

Estimating Volumes of Simulated Lung Cancer Nodules*

David E. Gilsinn[†]
Bruce R. Borchardt[‡]
Amelia Tebbe[§]

July 15, 2009

Abstract

Lung cancer is a disease of uncontrolled cell growth in tissues of the lung. Computed tomography (CT) shows promise in detecting lung cancers at earlier, more operable stages, when survival is better. CT scans generate multiple 2-D slice images of the lung and digital image processing software is used to combine these images into a 3-D representation of the lung and, in particular, an identified cancer lesion. Various CT scanners use, often different and usually proprietary, software to develop these 3-D images and generate parameters such as lesion volume. Tracking lesion volume is considered a good diagnostic tool for evaluating the results of patient treatment. The Food and Drug Administration (FDA) is conducting research on developing reference cancer lesions, called phantoms, to test CT scanners and their proprietary software. FDA loaned two semi-spherical phantoms to the National Institute of Standards and Technology (NIST), called Green and Pink, and asked to have the phantoms measured by a coordinate measuring machine (CMM) and the volumes estimated. This report describes in detail both the experimental and computational methods used to estimate the phantoms' volumes as well as a bootstrap method for estimating the uncertainties of the computed volumes. Three sets of CMM measured data were produced. One set of data involved data density measurements of a known calibrated metal sphere. The other two sets were measurements of the two FDA phantoms at two densities, called the coarse set and the dense set. Two computational approaches were applied to the data. In the first approach spherical models were fit to the calibrated sphere data and to the phantom data. The second approach was to model the data points on the boundaries of the spheres with surface B-splines and then use the Divergence Theorem to estimate the volumes. The results for the coarse data set tended to predict the volumes as expected with low expanded uncertainties and the results did show that the Green phantom was very near spherical. This was confirmed by both computational methods. The spherical model did not fit the Pink phantom as well and the B-spline approach provided a better estimate of the volume in that case. The results for the dense data set did not provide as nice a prediction of volumes as the coarse data set and produced larger expanded uncertainties. A discussion of some possible reasons for these results will be given in the conclusion section. The report includes the MATLAB codes used in the study.

Keywords: B-splines, computed tomography, coordinate measuring machine, divergence theorem, lung cancer, lung cancer phantoms, nonlinear least squares.

*NISTIR 7571. Contribution of the National Institute of Standards and Technology, a Federal agency, not subject to copyright.

[†]Mathematical and Computational Sciences Division, Information Technology Laboratory, National Institute of Standards and Technology, Gaithersburg MD 20899-8910.

[‡]Precision Engineering Division, Manufacturing Engineering Laboratory, National Institute of Standards and Technology, Gaithersburg MD 20899-8211.

[§]Department of Mathematics, St. Mary's College of Maryland, St. Mary's City, MD 20686-3001.

1 Introduction

Lung cancer is a disease of uncontrolled cell growth in tissues of the lung. It is the most common cause of cancer related deaths in men and second most in women and is responsible for an estimated 560 000 deaths in the US in 2007. It is classified as small cell (13 %) and non-small cell (87 %) [12]. The most common cause is long term exposure to tobacco smoke. Lung cancer lesions may be seen in chest X-ray and computed tomography (CT), although chest X-ray analysis has shown limited effectiveness in improving survival. Computed tomography has shown promising results in detecting lung cancers at earlier, more operable stages, when survival is better [1].

Tomography is a method of imaging a single slice of the body, in this case the lung. Modern CT is a medical imaging method that uses tomography but also employs digital image processing techniques to generate three dimensional images built from a large sequence of two-dimensional X-ray images made around a single axis. There are a number of current CT methods but the one that is of most interest in this study is called a helical or spiral CT in which the X-ray source is connected to a rotating gantry. The patient, on a table, moves smoothly through the scanner. The name, helical or spiral CT, comes from the helical path traced out by the X-ray beam. The helical trajectory of the data is converted into two-dimensional image slices of the patient's body. The advantage of the helical or spiral CT is speed, since the patients must remain perfectly still and hold their breath. A large portion of the body can be scanned in 20-60 seconds [11]. CT provides excellent diagnostics for detecting changes in the lungs, where two-dimensional X-rays do not clearly show the lung cancer changes. A diagnosis is usually confirmed by biopsy. Treatments include surgery and chemotherapy. With treatment, the one-year survival was 42 % in 2002 but the five-year survival rate was only 16 % [1].

Evaluating the rate of growth/shrinkage of lung cancer lesions is important for pharmaceutical research in order to compare drug effects, for clinical practice to determine time to use drugs or surgery, and for patient follow up to quantify the effects of drugs or surgery. A goal then is to enable CT scans to become a measurement device, providing quantitative rather than only qualitative information. The current guidelines for measuring the increase or decrease of tumors involve linear measurements of the greatest diameter of a lesion from the sequence of CT slices [19]. However, the volume change of lesions is found to show more significant relative change than a relative diameter change. As an example, for a sphere a 30 % decrease in the relative diameter would produce a 65.7 % decrease in relative volume whereas a 30 % increase in the relative diameter would produce approximately a 120 % increase in relative volume.

The Food and Drug Administration (FDA) is conducting research on developing reference cancer lesions, called phantoms, to test CTs and their proprietary software. Two samples were loaned to NIST to estimate volumes. The material, of which the phantoms are composed, simulates lung cancer material. The phantoms can be inserted into a simulated body torso for CT scans. The two phantoms are shown in Fig. 1. Although they seem spherical, they are slightly non-spherical. The larger one on the right is referred to as the Green phantom and the one on the left is called the Pink phantom due to the material colors

One approach to estimating the volume of the phantoms is to use, what is called, an Archimedes test in which the phantoms would be immersed in a liquid bath in a well calibrated container with fine measurement gradations to determine liquid displacement. However, in the case of these phantoms, the material used to manufacture them is porous and the phantoms would have to be coated. This, of course, would affect the "ground truth" volume estimate. Furthermore, the potential expense involved with the process was found to be sufficiently significant that another approach was chosen.

The approach chosen for this study is based on a fundamental theorem in calculus, called the Divergence Theorem (see Taylor [17]), which is an analogue of Green's Theorem in two dimensional



Figure 1: Two simulated lung cancers, called phantoms.

space. In the Divergence Theorem a volume integral is shown to be equal to a surface integral. Therefore, we surmised that, if a model of the surfaces of the phantoms could be developed, we could use the Divergence Theorem to estimate their volumes. In order to develop a surface model we needed to obtain data about the surface. This was done using a coordinate measuring machine (CMM) in the Manufacturing Engineering Lab (MEL) at NIST. This machine produced a set of (x, y, z) points on the surface of each phantom. The data were then transformed to spherical coordinates and modeled using a set of basis functions, called B-splines. After fitting, the B-spline model was then used to generate a grid of values on the surface. These values were used to form surface triangles that were then used to compute the necessary surface integrals and finally the volume. The quality of the volume estimates depended on the surface grid sizes, as will be clear from the discussion below. The method is not new, in that it was suggested in the book by Dierckx [6]. A reader can also consult the Dierckx references [4] and [5].

The report is divided as follows. Section 2 introduces B-splines and tensor products of B-splines. Section 3 describes the surface point generation experiments using the CMM. Section 4 describes the methods and results from applying the spherical and B-spline models to the CMM data. Appendices include the MATLAB codes used in the study.

2 Introduction to B-splines

In this section we will define cubic B-splines and show the construction of bivariate or tensor products of B-splines on a two dimensional array. We will then give a discussion of how B-splines are computed. These specific computational algorithms will not be implemented in code, since they are available as functions in the MATLAB spline toolbox. The algorithmic discussions are given, though, as background to understanding the output from the MATLAB functions.

2.1 Cubic B-Splines in One Variable

Suppose that a function $y = f(x)$ is known at the n points $(x_1, y_1), \dots, (x_n, y_n)$, where $a < x_1 < x_2 < \dots < x_n < b$, $y_q = f(x_q)$, $q = 1, \dots, n$. It is known that a polynomial of degree $n - 1$, $P(x)$, can be constructed to pass through these n points. In the case of highly accurate data points this polynomial can be constructed to interpolate these n points by, for example, Lagrange polynomial or Newton divided difference algorithms. But, in the case of large n , it is also well known that

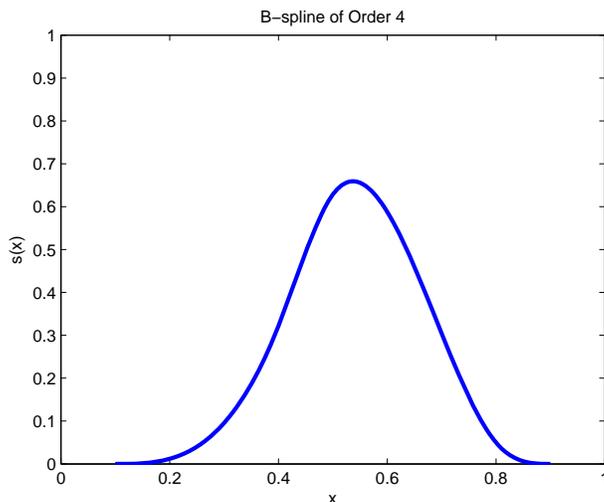


Figure 2: Plot of a B-spline of Order 4.

polynomials of high degree can produce unwanted oscillations between the interpolated points. It is crucial then to approximate sets of data with as low degree polynomials as possible. Of course these polynomials may or may not interpolate the data points but may be made to come as close to them as possible. The ability to create highly flexible approximants from low-degree polynomials is a significant advantage of functions called splines.

Given a set of r real numbers, satisfying $a < \xi_1 < \xi_2 < \dots < \xi_r < b$, a spline function, $s(x)$, of order p (or degree $p - 1$) with knots $\xi_1, \xi_2, \dots, \xi_r$ is a function that satisfies two properties: (1) in each of the intervals $x \leq \xi_1$; $\xi_{j-1} \leq x \leq \xi_j$, ($j = 2, 3, \dots, r$); $\xi_r \leq x$, $s(x)$ is a polynomial of degree $p - 1$ or less. (2) $s(x)$ and its derivatives of orders $1, 2, \dots, p - 2$ are continuous. This would mean, for example, a spline function of order four would be constructed from polynomials of degree three (cubic) or less on the intervals $x \leq \xi_1$; $\xi_{j-1} \leq x \leq \xi_j$, ($j = 2, 3, \dots, r$); $\xi_r \leq x$ with continuous derivatives of orders one and two.

A well known mathematical technique to construct complex functions is to form linear combinations of simpler functions, called basis functions. In the current report, data sets will be approximated by linear combinations of special spline functions, called B-splines, for Basis splines. It is known that any spline function can be represented in terms of B-splines. This particular basis has the advantage of leading to computational algorithms which are elegant, efficient, and stable. Only B-splines of order four will be considered here. B-splines of order four are cubic splines that are non-zero only over four adjacent intervals between knots (see Fig. 2). The notation for a B-spline is $N_{p,i}(x)$, where $N_{p,i}(x)$ is zero everywhere except in the range $\xi_{i-p} < x < \xi_i$, where in this work $p = 4$. As a note, sometimes B-splines are denoted in the literature by $B_{p,i}(x)$, but in this report we will use $N_{p,i}(x)$. To simplify notation let $N_i(x) = N_{4,i}(x)$. Then a B-spline of order four, or cubic B-spline, is a cubic spline with knots $\xi_{i-4}, \xi_{i-3}, \xi_{i-2}, \xi_{i-1}, \xi_i$ that is zero everywhere except in the range $\xi_{i-4} < x < \xi_i$. $N_i(x)$ is defined uniquely except for a scaling factor and is conventionally taken to be positive throughout the range $\xi_{i-4} < x < \xi_i$ and has a single maximum value. Since $N_i(x)$ is a cubic spline it has continuous derivatives of order one and two at ξ_{i-4} and ξ_i . These derivatives are zero at the endpoint knots.

To define a complete set of B-splines on the set of points $a < \xi_1 < \xi_2 < \dots < \xi_r < b$ it is

necessary to introduce eight additional points at the boundaries given by $\xi_{-3}, \xi_{-2}, \xi_{-1}, \xi_0, \xi_{r+1}, \xi_{r+2}, \xi_{r+3}, \xi_{r+4}$. It is usual to have $\xi_0 = a, \xi_{r+1} = b$. With this augmented set of knots one can define $m+4$ fundamental cubic B-splines, $N_i(x), i = 1, 2, \dots, m+4$. Then the general cubic B-spline has a unique representation in the range $a \leq x \leq b$ of the form

$$s(x) = \sum_{i=1}^{r+4} c_i N_i(x). \quad (1)$$

There are computational advantages of cubic B-splines. For any given x , all but four adjacent $N_i(x)$ are zero. In particular, if $x \in [\xi_{i-1}, \xi_i]$, the four non-zero cubic B-splines are $N_i(x), N_{i+1}(x), N_{i+2}(x), N_{i+3}(x)$.

A least squares curve fitting problem to the data $a < x_1 < x_2 < \dots < x_n < b, y_q = f(x_q), q = 1, \dots, n$, becomes one of determining the coefficients c_i as the least squares solution to the equations

$$\sum_{i=1}^{r+4} c_i N_i(x_q) = f_q, \quad q = 1, 2, \dots, n. \quad (2)$$

These may be written in matrix notation as

$$Ac = f, \quad (3)$$

where A is the $n \times (r+4)$ matrix whose element in column i of row q is $N_i(x_q)$ and c, f are column vectors with elements $c_i, f_q, q = 1, \dots, n$, respectively. If the data points are arranged in increasing order of x , then the matrix A becomes a banded matrix with bandwidth four. For a more thorough discussion of B-splines see de Boor [3].

2.2 Evaluation of B-splines

Let p be the order of the desired B-splines and let $\xi_1, \xi_2, \dots, \xi_r$ be the knots. In order to develop a complete set of B-splines we need to extend the knots by p points on each end as described in the previous section. On the left end, extend the knots by setting

$$\xi_{-(p-1)} = \xi_{-(p-2)} = \dots = \xi_0 = a, \quad (4)$$

and, on the right set

$$\xi_{r+1} = \xi_{r+2} = \dots = \xi_{r+p} = b. \quad (5)$$

Now define a working array, w , that incorporates all of the knots and the extended knot points as

$$w_i = \xi_{-p+i}, \quad i = 1, \dots, r+2p \quad (6)$$

In this section the double subscript for B-splines will be used since it is needed to define the computational algorithm. The B-splines, $N_{ji}(x)$, are constructed by a recursion formula given by

$$N_{ji}(t) = \begin{cases} 1 & \text{if } \xi_i \leq t \leq \xi_{i+1}, i = 1, \dots, r, \\ 0 & \text{otherwise,} \end{cases} \quad (7)$$

and

$$N_{ji}(t) = \frac{(t - \xi_i) N_{j-1,i}(t)}{\xi_{i+j-1} - \xi_i} + \frac{(\xi_{i+j} - t) N_{j-1,i+1}(t)}{\xi_{i+j} - \xi_{i+1}}, j = 1, \dots, p. \quad (8)$$

This formula is referred to in the literature as the Cox-deBoor formula (see Cox [2]).

To evaluate a B-spline of order p for a given $t \in [\xi_0, \xi_{r+1}]$ the first step is to find $i \in [k+1, r+k]$ so that $t \in [w_i, w_{i+1}]$. Note that if $t = \xi_{r+1}$ then $i = r+p$. Begin by initializing the array $N(j, i) = 0$ for $i = 1, \dots, r+p$, $j = 1, \dots, p$. By the Cox-deBoor formula set $N(1, i) = 1$, $N(1, q) = 0$, $q \neq i$ and

$$N(j, i) = \frac{(t - w(i))N(j-1, i)}{w(i+j-1) - w(i)} + \frac{(w(i+j) - t)N(j-1, i+1)}{w(i+j) - w(i+1)}. \quad (9)$$

We have dropped the function representation here and chosen an array form for the ease of algorithmic representation.

Once an interval index i has been identified with a given t , the evaluation of the final B-spline begins by setting $N(1, i) = 1$. It will happen that only p values $N(p, i-p+1)$, $N(p, i-p+2)$, \dots , $N(p, i)$ will be nonzero. This occurs since there is a triangular dependency of B-splines as

$$\begin{array}{cccc} & & & N(1, i) \\ & & & N(2, i-1) \quad N(2, i) \\ & & N(3, i-2) \quad N(3, i-1) \quad N(3, i) \quad . \\ & & \dots & \dots \quad \dots \\ N(p, i-p+1) & \dots & \dots & N(p, i) \end{array} \quad (10)$$

From the discussion here it will become clear why the computations for the relevant B-splines can be reduced to computing (10). In fact, the computation of the B-splines along the right leg of the triangle depends only on the B-splines just above on the right leg. The computation of the B-splines along the hypotenuse of the triangle depends only on the B-splines directly to the right and above along the hypotenuse. Finally, the B-splines within the triangle depend on the B-splines just above and to the right. This can easily be seen from the Cox-deBoor formula and (10). First of all $N(1, i) = 1$ initializes the recursion. By Cox-deBoor, $N(2, i)$ depends on $N(1, i)$ and $N(1, i+1)$, but $N(1, i+1) = 0$, so $N(2, i)$ depends only on $N(1, i)$. Now $N(2, i-1)$ depends on $N(1, i-1)$ and $N(1, i)$, but $N(1, i-1) = 0$, so $N(2, i-1)$ depends only on $N(1, i-1)$. Now consider $N(3, i)$. This depends on $N(2, i)$ and $N(2, i+1)$, but $N(2, i+1)$ depends on $N(1, i+1)$ and $N(1, i+2)$, which are both zero. So $N(3, i)$ depends only on $N(2, i)$. $N(3, i-1)$ depends, by Cox-deBoor, on $N(2, i-1)$ and $N(2, i)$, which are both nonzero. Finally, $N(3, i-2)$ depends on $N(2, i-2)$ and $N(2, i-1)$, but $N(2, i-2) = 0$ since it depends on $N(1, i-2) = 0$ and $N(1, i-1) = 0$. This argument can easily be extended down the triangle, but the pattern is clear. Any implementation of the Cox-deBoor formula can then be done by proceeding down the right leg, then down the hypotenuse and then fill in the interior of the triangle.

There is a property of B-splines that can be used to check the calculation of the final p nonzero B-splines. It is called a "partition of unity" property in that for any t

$$\sum_{q=i-p+1}^i N(p, q) = 1. \quad (11)$$

2.3 Tensor Products of Cubic B-splines in Two Variables

In Section 2.1 the problem of least squares approximation of data by one dimensional B-splines was discussed. In this section the B-spline concept will be extended to two dimensions in order to fit two dimensional scattered data by a surface function. In this surface-fitting problem data points (x_q, y_q) , $q = 1, 2, \dots, n$, and values at these points, $z_q = f(x_q, y_q)$, $q = 1, 2, \dots, n$, are given. The surface model used to fit these data points will involve sums of products of B-splines.

matrix might be rank deficient and it will not be possible to use the traditional normal equations to solve a least squares problem. This can potentially happen in the case of widely scattered data where some of the logical panels defined by the knots do not contain data points. This problem can be solved, though, by using the matrix singular value decomposition.

For the moment, assume that a tensor product spline has been computed as described in the previous section. For notation, designate it by NM . An implementation of a tensor product spline algorithm is given by the MATLAB function `tensor_prod_spl` in Appendix C.4. A least squares fit can be done to the scattered data as follows. Since the matrix NM could be rank deficient, with potentially zero rows or columns, we cannot rely on the standard normal equations for determining the coefficients but using the matrix singular value decomposition provides a convenient substitute. This is done in the MATLAB function `lsq_2D_spline` in Appendix C.5. The input to this function is the tensor product matrix NM and the array of $z_q, q = 1, \dots, n$, values at the scattered data points, along with a desired data tolerance. The data tolerance is used to determine which of the small singular values in the decomposition should be considered zero. In the function `lsq_2D_spline` the matrix NM , with the number of rows larger than the number of columns, is decomposed by the MATLAB function `svd` into the product of a column-orthogonal matrix U , a diagonal matrix S , with diagonal element S_i , and the transpose V' of a square orthogonal matrix, so that $NM = USV'$. In order to solve the problem $NMc = z$ we compute $c = VS^+U'z$, where

$$S_i^+ = \begin{cases} \frac{1}{S_i} & S_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

defines the generalized inverse of S and U' is the transpose of U .

2.5 A Knot Selection Algorithm

In a least squares data fitting process involving B-spline basis functions, the resulting model residuals are sensitive to the knot placement for the B-splines. The selection of B-spline knots in order to achieve as small a residual as possible during the least squares process is a nontrivial task. One would be extremely lucky to manually select a set of knots that could achieve a very small least squares residual. In this section we will describe a heuristic algorithm that, in practice, generates a set of knots that produces small least squares residuals. The strategy is a knot insertion algorithm and is iterative. An initial set of knots is selected by the algorithm user and at each iteration of the algorithm new knots are inserted in the vicinity of the previous fit where the local residuals are the largest. The knots are not moved once they have been inserted. The iterative algorithm has a stopping criteria that will also be discussed below.

First we will discuss the knot placement algorithm. It is based on a strategy suggested by Dierckx [6]. At the beginning of each iteration the assumption is that there exists a current set of knots. In the first iteration of the algorithm these would be an initial set chosen by the user. The tensor product of the B-splines is computed at the data points, a least squares fit is made to the data, and the current absolute residuals of the fit are computed at each data point. The current knots divide the (x, y) plane into logical rectangles. Some rectangles have data points and others don't. Let the knots at the k -th iteration be labeled $a < \xi_1^{(k)} < \xi_2^{(k)} < \dots < \xi_{r_k}^{(k)} < b$ along the x -axis and $c < \eta_1^{(k)} < \eta_2^{(k)} < \dots < \eta_{s_k}^{(k)} < d$ along the y -axis. The $(k + 1)$ iteration begins by associating all of the data points with the knot panels in which they fall. Let the index pair ij indicate the R_{ij} -th panel defined by $\xi_i \leq x \leq \xi_{i+1}$, $\eta_j \leq y \leq \eta_{j+1}$. Then suppose the data values $(x_1^{(ij)}, y_1^{(ij)})$, \dots , $(x_{r_{ij}}^{(ij)}, y_{r_{ij}}^{(ij)})$ fall into the R_{ij} -th panel. Let $s_k(x, y)$ be the least squares B-spline function fit to the data at the k -th iteration. We next compute the residuals of the fit at all of the

data points

$$\text{Resid}_q = z_q - s_k(x_q, y_q), \quad q = 1, \dots, n. \quad (17)$$

From these residuals we form the sums of squares of the residuals that fall within the knot intervals. The sums are separated into the x direction and the y direction as follows.

$$\begin{aligned} \delta_i^{(k)}(x) &= \sum_q \left\{ \text{Resid}_q^2 : \xi_i^{(k)} \leq x_q < \xi_{i+1}^{(k)} \right\}, \quad i = 1, 2, \dots, r_k, \\ \delta_j^{(k)}(y) &= \sum_q \left\{ \text{Resid}_q^2 : \eta_j^{(k)} \leq y_q < \eta_{j+1}^{(k)} \right\}, \quad j = 1, 2, \dots, s_k. \end{aligned} \quad (18)$$

We find the maximums of these sums of squared residuals.

$$\begin{aligned} \delta_u^{(k)}(x) &= \max \left\{ \delta_i^{(k)}(x) : i = 1, 2, \dots, r_k \right\}, \\ \delta_v^{(k)}(y) &= \max \left\{ \delta_j^{(k)}(y) : j = 1, 2, \dots, s_k \right\}, \end{aligned} \quad (19)$$

where $u = i$ for some $i = 1, 2, \dots, r_k$ and $v = j$ for some $j = 1, 2, \dots, s_k$. The next step is to add one knot at a time at each iteration. In particular, a knot is added in the x direction if $\delta_u^{(k)}(x) > \delta_v^{(k)}(y)$ or added in the y direction if $\delta_u^{(k)}(x) < \delta_v^{(k)}(y)$. The knot added is positioned based upon a weighted average of x or y data points in the columns or rows determined by the knot intervals with the maximum residual errors determined by (19). In particular, the positions are given by

$$\begin{aligned} \xi_{new}^{(k+1)} &= \sum_{\xi_u^{(k)} \leq x_q < \xi_{u+1}^{(k)}} \left\{ \frac{\text{Resid}_q^2 x_q}{\delta_u^{(k)}(x)} \right\}, \\ \eta_{new}^{(k+1)} &= \sum_{\eta_v^{(k)} \leq y_q < \eta_{v+1}^{(k)}} \left\{ \frac{\text{Resid}_q^2 y_q}{\delta_v^{(k)}(y)} \right\}. \end{aligned} \quad (20)$$

The knots are then reindexed as necessary and $s_{k+1}(x, y)$ is computed as the least squares B-spline function fit to the data at the $(k + 1)$ iteration based on the new knot set.

