10.05	10.05
40	-6.66	7.65	10.18	10.1	10.1
40	-7.4	6.22	9.71	9.63	9.63
40	-8.93	5.6	10.58	10.51	10.51
40	-9.18	4.51	10.27	10.2	10.2
40	-8.34	4.74	9.63	9.56	9.56
40	-9.71	5.38	11.14	11.08	11.08
40	-9.27	4.54	10.36	10.29	10.29
40	-9.6	2.24	9.89	9.84	9.84
40	-8.87	1.87	9.09	9.04	9.04

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
40	-10.46	2.2	10.72	10.67	10.67
40	-9.07	-0.13	9.1	9.07	9.07
40	-10.48	0.09	10.51	10.47	10.47
40	-9.91	0.42	9.94	9.91	9.91
40	-8.96	0.59	9.01	8.97	8.97
40	-10.72	-1.65	10.86	10.84	10.84
40	-9.12	-1	9.2	9.17	9.17
40	-9.65	-2.98	10.12	10.11	10.11
40	-10.31	-3.38	10.87	10.86	10.86
40	-9.79	-4.87	10.95	10.95	10.95
40	-9.07	-5.46	10.6	10.61	10.61
40	-9.04	-4.04	9.91	9.91	9.91
40	-8.23	-4.96	9.62	9.63	9.63
40	-8.05	-5.07	9.52	9.53	9.53
40	-7.68	-7.03	10.41	10.44	10.44
40	-7.42	-5.73	9.38	9.4	9.4
40	-7.48	-8.1	11.02	11.05	11.05
40	-4.91	-8.74	10.01	10.06	10.06
40	-5.96	-7.53	9.6	9.63	9.63
40	-6.19	-8.42	10.45	10.48	10.48
40	-5.6	-10.15	11.58	11.63	11.63
40	-4.87	-8.95	10.18	10.22	10.22
40	-3.57	-10.02	10.62	10.67	10.67
40	-2.82	-9.31	9.71	9.77	9.77
40	-4.1	-9.87	10.67	10.72	10.72
40	-2.05	-10.77	10.93	11	11
40	-2.03	-10.16	10.34	10.41	10.41
40	-0.75	-10.06	10.06	10.14	10.14
40	-2.23	-9.78	10.01	10.07	10.07
40	-0.17	-10.55	10.52	10.6	10.6
40	1.9	-10.47	10.61	10.68	10.68

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
40	0.81	-9.26	9.26	9.34	9.34
40	2.22	-8.48	8.72	8.8	8.8
40	3.19	-9.91	10.38	10.46	10.46
40	4.98	-8.62	9.92	10	10
40	3.97	-9.42	10.18	10.26	10.26
40	3.45	-8.04	8.71	8.79	8.79
40	3.19	-8.56	9.1	9.18	9.18
40	6.14	-7.51	9.66	9.74	9.74
40	6.56	-7.42	9.86	9.94	9.94
40	7.01	-6.56	9.56	9.63	9.63
40	6.44	-6.58	9.16	9.24	9.24
40	7.94	-6.34	10.12	10.19	10.19
40	8.28	-5.26	9.77	9.83	9.83
40	9.64	-4.07	10.42	10.49	10.49
40	8.01	-4.21	9.01	9.08	9.08
40	9.66	-4.3	10.54	10.6	10.6
40	9.45	-3.7	10.11	10.17	10.17
40	9.7	-1.81	9.83	9.88	9.88
40	9.62	-2.33	9.87	9.92	9.92
40	11.37	-0.77	11.37	11.41	11.41
40	11.37	-2.33	11.58	11.63	11.63
40	10.14	-0.18	10.11	10.15	10.15
60	9.29	-0.38	9.39	9.25	9.25
60	11.01	-0.23	11.1	10.96	10.96
60	9.41	1.18	9.58	9.35	9.35
60	11.03	1.91	11.29	11.03	11.03
60	11.81	1.49	12.01	11.77	11.77
60	8.82	3.14	9.47	9.12	9.12
60	8.94	4.27	10.02	9.62	9.62
60	10.12	4.49	11.18	10.8	10.8
60	8.55	4.63	9.84	9.42	9.42

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
60	8.09	5.09	9.67	9.22	9.22
60	8.79	5.05	10.25	9.82	9.82
60	7.99	5.65	9.9	9.43	9.43
60	7.06	7.67	10.54	9.99	9.99
60	7.01	6.65	9.78	9.26	9.26
60	6.3	8.36	10.58	10.01	10.01
60	6.2	9.05	11.08	10.5	10.5
60	4.75	8.41	9.77	9.17	9.17
60	5.04	8.6	10.08	9.48	9.48
60	3.19	8.81	9.47	8.86	8.86
60	4.27	10.42	11.37	10.76	10.76
60	3.77	9.11	9.96	9.35	9.35
60	2.63	9.39	9.85	9.23	9.23
60	1.16	10.55	10.7	10.09	10.09
60	0.72	9.76	9.87	9.27	9.27
60	-0.1	9.18	9.25	8.66	8.66
60	0.23	10.11	10.18	9.59	9.59
60	-0.53	9.52	9.61	9.03	9.03
60	-1.67	10.95	11.14	10.58	10.58
60	-2.24	10.53	10.82	10.28	10.28
60	-2.86	9.48	9.95	9.43	9.43
60	-3.14	9.51	10.06	9.56	9.56
60	-3.73	8.82	9.61	9.14	9.14
60	-5.06	8.18	9.63	9.22	9.22
60	-4.62	9.58	10.66	10.21	10.21
60	-5.23	8.74	10.21	9.79	9.79
60	-5.22	8.37	9.88	9.48	9.48
60	-6.43	8.9	10.99	10.61	10.61
60	-6.03	8.16	10.15	9.79	9.79
60	-7.73	5.76	9.61	9.41	9.41
60	-7.66	6.69	10.15	9.9	9.9

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
60	-7.9	6.03	9.92	9.7	9.7
60	-9.91	5.72	11.4	11.27	11.27
60	-8.31	4.96	9.64	9.49	9.49
60	-8.17	4.06	9.08	8.98	8.98
60	-9.27	3.5	9.85	9.81	9.81
60	-9.31	2.9	9.69	9.69	9.69
60	-9.39	1.79	9.49	9.55	9.55
60	-9.41	1.66	9.48	9.56	9.56
60	-8.9	1.02	8.88	8.99	8.99
60	-10.95	0.58	10.88	11.03	11.03
60	-10.76	0.28	10.67	10.84	10.84
60	-10.44	-1.29	10.42	10.67	10.67
60	-10.28	-1.02	10.23	10.47	10.47
60	-9.88	-1.94	9.97	10.26	10.26
60	-10.59	-2.19	10.71	11.01	11.01
60	-10	-2.77	10.27	10.6	10.6
60	-9.88	-3.65	10.42	10.79	10.79
60	-10.81	-4.58	11.63	12.02	12.02
60	-8.78	-5.49	10.24	10.7	10.7
60	-8.7	-5.08	9.96	10.41	10.41
60	-7.74	-7.63	10.75	11.29	11.29
60	-7.74	-7.81	10.88	11.43	11.43
60	-6.36	-7.16	9.46	10.02	10.02
60	-7.49	-7.95	10.8	11.35	11.35
60	-5.99	-7.25	9.29	9.86	9.86
60	-5.32	-8.73	10.12	10.71	10.71
60	-6.33	-8.2	10.25	10.82	10.82
60	-6.3	-7.78	9.9	10.46	10.46
60	-3.91	-9.48	10.15	10.76	10.76
60	-6.24	-9.53	11.28	11.87	11.87
60	-2.61	-9.61	9.86	10.48	10.48