The stopping criteria used in this algorithm is based on the use of the R^2 -statistic, called the coefficient of multiple determination (see Draper and Smith [7]). R^2 is the square of the correlation between the vector of observed data, r_k , $k = 1, \dots, n$, and the least squares predicted data, \hat{r}_k , $k = 1, \dots, n$. The statistic satisfies $0 \leq R^2 \leq 1$. This statistic is often used as a measure of how well the regression equation explains the variation in the data. It is known that building models based on adding terms in the regression equation, care must be taken in using this statistic. However, in the current algorithm the statistic is used in a somewhat non-conventional manner. We use it as a measure of the benefit of adding more knots to the tensor product spline model. The knot selection algorithm is terminated once $R^2 > 0.98$.

Although the knot placement algorithm is heuristic, coupling it with the R^2 -statistic has shown good convergence in practice. It is reasonable to expect this, since knots are placed in intervals in which the fits at a previous iteration showed the largest local error. Placing a knot there allows extra terms to be inserted into the tensor product for those areas.

3 Surface Point Data Generation

In this section we discuss the experimental method by which boundary data was generated in Euclidean coordinates on each of the phantoms. Figure 3 shows the CMM system used to measure



Figure 3: Computer controlled CMM.

the phantoms and a calibrated sphere used to check the probe. The system is computer controlled and touches an object to be measured at programmed points in order to produce (x, y, z) values at the probed points.

The effective diameter of the CMM probe was measured by first measuring the high quality steel sphere, with a well-calibrated diameter. This sphere is separate from the calibrated one used to generate the sphere data in Section 4.2.1 below. The calibration sphere can be seen in the background of Fig. 4. The difference between the known diameter and the apparent diameter of the measured reference metal sphere gave the calibrated effective probe diameter. A similar pattern of probe touches was used for the measurement of the calibrated sphere and for all of the phantom sphere measurements.

The FDA phantoms were created by a molding process and the mold marks on both spheres were visible. These mold marks were used to align the phantoms with the coordinate system of the CMM. The marks were laid out like lines of latitude, leading to the use of a natural nomenclature of latitude and longitude like those of the Earth. For each phantom, the North Pole was chosen to be the one with darker, deeper, or more obvious mold marks.

For the purpose of understanding the measurement process we will describe the physical fixturing of the phantoms. They were held by a vacuum chuck (see Fig. 4) to minimize distortion and reduce the chance of damaging the spheres' surfaces. As Fig. 4 shows the chuck has a shallow cone to hold the phantoms. The phantoms contacted the cone around a circle of latitude at about -45° . This was measured from the equatorial circle around the middle of the phantoms. For example, the point at the top of the phantoms would be at $+90^\circ$. The wall vacuum of the cone was strong enough to hold the spheres sufficiently that they did not move significantly, as shown by the repeatable results from run to run at the $1 \mu\text{m}$ level. The setup of the phantoms in the vacuum chuck was accomplished by eye alignment using the visible mold marks and minor imperfections as guides to the eye. The expanded uncertainty of alignment ($k = 2$, according to the guidelines of Taylor and Kuyatt [18]) was estimated to be approximately $\pm 2^\circ$ for the vertical angle (i. e. keeping the equator horizontal) and $\pm 4^\circ$ for the azimuthal angle.

Three sets of measurements were planned for each of the phantoms. In each, points were probed



Figure 4: Lung nodule phantom on the vacuum chuck.

	Equatorial	Polar
Pink	20.0 mm	18.4 mm
Green	20.0 mm	20.1 mm

Table 1: Dial Caliper Measurements of Phantoms' Diameters

on only a hemisphere at a time and later the data sets were post-processed to form a spherical data set. In the first set of measurements the North Pole was set as the top point. A set of points were programmed to probe the top hemisphere down to the equator. The phantom was then re-positioned in the chuck so that the South pole was the top point. The same hemispherical points were probed. The rotation was done in such a manner that the mold marks representing the longitudinal lines were kept as aligned as possible. Post-processing of the data associated the correct signs with the measured coordinates relative to the CMM coordinate system. The third measurement involved re-positioning the phantoms so that the equatorial circle was vertical and the North-South axis was horizontal. Again two hemisphere sets of probe points were measured. This position was not feasible for the Pink phantom in one of the experiments described below.

Visually, the pink sphere was noticeably out-of-round, in the shape of an oblate spheroid. A dial caliper gave diameter measurements given in Table 1.

The uncertainty of caliper measurements on hard steel surfaces is about 0.1 mm, and is estimated to be about 0.3 mm on the sample spheres due to the potential that the contact force would distort the soft surfaces of the spheres (all estimated uncertainties were $k = 2$ expanded uncertainties). Since the CMM is primarily used to measure metallic artifacts produced by machining operations, we will see in this report some of the possible consequences of using the CMM to measure the non-metallic phantoms.

Two probing experiments were performed on each of the phantoms and the calibrated sphere. They created what we will call a coarse data set and a dense data set. In the next two sections we will discuss the methods used in the two experiments since they were slightly different.

3.1 Experiment 1: Coarse Data Set

In this experiment the plan was to measure each phantom on the CMM three times in each position, with 61 coordinate points per hemisphere. For the green sphere, each measurement set consisted of three separate sets of points: North pole up, South pole up, and prime meridian/equator intersection

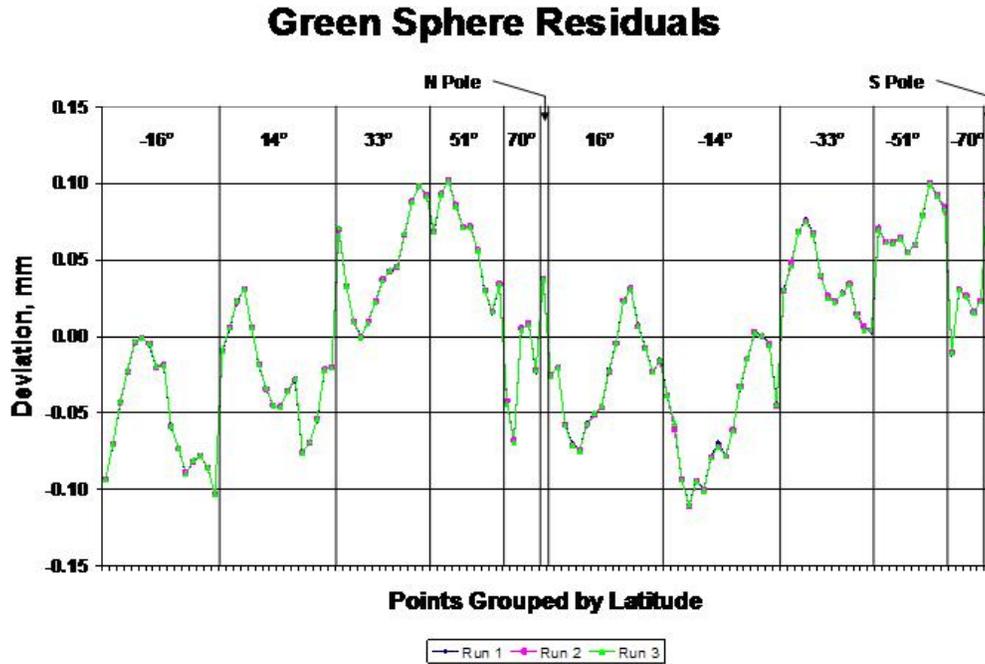


Figure 5: Residual Measure CMM Errors for the Green Phantom Sphere: Experiment 1.

up. The pink sphere could not be held sideways in the first experiment, as the out-of-roundness prevented an effective vacuum seal. Therefore, a measurement data set for this sphere had only the North Pole up and South Pole up data. The third data set in this case was a re-measure of the North Pole up position.

Although 183 points were measured on the green sphere, 121 were selected in order to make the data sets for the green and pink phantoms consistent in terms of probed points. All 121 points from both hemispheres were merged together, and the plots in Fig. 5 and 6 show the radial deviation from a best-fit sphere for the full data sets. The figures show the radial residuals obtained by fitting sphere models to the Green and Pink data with an algorithm ordinarily used during sphere calibration work. In particular, they represent the residual errors between the distance from the fitted sphere center to the probed points and the fitted radius of the sphere model. The residuals, in the case of the Green phantom, range from approximately -0.1 mm to $+0.1$ mm, whereas the residuals, in the case of the Pink phantom, range from approximately -0.57 mm to $+0.23$ mm. This indicates the non-spherical nature of the Pink phantom. A separate nonlinear algorithm for sphere fitting will be described in Section 4. It produces consistent results. Figure 7 shows the typical distribution of the probe points on a sphere. The plot is a transparency so that probe points on the opposite side of the sphere are visible. The calibrated sphere on the shaft, with a diameter of 19.05 mm, was also measured with 121 points, repeated five times.

3.2 Experiment 2: Dense Data Set

In this experiment 181 points were taken on the phantoms and the calibrated sphere, with five repeats in each position. The positions were taken the same as those in Experiment 1. That is,

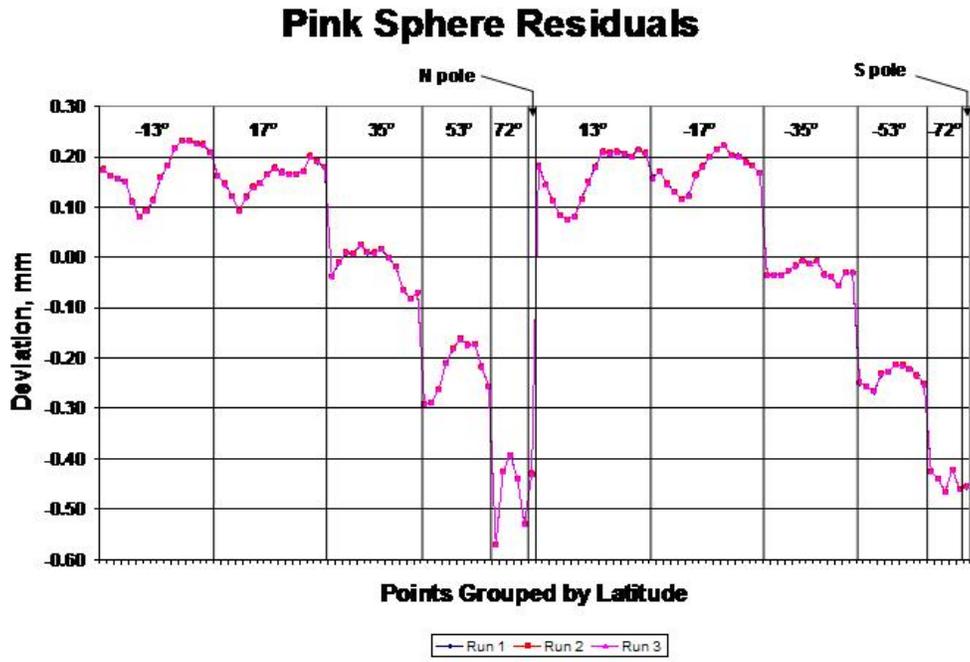


Figure 6: Residual Measure CMM Errors for the Pink Phantom Sphere: Experiment 1.

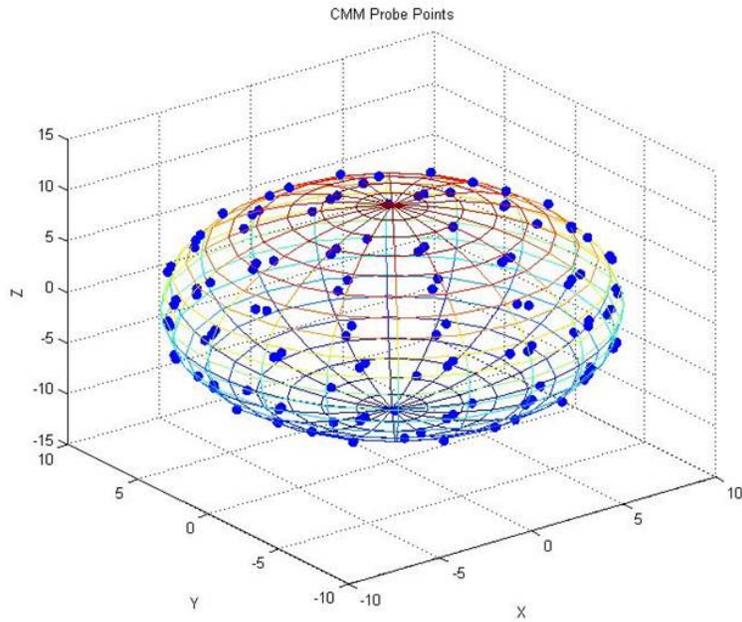


Figure 7: Distribution of the CMM Probe Points on a Sphere: Experiment 1.

the alignment was selected with North pole up (position 1), South pole up (position 2), and prime meridian/equator intersection up (position 3). In this case it was possible to hold the Pink phantom in the sideways position 3. The calibrated sphere was measured in only one position, with five repeats, since it was permanently mounted on a support shaft.

4 Data Modeling and Volume Estimation

In this section two forms of data modeling will be discussed. Since the phantoms seemed to be nearly spherical the natural tendency was to first consider fitting a spherical model to each phantom and estimating the volume of the fitted spheres. However, in order to develop a potentially more accurate volume estimation model the surface data was also fit using tensor products of B-splines and the volumes estimated by the Divergence Theorem.

4.1 A Spherical Model

Since the phantoms were nearly spherical it was natural, as noted above, to consider how close the data could be modeled by first assuming spherical models for the data. The calibrated sphere, of course, could clearly be modeled with a spherical model.

In particular, let $c = (c_1, c_2, c_3, c_4)$ and set

$$f(x, y, z, c) = (x - c_1)^2 + (y - c_2)^2 + (z - c_3)^2 - c_4^2. \quad (21)$$

The unknown parameters c_1, c_2, c_3 represent the center of the sphere and c_4 is the radius. All of the data were measured in millimeters so that the parameters naturally have millimeter units. Define

$$F(c) = \sum_{i=1}^n \left\{ (x_i - c_1)^2 + (y_i - c_2)^2 + (z_i - c_3)^2 - c_4^2 \right\}^2, \quad (22)$$

where F is a measure of the residual for the fitted sphere defined by the coefficients c . Since the function F is a nonlinear implicit function of the parameters we needed to use a nonlinear minimization algorithm to find the best fit, i.e. to solve the problem

$$\min_c F(c). \quad (23)$$

There are a number of algorithms for fitting least squares models to data on geometric shapes. The reader can consult Shakarji [16]. There are also various algorithms for minimizing general nonlinear functions, such as (22). All of the algorithms involve iterative minimization of some form. Many require computing derivatives of the objective function in order to generate search directions along which to identify a minimum. Others do not involve derivatives but may be somewhat slower in the minimization search. The algorithm selected here, because of the relatively few parameters involved, and the fact that derivatives are not required, is a form of polygon search method called the Nelder-Mead method (see Sauer [14]). It should be noted here that the Newton's method, requiring derivatives, was initially used to estimate the parameters, but the Nelder-Mead tended to produce the smallest value to (22).

The Nelder-Mead method begins with an initial vector guess for the minimum in R^4 (although the method works for R^n , but due to the "curse of dimension", becomes prohibitive beyond R^{10}). From the initial vector guess a polygon of five vertices is built. The function evaluations at the vertices, $w_i = F(v_i)$, of the polygon are tested and put into ascending order, $w_1 < w_2 < \dots < w_5 = w_h$. The

polygon vertex, v_h , with the largest function value, w_h , is replaced as follows. The centroid, \bar{v} , of the face of the polygon that does not contain v_h is chosen. The function (22) is evaluated at, v_r , the reflection point about the centroid, given by $v_r = 2\bar{v} - v_h$, is selected. If the value $w_r = F(v_r)$ lies in the range $w_1 < w_r < w_4$, then v_r replaces v_h , the function values of the vertices are again sorted, and the iterative step is repeated. If $w_r < w_1$, extrapolation is continued in the direction of w_r by setting $v_e = 3\bar{v} - 2v_h$. The better of v_e or v_r is selected to replace v_h . Finally, if $w_r > w_n$ then a contraction of points is attempted. Two points are selected. The outer contraction point is taken as $v_{oc} = 1.5\bar{v} - 0.5v_r$ and the inner contraction is $v_{ic} = 0.5\bar{v} + 0.5v_h$. The function (22) is evaluated at both contraction points. If there is no reduction in the value of (22) then the polygon is shrunk by a factor of 2 in the direction of the current minimum, v_1 . The iterations of sorting and testing continue until the volume of the polygon becomes less than some prespecified tolerance.

A MATLAB code to estimate the center and radius of the best fit sphere, given a set of (x, y, z) points, is given in Appendix A. The output of the program produces a vector that is the median value of the vertex values of the final polyhedron, whose volume satisfies the tolerance specification. The volume is computed as a determinant involving the polygon vertices.

The uncertainties of the estimated center and radius were computed using the methods proposed in Draper and Smith [7] for nonlinear regression. In particular, if $\hat{c} = (\hat{c}_1, \hat{c}_2, \hat{c}_3, \hat{c}_4)$, then define the matrix with elements

$$\hat{Z}_{ij} = \frac{\partial F}{\partial c_j}(x_i, y_i, z_i, \hat{c}), \quad i = 1, \dots, n, \quad j = 1, 2, 3, 4. \quad (24)$$

The matrix \hat{Z} is an $n \times 4$ matrix. The i -th row is given by

$$\hat{Z}_i = [-2(x_i - \hat{c}_1) \quad -2(y_i - \hat{c}_2) \quad -2(z_i - \hat{c}_3) \quad -2\hat{c}_4]. \quad (25)$$

If we then form $\hat{Z}'\hat{Z}$ the standard error of \hat{c}_i , s.e. (\hat{c}_i) , is given by the square root of the i -th diagonal element of $(\hat{Z}'\hat{Z})^{-1} s^2$ where $s = S(\hat{c}) / (n - 4)$. The expanded uncertainty is given by $2\text{s.e.}(\hat{c}_i)$ as defined in Taylor and Kuyatt [18]. The uncertainty limits of \hat{c}_i are $\hat{c}_i \pm 2\text{s.e.}(\hat{c}_i)$. All units are in millimeters except volume, which is in cubic millimeters.

4.2 Computational Results for the Spherical Model

Tables 2, 3, and 4 present the results obtained by fitting spherical models to the data from the calibrated sphere and the two phantoms. In Experiment 1 there were three data sets for each of the Green and Pink phantoms, designated Green 4, 5, and 6 and Pink 5, 6, and 7. Since the output of the Nelder-Mead algorithm was a final polytope surrounding the minimum with the polytope vertices representing the final parameter estimates, the final reported parameters were selected as the median values of the vertex values. These are the first four entries in the table: Center x, Center y, Center z, and Radius. The units are millimeters. The fifth table entry is the spherical volume based on the median radius value in cubic millimeters. The radius residuals were computed as

$$\text{resid}_i = \sqrt{(x_i - \hat{c}_1)^2 + (y_i - \hat{c}_2)^2 + (z_i - \hat{c}_3)^2} - \hat{c}_4. \quad (26)$$

The sixth and eighth entries in the tables give the mean value and standard deviation of these residuals. The units are millimeters for these entries. The seventh and ninth table entries give the mean and standard deviation of the absolute values of the residuals in millimeters. The tenth through the thirteenth entries are the expanded uncertainties for the estimated Center x, Center y, Center z, and radius.

Spherical Fit Results for Calibrated Sphere		
Properties	Coarse	Dense
Center x	0.2803×10^{-4}	-0.1331×10^{-2}
Center y	-0.4881×10^{-3}	-0.2034×10^{-2}
Center z	-0.6642×10^{-2}	-0.2564×10^{-2}
Radius	9.5326	9.5314
Est. Volume	3628.4	3627.1
Mean Rad. Residual	0.3232×10^{-4}	0.8720×10^{-4}
Mean Abs. Rad. Residual	0.1646×10^{-2}	0.1700×10^{-2}
Stand. Dev. Rad. Residual	0.2156×10^{-2}	0.22302×10^{-2}
Stand. Dev. Abs. Rad. Residual	0.1384×10^{-2}	0.1441×10^{-2}
Expanded Uncert. Center x	0.2711×10^{-4}	0.2357×10^{-4}
Expanded Uncert. Center y	0.2711×10^{-4}	0.2357×10^{-4}
Expanded Uncert. Center z	0.4316×10^{-4}	0.3757×10^{-4}
Expanded Uncert. Radius	0.2188×10^{-4}	0.1909×10^{-4}

Table 2: Results of a Spherical Fit to Coarse and Dense Data for Calibrated Sphere

The results in these sections on spherical model fitting involve strictly fitting the non-linear model (26) to the sparse data sets generated by the CMM. This process does not involve the B-spline algorithm used later in this report. Therefore any differences between coarse and dense results are most likely the consequence of the point selection in the metrology of the artifacts.

4.2.1 Results for the Spherical Model Fit to the Calibrated Sphere with Coarse and Dense Data

Table 2 gives the results of the fit of the sphere model to both the coarse (121 points) and the dense (181 points) data sets for the calibrated sphere. In both cases the average of the five repeat data sets was used for the fitting process. As can be seen, the two volume estimates differ by approximately 0.04%. The radii estimates differ by about 0.01%. There is a larger difference when the spherical fit results are compared to the volume estimate based on the calibrated sphere diameter of 19.5 mm. This would lead to a volume estimate of 3619.8 mm^3 . The measured radius differs from the computed radii in Table 2 by about 0.08% whereas the volume estimate based on the measured radius differs from the volume estimates in Table 2 by about 0.2%. It would seem that these differences are within the operating regime of the CMM. Although the estimated centers are slightly different, all other values are of the same order of magnitude. Based on these results we can accept that the CMM is producing accurate position data for spherical metallic artifacts. Tables 3 through 6, however, begin to show the consequences involved with attempting to measure slightly non-spherical and non-metallic artifacts with the CMM.

4.2.2 Results for the Spherical Model Fit to the Phantoms with Coarse Data

From Tables 3 and 4 it is clear that the Green phantom is more spherical than the Pink phantom as indicated by the residuals and the expanded uncertainties. This simply confirms the fact that the Pink phantom was more difficult to measure using the vacuum chuck due to its lack of sphericity. For example, the Mean Radial Residual for the Pink data is two orders of magnitude larger than for the Green data. The Standard Deviations for the Pink data are an order of magnitude greater,

Spherical Fit Results			
Properties	Green 4	Green 5	Green 6
Center x	0.2150×10^{-2}	0.2128×10^{-2}	0.2579×10^{-2}
Center y	-0.4694×10^{-2}	-0.4813×10^{-2}	-0.4611×10^{-2}
Center z	-0.5482×10^{-1}	-0.5489×10^{-1}	-0.5442×10^{-1}
Radius	10.1124	10.1121	10.1118
Est. Volume	4331.7	4331.3	4330.8
Mean Rad. Residual	-1.7530×10^{-4}	-1.764×10^{-4}	-1.4684×10^{-4}
Mean Abs. Rad. Residual	0.4592×10^{-1}	0.4607×10^{-1}	0.4586×10^{-1}
Stand. Dev. Rad. Residual	0.5521×10^{-1}	0.5538×10^{-1}	0.5518×10^{-1}
Stand. Dev. Abs. Rad. Residual	0.3038×10^{-1}	0.3043×10^{-1}	0.3041×10^{-1}
Expanded Uncert. Center x	0.1923×10^{-1}	0.1934×10^{-1}	0.1920×10^{-1}
Expanded Uncert. Center y	0.1923×10^{-1}	0.1934×10^{-1}	0.1920×10^{-1}
Expanded Uncert. Center z	0.2121×10^{-1}	0.2133×10^{-1}	0.2118×10^{-1}
Expanded Uncert. Radius	0.1145×10^{-1}	0.1151×10^{-1}	0.1143×10^{-1}

Table 3: Results of a Spherical Fit to Coarse Data for the Green Phantom

as are the Expanded Uncertainties. It is not clear how much the phantom material affected the results since it was difficult to set up a specific probe test for metallic versus phantom material. The artifacts would have to have been exactly the same size, positioned at exactly the right location, and probed at exactly the same coordinates to separate those factors from the material factor difference. There were no such comparable artifacts. We can make a guess, though, that there might be some effect due to probe force against the non-metallic material if we look at Table 2 and Table 3 for the Green phantom, the most spherical of the two phantoms. The Expanded Uncertainties for the sphere fit to the calibrated metallic sphere and the Expanded Uncertainties for the sphere fit to the Green phantom data differ by one to three orders of magnitude. Since the repeatability of the CMM measurements is at the $1 \mu\text{m}$ level, it is likely then that material difference had some significant affect on the difference in the uncertainties. It is a conjecture that this and the non-spherical shape of the Pink phantom account for a large part of the differences between the Pink phantom Expanded Uncertainties in Table 4, the Green phantom Expanded Uncertainties in Table 3, and the calibrated sphere Expanded Uncertainties in Table 2.

4.2.3 Results for the Spherical Model Fit to the Phantoms with Dense Data

We want to make some observations about the resulting measurements and the spherical fits to the dense data. First of all Figures 8 and 9 for the Green phantom have similar residual patterns, as do Figures 11 and 12 for the Pink phantom. If we look at the residuals for the sphere fits to the dense data sets in Figures 10 and 13, we see an interesting phenomenon occurring in the fits for both the Green and Pink phantoms in position 3 (prime meridian/equator intersection up). There appears to be a definite periodic oscillation in the residual data in Figures 10 and 13. It is not clear what in the measurement process produced these residual oscillations, although the points selected would have been chosen along longitudes as the CMM moved from top to bottom of the positioned phantom. These periodicities are possible considering the North to South symmetry across the equator in the sphere model fit. Next we examined the possible effects of these residual oscillations on the sphere volume estimates.

If we compare the volume results for the coarse Green and dense Green data we get the following.

Spherical Fit Results			
Properties	Pink 5	Pink 6	Pink 7
Center x	-0.8612×10^{-2}	-0.8235×10^{-2}	-0.8373×10^{-2}
Center y	0.1572×10^{-1}	0.1547×10^{-1}	0.1555×10^{-1}
Center z	0.2315×10^{-1}	0.2365×10^{-1}	0.2359×10^{-1}
Radius	9.7337	9.7334	9.7329
Est. Volume	3863.02	3862.61	3862.03
Mean Rad. Residual	-0.2222×10^{-2}	-0.2721×10^{-2}	-0.2356×10^{-2}
Mean Abs. Rad. Residual	0.1748	0.1746	0.1746
Stand. Dev. Rad. Residual	0.2128	0.2127	0.2126
Stand. Dev. Abs. Rad. Residual	0.1203	0.1203	0.1203
Expanded Uncert. Center x	0.2690	0.2686	0.2684
Expanded Uncert. Center y	0.2696	0.2693	0.2691
Expanded Uncert. Center z	0.2976	0.2971	0.2969
Expanded Uncert. Radius	0.1604	0.1602	0.1601

Table 4: Results of a Spherical Fit to Coarse Data for the Pink Phantom

From Table 3 and Table 5 there appears to be a decrease of about 1 % from Green 4 volume to Dense Green 1 and similarly for Green 5 to Dense Green 2. Green 6 and Dense Green 3 volumes differ by about 0.08 %. Therefore the oscillations in the residuals for the Dense Green 3 data do not seem to have affected the volume estimate. The residuals themselves in Fig. 10 oscillate between -0.15 mm and 0.1 mm. However, when we compare the Pink phantom volume estimates we get a vastly different result. Whereas the residuals in Figures 11 and 12 fall within the range of -0.4 to 0.15 mm, the residuals as shown in Fig. 13 oscillate between -0.6 mm and 0.6 mm, about six times the residual oscillations of the Dense Green 3 data. The damping of the oscillations may have been due to the slight non-spherical shape of the Pink phantom, although this is a conjecture. The volume differences between Pink 5 in Table 4 and Dense Pink 1 in Table 6 as well as Pink 6 and Dense Pink 2 in the same tables are approximately 6 %. The volume estimates between the Pink 7 results in Table 4 and the Dense Pink 3 results in Table 6, however, show about a 7 % decrease. This last comparison may not be a fair one in that the Pink 7 data did not have data for the Pink phantom in position 3. Table 1 shows that the diameter of the Pink phantom falls approximately between 20 mm and 18.5 mm. This would imply that the sphere model should have a volume between 4188.8 mm^3 and 3315.2 mm^3 . Table 4 for the Pink phantom shows that for the coarse data the volume estimate falls from 3862 mm^3 to about 3863 mm^3 . These volumes are within the expected range. Whereas the volume estimates for Dense Pink 1, Dense Pink 2, and Dense Pink 3 from Table 6 are also within the expected range. The Pink phantom in position 3 may have shown difficulties in the data acquisition phase for the CMM. This could explain the lower volume estimate. In particular, an approximate 4 % difference in radius estimate between Dense Pink 3 and Dense Pink 2 causes an approximate 13 % decrease in volume estimate. This points out how seemingly small differences in linear parameter estimates can make significant volumetric differences.

4.3 Data Modeling by B-Splines

In the following sections we introduce a more general approach to estimate the phantom volumes by modeling the surfaces with B-splines and using the Divergence Theorem to estimate the volume. In the case of the Green phantom, the sphere model and the B-spline model will confirm its near

Spherical Fit Results			
Properties	Dense Green 1	Dense Green 2	Dense Green 3
Center x	-0.1073×10^{-1}	0.2112×10^{-1}	-0.1321×10^{-1}
Center y	0.5112×10^{-1}	0.8661×10^{-2}	-0.4700×10^{-1}
Center z	0.2550	0.4723×10^{-1}	0.2139
Radius	10.0706	10.0710	10.1146
Est. Volume	4278.1	4278.7	4334.5
Mean Rad. Residual	-0.9344×10^{-4}	-0.7116×10^{-4}	-0.8881×10^{-4}
Mean Abs. Rad. Residual	0.3502×10^{-1}	0.3267×10^{-1}	0.3980×10^{-1}
Stand. Dev. Rad. Residual	0.4566×10^{-1}	0.3935×10^{-1}	0.4904×10^{-1}
Stand. Dev. Abs. Rad. Residual	0.2918×10^{-1}	0.2181×10^{-1}	0.2850×10^{-1}
Expanded Uncert. Center x	0.1032×10^{-1}	0.7720×10^{-2}	0.1200×10^{-1}
Expanded Uncert. Center y	0.1032×10^{-1}	0.7720×10^{-2}	0.1196×10^{-1}
Expanded Uncert. Center z	0.1570×10^{-1}	0.1184×10^{-1}	0.1828×10^{-1}
Expanded Uncert. Radius	0.7820×10^{-2}	0.5963×10^{-2}	0.9119×10^{-2}

Table 5: Results of a Spherical Fit to Dense Data for the Green Phantom

Spherical Fit Results			
Properties	Dense Pink 1	Dense Pink 2	Dense Pink 3
Center x	0.1472×10^{-1}	0.4573×10^{-1}	0.1496×10^{-1}
Center y	0.1448×10^{-1}	0.1557×10^{-1}	0.6815×10^{-1}
Center z	0.6848	0.7473	-0.3443
Radius	9.9234	9.9370	9.5045
Est. Volume	4093.2	4110.1	3596.5
Mean Rad. Residual	-0.4393×10^{-3}	-0.4859×10^{-3}	-0.3282×10^{-2}
Mean Abs. Rad. Residual	0.7993×10^{-1}	0.7989×10^{-1}	0.2149
Stand. Dev. Rad. Residual	0.9764×10^{-1}	0.1001	0.2567
Stand. Dev. Abs. Rad. Residual	0.5576×10^{-1}	0.5996×10^{-1}	0.1397
Expanded Uncert. Center x	0.4604×10^{-1}	0.4820×10^{-1}	0.3061
Expanded Uncert. Center y	0.4599×10^{-1}	0.4811×10^{-1}	0.3185
Expanded Uncert. Center z	0.6926×10^{-1}	0.7224×10^{-1}	0.4854
Expanded Uncert. Radius	0.3365×10^{-1}	0.3501×10^{-1}	0.2505

Table 6: Results of a Spherical Fit to Dense Data for the Pink Phantom

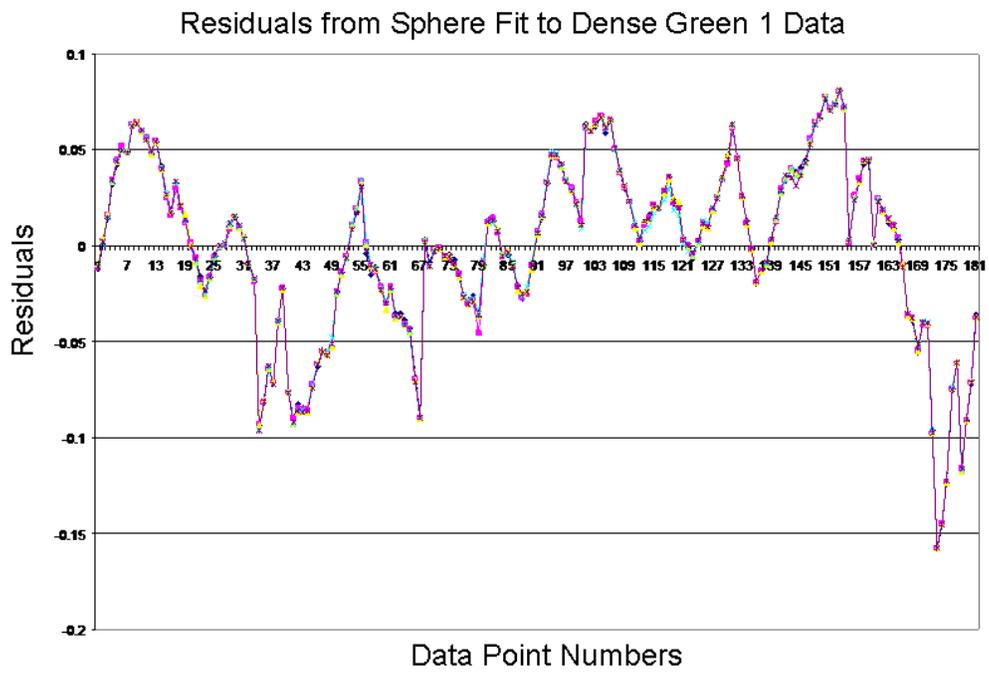


Figure 8: Residuals for the Sphere Fit to the Dense Green Data Set 1.

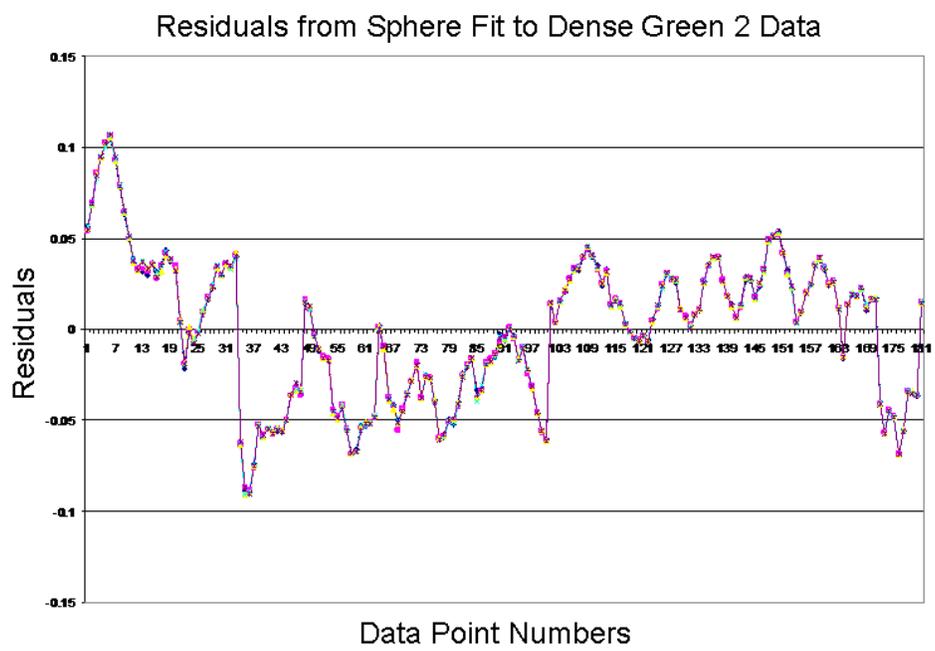


Figure 9: Residuals for the Sphere Fit to the Dense Green Data Set 2.

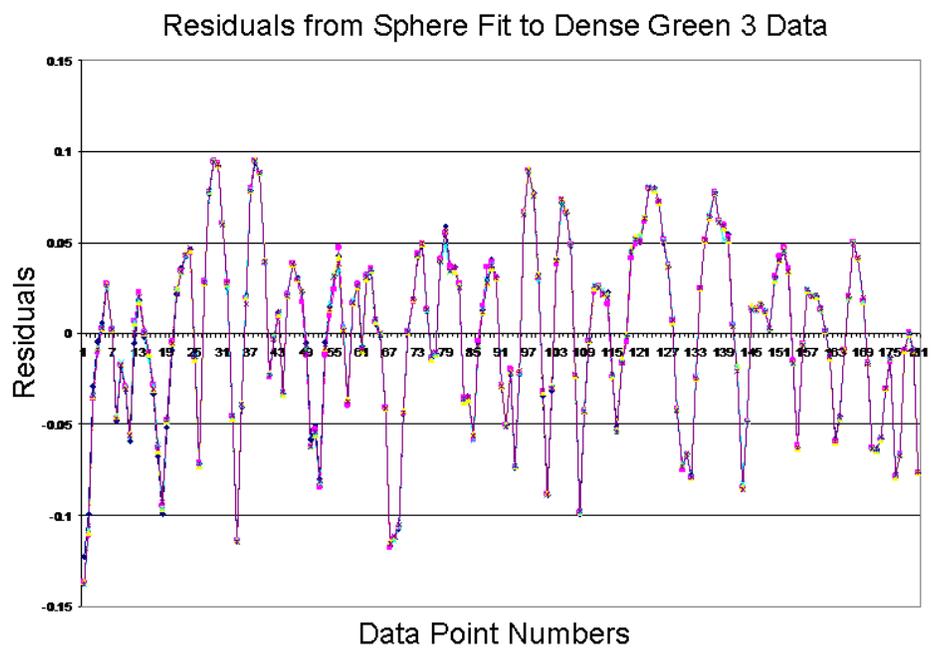


Figure 10: Residuals for the Sphere Fit to the Dense Green Data Set 3.

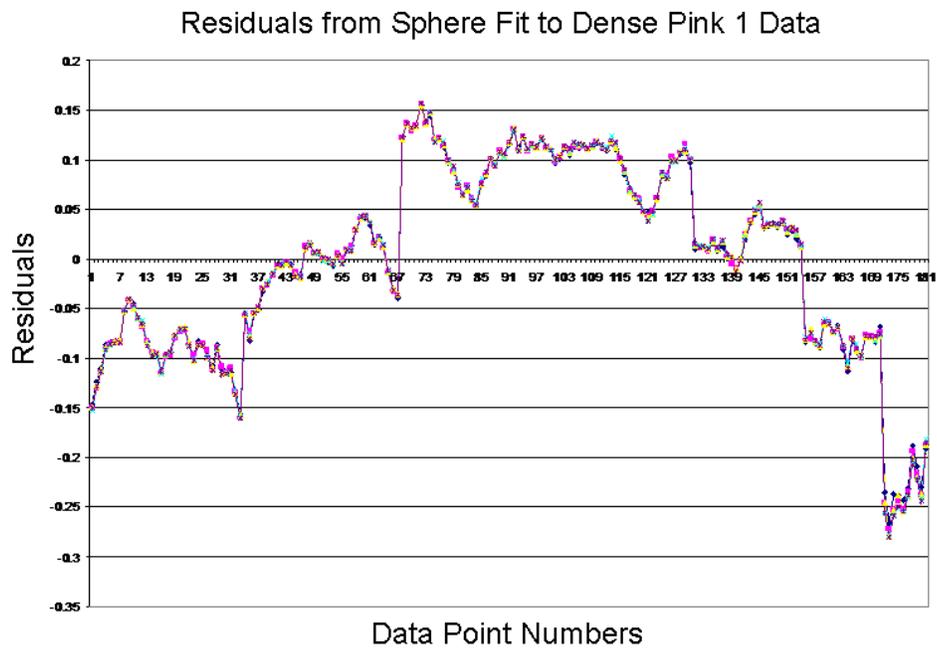


Figure 11: Residuals for the Sphere Fit to the Dense Pink Data Set 1.

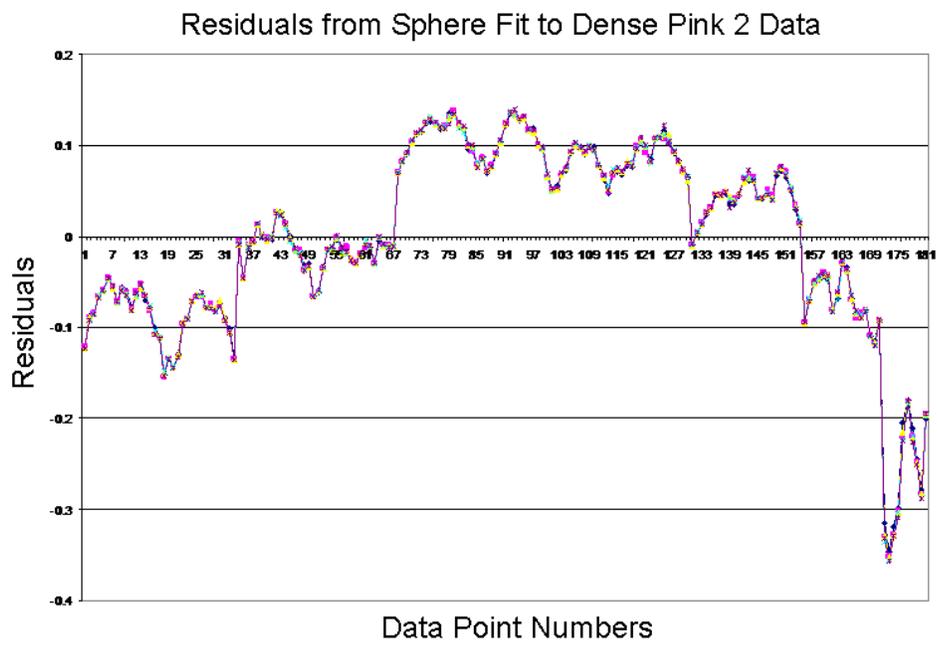


Figure 12: Residuals for the Sphere Fit to the Dense Pink Data Set 2.

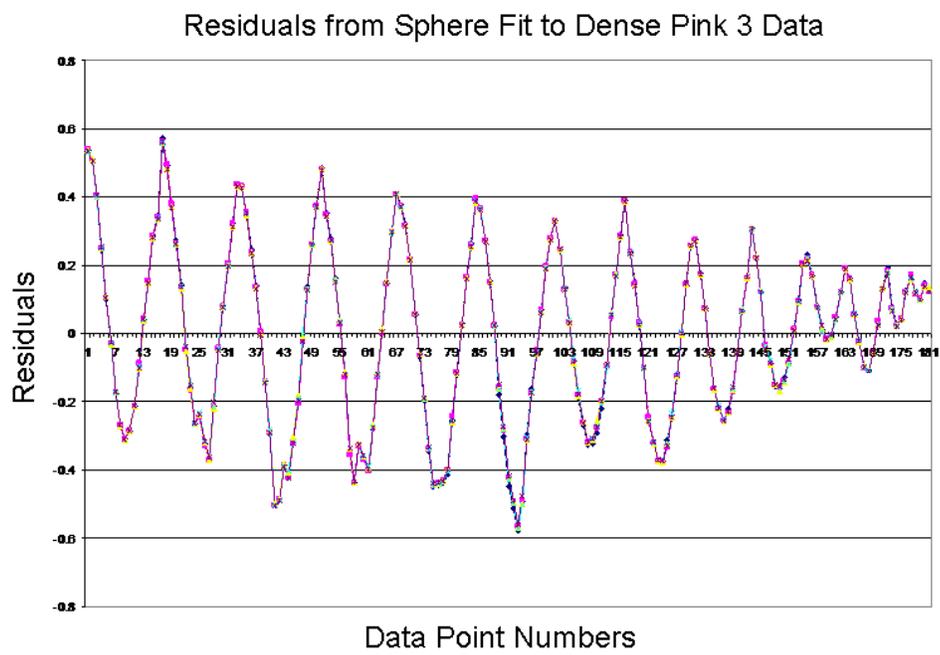


Figure 13: Residuals for the Sphere Fit to the Dense Pink Data Set 3.

spherical nature.

4.3.1 Euclidean to Spherical Coordinate Transformation

The current study involves the modeling of the sets of CMM data points. These unstructured sets of (x, y, z) data points are called point clouds. Therefore, in this section we assume that we are given a set of n points, (x_i, y_i, z_i) , $i = 1, 2, \dots, n$, for some n , on a surface. As measured, these points are given relative to the CMM origin. The first step is to center the data by using the center-of-data mass of the point cloud. This is done in order to establish a common reference point interior to the measured data points. It simplifies writing vectors from the origin to the data points and allows the introduction of spherical coordinates. The center-of-data point is given by

$$\begin{aligned}\bar{x} &= \frac{1}{n} \sum_{i=1}^n x_i \\ \bar{y} &= \frac{1}{n} \sum_{i=1}^n y_i \\ \bar{z} &= \frac{1}{n} \sum_{i=1}^n z_i\end{aligned}\tag{27}$$

Since the point cloud for each data set is enclosed in a near-spherical bounded region, it is reasonable to identify the Euclidean data points with spherical coordinates. This will map the points on the sphere to a rectangular surface where the surface coordinates are designated by θ , ϕ and the height of a surface point is given by $r(\theta, \phi)$. In order to use spherical coordinates to represent points on the boundary of a surface we need to restrict our analysis to surfaces that are called star-shaped. These are surfaces in which a ray drawn from the center-of-data mass intersects the boundary in a unique point. Whereas in the definition of the B-splines we used the coordinates (x, y) we will now use (θ, ϕ) and build B-splines in terms of these spherical coordinates. Euclidean coordinates will now refer to the measured data points. This is a switch in notation but should not, it is hoped, cause too much confusion.

To each point in the point cloud there is a vector from $(\bar{x}, \bar{y}, \bar{z})$ to (x_i, y_i, z_i) given by $V_i = (x_i, y_i, z_i) - (\bar{x}, \bar{y}, \bar{z})$. Furthermore, any point within the bounding sphere can be identified by spherical coordinates of the form $Q(r, \theta, \phi) = (r \sin \theta \cos \phi, r \sin \theta \sin \phi, r \cos \theta)$, where $x = r \sin \theta \cos \phi$, $y = r \sin \theta \sin \phi$, $z = r \cos \theta$, for $0 \leq \theta \leq \pi$, $0 \leq \phi \leq 2\pi$. θ is referred to as the colatitude and ϕ is referred to as the azimuth. Table 7 associates the three dimensional octants with their spherical coordinates, where we have suppressed r . The conversion to spherical coordinates was necessary since tensor product B-splines, which will be used to fit the surface, need to be defined on logically rectangular regions. Therefore, the Euclidean coordinates were converted to θ , ϕ angles on a rectangle $[0, \pi] \times [0, 2\pi]$. The height, $r(\theta, \phi)$, at each θ , ϕ was taken as the estimated radius from the center-of-data mass of all of the Euclidean coordinates.

Points, not necessarily on a spherical surface of radius r , will in general be a function of θ and ϕ . At this point we will start with an (x, y, z) point from the point cloud and compute the value of $(r(\theta, \phi), \theta, \phi)$ relative to the center-of-data mass. There are eight cases to consider. In all cases we assume that $r(\theta, \phi) > 0$. r will always be taken so that $r^2 = x^2 + y^2 + z^2$.