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
60	-2.71	-10.05	10.32	10.93	10.93
60	-2.47	-9.11	9.35	9.96	9.96
60	-0.9	-8.09	8.06	8.66	8.66
60	0.24	-9.53	9.46	10.05	10.05
60	0.06	-9.25	9.17	9.76	9.76
60	-0.01	-10.54	10.47	11.06	11.06
60	0.84	-11.44	11.4	11.98	11.98
60	1.6	-9.93	10	10.56	10.56
60	2.45	-10.54	10.77	11.3	11.3
60	3.84	-9.58	10.29	10.77	10.77
60	4.13	-9.07	9.94	10.41	10.41
60	3.66	-9.58	10.22	10.71	10.71
60	6.26	-8.54	10.59	10.97	10.97
60	6.47	-7.53	9.93	10.27	10.27
60	6.08	-7.97	10.02	10.39	10.39
60	5.62	-8.2	9.93	10.32	10.32
60	6.6	-7.35	9.88	10.22	10.22
60	8.64	-7.19	11.26	11.52	11.52
60	8.23	-8.32	11.71	12.02	12.02
60	7.76	-6.01	9.84	10.08	10.08
60	8.2	-5.5	9.91	10.1	10.1
60	9.09	-4.06	10.01	10.1	10.1
60	8.61	-3.25	9.27	9.33	9.33
60	8.56	-3.64	9.36	9.44	9.44
60	8.94	-1.99	9.23	9.21	9.21
60	9.25	-2.53	9.66	9.66	9.66
60	9.84	-0.91	9.97	9.87	9.87
60	9.19	-1.02	9.32	9.23	9.23
60	10.03	-1.84	10.27	10.23	10.23
80	10.58	0.65	10.46	10.59	10.59
80	10.05	0.3	9.92	10.05	10.05

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
80	11.01	2.25	11.12	11.27	11.27
80	9.79	2	9.88	10.02	10.02
80	9.36	2.2	9.51	9.65	9.65
80	10.21	3.05	10.56	10.7	10.7
80	9.24	4.38	10.15	10.29	10.29
80	9.68	3.67	10.26	10.41	10.41
80	9.84	4.8	10.87	11.01	11.01
80	8.03	5.63	9.76	9.9	9.9
80	7.99	5.62	9.72	9.86	9.86
80	7.41	6.73	9.98	10.12	10.12
80	8.51	7.44	11.27	11.42	11.42
80	7.34	7.23	10.28	10.42	10.42
80	4.95	6.57	8.24	8.37	8.37
80	5.27	8.83	10.31	10.44	10.44
80	4.7	9.33	10.49	10.61	10.61
80	5.53	7.18	9.07	9.2	9.2
80	3.85	8.82	9.68	9.79	9.79
80	4.38	7.47	8.69	8.81	8.81
80	4.49	9.63	10.68	10.79	10.79
80	3.26	9.76	10.36	10.47	10.47
80	1.55	9.45	9.67	9.76	9.76
80	0.07	9.59	9.71	9.77	9.77
80	0.91	10.72	10.87	10.94	10.94
80	0.21	10.5	10.62	10.69	10.69
80	-0.74	11.18	11.33	11.39	11.39
80	0.2	9.37	9.49	9.56	9.56
80	-1.4	8.95	9.2	9.25	9.25
80	-1.54	9.99	10.25	10.29	10.29
80	-3.04	8.87	9.54	9.56	9.56
80	-3.37	8.82	9.61	9.63	9.63
80	-3.93	9.1	10.08	10.09	10.09

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
80	-4.74	8.37	9.8	9.79	9.79
80	-5.34	7.73	9.57	9.56	9.56
80	-6.28	9.45	11.53	11.51	11.51
80	-7.4	7.31	10.59	10.55	10.55
80	-7.3	6.39	9.89	9.84	9.84
80	-5.98	6.16	8.77	8.73	8.73
80	-7	6.38	9.66	9.61	9.61
80	-9.52	4.66	10.78	10.7	10.7
80	-8.99	6.38	11.21	11.14	11.14
80	-7.83	4.38	9.16	9.08	9.08
80	-8.31	4.43	9.6	9.52	9.52
80	-8.02	2.56	8.6	8.5	8.5
80	-9.41	3.78	10.32	10.22	10.22
80	-9.65	2.25	10.08	9.97	9.97
80	-8.9	2.29	9.36	9.25	9.25
80	-11.25	1.8	11.56	11.44	11.44
80	-9.52	0.64	9.7	9.58	9.58
80	-9.24	0.53	9.41	9.29	9.29
80	-9.98	-2.28	10.35	10.21	10.21
80	-10.42	-0.93	10.6	10.47	10.47
80	-9.66	-1.56	9.91	9.77	9.77
80	-10.55	-2.46	10.95	10.81	10.81
80	-9.31	-2.49	9.75	9.61	9.61
80	-9.11	-4.41	10.2	10.05	10.05
80	-9.11	-3.77	9.94	9.8	9.8
80	-10.24	-3.81	11.02	10.87	10.87
80	-6.87	-5.05	8.58	8.43	8.43
80	-7.67	-5.32	9.39	9.24	9.24
80	-8.3	-6.33	10.48	10.34	10.34
80	-7.59	-6.71	10.16	10.02	10.02
80	-7.04	-7.09	10.01	9.87	9.87