Before considering individual cases there are some general relations that apply to multiple cases. For example, in cases one through four in Table 7 we have $z \geq 0$ so that $0 \leq z \leq r$ and we will always have $0 \leq z/r \leq 1$. For MATLAB this implies that

$$0 \leq \theta = \arccos\left(\frac{z}{r}\right) \leq \frac{\pi}{2}.\tag{28}$$

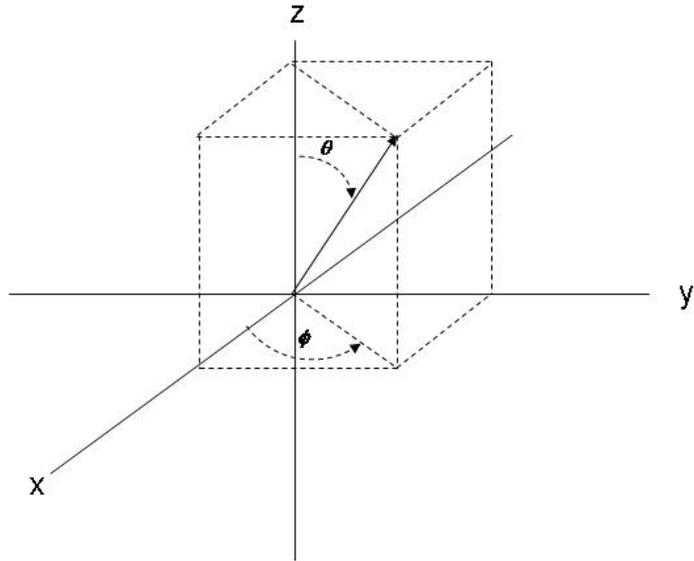


Figure 14: Spherical Coordinate angles. θ is the colatitude and ϕ is the azimuth.

3-D Octants to Spherical Coordinates		
Octant Number	Cartesian Coordinate	Spherical Coordinate
1	$x \geq 0, y \geq 0, z \geq 0$	$0 \leq \theta \leq \pi/2, 0 \leq \phi \leq \pi/2$
2	$x < 0, y \geq 0, z \geq 0$	$0 \leq \theta \leq \pi/2, \pi/2 \leq \phi \leq \pi$
3	$x < 0, y < 0, z \geq 0$	$0 \leq \theta \leq \pi/2, \pi \leq \phi \leq 3\pi/2$
4	$x \geq 0, y < 0, z \geq 0$	$0 \leq \theta \leq \pi/2, 3\pi/2 \leq \phi \leq 2\pi$
5	$x \geq 0, y \geq 0, z < 0$	$\pi/2 \leq \theta \leq \pi, 0 \leq \phi \leq \pi/2$
6	$x < 0, y \geq 0, z < 0$	$\pi/2 \leq \theta \leq \pi, \pi/2 \leq \phi \leq \pi$
7	$x < 0, y < 0, z < 0$	$\pi/2 \leq \theta \leq \pi, \pi \leq \phi \leq 3\pi/2$
8	$x \geq 0, y < 0, z < 0$	$\pi/2 \leq \theta \leq \pi, 3\pi/2 \leq \phi \leq 2\pi$

Table 7: Octant Equivalence between Euclidean and Spherical Coordinates

In cases five through eight in Table 7 $z < 0$ so that $-1 \leq z/r < 0$. Then

$$\frac{\pi}{2} \leq \theta = \arccos\left(\frac{z}{r}\right) \leq \pi. \quad (29)$$

There are also some special cases. In particular, if $x = 0, y = 0, z = r$, then we set $\theta = 0$, and, if $x = 0, y = 0, z = -r$, then we set $\theta = \pi$.

In Octant 1, $x \geq 0, y \geq 0, z \geq 0$. We assume $(0, 0, 0)$ will not be considered. We estimate $r = \sqrt{x^2 + y^2 + z^2}$. We have θ from (28), so now we want to compute the associated ϕ . For $\theta \in (0, \pi/2)$ we have $0 < x/(r \sin \theta) < 1$. Then

$$0 < \phi = \arccos\left(\frac{x}{r \sin \theta}\right) < \frac{\pi}{2}. \quad (30)$$

In Octant 2, $x < 0, y \geq 0, z \geq 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and θ from (28). Since $0 \leq \theta < \pi/2$, then $-1 < x/(r \sin \theta) < 0$. Therefore

$$\frac{\pi}{2} < \phi = \arccos\left(\frac{x}{r \sin \theta}\right) < \pi, \quad (31)$$

the desired interval.

In Octant 3, $x < 0, y < 0, z \geq 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and θ from (28). We want $\pi \leq \phi \leq 3\pi/2$. Since $-1 < x/(r \sin \theta) < 0$ then $\pi/2 < \phi_1 = \arccos(x/(r \sin \theta)) < \pi$ and

$$\pi < \phi = \phi_1 + \frac{\pi}{2} < \frac{3\pi}{2}. \quad (32)$$

In Octant 4, $x \geq 0, y < 0, z \geq 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and θ from (28). We want $3\pi/2 \leq \phi \leq 2\pi$. Since $0 < x/(r \sin \theta) < 1$, then $0 < \phi_1 = \arccos(x/(r \sin \theta)) < \pi/2$

$$\frac{3\pi}{2} \leq \phi = \phi_1 + \frac{3\pi}{2} \leq 2\pi. \quad (33)$$

In Octant 5, $x \geq 0, y \geq 0, z < 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and solve for θ from (28). Since $0 < x/(r \sin \theta) < 1$,

$$0 < \phi = \arccos\left(\frac{x}{r \sin \theta}\right) \leq \frac{\pi}{2}. \quad (34)$$

In Octant 6, $x < 0, y \geq 0, z < 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and solve for θ from (28). We want $\pi/2 \leq \phi \leq \pi$. Since $-1 < x/(r \sin \theta) < 0$

$$\frac{\pi}{2} < \phi = \arccos\left(\frac{x}{r \sin \theta}\right) \leq \pi. \quad (35)$$

In Octant 7, $x < 0, y < 0, z < 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and solve for θ from (28). We want $\pi \leq \phi \leq 3\pi/2$, but $\pi/2 \leq \phi_1 = \arccos(x/(r \sin \theta)) \leq \pi$ so we set

$$\pi \leq \phi = \phi_1 + \frac{\pi}{2} \leq \pi. \quad (36)$$

In Octant 8, $x \geq 0, y < 0, z < 0$. Set $r = \sqrt{x^2 + y^2 + z^2}$ and solve for θ from (28). We want $3\pi/2 \leq \phi \leq 2\pi$, but $0 \leq \phi_1 = \arccos(x/(r \sin \theta)) \leq \pi/2$ so we set

$$\frac{3\pi}{2} < \phi = \phi_1 + \frac{3\pi}{2} < 2\pi. \quad (37)$$

4.3.2 Transformation Algorithm

The algorithm is relatively straightforward.

1. Load the measured points (x_i, y_i, z_i) into three arrays.
2. Compute length of the arrays and set n . This assumes the arrays are of equal length.
3. Compute the center of data mass

$$\begin{aligned}\bar{x} &= \frac{1}{n} \sum_{i=1}^n x_i, \\ \bar{y} &= \frac{1}{n} \sum_{i=1}^n y_i, \\ \bar{z} &= \frac{1}{n} \sum_{i=1}^n z_i.\end{aligned}\tag{38}$$

4. Reset the (x_i, y_i, z_i) points as

$$\begin{aligned}x_i &= x_i - \bar{x}, \\ y_i &= y_i - \bar{y}, \\ z_i &= z_i - \bar{z}.\end{aligned}\tag{39}$$

5. Begin loop from $i = 1$ to n

5.1. Compute $r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$.

5.2. Test for the special cases.

5.2.1. If $x_i = 0, y_i = 0, z_i = r_i$, set $\theta_i = 0, \phi_i = 0$.

5.2.1. If $x_i = 0, y_i = 0, z_i = -r_i$, set $\theta_i = \pi, \phi_i = 0$.

5.3. Test for the Octants

5.3.1. If $x_i \geq 0, y_i \geq 0, z_i \geq 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{40}$$

5.3.2. If $x_i < 0, y_i \geq 0, z_i \geq 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{41}$$

5.3.3. If $x_i < 0, y_i < 0, z_i \geq 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \frac{\pi}{2} + \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{42}$$

5.3.4. If $x_i \geq 0, y_i < 0, z_i \geq 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \frac{3\pi}{2} + \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{43}$$

5.3.5. If $x_i \geq 0, y_i \geq 0, z_i < 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{44}$$

5.3.6. If $x_i < 0, y_i \geq 0, z_i < 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{45}$$

5.3.7. If $x_i < 0, y_i < 0, z_i < 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \frac{\pi}{2} + \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{46}$$

5.3.8. If $x_i \geq 0, y_i < 0, z_i < 0$

$$\begin{aligned}\theta_i &= \arccos\left(\frac{z_i}{r_i}\right), \\ \phi_i &= \frac{3\pi}{2} + \arccos\left(\frac{x_i}{r_i \sin \theta_i}\right).\end{aligned}\tag{47}$$

6. End loop on i.

This algorithm is implemented in the MATLAB function program in Appendix C.2. The results of the conversion of the probe points for the Green phantom from Euclidean coordinates to θ, ϕ coordinates are shown in Fig. 15. We note the density of data points near the equator is higher than towards the poles at $\theta = 0$ and $\theta = \pi$. Unfortunately the distribution of data points is dictated by the software controlling the CMM. This lack of data points near the poles leads to a well known problem, called the Pole Problem in the literature. It leads to rank deficient matrices during the least squares fitting process and was discussed in Section C.5.

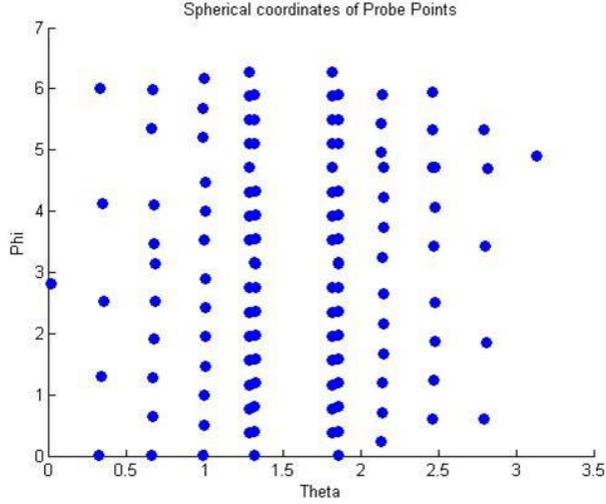


Figure 15: Plot of the θ , ϕ Coordinates of the Probe Points for the Coarse Data.

4.4 Volume Estimation by the Divergence Theorem

In this section we will state the Divergence Theorem and indicate how it can be used to estimate the volume of a polyhedron.

4.4.1 Divergence Theorem in 3-D and Volume Computation

Let R be a simply connected region in space and S the surface boundary. Then

$$\int \int \int_R \vec{\nabla} \cdot F \, dx dy dz = \int \int_S F \cdot \hat{n} \, d\sigma, \quad (48)$$

where $F(x, y, z) = (F_1(x, y, z), F_2(x, y, z), F_3(x, y, z))$ is a differentiable vector field,

$$\vec{\nabla} \cdot F = \frac{\partial F_1}{\partial x} + \frac{\partial F_2}{\partial y} + \frac{\partial F_3}{\partial z}, \quad (49)$$

is the divergence of the vector field, \hat{n} is an outward-pointing unit normal vector to S , and $d\sigma$ is an infinitesimal element of the surface.

If we select $F(x, y, z) = (1/3)(x, y, z)$ then $\vec{\nabla} \cdot F = 1$ and we can write

$$\int \int \int_R dx dy dz = \frac{1}{3} \int \int_S \hat{n} \cdot (x, y, z) d\sigma. \quad (50)$$

Note that the left hand side is simply the volume of the region R . The $\frac{1}{3}$ factor comes from the definition of $F(x, y, z)$ so that $\vec{\nabla} \cdot F = 1$. Now, if S is approximated by disjoint polyhedra (linear surface patches), S_i , then

$$S = \bigcup_{i=1}^N S_i, \quad (51)$$

and

$$\frac{1}{3} \int_S \hat{n} \cdot (x, y, z) d\sigma = \frac{1}{3} \sum_{i=1}^N \int_{S_i} \hat{n}_i \cdot (x, y, z) d\sigma, \quad (52)$$

where \hat{n}_i is the unit outer normal to S_i . Here we will model the surface patches by planar facets and, in particular, triangular facets. The plane S_i is given by $\hat{n}_i \cdot (x, y, z) = c_i$ for a constant c_i associated with each triangular facet. Since this constant applies for all points on the triangular facet, it applies for any vertex point P_i of the triangular facet. The sum of the integrals over the facets thus reduces to

$$\frac{1}{3} \sum_{i=1}^N \int_{S_i} \hat{n}_i \cdot (x, y, z) d\sigma = \frac{1}{3} \sum_{i=1}^N \int_{S_i} c_i d\sigma = \frac{1}{3} \sum_{i=1}^N c_i \text{Area}(S_i). \quad (53)$$

This method of computing the volume of an object from a surface integral can be found in Schneider and Eberly [15].

4.4.2 Area of a Planar Polyhedron in 3-D

We will describe the process of computing the area of a planar polyhedron in space by use of Stokes' Theorem and then we will particularize it to a planar triangle in space. Stokes' Theorem assumes that if C is a piecewise smooth boundary curve, oriented positively, of a surface S , and if F is a differentiable vector field defined on S and \hat{n} is a unit normal satisfying the right-hand rule relative to the boundary orientation, then

$$\int_S \int (\vec{\nabla} \times F) \cdot \hat{n} d\sigma = \int_C F \cdot d\vec{R}, \quad (54)$$

where the curl of $F(x, y, z) = (F_1(x, y, z), F_2(x, y, z), F_3(x, y, z))$ is

$$\vec{\nabla} \times F = \left(\frac{\partial F_3}{\partial y} - \frac{\partial F_2}{\partial z}, \frac{\partial F_1}{\partial z} - \frac{\partial F_3}{\partial x}, \frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} \right), \quad (55)$$

and $d\vec{R} = (dx, dy, dz)$. If we set $F(x, y, z) = (1/2)\hat{n} \times (x, y, z)$ then $\vec{\nabla} \times F = \hat{n}$ and $\hat{n} \cdot \hat{n} = 1$ so that

$$\begin{aligned} \text{Area}(S) &= \int_S \int d\sigma = \frac{1}{2} \oint_C (\hat{n} \times (x, y, z)) \cdot (dx, dy, dz), \\ &= \frac{1}{2} \oint_C \hat{n} \cdot ((x, y, z) \times (dx, dy, dz)), \\ &= \frac{1}{2} \oint_C \hat{n} \cdot ((x(t), y(t), z(t)) \times (x'(t), y'(t), z'(t))) dt, \\ &= \frac{1}{2} \oint_C \hat{n} \cdot (y(t)z'(t) - z(t)y'(t), z(t)x'(t) - x(t)z'(t), x(t)y'(t) - y(t)x'(t)) dt. \end{aligned} \quad (56)$$

The $\frac{1}{2}$ comes from the definition of $F(x, y, z)$ so that $\vec{\nabla} \times F = \hat{n}$. We will now specialize the Stokes formula to find the area of a spatial triangle in terms of its three vertices oriented positively. Let the three vertices, in positive orientation, be specified by $v_1 = (x_1, y_1, z_1)$, $v_2 = (x_2, y_2, z_2)$, $v_3 =$

(x_3, y_3, z_3) . Parameterize the boundary of the triangle as follows. On $[0, 1]$ let $v = v_1 + t(v_2 - v_3) = (x_1 + t(x_2 - x_1), y_1 + t(y_2 - y_1), z_1 + t(z_2 - z_1))$. On $[1, 2]$ let $v = v_2 + (t - 1)(v_3 - v_2) = (x_2 + (t - 1)(x_3 - x_2), y_2 + (t - 1)(y_3 - y_2), z_1 + (t - 1)(z_3 - z_2))$. Finally on $[2, 3]$ let $v = v_3 + (t - 2)(v_1 - v_3) = (x_3 + (t - 2)(x_1 - x_3), y_3 + (t - 2)(y_1 - y_3), z_3 + (t - 2)(z_1 - z_3))$. The area of the spatial triangle, in terms of the parameterized boundary, can be written as

$$\begin{aligned} \text{Area}(S) &= \frac{1}{2} \oint_C \hat{n} \cdot (y(t)z'(t) - z(t)y'(t), z(t)x'(t) - x(t)z'(t), x(t)y'(t) - y(t)x'(t)) dt, \\ &= \frac{1}{2} \hat{n} \cdot \left(\int_0^3 (y(t)z'(t) - z(t)y'(t)) dt, \int_0^3 (z(t)x'(t) - x(t)z'(t)) dt, \right. \\ &\quad \left. \int_0^3 (x(t)y'(t) - y(t)x'(t)) dt \right). \end{aligned} \quad (57)$$

By straightforward integration over each of the parameterized segments it is easy to show that

$$\begin{aligned} \text{Area}(S) &= \frac{1}{2} \hat{n} \cdot ((y_1 z_2 - y_2 z_1) + (y_2 z_3 - y_3 z_2) + (y_3 z_1 - y_1 z_3), \\ &\quad (z_1 x_2 - z_2 x_1) + (z_2 x_3 - z_3 x_2) + (z_3 x_1 - z_2 x_3), \\ &\quad (x_1 y_2 - x_2 y_1) + x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3) \end{aligned} \quad (58)$$

The normal vector to the oriented triangle can be computed as follows. Let $\vec{v} = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$ and $\vec{w} = (x_3 - x_1, y_3 - y_1, z_3 - z_1)$. Then $\vec{n} = \vec{v} \times \vec{w} = ((y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1), (z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_1), (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1))$. If we set $n_1 = (y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1)$, $n_2 = (z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_1)$, $n_3 = (x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)$ then $\|\vec{n}\| = \sqrt{n_1^2 + n_2^2 + n_3^2}$ and $\hat{n} = \vec{n}/\|\vec{n}\|$.

4.5 Volume Estimation for the Phantom Grid Data

We divided the phantom surfaces into triangular surface patches as follows. First of all we partitioned the phantom surfaces at grid points located at the colatitude angles $\theta_1 = 0 < \theta_2 < \dots < \theta_v < \theta_{v+1} = \pi$ from the north pole to the south pole and azimuthal angles $\phi_1 = 0 < \phi_2 < \dots < \phi_h < \phi_{h+1} = 2\pi$ around the phantom surface. Since these grid points did not necessarily fall at the measured data points, a radius, $r(\phi, \theta)$, was calculated at the grid points from the surface B-spline model. At the north and south poles the radii was taken as the median value, r_n , of the grid point values $r(\phi_i, 0)$, $i = 1, \dots, h$, for the north pole and r_s , the median value of $r(\phi_i, \pi)$, $i = 1, \dots, h$. The spherical coordinates of all of the grid points were converted to Euclidean coordinates on the surface by

$$\begin{aligned} x &= r \sin(\theta) \cos(\phi) \quad 0 \leq \theta \leq \pi, \\ y &= r \sin(\theta) \sin(\phi) \quad 0 \leq \phi \leq 2\pi, \\ z &= r \cos(\theta). \end{aligned} \quad (59)$$

For triangular patches at the north poles were easily constructed to be triangular as part of the process of determining the contribution of each patch to the phantom volumes. In particular, at the north pole we designated the point as $(x_1, y_1, z_1) = (0, 0, r_n)$. We then iterated through the spherical coordinate points (ϕ_i, θ_2) , $i = 1, \dots, h$. At each of these angle pairs there was a value $r(\phi_i, \theta_2)$, $i = 1, \dots, h$. We generated the volume by the Divergence Theorem algorithms described in Sections 4.4.1 and 4.4.2 by adding up the contributions of each patch to the volume total. We

Sample Surface Triangulation

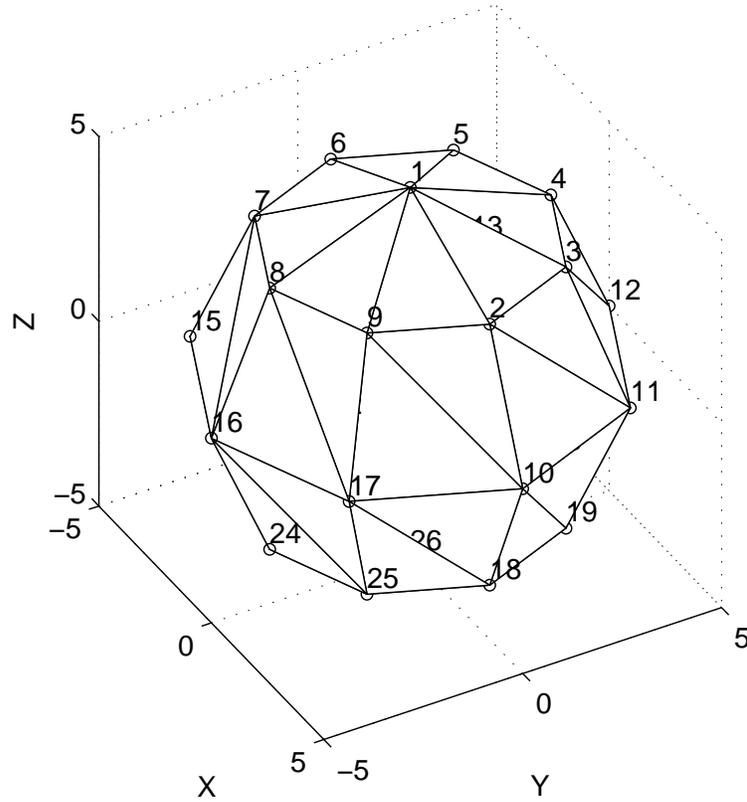


Figure 16: A Sample Triangulation of a Sphere Viewed from the North Pole.

did this by initializing a variable, vol , for the volume to zero. We then started to generate the contribution from the first layer of patches at the north pole. As noted above, we set the north pole to (x_1, y_1, z_1) and then set

$$\begin{aligned} x_2 &= r(\phi_1, \theta_2) \sin(\theta_2) \cos(\phi_1), \\ y_2 &= r(\phi_1, \theta_2) \sin(\theta_2) \sin(\phi_1), \\ z_2 &= r(\phi_1, \theta_2) \cos(\theta_2), \end{aligned}$$

and

$$\begin{aligned} x_3 &= r(\phi_2, \theta_2) \sin(\theta_2) \cos(\phi_2), \\ y_3 &= r(\phi_2, \theta_2) \sin(\theta_2) \sin(\phi_2), \\ z_3 &= r(\phi_2, \theta_2) \cos(\theta_2). \end{aligned}$$

These were set in a positive orientation. In order to make this process more concrete we have included a sample surface triangulation with $v = 5$ colatitude angles and $h = 8$ azimuthal angles

in Fig. 16. The MATLAB program that generated the figure is given in Appendix C.8. The Euclidean grid points have been indexed. We note that the plotting software has allowed some of the indices on the reverse side to partially show through, although this should not cause trouble in the explanations below. The North Pole (x_1, y_1, z_1) is designated by the index 1. In the first step described above the points (x_2, y_2, z_2) and (x_3, y_3, z_3) are indexed in Fig. 16 by points 2 and 3 respectively. Next we computed the outward normal to the triangle patch formed the vectors $v_1 = (x_2, y_2, z_2) - (x_1, y_1, z_1)$, $v_2 = (x_3, y_3, z_3) - (x_1, y_1, z_1)$, for triangle 123 in Fig. 16 and then formed the normalized cross product

$$\hat{n} = \frac{v_1 \times v_2}{\|v_1 \times v_2\|} \quad (60)$$

We then computed the contribution that this patch made to the volume as

$$\text{vol} = \text{vol} + (\hat{n} \cdot (x_1, y_1, z_1)) \left\{ \hat{n} \cdot \left(\sum_{j=1}^2 (y_j z_{j+1} - y_{j+1} z_j), (z_j x_{j+1} - z_{j+1} x_j), (x_j y_{j+1} - x_{j+1} y_j) \right) \right\}. \quad (61)$$

We proceeded to the next patch in the North Pole layer. Again (x_1, y_1, z_1) was the North Pole, indexed by 1 in Fig. 16, and we then used the previous computation to get $x_2 = x_3$, $y_2 = y_3$, $z_2 = z_3$ and set

$$\begin{aligned} x_3 &= r(\phi_3, \theta_2) \sin(\theta_2) \cos(\phi_3), \\ y_3 &= r(\phi_3, \theta_2) \sin(\theta_2) \sin(\phi_3), \\ z_3 &= r(\phi_3, \theta_2) \cos(\theta_2). \end{aligned}$$

In Fig. 16 the new point (x_3, y_3, z_3) is indexed by 4 in Fig. 16. The triangle of interest is now 134 in terms of indices. We computed the normalized cross product as for the first patch and then computed the contribution of the second patch to the volume using equation (61). We continued this process for $\theta_2, \phi_i, i = 1, \dots, h$. In Fig. 16 we would have proceeded with computing contributions to the volume by working through the indexed triangles 123, 134, 145, 156, 167, 178, 189, and 192.