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
80	-6.75	-8.56	10.9	10.76	10.76
80	-5.22	-6.71	8.5	8.37	8.37
80	-4.84	-8.73	9.95	9.82	9.82
80	-5.25	-9.74	11.03	10.9	10.9
80	-4.89	-8.06	9.4	9.28	9.28
80	-3.29	-10.19	10.63	10.53	10.53
80	-3.91	-9.46	10.18	10.07	10.07
80	-2.33	-9.93	10.12	10.02	10.02
80	-3.43	-9.98	10.49	10.38	10.38
80	-0.17	-10.56	10.45	10.37	10.37
80	-1.15	-9.56	9.53	9.44	9.44
80	1.33	-10.25	10.2	10.15	10.15
80	1.5	-8.82	8.81	8.76	8.76
80	0.97	-11.66	11.57	11.51	11.51
80	0.64	-10.19	10.08	10.02	10.02
80	2.26	-10.66	10.75	10.71	10.71
80	3.41	-8.94	9.4	9.38	9.38
80	3.23	-9.25	9.63	9.61	9.61
80	3.9	-8.14	8.86	8.85	8.85
80	5.86	-8.33	10	10.02	10.02
80	5.32	-9.81	10.98	10.99	10.99
80	5.55	-6.98	8.73	8.76	8.76
80	5.6	-7.53	9.21	9.23	9.23
80	6.61	-7.02	9.46	9.5	9.5
80	8.11	-6.9	10.46	10.51	10.51
80	7.93	-6.25	9.91	9.97	9.97
80	8.27	-6.49	10.33	10.39	10.39
80	9.05	-5.3	10.3	10.38	10.38
80	9.91	-4.01	10.51	10.61	10.61
80	9.77	-4.81	10.71	10.8	10.8
80	8.49	-4.44	9.4	9.48	9.48

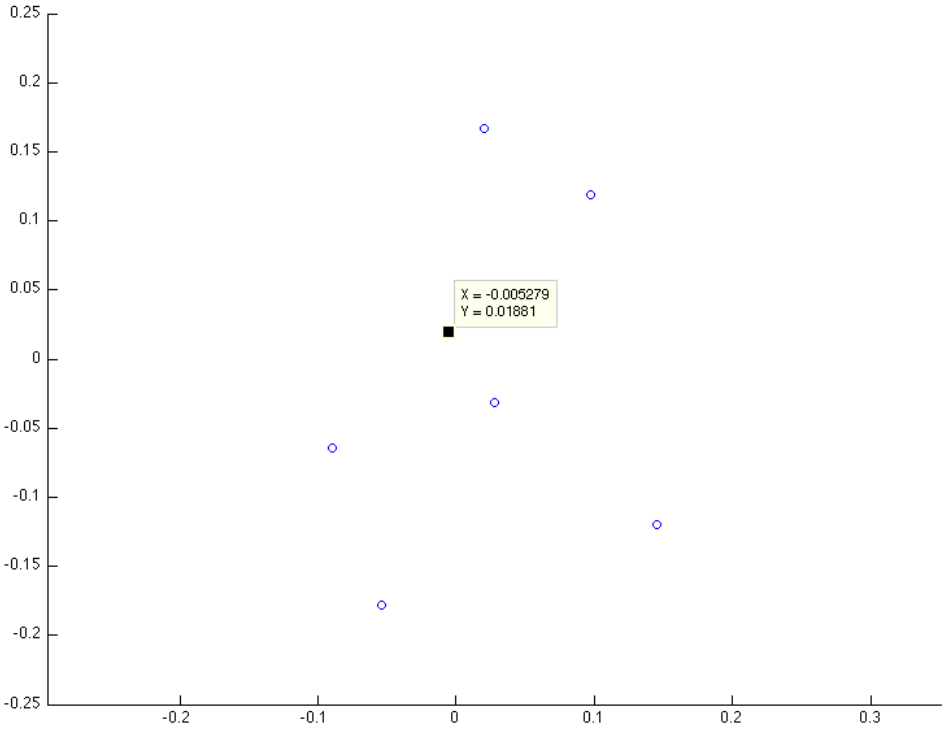
Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
80	10.52	-2.85	10.72	10.83	10.83
80	8.93	-2.71	9.16	9.27	9.27
80	10.26	-1.09	10.16	10.28	10.28
80	9.8	-0.39	9.66	9.79	9.79
80	10.05	-0.66	9.92	10.04	10.04
100	10.31	1.38	10.29	10.3	10.47
100	10.67	1.3	10.64	10.65	10.82
100	10.56	1.32	10.53	10.54	10.71
100	10.19	3.3	10.58	10.67	10.8
100	10.25	1.89	10.31	10.34	10.49
100	9.32	2.31	9.48	9.54	9.68
100	8.7	3.31	9.17	9.28	9.39
100	9.88	5.35	11.09	11.25	11.33
100	9.27	4.71	10.26	10.41	10.49
100	7.77	6.56	10.02	10.26	10.28
100	8.31	4.55	9.33	9.49	9.57
100	7.88	6.05	9.78	10.01	10.04
100	6.83	7.63	10.09	10.38	10.36
100	6.34	6.87	9.19	9.48	9.46
100	6.49	8.43	10.49	10.8	10.76
100	6.34	8.04	10.08	10.39	10.36
100	6.58	8.43	10.54	10.85	10.81
100	4.02	9.77	10.41	10.79	10.68
100	4.22	8.93	9.73	10.1	10
100	4.1	9.12	9.85	10.23	10.12
100	2.39	9.25	9.41	9.82	9.67
100	4.04	9.48	10.16	10.53	10.43
100	2.13	10.13	10.22	10.63	10.47
100	1.33	9.76	9.72	10.14	9.96
100	1.34	10.13	10.09	10.51	10.34
100	-0.33	10.48	10.37	10.8	10.6

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
100	-0.71	10.03	9.95	10.38	10.16
100	-2.17	10.15	10.28	10.71	10.47
100	-0.79	11.17	11.09	11.52	11.3
100	-2.62	10.93	11.15	11.57	11.33
100	-1.88	9.49	9.58	10.01	9.77
100	-3.63	9.29	9.9	10.31	10.05
100	-4.55	9.42	10.4	10.81	10.54
100	-3.79	8.99	9.68	10.1	9.83
100	-4.88	8.44	9.7	10.09	9.82
100	-5.77	7.35	9.31	9.68	9.4
100	-6.29	9.18	11.08	11.46	11.18
100	-5.95	9.09	10.82	11.21	10.93
100	-5.55	6.08	8.21	8.56	8.28
100	-8.86	6.4	10.95	11.24	10.96
100	-8.84	5.18	10.27	10.53	10.25
100	-8	5.57	9.76	10.04	9.76
100	-9.04	4.17	9.99	10.22	9.95
100	-8.19	3.62	9	9.22	8.95
100	-9.22	1.89	9.48	9.62	9.38
100	-9.43	2.6	9.84	10.01	9.76
100	-10.51	1.88	10.76	10.88	10.65
100	-8.98	-0.17	9.08	9.13	8.93
100	-11.11	2.38	11.43	11.57	11.33
100	-9.64	-0.03	9.74	9.79	9.59
100	-9.31	-0.28	9.42	9.46	9.26
100	-9	0.69	9.12	9.2	8.98
100	-9.41	-1.79	9.7	9.68	9.51
100	-10.47	-0.22	10.58	10.62	10.42
100	-8.91	-2.18	9.3	9.25	9.1
100	-8.99	-3.81	9.9	9.79	9.68
100	-9.5	-3.32	10.19	10.1	9.97