We next computed the contributions of the middle layer patches in a two step process. We iterated through each $\theta_j, j = 2, \dots, v-1$. The triangles at the South Pole were handled separately. For each $\theta_j, j = 2, \dots, v-1$ and $\phi_i, i = 1, \dots, h$ the patches were defined first in terms of four vertices to create four sided patches. Each of these patches was then divided into two triangles. The four vertices of a rectangular patch were identified counterclockwise as $(\theta_j, \phi_i, r(i, j)), (\theta_{j+1}, \phi_i, r(i, j+1)), (\theta_{j+1}, \phi_{i+1}, r(i+1, j+1)), (\theta_j, \phi_{i+1}, r(i+1, j))$. As an example, in Fig. 16 one of the patches is identified by indices, in counterclockwise order, as 2 10 11 3. These indices would be associated with vertices $(\theta_2, \phi_1, r(1, 2)), (\theta_3, \phi_1, r(1, 3)), (\theta_3, \phi_2, r(2, 3)), (\theta_2, \phi_2, r(2, 2))$. The four vertex patches were then divided into two triangles. For the first triangle in the rectangular patch we set

$$\begin{aligned} x_1 &= r(i, j) \sin(\theta_j) \cos(\phi_i), \\ y_1 &= r(i, j) \sin(\theta_j) \sin(\phi_i), \\ z_1 &= r(i, j) \cos(\theta_j). \end{aligned}$$

$$\begin{aligned} x_2 &= r(i, j+1) \sin(\theta_{j+1}) \cos(\phi_i), \\ y_2 &= r(i, j+1) \sin(\theta_{j+1}) \sin(\phi_i), \\ z_2 &= r(i, j+1) \cos(\theta_{j+1}). \end{aligned}$$

$$\begin{aligned}
x_3 &= r(i+1, j+1) \sin(\theta_{j+1}) \cos(\phi_{i+1}), \\
y_3 &= r(i+1, j+1) \sin(\theta_{j+1}) \sin(\phi_{i+1}), \\
z_3 &= r(i+1, j+1) \cos(\theta_{j+1}).
\end{aligned}$$

This triangle in the example Fig. 16 would be 2 10 11. Again, the volume increment was computed as discussed previously. For the second triangle of the rectangular patch we maintained the same (x_1, y_1, z_1) and set $x_2 = x_3$, $y_2 = y_3$, $z_2 = z_3$ and then set

$$\begin{aligned}
x_3 &= r(i+1, j) \sin(\theta_j) \cos(\phi_{i+1}), \\
y_3 &= r(i+1, j) \sin(\theta_j) \sin(\phi_{i+1}), \\
z_3 &= r(i+1, j) \cos(\theta_j).
\end{aligned}$$

This triangle in the example Fig. 16 would be 2 11 3. Again the volume increment was computed as before. We continued the process for θ_j , $j = 2, \dots, v-1$ and ϕ_i , $i = 1, \dots, h$.

Finally, at the South Pole there were h triangles to include in the volume calculation. Their vertices were identified as follows.

$$\begin{aligned}
x_1 &= r(i, v) \sin(\theta_v) \cos(\phi_i), \\
y_1 &= r(i, v) \sin(\theta_v) \sin(\phi_i), \\
z_1 &= r(i, v) \cos(\theta_v).
\end{aligned}$$

$$\begin{aligned}
x_2 &= 0, \\
y_2 &= 0, \\
z_2 &= -r_s.
\end{aligned}$$

$$\begin{aligned}
x_3 &= r(i+1, v) \sin(\theta_v) \cos(\phi_{i+1}), \\
y_3 &= r(i+1, v) \sin(\theta_v) \sin(\phi_{i+1}), \\
z_3 &= r(i+1, v) \cos(\theta_v).
\end{aligned}$$

As an example in Fig. 17 one of these triangles would be indexed by the points 18 26 19, where the South Pole is point number 26. The contribution of these triangles to the volume was computed as above. After all of the triangle contributions to the volume were computed the final volume was then computed as $\text{vol} = (1/6)\text{vol}$. The factor 1/6 comes from the product of 1/3 from Section 4.4.1 and 1/2 from Section 4.4.2. The reader can refer back to these sections to see how the factors arose.

4.6 Computational Results for the B-spline Model

In this section we will describe the method used to estimate volume uncertainties and discuss the results of using a B-spline surface model and the Divergence Theorem to estimate the phantom volumes. A nonparametric method to estimate the volume uncertainties was chosen since there did not exist any "ground truth" values for the Green and Pink FDA phantom volumes.

All of the results in this section will be displayed in terms of a table and two graphs for each data set. There are fourteen data sets used for volume and volume uncertainty estimates. Two data sets are for the CMM data acquired on the calibrated steel spheres, one for the coarse (121) data distribution and one for the dense (181) data distribution. There are data sets for the measurements of the Green phantoms in three positions (North Pole up, South Pole up, prime meridian/equator

Sample Surface Triangulation

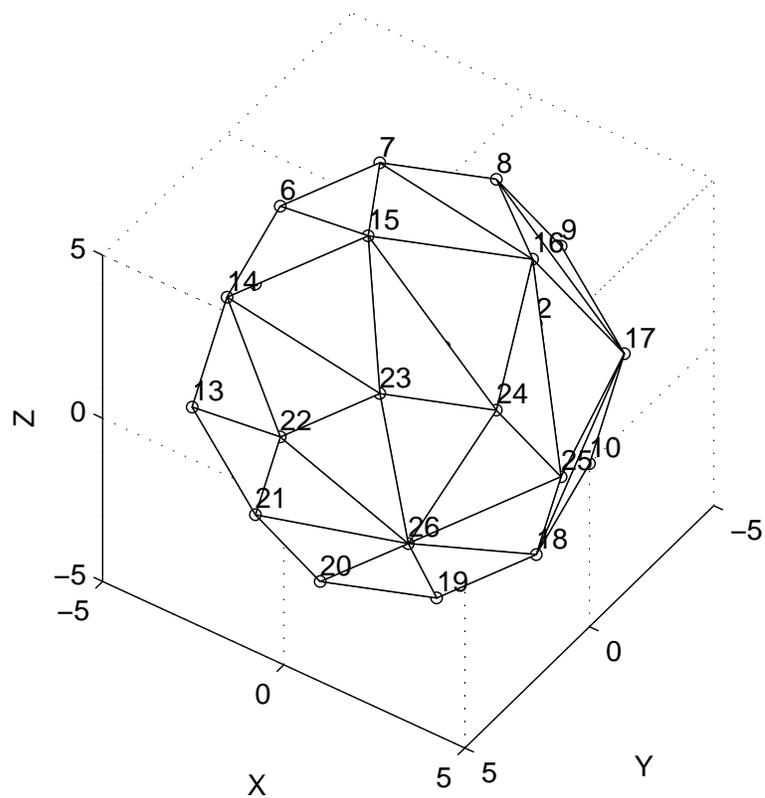


Figure 17: A Sample Triangulation of a Sphere Viewed from the South Pole.

up) for both the coarse and dense data distributions. Similarly for the Pink phantom, recalling that for the coarse data the prime meridian/equator position was not possible and a re-measure of the North Pole up position was done.

For each data set we conducted sixteen estimates of the volumes and volume uncertainties by increasing the density of the surface triangulation. The sixteen were selected since beyond that point the software used began to run into memory problems. In order to plot the results in terms of grid density we use the term grid size by which we mean the product of the number of θ values times the number of ϕ values for a particular grid density. As shown in Tables 8 through 21 we begin with a grid of ten θ and twenty ϕ values and compute the volume and uncertainty. The grid size in this 10×20 grid is then 200. We then increment the θ values by ten and the ϕ values by twenty for each grid case until the last case of 160 θ values and 320 ϕ values, giving a grid size of 51200 (5.12×10^4). The volume Figures 18 through 44 show graphically how the volume estimates in all cases rise rapidly and appear to approach a fixed value as the grid size increases. They simply reflect the volume data in the Tables 8 through 21. The uncertainty Figures 19 through 45 show graphically how the volumetric uncertainties decrease and appear to stabilize as the grid size increases. Again, these graphically represent the uncertainties in Tables 8 through 21.

4.6.1 Estimating Volume Uncertainties

The nonparametric "bootstrap" method is a computer intensive technique for estimating uncertainties. It involves repeated Monte Carlo resampling from the spherical coordinates of the original measured data sets with radii values modified by the fitting residuals, refitting the model to estimate new volumes, and finally computing an uncertainty for the process from the set of computed volumes. For a full discussion of the bootstrap see Efron and Tibshirani [9] but we will give a brief description here of how we applied the bootstrap method in the current study.

The object of our application of the bootstrap was to develop volume uncertainty as a function of grid size. The process began with the conversion of the Euclidean data points to spherical coordinates. An automatic selection of knots was done. These steps were done once for a given data set. It was assumed that the modifications of the data made during the bootstrap would be small and not affect the knot selection. During the first pass of the bootstrap algorithm the radii of the spherical coordinates were fit by a tensor product of B-splines. The predicted values and residuals from this initial fit were called the master predicted values and master residuals respectively. The computed volume was then put in a list of volumes that would be added to in subsequent passes of the algorithm. The algorithm then iterated two hundred times as follows. In the first iteration the master residuals were sampled randomly uniformly with replacement. The resampled residuals were then added to the master predicted radii and a new fit was performed. The new computed volume was added to the volume list and the master residuals sampled randomly uniformly with replacement for the next iteration. The process continued for all two hundred iterations. The standard deviation of the volumes in the volume list was then used as an estimate of the volume uncertainty and the average volume was taken as the reference volume for the chosen grid size.

4.6.2 Calibrated Sphere Volume Estimates and Uncertainties for the Coarse and Dense Data

If we compare the volumes and radii computed by the B-spline method in Tables 8 and 9 with the sphere fit results in Table 2, we see that the B-spline method underestimates the volume in both the coarse and dense date experiments with the calibrated sphere by about 10 %. From Tables 8 and 9 the radii estimates differ from the radii estimates in Table 2 by about 4 %. This just points out how small differences in linear measurements can produce large differences in volume estimates. Whether

Calibrate Sphere Coarse Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3100.1	94.9
2	20	40	3216.9	54.5
3	30	60	3240.6	43.1
4	40	80	3247.1	39.5
5	50	100	3249.3	36.5
6	60	120	3251.2	36.5
7	70	140	3254.3	34.9
8	80	160	3254.2	35.5
9	90	180	3255.4	34.8
10	100	200	3254.9	35.4
11	110	220	3255.2	34.5
12	120	240	3254.4	34.3
13	130	260	3254.8	34.1
14	140	280	3255.2	34.8
15	150	300	3254.8	34.0
16	160	320	3255.6	34.4

Table 8: Estimated Volumes and Uncertainties for Calibrated Sphere 121 Data

these percentage differences are due to data point distribution or algorithm weakness is beyond the scope of this report to examine and will be addressed in a separate study. These differences will no doubt affect the volume estimates for the phantoms as will be discussed in the next sections. We do expect in all cases that there will be algorithmic difficulties due to the lack of data points near the poles. Since this study depended on the CMM produced data points we accept and deal with the data as produced by the CMM.

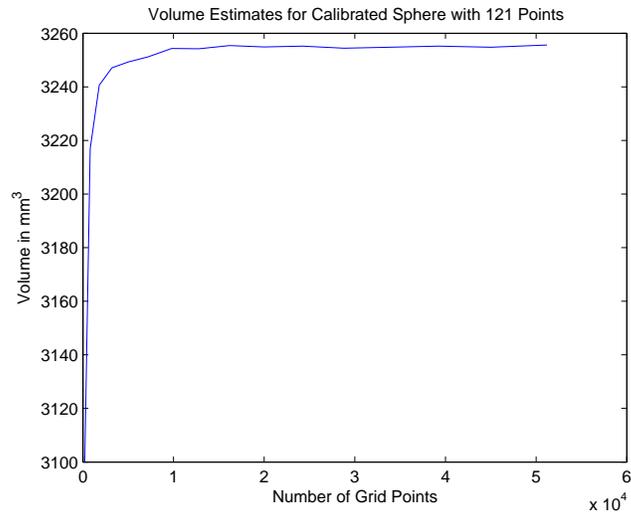


Figure 18: Volume Estimates for 121 Calibrated Sphere Data Set vs. Grid Size.

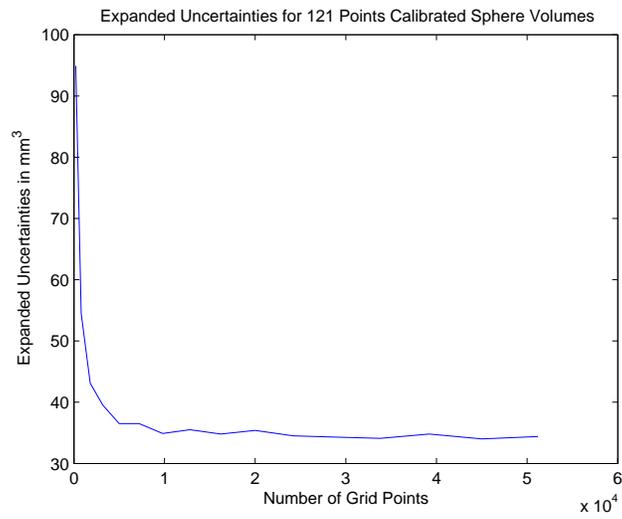


Figure 19: Uncertainty Estimates for 121 Calibrated Sphere Data Set vs. Grid Size.

Calibrated Sphere Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3102.4	99.2
2	20	40	3212.5	58.7
3	30	60	3235.0	46.3
4	40	80	3242.3	43.1
5	50	100	3246.1	40.9
6	60	120	3248.1	40.1
7	70	140	3249.5	38.0
8	80	160	3247.7	39.4
9	90	180	3249.3	38.2
10	100	200	3250.7	38.0
11	110	220	3250.1	38.0
12	120	240	3250.3	38.6
13	130	260	3250.8	38.4
14	140	280	3251.2	37.5
15	150	300	3250.6	38.2
16	160	320	3251.2	37.5

Table 9: Estimated Volumes and Uncertainties for Calibrated Sphere 181 Data

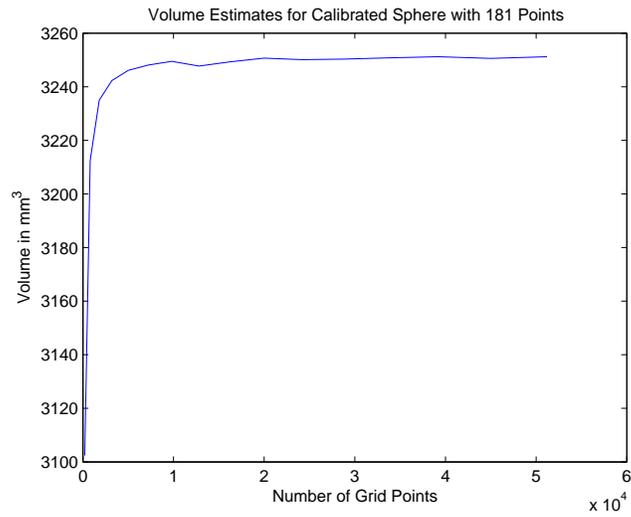


Figure 20: Volume Estimates for 181 Calibrated Sphere Data Set vs. Grid Size.

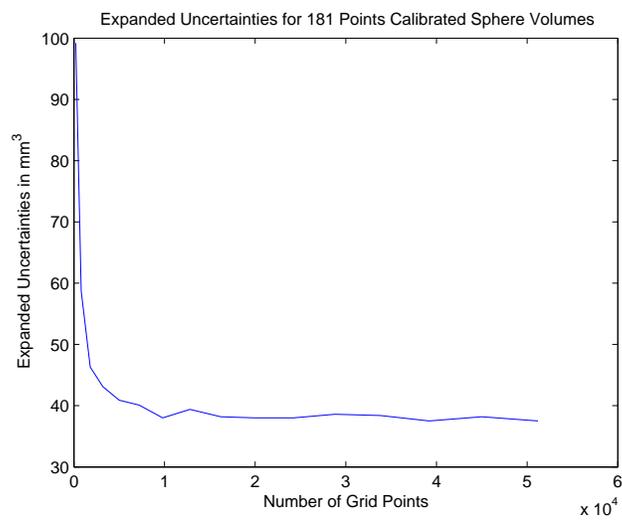


Figure 21: Uncertainty Estimates for 181 Calibrated Sphere Data Set vs. Grid Size.

Green 4 Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	4160.2	6.0
2	20	40	4320.5	4.0
3	30	60	4347.5	2.9
4	40	80	4357.0	2.8
5	50	100	4361.3	2.4
6	60	120	4363.6	2.4
7	70	140	4365.1	2.3
8	80	160	4365.9	2.2
9	90	180	4366.5	2.3
10	100	200	4366.9	2.2
11	110	220	4367.3	2.2
12	120	240	4367.5	2.2
13	130	260	4367.7	2.2
14	140	280	4367.8	2.2
15	150	300	4368.0	2.2
16	160	320	4368.1	2.2

Table 10: Estimated Volumes and Uncertainties for Green 4 Data

4.6.3 Phantom Volume Estimates and Uncertainties for the Coarse Data

In this section we discuss briefly the results of computing the volumes and volume uncertainties for the two phantoms using the six coarse (121) data sets. As described earlier, sixteen surface grid cases were used and the results are shown in Tables 10 through 15. Figures 22 through 33 show the trends of the volume estimates and uncertainty estimates as the grid sizes increase. They both tend to what appears to be stabilized values. The figures simply display the values in the tables.

The spherical fit to the Green phantom data in Table 3 indicates an average value of 4331.3 mm³. Tables 10 through 12 indicate an average value of 4367.8 mm³ for the largest grid size of 160 θ by 320 ϕ . This is approximately a 1% difference suggesting that the Green phantom is very nearly spherical. However, Table 1 indicates a diameter of approximately 20 mm. The volume then would be 4188.8 mm³ indicating that the B-spline method might be producing a 4% larger value. This is interesting, considering the B-spline method produced about a 10% smaller value to the calibrated sphere for the coarse data points.

The spherical fit to the Pink phantom data in Table 4 indicates an average volume of 3862.6 mm³. Tables 13 through 15 indicate an average volume of 3947.4 mm³ for the largest grid size. The difference is approximately 2%, suggesting the slight non-spherical shape of the Pink phantom. Table 1 indicates an average diameter of 19.2 mm or a volume of 3706 mm³. This is about a 4% smaller volume than the average spherical fit volume or about a 6% smaller volume than the B-spline volume estimate.

From Tables 10 to 12 the uncertainties range from 6.7 mm³ on the coarse surface grid to 2.2 mm³ on the finer grid for the Green phantom. Tables 13 through 15 show a range of uncertainties from 24 mm³ to 2.9 mm³. This wider range might be due to the non-spherical nature of the Pink phantom.

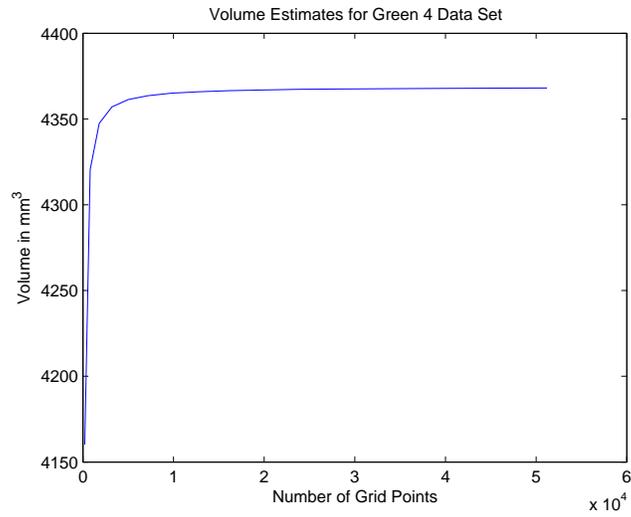


Figure 22: Volume Estimates for Green 4 Data Set vs. Grid Size.

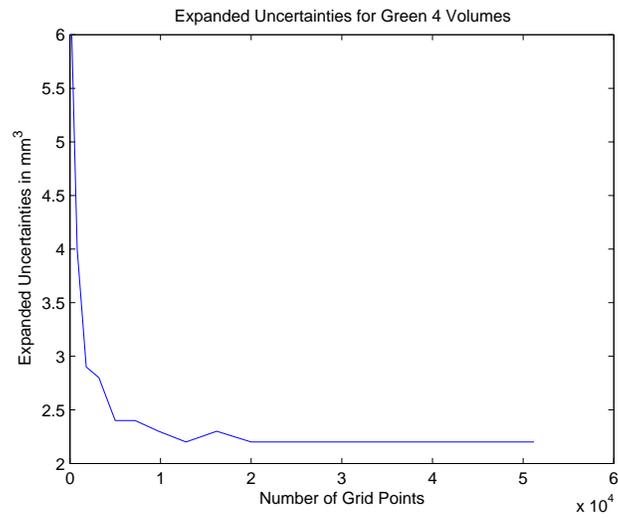


Figure 23: Uncertainty Estimates for Green 4 Data Set vs. Grid Size.

Green 5 Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	4159.7	6.3
2	20	40	4319.8	3.9
3	30	60	4347.3	3.2
4	40	80	4356.6	2.8
5	50	100	4360.9	2.6
6	60	120	4363.3	2.6
7	70	140	4364.6	2.5
8	80	160	4365.6	2.5
9	90	180	4366.1	2.4
10	100	200	4366.6	2.4
11	110	220	4366.9	2.5
12	120	240	4367.1	2.4
13	130	260	4367.4	2.4
14	140	280	4367.4	2.4
15	150	300	4367.6	2.4
16	160	320	4367.7	2.4

Table 11: Estimated Volumes and Uncertainties for Green 5 Data

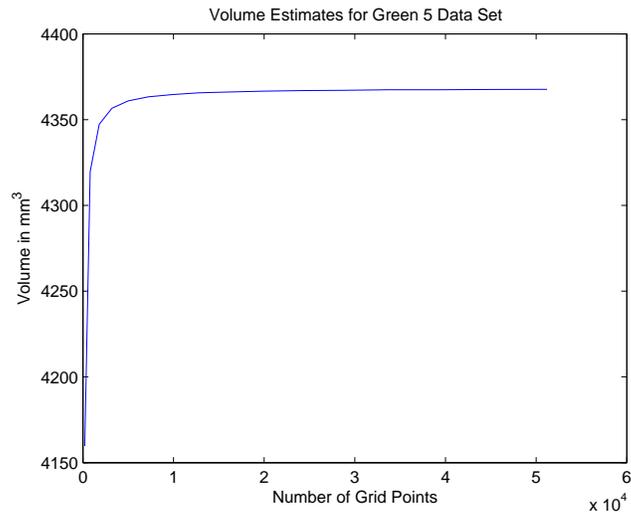


Figure 24: Volume Estimates for Green 5 Data Set vs. Grid Size.

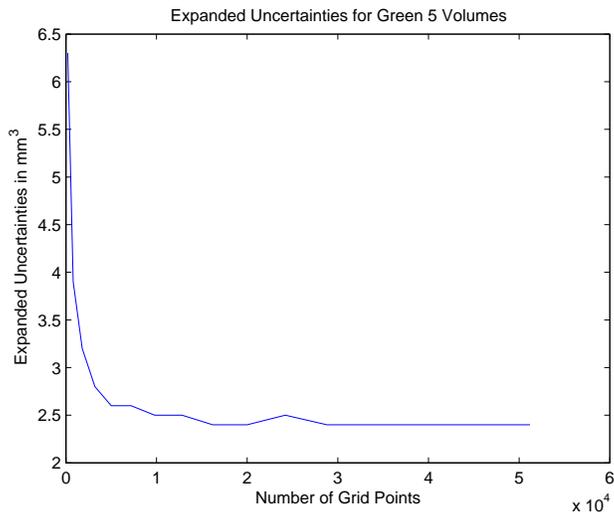


Figure 25: Uncertainty Estimates for Green 5 Data Set vs. Grid Size.