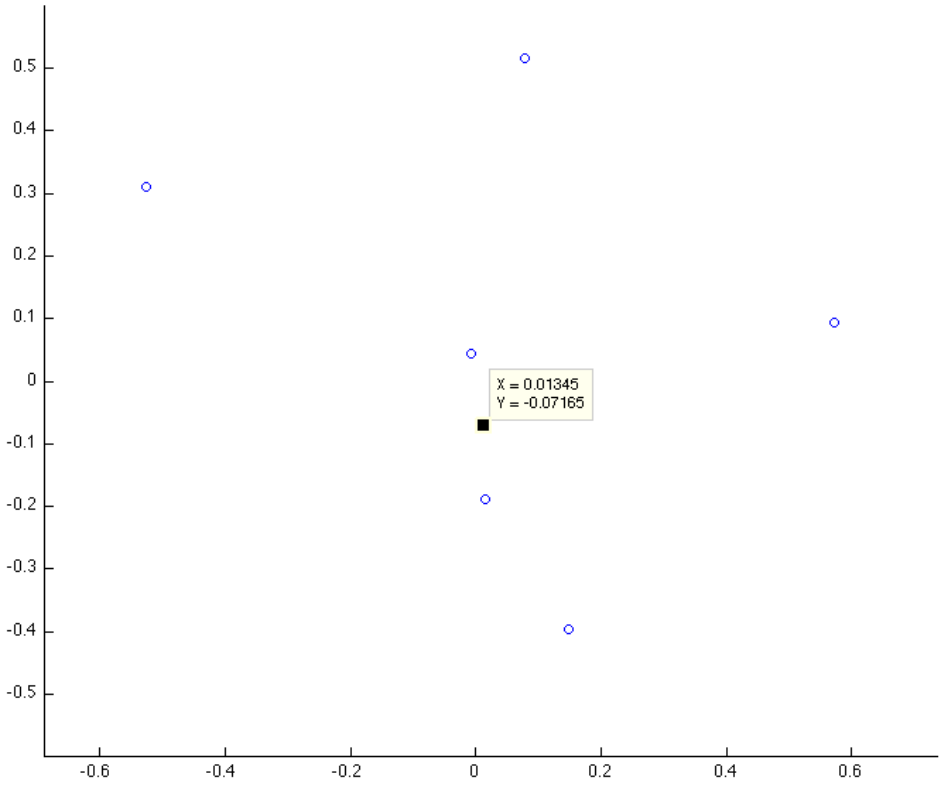
Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
100	-8.38	-4.18	9.51	9.37	9.27
100	-9.59	-5.12	11.01	10.86	10.77
100	-9.17	-6.49	11.38	11.18	11.13
100	-8.7	-6.45	10.98	10.77	10.72
100	-8.02	-6.39	10.4	10.18	10.14
100	-6.96	-5.74	9.17	8.95	8.91
100	-6.27	-7.55	9.97	9.68	9.7
100	-6.95	-7.78	10.58	10.3	10.31
100	-5.83	-8.39	10.37	10.05	10.1
100	-5.09	-8.19	9.79	9.46	9.52
100	-4.47	-9.08	10.27	9.91	10
100	-4.8	-9.5	10.79	10.44	10.52
100	-3.93	-9.42	10.35	9.98	10.08
100	-2.46	-9.69	10.14	9.74	9.88
100	-3.11	-9.26	9.91	9.52	9.64
100	-2.52	-10.51	10.95	10.54	10.69
100	-1.58	-9.2	9.47	9.06	9.22
100	-0.48	-9.19	9.33	8.91	9.09
100	-0.88	-8.96	9.13	8.71	8.89
100	1.41	-9.81	10.02	9.58	9.81
100	2.07	-10.31	10.62	10.19	10.42
100	1.56	-9.52	9.74	9.31	9.54
100	3.58	-8.29	9.11	8.69	8.96
100	3	-9.91	10.45	10.02	10.27
100	3.63	-8.29	9.12	8.71	8.97
100	4.04	-9	9.94	9.52	9.79
100	6.75	-8.89	11.2	10.82	11.11
100	6.05	-8.47	10.45	10.07	10.35
100	6.29	-8.77	10.84	10.46	10.74
100	5.95	-7.92	9.94	9.57	9.85
100	7.65	-8.27	11.29	10.94	11.22