Green 6 Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	4159.8	6.7
2	20	40	4319.8	3.9
3	30	60	4347.3	3.1
4	40	80	4356.7	2.7
5	50	100	4360.9	2.4
6	60	120	4363.2	2.4
7	70	140	4364.6	2.3
8	80	160	4365.5	2.3
9	90	180	4366.0	2.3
10	100	200	4366.6	2.3
11	110	220	4366.8	2.2
12	120	240	4367.1	2.2
13	130	260	4367.3	2.2
14	140	280	4367.4	2.2
15	150	300	4367.6	2.2
16	160	320	4367.7	2.2

Table 12: Estimated Volumes and Uncertainties for Green 6 Data

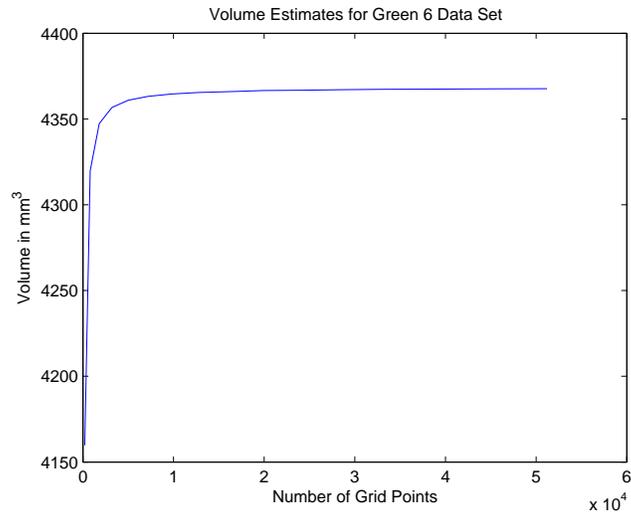


Figure 26: Volume Estimates for Green 6 Data Set vs. Grid Size.

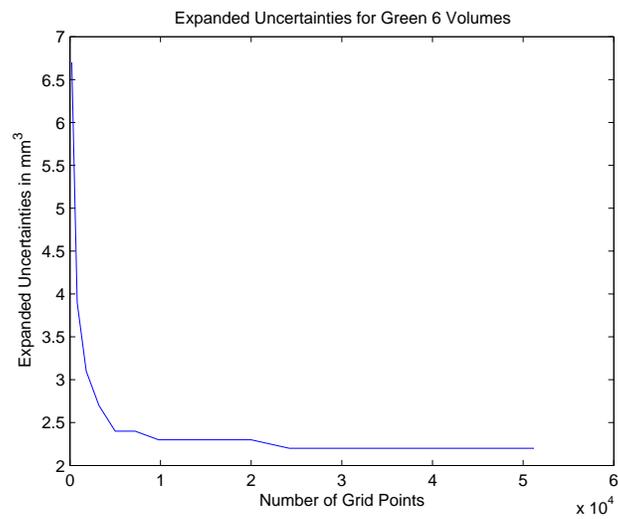


Figure 27: Uncertainty Estimates for Green 6 Data Set vs. Grid Size.

Pink 5 Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3762.4	24.0
2	20	40	3905.6	11.6
3	30	60	3928.9	8.6
4	40	80	3939.0	6.2
5	50	100	3942.2	5.5
6	60	120	3944.8	4.5
7	70	140	3946.1	4.2
8	80	160	3946.6	3.9
9	90	180	3947.1	3.6
10	100	200	3947.3	3.4
11	110	220	3947.8	3.3
12	120	240	3947.9	3.3
13	130	260	3948.1	3.0
14	140	280	3948.3	2.9
15	150	300	3948.4	3.0
16	160	320	3948.5	3.0

Table 13: Estimated Volumes and Uncertainties for Pink 5 Data

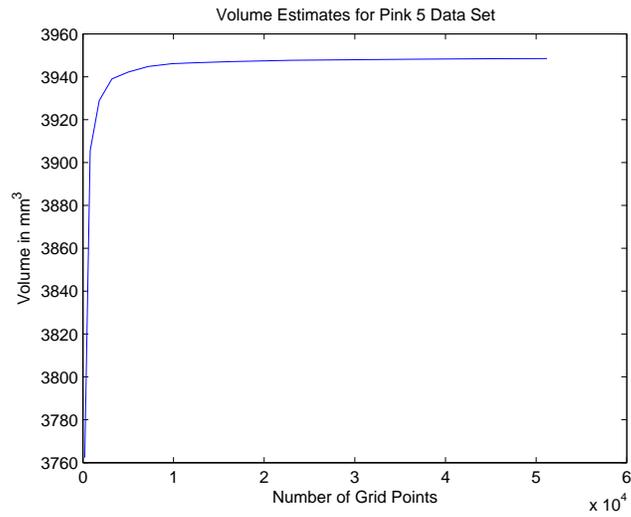


Figure 28: Volume Estimates for Pink 5 Data Set vs. Grid Size.

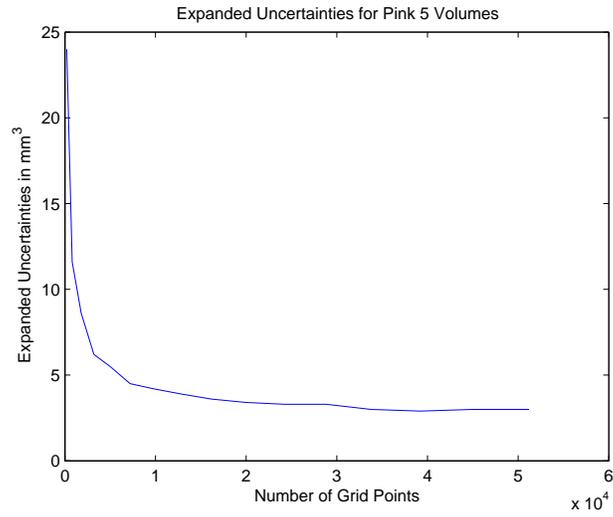


Figure 29: Uncertainty Estimates for Pink 5 Data Set vs. Grid Size.

Pink 6 Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3757.7	23.9
2	20	40	3903.1	11.2
3	30	60	3927.9	7.6
4	40	80	3937.2	6.1
5	50	100	3940.3	5.3
6	60	120	3942.7	4.6
7	70	140	3944.4	3.8
8	80	160	3944.5	3.8
9	90	180	3945.6	3.4
10	100	200	3945.6	3.5
11	110	220	3946.0	3.2
12	120	240	3946.4	3.3
13	130	260	3946.4	3.1
14	140	280	3946.6	2.9
15	150	300	3946.7	2.9
16	160	320	3946.8	2.9

Table 14: Estimated Volumes and Uncertainties for Pink 6 Data

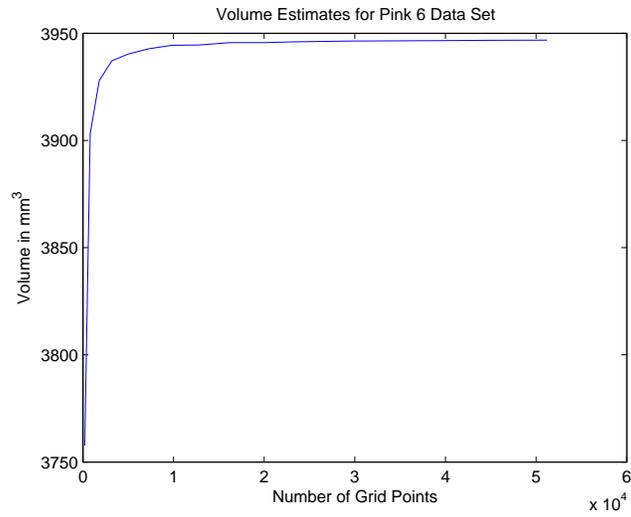


Figure 30: Volume Estimates for Pink 6 Data Set vs. Grid Size.

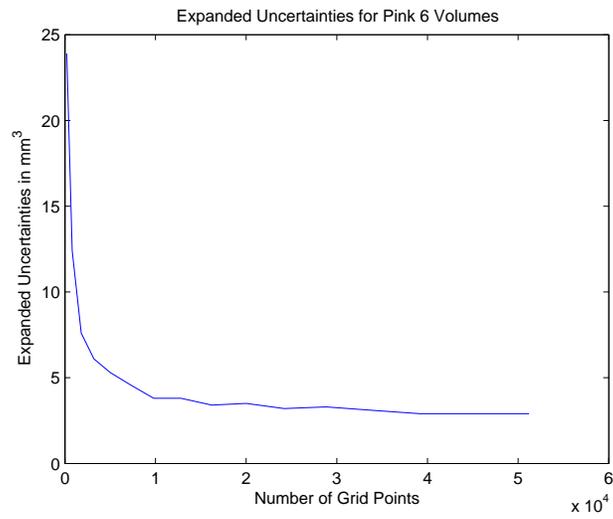


Figure 31: Uncertainty Estimates for Pink 6 Data Set vs. Grid Size.

Pink 7 Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3761.1	23.0
2	20	40	3904.2	11.9
3	30	60	3928.9	8.1
4	40	80	3937.1	6.1
5	50	100	3940.5	4.8
6	60	120	3942.9	4.7
7	70	140	3943.9	4.1
8	80	160	3945.0	3.9
9	90	180	3945.4	3.7
11	100	200	3946.2	3.3
10	110	220	3946.2	3.4
12	120	240	3946.2	3.1
13	130	260	3946.6	3.0
14	140	280	3946.7	3.0
15	150	300	3946.9	3.0
16	160	320	3946.9	2.9

Table 15: Estimated Volumes and Uncertainties for Pink 7 Data

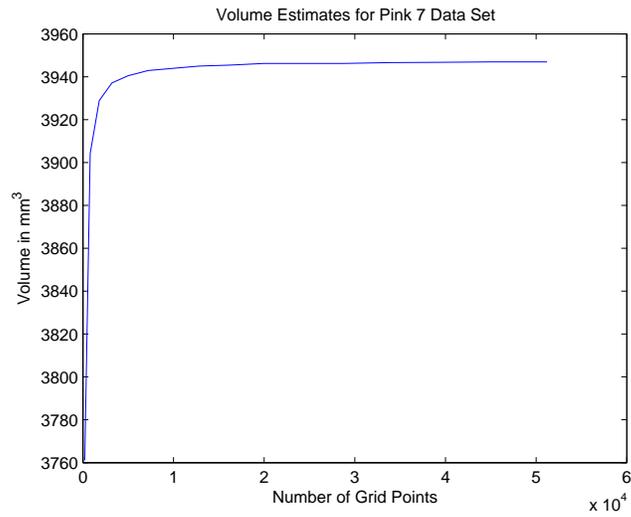


Figure 32: Volume Estimates for Pink 7 Data Set vs. Grid Size.

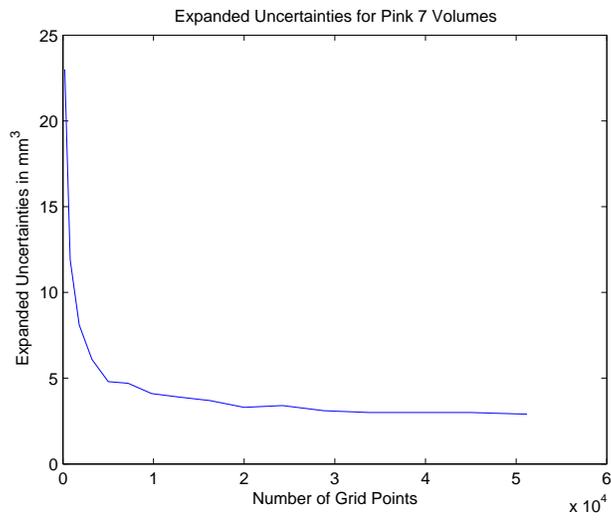


Figure 33: Uncertainty Estimates for Pink 7 Data Set vs. Grid Size.

Green 1 Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3886.3	115.2
2	20	40	4031.0	64.3
3	30	60	4057.4	53.7
4	40	80	4065.3	50.9
5	50	100	4068.8	47.9
6	60	120	4070.9	47.0
7	70	140	4073.5	46.1
8	80	160	4073.8	45.1
9	90	180	4074.1	45.4
10	100	200	4074.8	45.1
11	110	220	4076.0	45.1
12	120	240	4074.7	44.0
13	130	260	4075.1	44.2
14	140	280	4076.2	44.2
15	150	300	4075.5	43.7
16	160	320	4075.7	43.7

Table 16: Estimated Volumes and Uncertainties for Green 1 Dense Data

4.6.4 Volume Estimates and Uncertainties for the Dense Data

In this section we report the results of volume estimates and the extended uncertainties for the dense data sets of 181 probed points on the phantom surfaces. The results do not seem to be what one would have expected. The first thing of interest is that the stabilized volume estimates for the Green dense data in Tables 16 through 18 differ from the volume estimates for the Green coarse data by about 7 %. This is in the negative direction in that the volume estimates for the Green dense data were smaller than for the Green coarse data. In the case of the volume estimates for the Pink dense data in Tables 19 through 21, the differences with the volume estimates for the Pink coarse data vary from 4 % to 11 % smaller. Another thing to notice is that the uncertainties are significantly higher as shown in Tables 16 through 21 compared to those estimated from the coarse data. In fact they appear to be an order of magnitude larger. It is not clear why the results for the dense data differ from the results for the coarse data, but we are reporting it nonetheless. The complete analysis of these differences is beyond the scope of this report but will be pursued in a coming study related to a redesign of the fitting algorithm.

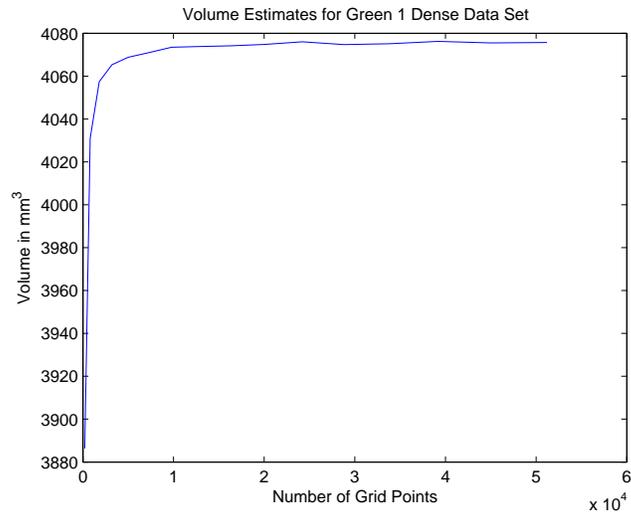


Figure 34: Volume Estimates for Green 1 Dense Data Set vs. Grid Size.

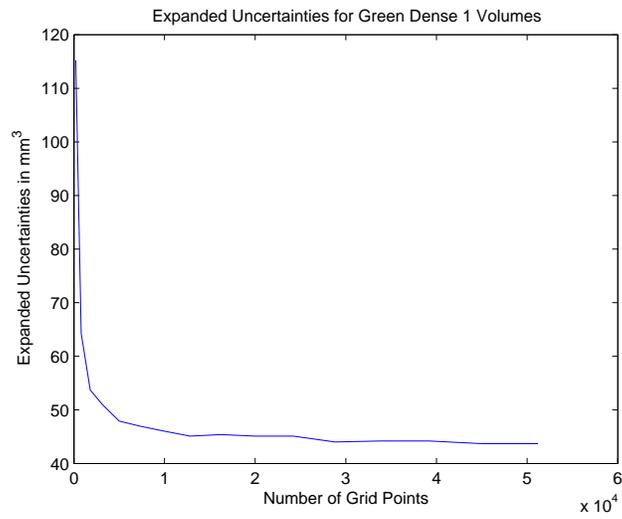


Figure 35: Uncertainty Estimates for Green 1 Dense Data Set vs. Grid Size.

Green 2 Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3863.7	112.9
2	20	40	4000.5	74.0
3	30	60	4028.9	61.6
4	40	80	4040.0	54.7
5	50	100	4039.9	51.8
6	60	120	4044.5	51.8
7	70	140	4042.3	50.1
8	80	160	4044.2	50.2
9	90	180	4045.3	49.6
10	100	200	4044.9	49.3
11	110	220	4044.8	47.8
12	120	240	4045.9	48.1
13	130	260	4045.8	49.0
14	140	280	4046.0	47.8
15	150	300	4046.2	47.8
16	160	320	4046.9	48.3

Table 17: Estimated Volumes and Uncertainties for Green 2 Dense Data

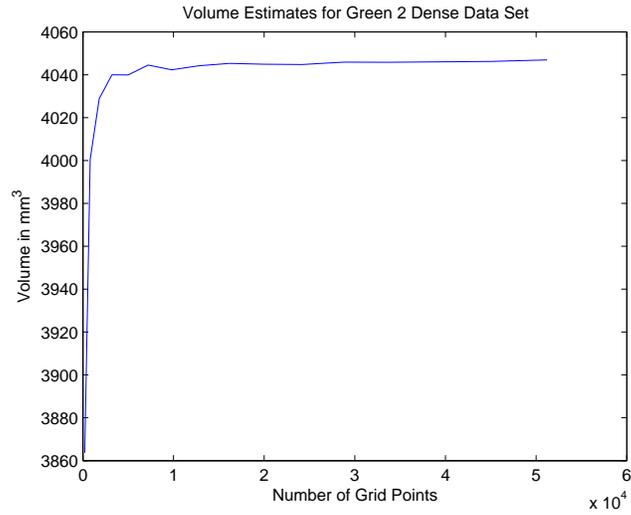


Figure 36: Volume Estimates for Green 2 Dense Data Set vs. Grid Size.

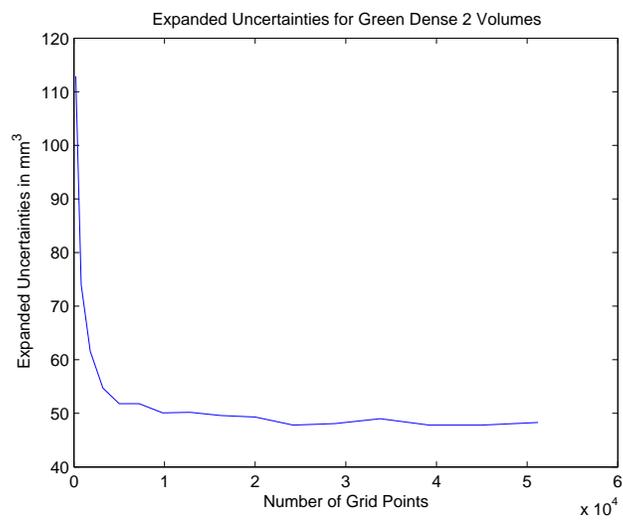


Figure 37: Uncertainty Estimates for Green 2 Dense Data Set vs. Grid Size.

Green 3 Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3921.9	111.2
2	20	40	4076.5	71.7
3	30	60	4103.1	64.1
4	40	80	4107.6	56.9
5	50	100	4111.2	53.5
6	60	120	4113.1	53.4
7	70	140	4114.9	52.0
8	80	160	4117.1	50.0
9	90	180	4118.8	50.8
10	100	200	4119.9	49.8
11	110	220	4118.5	49.6
12	120	240	4118.3	49.7
13	130	260	4120.0	49.6
14	140	280	4119.3	50.4
15	150	300	4120.1	49.5
16	160	320	4119.7	49.4

Table 18: Estimated Volumes and Uncertainties for Green 3 Dense Data

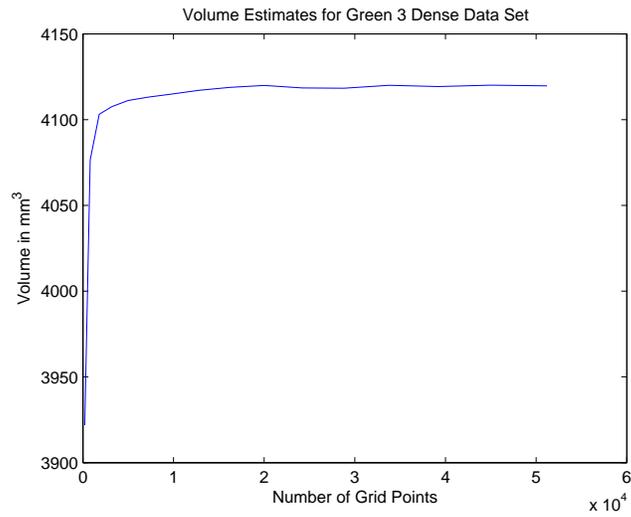


Figure 38: Volume Estimates for Green 3 Dense Data Set vs. Grid Size.

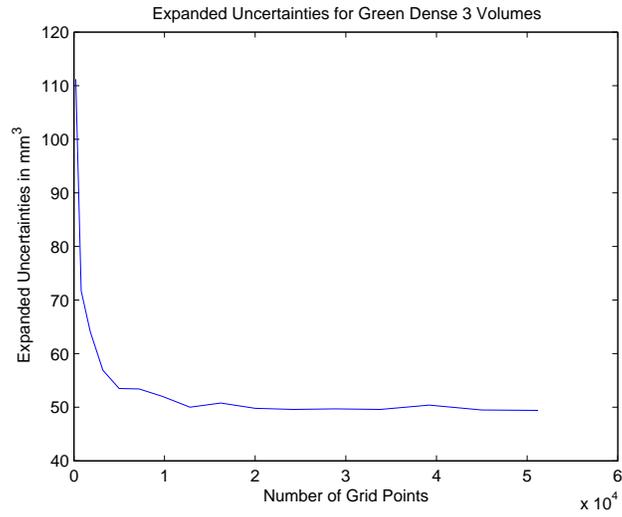


Figure 39: Uncertainty Estimates for Green 3 Dense Data Set vs. Grid Size.

Pink 1 Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3719.8	87.6
2	20	40	3865.9	56.1
3	30	60	3887.5	46.1
4	40	80	3895.4	41.8
5	50	100	3900.1	38.5
6	60	120	3902.4	37.4
7	70	140	3903.4	37.0
8	80	160	3904.2	37.2
9	90	180	3904.5	35.8
10	100	200	3903.8	35.1
11	110	220	3905.0	35.1
12	120	240	3905.0	35.1
13	130	260	3904.2	34.6
14	140	280	3905.0	34.5
15	150	300	3905.7	34.6
16	160	320	3905.4	34.3

Table 19: Estimated Volumes and Uncertainties for Pink 1 Dense Data

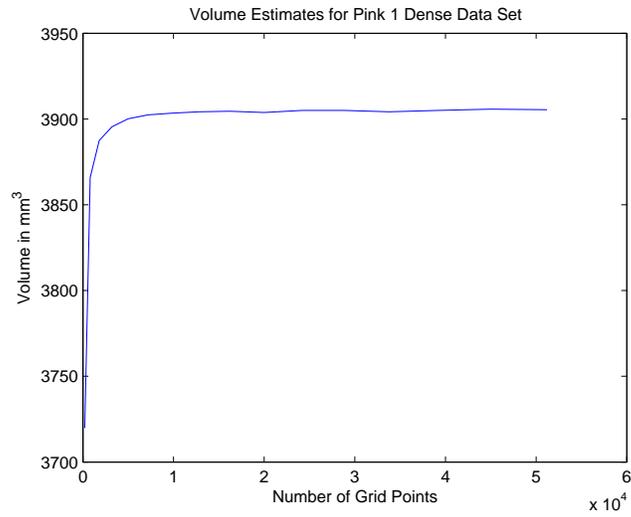


Figure 40: Volume Estimates for Pink 1 Dense Data Set vs. Grid Size.

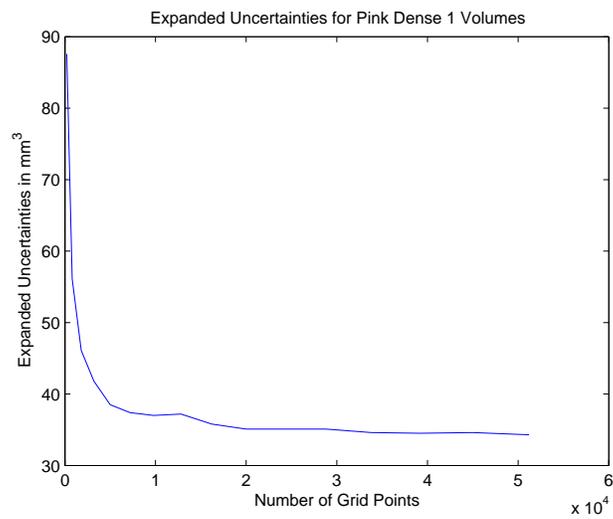


Figure 41: Uncertainty Estimates for Pink 1 Dense Data Set vs. Grid Size.

Pink 2 Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3752.0	95.3
2	20	40	3891.8	53.6
3	30	60	3918.3	44.7
4	40	80	3926.3	40.0
5	50	100	3928.7	37.8
6	60	120	3931.9	36.8
7	70	140	3934.8	37.3
8	80	160	3934.5	35.9
9	90	180	3935.2	36.1
10	100	200	3935.4	35.9
11	110	220	3936.7	35.7
12	120	240	3936.5	35.9
13	130	260	3935.9	35.2
14	140	280	3936.9	34.5
15	150	300	3935.8	35.1
16	160	320	3937.1	34.9

Table 20: Estimated Volumes and Uncertainties for Pink 2 Dense Data

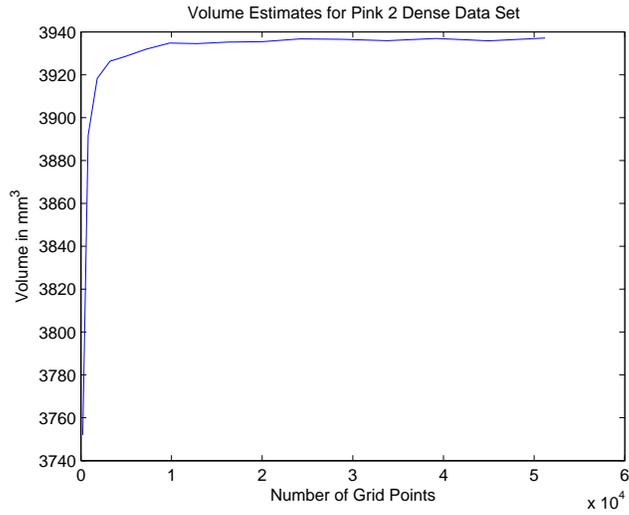


Figure 42: Volume Estimates for Pink 2 Dense Data Set vs. Grid Size.

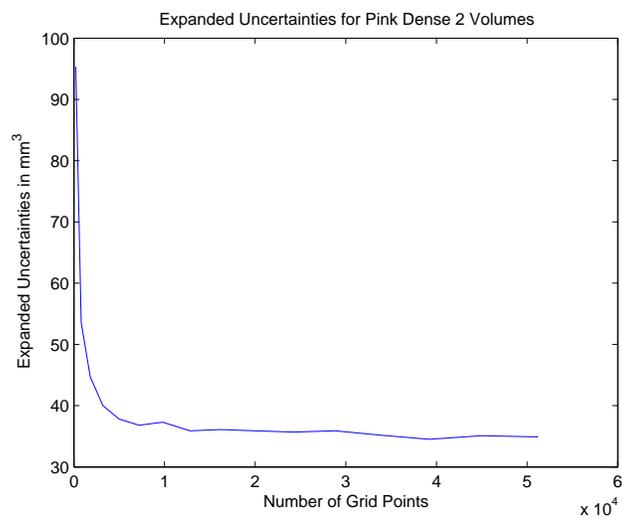


Figure 43: Uncertainty Estimates for Pink 2 Dense Data Set vs. Grid Size.

Pink 3 Dense Data Set				
Volumes and Expanded Uncertainties vs Grid Size				
Grid Case	θ	ϕ	Volume (mm ³)	Uncertainty (mm ³)
1	10	20	3338.5	115.0
2	20	40	3467.2	78.3
3	30	60	3491.0	64.4
4	40	80	3499.1	58.6
5	50	100	3500.1	56.6
6	60	120	3500.5	56.2
7	70	140	3504.1	53.4
8	80	160	3505.1	53.1
9	90	180	3504.1	51.7
10	100	200	3506.4	52.1
11	110	220	3506.6	52.8
12	120	240	3507.1	52.3
13	130	260	3508.5	51.8
14	140	280	3507.6	52.1
15	150	300	3508.7	51.8
16	160	320	3507.1	51.7

Table 21: Estimated Volumes and Uncertainties for Pink 3 Dense Data

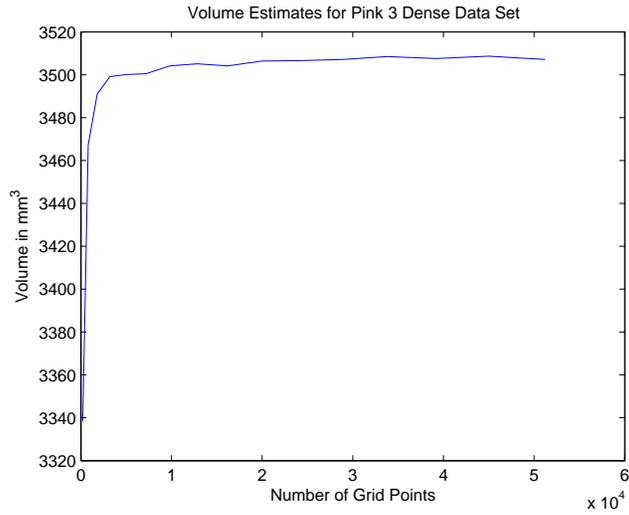


Figure 44: Volume Estimates for Pink 3 Dense Data Set vs. Grid Size.

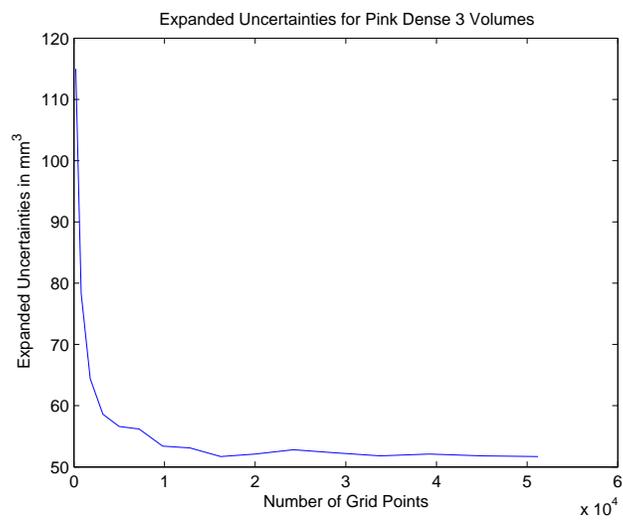


Figure 45: Uncertainty Estimates for Pink 3 Dense Data Set vs. Grid Size.

5 Summary

The B-spline surface model joined with the Divergence Theorem seems to be a viable approach for estimating volumes of near spherical star-shaped artifacts, but the results seem to depend strongly on the distribution of the data points on the surface. In the case of sparse data the problem of extrapolating a surface fitted to the data to a grid on the surface leads potentially to unwanted oscillations in the neighborhood of some of the sparse data points. In the current algorithm this problem was dealt with for convenience by a chopping operation, but this may not be a good procedure and may have affected the differences between the coarse data results and the dense data results, although this needs to be investigated further.

The results obtained from the coarse data distribution appear to be consistent with the results from the spherical model fits. As shown in Tables 10, 11, and 12 in Section 4.6.3 the volume estimates for the Green sphere are very close to the estimates obtained by the B-spline model suggesting that the Green sphere is essentially spherical in shape. As expected the volume estimates in Table 4 for the Pink sphere are significantly lower than the B-spline model estimates in Tables 13, 14, and 13, in this case suggesting that the Pink sphere has a significant non-spherical shape.

The results from the dense data distribution were not what would have been expected. The uncertainties were significantly higher than those for the coarse data. Tables 16 to 18 show volumes smaller than those from 10, 11, and 12. Tables 19 and 20 show similar volume estimates but Table 21 shows significantly lower volume estimates. This, of course, could have been brought about by any difficulty in holding the Pink phantom in position 3.

In summary we conclude that the combination of using the CMM to obtain surface data and the B-spline/Divergence Theorem volume estimation method can produce useful results but that there are some limitations. First, the CMM is primarily used for probing manufactured metallic artifacts and its use in probing non-metallic artifacts such as the FDA phantoms can lead to some large uncertainties in volume estimation. Second, the distribution of probed points can lead to unexpected results. So there is a problem of consistency here. Third, the methods used may not provide useful volume estimation for more complex artifacts that surely would arise in developing simulated lung cancers. The current artifacts were near-spherical and even these did not provide fully expected results based on data distribution. Further research on the affect of surface data distribution is required as well as further testing of the volume estimation program. The Pole Problem is definitely a limitation. As a result of these limitations we think other means of estimating "ground truth" volumes for the current and any future phantoms be investigated by the FDA.

6 Acknowledgement

The authors wish to acknowledge the generous assistance of Stefan Leigh of the Statistical Engineering Division of ITL at NIST in designing the volume uncertainty estimation algorithm by the bootstrap method. We also wish to acknowledge the gracious assistance of Prof. Dianne P. O'Leary, University of Maryland, in writing an efficient version of the least squares program using SVD. Finally, we acknowledge the use of the Nelder-Mead code by Prof. Timothy Sauer, George Mason University.

7 Disclaimer

Certain commercial software products are identified in this paper in order to adequately specify the computational procedures. Such identification does not imply recommendation or endorsement by

the National Institute of Standards and Technology nor does it imply that the software products identified are necessarily the best available for the purpose.

References

- [1] American Cancer Society, *Cancer Facts & Figures 2007*. Atlanta: American Cancer Society, 2007.
- [2] Cox, M. G., 'The Numerical Evaluation of B-splines', *J. Inst. Maths Applics*, **10**, (1972), 134-149.
- [3] de Boor, C., *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
- [4] Dierckx, P., 'An Algorithm for Surface-Fitting with Spline Functions', *IMA Journal of Numerical Analysis*, **1**, (1981), 267-283.
- [5] Dierckx, P., 'Algorithms for Smoothing Data on the Sphere with Tensor Product Splines', *Computing*, **32**, 319-342.
- [6] Dierckx, P., *Curve and Surface Fitting with Splines*, Clarendon Press, Oxford, 2006.
- [7] Draper, N. and Smith, H., *Applied Regression Analysis*, John Wiley & Sons, New York, 1981.
- [8] Eberly, D. H., *Game Physics*, Morgan Kaufmann Publishers, Amsterdam, 2004.
- [9] Efron, B. and Tibshirani, R. J., *An Introduction to the Bootstrap*, Chapman and Hall, New York, 1993.
- [10] Hayes, J. G. and Halliday, J., 'The Least-squares Fitting of Cubic Spline Surfaces to General Data Sets', *J. Inst. Maths Applics*, **14**, (1974), 89-103.
- [11] http://en.wikipedia.org/wiki/Computer_tomography
- [12] Jenal, A., Siegel, R., Ward, E., Murray, T., Xu, J., Thun, J., 'Cancer Statistics, 2007', *CA Cancer J Clin*, **57**, 1,(2007), 43-66.
- [13] Rogers, D. F. and Adams, J. A., *Mathematical Elements for Computer Graphics*, McGraw-Hill, Boston, 1990.
- [14] Sauer, T., *Numerical Analysis*, Pearson, Boston, 2006.
- [15] Shakarji, C. M., 'Least-Squares Fitting Algorithms of the NIST Algorithm Testing System', *J. Res. Natl. Inst. Stand. Technol.*, **103**, (1998), 633-641.
- [16] Taylor, A. E., *Advanced Calculus*, Ginn and Company, Boston, 1955.
- [17] Taylor, B. N. and Kuyatt, C. E., 'Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results', *NIST Technical Note 1297, 1994 Edition*, National Institute of Standards and Technology, 1994.
- [18] Therasse, P., Arbuck, S. G., Eisenhauer, E. A., Wanders, J., Kaplan, R. S., Rubinstein, L., Verweij, J. Van Glabbeke, M., van Oosterom, A. T., Christian, M. C., Gwyther, S. G., 'New Guidelines to Evaluate the Response to Treatment in Solid Tumors', *Journal of the National Cancer Institute*, **92**, 3, (2000), 205-216.

A Fitting a Sphere to Probe Data

```
function sphere_fit_to_phantoms_simplex
%*****
%This functions uses a nonlinear optimization procedure to fit a set of
%(x,y,z) data points determined by a CMM on the surface of two phantom lung
%cancer lesions supplied to NIST by the FDA.
%*****
global x y z
format long;
%Open the CMM data file
%select the file desired by removing the % sign
%fid= fopen('Tumor_Green_and_Pink_Results_Green_4.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Green_5.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Green_6.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Pink_5.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Pink_6.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Pink_7.txt','r');
%fid= fopen('Green_Results_dense_data_1.txt','r');
%fid= fopen('Green_Results_dense_data_2.txt','r');
%fid= fopen('Green_Results_dense_data_3.txt','r');
%fid= fopen('Pink_Results_dense_data_1.txt','r');
%fid= fopen('Pink_Results_dense_data_2.txt','r');
fid= fopen('Pink_Results_dense_data_3.txt','r');
%fid= fopen('Cal_121_sphere.txt','r');
%fid= fopen('Cal_181_sphere.txt','r');
%reads file into a 3-by-inf matrix and takes the transpose. This is
%required due to the way Matlab reads an array.
A=fscanf(fid, '%f %f %f', [3 inf])';
[n,m] = size(A);
x=A(:,1);
y=A(:,2);
z=A(:,3);
A
fclose(fid);
mnx = mean(x);
mny = mean(y);
mnz = mean(z);
%Set initial center and radius
c = [mnx mny mnz 10]';
[c_final,val_final,iter]=neldermead_mod(@f2,c,1,1.0e-14,1000)
%Choose the median values of the tetrahedron vertex coordinates
c_final_median = median(c_final)';
Estimated_volume = (4/3)*pi*(c_final_median(4))^3
%Choose the median of the sum of squares of the function values at the
%vertices.
val_final_median = median(val_final)';
%Compute the residual radial value between the data points to the computed
```

```

%center and the estimated sphere radius.
%First get the radial values to the data points
radius = sqrt((x-c_final_median(1)).^2 + (y-c_final_median(2)).^2 + (z-c_final_median(3)).^2);
Residual_radius = radius - c_final_median(4);
for i = 1:length(x)
    disp(sprintf('%f    %f    %f    %f    %f ',x(i),y(i),z(i), radius(i), Residual_radius(i)));
end
Mean_residual_radius = mean(Residual_radius)
Mean_abs_residual_radius = mean(abs(Residual_radius))
Std_residual_radius = std(Residual_radius)
Std_abs_residual_radius = std(abs(Residual_radius))

%Estimating coefficient uncertainties
Z_hat = [-2*(x(:)-c_final_median(1)) -2*(y(:)-c_final_median(2)) -2*(z(:)-c_final_median(3)) -2*c
s = f2(c_final_median)/(length(x) - 4);
SE = inv(Z_hat'*Z_hat)*s^2;
for i = 1:4
    se_c(i) = sqrt(SE(i,i));
    expand_uncert_c(i) = 2*se_c(i);
end
se_c
expand_uncert_c

%*****
%This function computes the sum of ((x(i) - c(1))^2 + (y(i) - c(2))^2
%+ (z(i) - c(3))^2 - c(4)^2)^2 for i = 1:n
%*****
function g = f2(c)
%*****
%F2 - Forms the sum of squares of the sphere model equations to be solved
%
%Input:
%  c - Parameters defining a sphere. (c(1),c(2),c(3)) the center, c(4) the
%      radius
%
%*****

global x y z
n = length(x);
g = zeros(n,1);
for i = 1:n
    p = x(i) - c(1);
    q = y(i) - c(2);
    r = z(i) - c(3);
    p2 = p^2;
    q2 = q^2;
    r2 = r^2;
    s2 = c(4)^2;

```

```

    pqrs = p2 + q2 + r2 - s2;
    g(i,1) = pqrs^2;
end
g = sum(g);

%*****
%Nelder-Mead search algorithm for minimizing a nonlinear function without
%derivatives.
%*****
function [x,y,iter]=neldermead_mod(f,xbar,rad,tol,maxit)
%*****
% Modified Nelder-Mead Search (modified with comments and convergence test)
% Input: function handle f, best guess xbar (column vector),
%       initial search radius rad , tolerance on iterative polyhedron volume,
%       and maximum iterations allowed.
% Output: matrix x whose columns are vertices of simplex,
%         function values y of those vertices, and iterations used.
%
%Author: T. Sauer, George Mason University
%Modified: D. Gilsinn, National Institute of Standards and Technology
%
%*****
n=length(xbar);
x(:,1)=xbar;          % each column of x is a simplex vertex
x(:,2:n+1)=xbar*ones(1,n)+rad*eye(n,n);
%determinant related to the volume of a polyhedron
v = zeros(n+1,n+1);
v(1,:) = 1;
for j = 1:n+1
    v(2:n+1,j) = x(:,j);
end
d = det(v);
for j=1:n+1
    y(j)=f(x(:,j)); % evaluate obj function f at each vertex
end
%The output array r is the ascending order of the values in the original y
[y,r]=sort(y);      % sort the function values in ascending order
x=x(:,r);           % and rank the vertices the same way
iter = 1;
while ((abs(d) > tol)& (iter <= maxit))
    iter = iter + 1;
    %The mean function of a matrix produces a row vector of the column
    %means. Since we want row means across vertices we have to
    %transpose the matrix of vertices. We then transpose the row of
    %means to form a column vector.
    xbar=mean(x(:,1:n)')'; % xbar is the centroid of the face
    xh=x(:,n+1);          % omitting the worst vertex xh
    %First step in each iteration is to reflect through the centroid

```

```

%of the opposite face to the worst point and compute the function
%value at the reflected point
xr = 2*xbar - xh; yr = f(xr);
%There are two major cases to test: yr < y(n), y(n)<= yr
if yr < y(n)
    %If yr is better than the current best value then this indicates
    %that we might get better values by continuing in the same direction
    %by an expansion
    if yr < y(1)      % try expansion xe, i.e. xe = xbar + 2*(xbar-xh)
        xe = 3*xbar - 2*xh; ye = f(xe);
        %If the expansion leads to a better value accept the point
        %go to the end, resort the points and continue to the next iteration
        if ye < yr    % accept expansion
            x(:,n+1) = xe; y(n+1) = f(xe); %resort and go to the next iteration
        else          % accept reflection point as the next best if yr<=ye
            x(:,n+1) = xr; y(n+1) = f(xr);
        end
    else              % xr is middle of pack, accept reflection
        x(:,n+1) = xr; y(n+1) = f(xr);
    end
else                  %Now test y(n) <= yr, use contractions
    if yr < y(n+1)    % try outside contraction xoc
        xoc = 1.5*xbar - 0.5*xh; yoc = f(xoc);
        if yoc < yr   % accept outside contraction if the new point is better
            x(:,n+1) = xoc; y(n+1) = f(xoc);
        else          % otherwise shrink polytope toward best point
            for j=2:n+1
                x(:,j) = 0.5*x(:,1)+0.5*x(:,j); y(j) = f(x(:,j));
            end
        end
    else              %here y(n+1) <= yr and xr is even worse than the previous worst
        xic = 0.5*xbar+0.5*xh; yic = f(xic); %contract towards the previous worst point
        if yic < y(n+1) %if things are better accept inside contraction
            x(:,n+1) = xic; y(n+1) = f(xic);
        else          % if y(n+1) <= yic then shrink polytope toward best point
            for j=2:n+1
                x(:,j) = 0.5*x(:,1)+0.5*x(:,j); y(j) = f(x(:,j));
            end
        end
    end
end
end
[y,r] = sort(y);      % resort the new obj function values
x=x(:,r);            % and rank the vertices the same way
%determinant related to the volume of n-hedron
for j = 1:n+1
    v(2:n+1,j) = x(:,j);
end
d = det(v);

```

```
end                                     %go back for the next iteration.
```

B Main Volume Estimation Code

```
function Main_Phantom_Volume
%*****
%MAIN_PHANTOM_VOLUME - This function is an interactive function that reads
%
%       a data file of Cartesian coordinates, obtained from
%       a CMM, converts them to spherical coordinates,
%       adaptively selects a near optimum set of knots to
%       fit a tensor product of B-splines to the data,
%       generates a grid of points in theta and phi and
%       extrapolates the B-spline function values to the
%       grid and finally computes the volume of a phantom
%       for that grid. A statistical bootstrap method is
%       used to estimate volumetric uncertainties.
%
%Author:
%   David E. Gilsinn
%   Mathematical and Computational Sciences Division
%   Information Technology Laboratory
%   National Institute of Standards and Technology
%   Gaithersburg, MD 20899-8910
%*****

%*****
%Data initialization section, knot selection, and least squares fit.
%Executed once.
%*****
%Get the Cartesian data. Choose a file name included in the function

[x,y,z] = Get_CMM_data;
N = length(x);

%Convert to spherical coordinates. The term base is used to indicate
%results not affected by any scaling of the radial data, otherwise the term
%shift will be used.

[theta_base,phi_base,rad_base] = Get_Spherical_Coordinates(x,y,z);

%Take the mean out of the radial data. It will be added back later.

mean_rad_base = mean(rad_base);
max_rad_base = max(rad_base);
min_rad_base = min(rad_base);
%rad_shift = rad_base - mean_rad_base;
```

```

rad_shift = rad_base;

%Adaptively select an optimum set of knots for theta and phi
%The final fit is made to the shifted radial data.

[rad_pred_shift,c_shift,NM_shift,resid_shift,knts_theta,knts_phi,iter]...
    = adaptive_knot_selection(theta_base,phi_base,rad_shift);

disp('*****');
disp('Initial Fitting Statistics');
disp('*****');
disp(sprintf('Maximum fitting residual, shifted data = %g ', max(resid_shift)));
disp(sprintf('Minimum of fitting residual, shifted data = %g ', min(resid_shift)));
disp(sprintf('Mean of fitting residual, shifted data = %g ', mean(resid_shift)));
disp(sprintf('Standard deviation of fitting residual, shifted data = %g ', std(resid_shift)));

%*****
%End data initialization, knot selection, and initial least squares fit
%*****

%*****
%Begin interactive volume determination for selected grid sizes
%*****
disp('*****');
disp('Interactive Grid Selection and Volume Estimate');
disp('*****');

flag = 1;
while (flag > 0)
    flag = input('To continue with a new grid type 1, else type -1 > ');
    if (flag <= 0)
        break;
    end
    %Set up grid parameters for generating a longitudinal and latitudinal grid
    nt_grid = input('Enter desired number of theta grid points > ');
    np_grid = input('Enter desired number of phi grid points > ');
    %This function adjusts the extrapolation due to the shifted
    %coefficients by rescaling the predicted radial values on the grid.
    [t_grid,p_grid,rad1_base, N1M1]...
        = Generate_grid_data(c_shift,nt_grid,np_grid, knts_theta,knts_phi,rad_base);
    %Compute the volume
    vol(1) = volume(t_grid,p_grid,rad1_base);
    disp(sprintf('Initial volume in cubic mm = %g', vol(1)));
    Boot = 1;
    Boot = input('Type -1 to skip the bootstrap process > ');
    new_resid = zeros(N,1);
    data_tol = 1e-14;
    if(Boot > 0)

```

```

%Bootstrap process to estimate volume uncertainty and ref volume
for i = 2:200
    disp(sprintf('Bootstrap iteration %d out of 200.',i));
    for j = 1:N
        u = rand;
        ii = fix(u*N) + 1;
        new_resid(j) = resid_shift(ii);
    end
    new_pred_shift = rad_pred_shift + new_resid;
    %pred values are the extrapolated values to the grid based on
    %the new coefficients c
    c1 = lsq_2D_spline(NM_shift,new_pred_shift,data_tol);
    pred = N1M1*c1 + mean_rad_base;
    delta_plus = max_rad_base - mean_rad_base;
    delta_minus = mean_rad_base - min_rad_base;
    outlyer = find( (pred < mean_rad_base - delta_minus) | (pred > mean_rad_base + delta_plus) );
    rnd = rand(length(outlyer),1);
    for ii =1:length(outlyer)
        j = outlyer(ii);
        if (rnd(ii,1) <= 0.5)
            pred(j) = mean_rad_base - rnd(ii,1)*delta_minus;
        else
            pred(j) = mean_rad_base + rnd(ii,1)*delta_plus;
        end
    end
    end
    vol(i) = volume(t_grid,p_grid,pred);
    %disp(sprintf('Volume(%d) = %f',i,vol(i)));
end
vol_uncert = std(vol);
vol_ref = mean(vol);
disp(sprintf('Volume uncertainty = %f cubic mm',vol_uncert));
disp(sprintf('Reference Volume = %f cubic mm',vol_ref));
end
%clear variables for next pass
clear t_grid p_grid rad1_base
end

```

C Main_Phantom_Volume Support Programs

C.1 Function to Read CMM Data

```

function [x,y,z] = Get_CMM_data
%*****
%GET_CMM_DATA - This function opens a file of (x,y,z) data from the CMM
%                and recenters the data
%
%
%Output:
%  x,y,z - Re-centered CMM data