Coordinates			Distance from Estimated Center		
Z-axis	**X-axis**	**Y-axis**	**LSQ**	**NLP**	**SVR**
100	7.93	-6.7	10.38	10.07	10.35
100	7.86	-5.91	9.83	9.53	9.81
100	7.01	-6.33	9.45	9.12	9.41
100	7.24	-5.1	8.85	8.56	8.84
100	8.6	-5.02	9.93	9.67	9.94
100	8.9	-4.1	9.76	9.53	9.8
100	8.24	-4.08	9.17	8.93	9.2
100	8.89	-3.35	9.45	9.25	9.51
100	10.19	-1.13	10.17	10.08	10.29
100	10.06	-2.09	10.21	10.07	10.31
100	10.17	-0.32	10.08	10.02	10.22
100	10.09	-1.39	10.1	10	10.22

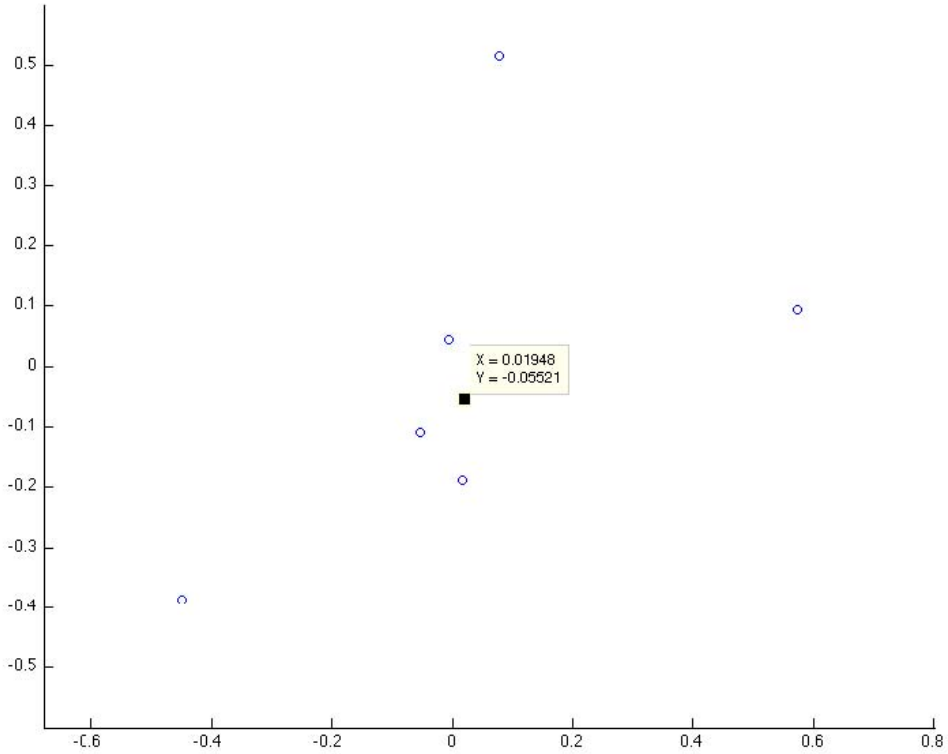
APPENDIX H: UNIFYING AXIS X-Y COORDINATES FOR EXAMPLE



LSQ Cylindricity Unifying Axis X-Y Coordinates



NLP Cylindricity Unifying Axis X-Y Coordinates



SVR Cylindricity Unifying Axis X-Y Coordinates

APPENDIX I: ADDITIONAL ANOVA TABLES

Analysis of Variance for Cylindricity, using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Points per Cross-Section	7	257.127	257.127	36.732	9.310	0.000
Method	2	24.173	24.178	12.089	3.060	0.047
Points*Method	14	18.062	18.062	1.290	0.330	0.991
Error	5160	20361.674	20361.674	3.946	0.870	
Total	5183	20661.036				

Table 6: ANOVA for Cylindricity: Points per Cross-section

Analysis of Variance for Cylindricity, using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Method	2	4.0309	4.0085	2.0043	2.1200	0.1220
Slices	9	19.1449	19.1449	2.1272	2.2400	0.0180
Method* Slices	9	14.0596	14.0596	0.7811	0.8200	0.6720
Error	406	420.7365	420.7365	0.9476		
Total	425	457.9719				

Table 7: ANOVA for Cylindricity: Number of Slices, Variance at 0.51

Analysis of Variance for CPU Time, using Adjusted SS for Tests						
Source	*DF*	*Seq SS*	*Adj SS*	*Adj MS*	*F*	*P*
Method	1	3.2700	2.9740	2.9740	1.4500	0.2290
Slices	9	491.7770	491.7770	54.6420	26.7200	0.0000
Method* Slices	9	2.4340	2.4340	0.2700	0.1300	0.9990
Error	406	830.3560	830.3560	2.0450		
Total	425	1327.8370				

Table 8: ANOVA for CPU Time: Num. of Cross-Sections, 100 pts per Cross-Section

APPENDIX J: ADDITIONAL STATISTICAL GRAPHS

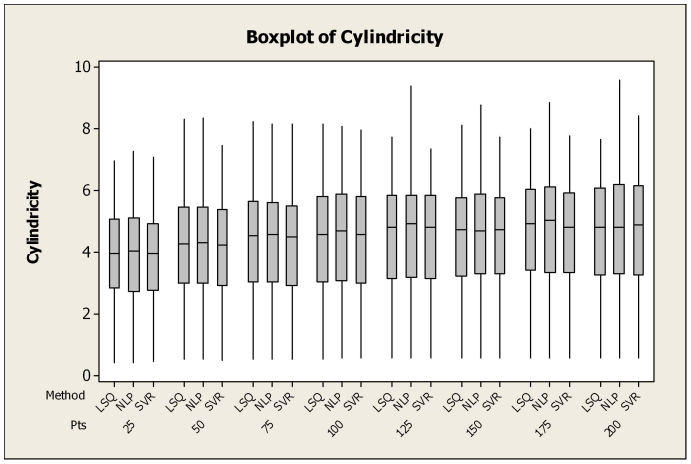


Figure 31: Boxplot of Cylindricity: Pts per Cross-Section

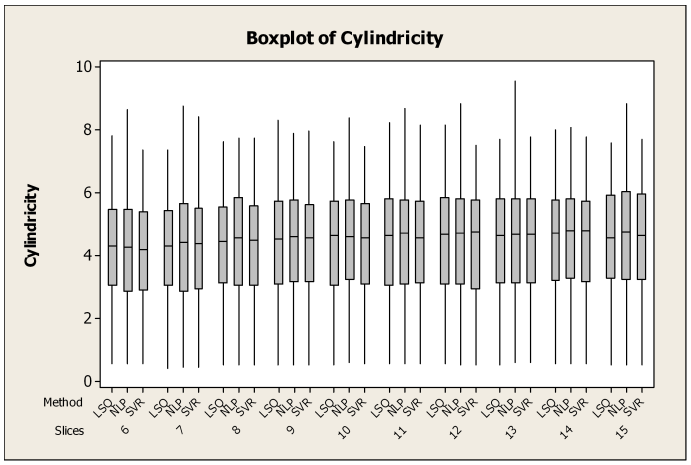


Figure 32: Boxplot of Cylindricity: Number of Cross-Sections, Variance: 0.51

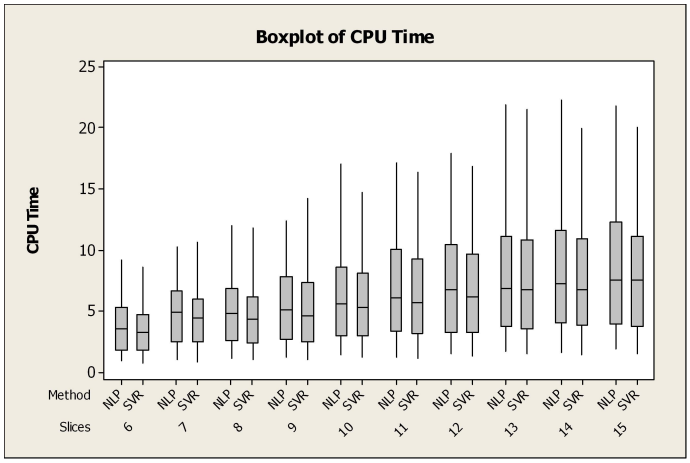


Figure 33: Boxplot of CPU Time: Num. of Cross-Sections, 100 Pts per Cross-Section