```

```

%
%Author:
%   David E. Gilsinn
%   Mathematical and Computational Sciences Division
%   Information Technology Laboratory
%   National Institute of Standards and Technology
%   Gaithersburg, MD 20899-8910
%*****
%Open the CMM data file
%fid= fopen('Tumor_Green_and_Pink_Results_Green_4.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Green_5.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Green_6.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Pink_5.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Pink_6.txt','r');
%fid= fopen('Tumor_Green_and_Pink_Results_Pink_7.txt','r');
%fid= fopen('Green_Results_dense_data_1.txt','r');
%fid= fopen('Green_Results_dense_data_2.txt','r');
%fid= fopen('Green_Results_dense_data_3.txt','r');
%fid= fopen('Pink_Results_dense_data_1.txt','r');
%fid= fopen('Pink_Results_dense_data_2.txt','r');
%fid= fopen('Pink_Results_dense_data_3.txt','r');
%fid= fopen('Cal_121_sphere.txt','r');
fid= fopen('Cal_181_sphere.txt','r');
%fid= fopen('pointcloud.txt','r');
%reads file into a 3-by-inf matrix and takes the transpose. This is
%required due to the way Matlab reads an array.
A=fscanf(fid, '%f %f %f', [3 inf]);
%Now center the data
n=length(A(:,1));
meanx = sum(A(:,1))/n;
meany = sum(A(:,2))/n;
meanz = sum(A(:,3))/n;
x = zeros(n,1);
y = zeros(n,1);
z = zeros(n,1);
for i=1:n
    x(i,1)=A(i,1)-meanx;
    y(i,1)=A(i,2)-meany;
    z(i,1)=A(i,3)-meanz;
end
fclose(fid);

```

C.2 Function to Euclidean to Spherical Coordinates

```

function [theta,phi,rad] = Get_Spherical_Coordinates(x,y,z)
%*****
%GET_SPHERICAL_COORDINATES - Convert Cartesian to Spherical Coord and
%                               returns the spherical coordinates of data

```

```

%Output:
%  theta (colatitude from +z axis)
%  phi (azimuth from + x axis)
%  rad (radial data)
%
%Author:
% Amelia Tebbe
% Department of Mathematics
% St. Mary's College of Maryland
% St. Mary's City, MD 20686-3001
%
%*****
B = sph_coord(x,y,z);
%Read the columns of B into separate vectors.
theta = B(:,1);
phi = B(:,2);
rad = B(:,3);

function D=sph_coord(a,b,c)
%*****
%SPH_COORD - Converts Cartesian coordinates to spherical coordinates
%INPUT:
%  a - Vector of x Cartesian coordinates of points
%  b - Vector of y Cartesian coordinates of points
%  c - Vector of x Cartesian coordinates of points
%
%OUTPUT:
%  D - Matrix of dimension length of a x 3 of Spherical coordinates
%      (r,theta,phi) where r is the radius,phi is the azimuth from the
%      positive x-axis,  $0 \leq \phi \leq 2\pi$ , and theta is the colatitude
%      from the positive z-axis,  $0 \leq \theta \leq \pi$ .  $D(:,1) = \theta$ ,  $D(:,2) = \phi$ ,
%       $D(:,3) = r$ .
%
%*****
n=length(a);

for i=1:n
    x=a(i);
    y=b(i);
    z=c(i);
    r=sqrt(x^2+y^2+z^2);

    % If the point lies on the positive z-axis
    if (abs(x)<= eps & abs(y) <=0 & abs(z-r)<=eps)
        theta=0;
        phi=0;
    % If the point lies on the negative z-axis
    elseif (abs(x)<=eps & abs(y)<=eps & abs(z+r)<=eps)

```

```

        theta=pi;
        phi=0;
% Cases for each octant
else
    % 1st octant
    if x>=0 & y>=0 & z>=0
        theta= acos(z/r);
        phi=acos(x/(r*sin(theta)));
    % 2nd octant
    elseif x<0 & y>=0 & z>=0
        theta=acos(z/r);
        phi=acos(x/(r*sin(theta)));
    % 3rd octant
    elseif x<0 & y<0 & z>=0
        theta=acos(z/r);
        phi=pi/2+acos(x/(r*sin(theta)));
    % 4th octant
    elseif x>=0 & y<0 & z>=0
        theta=acos(z/r);
        phi=3*pi/2+ acos(x/(r*sin(theta)));
    % 5th octant
    elseif x>=0 & y>=0 & z<0
        theta=acos(z/r);
        phi=acos(x/(r*sin(theta)));
    % 6th octant
    elseif x<0 & y>=0 & z<0
        theta=acos(z/r);
        phi=acos(x/(r*sin(theta)));
    % 7th octant
    elseif x<0 & y<0 & z<0
        theta=acos(z/r);
        phi=pi/2+acos(x/(r*sin(theta)));
    % 8th octant
    elseif x>=0 & y<0 & z<0
        theta=acos(z/r);
        phi=3*pi/2+acos(x/(r*sin(theta)));
    end
end

D(i,1)=theta;
D(i,2)=phi;
D(i,3)=r;
end

```

C.3 MATLAB Function to Adaptively Select θ and ϕ knots

```

function [rad_hat, c,NM, resid,knts_theta,knts_phi,iter]...
    = adaptive_knot_selection(theta,phi,rad_shift)

```

```

%*****
%ADAPTIVE_KNOT_SELECTION - Choose a good set of knots for the least squares
%                           fit of the scattered data.
%
%
%The selection algorithm is based on a suggested algorithm by P. Dierckx,
%Curve and Surface Fitting with Splines, Clarendon Press, Oxford, 2006.
%
%The object of the algorithm is to place knots in intervals where the
%residual errors are greatest.
%
%Input:
%  theta, phi - Vectors of spherical coordinates for the probe
%               points. theta and phi are in radians for theta in [0,pi]
%               and phi in [0,2pi]
%  rad_shift - rad_shift is in mm and is the radial data shifted by the
%               mean value
%
%Output:
%  rad_hat - Best fit radial values in mm. This is predicted values of
%             shifted data.
%  c       - Best fit B-spline basis function coefficients
%  resid   - Best fit residuals between rad_hat and rad_shift
%  knts_theta - Optimum theta knots
%  knts_phi  - Optimum phi knots
%  iter    - iteration counter
%
%Author:
%  David E. Gilsinn
%  Mathematical and Computational Sciences Division
%  Information Technology Laboratory
%  National Institute of Standards and Technology
%  Gaithersburg, MD 20899-8910
%
%*****
%Set a tolerance for computing the pseudo inverse for diagonal matrix S in
%the Singular Value Decomposition (SVD) solution of the least squares
%problem.
data_tol = 1e-14;
%set initial knots with 4 evenly spaced in theta and 8 in phi
knts_theta = [0.0 1.3 1.7 2.0 3 pi]'; %Specified knots for theta
knts_phi = [0 1.7 1.9 2.7 3.5 4 4.5 5.5 2*pi]'; %Specified knots for phi
%compute the fit for the initial data set and define the R_squared
%statistic
NM = tensor_prod_spl(theta,phi,knts_theta,knts_phi); %Get tensor product
c = lsq_2D_spline(NM,rad_shift,data_tol); %Do a least squares fit by SVD
%Get the initial residual
resid = abs(NM*c-rad_shift);

```

```

%Compute the initial predicted values
rad_hat = NM*c;
%Compute the initial R squared statistic (coefficient of determination)
rad_bar = mean(rad_shift);
R_square = (sum((rad_hat - rad_bar).^2))/(sum((rad_shift - rad_bar).^2))
%Compute the degrees of freedom
n_dat = length(rad_shift);
df = n_dat - length(c);
%Initialize iteration counter
iter = 1;
while (R_square < 0.98)&(df >= 1)
    iter = iter + 1; %Update iteration counter
    %Begin the automatic knot adjustment algorithm
    %Compute the sum of squares of the residuals of the fit to all points in
    %the interval (knts_theta(i),knts_theta(i+1))
    lkt = length(knts_theta);
    for i = 1:lkt-1
        t_index = find((knts_theta(i) <= theta) & (theta <= knts_theta(i+1)));
        Res = resid(t_index);
        del_t(i) = (norm(Res))^2;
    end
    %del_t
    u = find((del_t >= max(del_t))); %Get the maximum of the sums for theta
    %Compute the sum of squares of the residuals of the fit to all points in
    %the interval (knts_phi(i),knts_phi(i+1))
    lkp = length(knts_phi);
    for i = 1:lkp-1
        p_index = find((knts_phi(i) <= phi) & (phi <= knts_phi(i+1)));
        Res = resid(p_index);
        del_p(i) = (norm(Res))^2;
    end
    %del_p
    v = find((del_p >= max(del_p))); %Get the maximum of the sums for phi
    %Test whether the maximum residuals are with respect to theta or phi and
    %then form a weighted sum of the squares of the residuals. Update the knot
    %arrays.
    if del_t(u) >= del_p(v) %Enter here to select a new theta knot
        t_index = find((knts_theta(u) <= theta) & (theta <= knts_theta(u+1)));
        l_t_index = length(t_index);
        res_t_index = resid(t_index);
        theta_t_index = theta(t_index);
        del_u = del_t(u);
        new_t_knt = 0;
        for k = 1:l_t_index
            new_t_knt = new_t_knt + (res_t_index(k)^2)*...
(theta_t_index(k)/del_u);
        end
        knts_theta(lkt+1,1) = new_t_knt;
    end
end

```

```

        knts_theta = sort(knts_theta);
    else %Otherwise select a new phi knot
        p_index = find((knts_phi(v) <= phi) & (phi <= knts_phi(v+1)));
        l_p_index = length(p_index);
        res_p_index = resid(p_index);
        phi_p_index = phi(p_index);
        del_v = del_p(v);
        new_p_knt = 0;
        for k = 1:l_p_index
            new_p_knt = new_p_knt + (res_p_index(k)^2)*...
(phi_p_index(k)/del_v);
        end
        knts_phi(lkp+1) = new_p_knt;
        knts_phi = sort(knts_phi);
    end
    %Refit the original data with the new knots
    NM = tensor_prod_spl(theta,phi,knts_theta,knts_phi);
    c = lsq_2D_spline(NM,rad_shift,data_tol);
    resid = abs(NM*c-rad_shift); %Compute the current residual
    rad_hat = NM*c; %Compute the current predicted values
    %Compute the new R-squared statistic and degrees of freedom
    rad_bar = mean(rad_shift);
    R_square = (sum((rad_hat - rad_bar).^2))/(sum((rad_shift - rad_bar).^2))
    df = n_dat - length(c);
end

```

C.4 Function to Compute the Tensor Product of Two B-splines

```

function NM = tensor_prod_spl(theta,phi,knts_theta,knts_phi)
%*****
%TENSOR_PROD_SPL - This function creates the bicubic tensor product spline
%
%           matrix for the spherical coordinate data points in theta
%           and phi given specified knots in knts_theta and knts_phi
%
%Input:
%  theta, phi - Spherical coordinate angles for CMM data
%  knts_theta, knts_phi - theta and phi knots
%
%Output:
%  NM - Tensor product B-spline matrix
%
%Author:
%  David E. Gilsinn
%  Mathematical and Computational Sciences Division
%  Information Technology Laboratory
%  National Institute of Standards and Technology
%  Gaithersburg, MD 20899-8910
%

```

```

%*****
kstheta = augknt(knts_theta,4); %Augment the knots at both ends
%
%The next generates four nonzero values for the four cubic B-splines that
%overlap at the specified point. The output is a matrix with a row
%associated with the argument. This will be done one row at a time
lt = length(theta);
lp = length(phi);
for i = 1:lt
    t = theta(i,1);
    N(i,:) = spcol(kstheta,4,t);
end
[row_N,col_N] = size(N);
%
%Create the associated M array for each data point
ksphi = augknt(knts_phi,4); %Augment the knots at the beginning and end
for j = 1:lp
    p = phi(j,1);
    M(j,:) = spcol(ksphi,4,p);
end
[row_M,col_M] = size(M);
%Create the tensor product matrix
for r = 1:lt
    for i = 1:col_N
        for j = 1:col_M
            q = (i-1)*col_M + j;
            NM(r,q) = N(r,i)*M(r,j);
        end
    end
end
end

```

C.5 Least Squares by Singular Value Decomposition

```

function c = lsq_2D_spline(NM,rad_shift, data_tol)
%*****
%LSQ_2D_SPLINE - Given the tensor product spline matrix NM and the right
%                hand side data rad, this function produces the least
%                square solution of  $NM*c = rad$  with minimum norm for c
%                using the singular value decomposition. data_tol is the
%                uncertainty in the scattered data being approximated.
%
%Input:
%  NM - Tensor B-spline matrix
%  rad_shift - radial data shifted by the mean

% Compute the singular value decomposition of NM, "economy" style,
% dropping unneeded columns in U and V.
%Authors:

```

```

% David E. Gilsinn
% Mathematical and Computational Sciences Division
% Information Technology Laboratory
% National Institute of Standards and Technology
% Gaithersburg, MD 20899-8910
%
% Dianne P. O'Leary
% Department of Computer Science
% University of Maryland
% College Park, MD 20742
%*****
[U,S,V] = svd(NM,'econ');

%NM = U*S*V' and S is diagonal with nonnegative values [m,n] = size(S);
%Expect that ordinarily m > n

% Compute the pseudoinverse of S. It is a diagonal matrix
% with entries stored in the vector sinv.
% Elements of S less than data_tol are assumed to be zero.
[m,n] = size(S);
sdiag = diag(S);
sinv = zeros(n,1);
for i=1:n

    if (sdiag(i) < data_tol)
        break
    end

    sinv(i) = 1 / sdiag(i);

end

% Form the solution vector as the pseudoinverse of NM times rad_shift.

c = V*(sinv.*(U'*rad_shift));

```

C.6 Generating Data Points on a Surface Grid

```

function [t_grid, p_grid, rad1, N1M1]...
    = Generate_grid_data(c,nt_grid,np_grid,knts_theta,knts_phi,rad)
%*****
%GENERATE_GRID_DATA - This function generates a grid of theta/phi points
% and extrapolates the radial values to these points
% given the coefficient vector c, optimum knots, and
% the spherical coordinate data radial values, rad
%
%Input:
% c - The B-spline coefficients from the least squares fit to the

```

```

%      spherical coordinate data values
%      nt_grid - The number of desired theta grid values
%      np_grid - The number of desired phi grid values
%      knts_theta - The optimum theta knots
%      knts_phi - The optimum phi knots
%      rad - Unshifted radial data from the spherical coordinates
%
%Output:
%      t_grid - Theta values at the grid points along the theta coordinate
%      p_grid - Phi values at the grid points along the phi coordinate
%      rad1 - Extrapolated radial values at the (t_grid,p_grid) coordinates
%
%Author:
%      David E. Gilsinn
%      Mathematical and Computational Sciences Division
%      Information Technology Laboratory
%      National Institute of Standards and Technology
%      Gaithersburg, MD 20899-8910
%
%*****
%Select an evenly spaced grid of (theta, phi) pairs
t_grid = linspace(0,pi,nt_grid)';
p_grid = linspace(0,2*pi,np_grid)';

%Compute the tensor product matrix for the selected grid
lt = length(t_grid);
lp = length(p_grid);
kstheta = augknt(knts_theta,4); %Augment the knots at both ends
ksphi = augknt(knts_phi,4); %Augment the knots at the beginning and end
%Generate the B-spline tensor product at the grid points. Do it point at a
%time to take care of non-sorted arrays
for i = 1:lt
    t = t_grid(i);
    N1(i,:) = spcol(kstheta,4,t);
end
[row_N1,col_N1] = size(N1);
for j = 1:lp
    p = p_grid(j);
    M1(j,:) = spcol(ksphi,4,p);
end
[row_M1,col_M1] = size(M1);
for r1 = 1:row_N1
    for r2 = 1:row_M1
        r = (r1-1)*row_M1 + r2;
        for i = 1:col_N1
            for j = 1:col_M1
                q = (i-1)*col_M1 + j;
                N1M1(r,q) = N1(r1,i)*M1(r2,j);
            end
        end
    end
end

```

```

        end
    end
end
[r_n1m1,c_n1m1] = size(N1M1);

%Some important parameters of the unshifted radial data
mean_rad = mean(rad);
max_rad = max(rad);
min_rad = min(rad);

%Compute the new radii by adding back the mean of the original radial data
%rad1 = N1M1*c + mean_rad;
rad1 = N1M1*c;
%Adjust the values of rad1 values that fall outside of the original data
delta_plus = max_rad - mean_rad;
delta_minus = mean_rad - min_rad;
outlyer = find( (rad1 < mean_rad - delta_minus) | (rad1 > mean_rad + delta_plus));
rnd = rand(length(outlyer),1);
for i =1:length(outlyer)
    j = outlyer(i);
    if (rnd(i,1) <= 0.5)
        rad1(j) = mean_rad - rnd(i,1)*delta_minus;
    else
        rad1(j) = mean_rad + rnd(i,1)*delta_plus;
    end
end

```

C.7 Computing the Volume Based on Grid Values

```

function vol = volume(t_grid,p_grid,rad1)
%*****
%VOLUME - This function computes the volume of a phantom based on a
%          specified grid size using the Divergence Theorem. We accumulate
%          the facet contributions to the volume estimate of the phantom.
%
%Input:
%  t_grid - theta values at grid points
%  p_grid - phi values at grid points
%  rad1 - extrapolated radial values at grid points
%
%Output:
%  vol - volume in cubic millimeters
%
%Author:
%  David E. Gilsinn
%  Mathematical and Computational Sciences Division
%  Information Technology Laboratory
%  National Institute of Standards and Technology

```

```

% Gaithersburg, MD 20899-8910
%*****
lt = length(t_grid);
lp = length(p_grid);
%Transfer the rad1 array of estimated radii on a specified grid to a two
%dimensional array
nv = lt-1;
nh = lp-1;
for i = 1:nv+1
    for j = 1:nh+1
        p = (i-1)*(nh+1) + j;
        r(i,j) = rad1(p);
    end
end
r(1,1) = median(r(1,:));
r(nv+1,1) = median(r(nv+1,:));

%Initialize the volume
vol = 0;

%Loop through the layers of theta. Begin with the North Pole layer.
%Set up facet triangle vertices in counterclockwise manner. The top point
%will always be the north pole.
v1(1) = 0;
v1(2) = 0;
v1(3) = r(1,1);

%loop through each facet in the first layer
for j = 1:nh
    v2(1) = r(2,j)*sin(t_grid(2))*cos(p_grid(j));
    v2(2) = r(2,j)*sin(t_grid(2))*sin(p_grid(j));
    v2(3) = r(2,j)*cos(t_grid(2));
    v3(1) = r(2,j+1)*sin(t_grid(2))*cos(p_grid(j+1));
    v3(2) = r(2,j+1)*sin(t_grid(2))*sin(p_grid(j+1));
    v3(3) = r(2,j+1)*cos(t_grid(2));
    v = v2 - v1;
    w = v3 - v1;
    n = normal_vec(v,w);
    term = facet_term(n,v1,v2,v3);
    vol = vol + term;
end

%Next loop through layers 2 through nv-1. There are nh 4-vertex patches at
%each level. Each patch will be divided into two triangles. The four
%vertices of the patches will be identified in counterclockwise order as
%(t_grid(i),p_grid(j),r(i,j)),
%(t_grid(i+1),p_grid(j),r(i+1,j)), (t_grid(i+1),p_grid(j+1),r(i+1,j+1)),
%(t_grid(i),p_grid(j+1),r(i,j+1))

```

```

for i = 2:nv-1
    for j = 1:nh
        %Get contribution of left triangle
        v1(1) = r(i,j)*sin(t_grid(i))*cos(p_grid(j));
        v1(2) = r(i,j)*sin(t_grid(i))*sin(p_grid(j));
        v1(3) = r(i,j)*cos(t_grid(i));
        v2(1) = r(i+1,j)*sin(t_grid(i+1))*cos(p_grid(j));
        v2(2) = r(i+1,j)*sin(t_grid(i+1))*sin(p_grid(j));
        v2(3) = r(i+1,j)*cos(t_grid(i+1));
        v3(1) = r(i+1,j+1)*sin(t_grid(i+1))*cos(p_grid(j+1));
        v3(2) = r(i+1,j+1)*sin(t_grid(i+1))*sin(p_grid(j+1));
        v3(3) = r(i+1,j+1)*cos(t_grid(i+1));
        v = v2-v1;
        w = v3-v1;
        n = normal_vec(v,w);
        term = facet_term(n,v1,v2,v3);
        vol = vol + term;
        v2 = v3;
        v3(1) = r(i,j+1)*sin(t_grid(i))*cos(p_grid(j+1));
        v3(2) = r(i,j+1)*sin(t_grid(i))*sin(p_grid(j+1));
        v3(3) = r(i,j+1)*cos(t_grid(i));
        v = v2-v1;
        w = v3-v1;
        n = normal_vec(v,w);
        term = facet_term(n,v1,v2,v3);
        vol = vol + term;
    end
end
%Loop through the final layer at the South Pole
for j = 1:nh
    v1(1) = r(nv,j)*sin(t_grid(nv))*cos(p_grid(j));
    v1(2) = r(nv,j)*sin(t_grid(nv))*sin(p_grid(j));
    v1(3) = r(nv,j)*cos(t_grid(nv));
    v2(1) = 0;
    v2(2) = 0;
    v2(3) = -r(nv+1,1);
    v3(1) = r(nv,j+1)*sin(t_grid(nv))*cos(p_grid(j+1));
    v3(2) = r(nv,j+1)*sin(t_grid(nv))*sin(p_grid(j+1));
    v3(3) = r(nv,j+1)*cos(t_grid(nv));
    v = v2-v1;
    w = v3-v1;
    n = normal_vec(v,w);
    term = facet_term(n,v1,v2,v3);
    vol = vol + term;
end
%Get the final volume
vol = 1/6*vol;

```

```

function n = normal_vec(v,w)
%*****
%NORMAL_VEC - This function computes the unit normal vector n to the
%               plane determine by v and w in counterclockwise form. The
%               function computes the cross product of v and w using the
%               right hand rule between v and w.
%
%*****

n(1) = v(2)*w(3) - v(3)*w(2);
n(2) = v(3)*w(1) - v(1)*w(3);
n(3) = v(1)*w(2) - v(2)*w(1);
n = n/norm(n);

```

C.8 Computing the Sample Surface Triangular Mesh

```

function triangle_mesh
%*****
%TRIANGLE_MESH produces a sample triangular mesh on a sphere and plots the
%               results
%
%Author:
%   David E. Gilsinn
%   Mathematical and Computational Sciences Division
%   Information Technology Laboratory
%   National Institute of Standards and Technology
%   Gaithersburg, MD 20899-8910
%*****

r = 5;
theta = [0 45*pi/180 pi/2 135*pi/180 pi];
phi = [0 45*pi/180 pi/2 135*pi/180 pi 225*pi/180 270*pi/180 315*pi/180];
lt = length(theta);
lp = length(phi);
pt_index = zeros(lt,lp);
%Create point indices starting at North Pole
pt_index(1,1) = 1;
x(1,1) = 0;
y(1,1) = 0;
z(1,1) = r;
%Move down each theta from 2 to length(theta) -1
for i = 2:lt-1
    for j = 1:lp
        pt_index(i,j) = (i-2)*lp + j+1;
        x(pt_index(i,j),1) = r*sin(theta(i))*cos(phi(j));
        y(pt_index(i,j),1) = r*sin(theta(i))*sin(phi(j));
        z(pt_index(i,j),1) = r*cos(theta(i));
    end
end

```

```

end
%South pole
pt_index(lt,1) = (lt-2)*lp + 2;
pt_fin = (lt-2)*lp + 2;
x(pt_index(lt,1)) = 0;
y(pt_index(lt,1)) = 0;
z(pt_index(lt,1)) = -r;
plot3(x(1:pt_fin),y(1:pt_fin),z(1:pt_fin),'o','MarkerSize',4,'MarkerEdgeColor','black');
axis equal;
grid;
for k = 1:pt_fin
    text(x(k),y(k),z(k),num2str(k),'VerticalAlignment','bottom');
end
%Create the triangle mesh matrix
i = 1;
for j = 1:lp-1
    tri(j,1) = pt_index(1,1);
    tri(j,2) = pt_index(2,j);
    tri(j,3) = pt_index(2,j+1);
end
tri(lp,1) = pt_index(1,1);
tri(lp,2) = pt_index(2,lp);
tri(lp,3) = pt_index(2,1);
for i = 2:lt-2
    for j = 1:lp-1
        tri((2*i-3)*lp+2*j-1,1) = pt_index(i,j);
        tri((2*i-3)*lp+2*j-1,2) = pt_index(i+1,j);
        tri((2*i-3)*lp+2*j-1,3) = pt_index(i+1,j+1);
        tri((2*i-3)*lp+2*j,1) = pt_index(i,j);
        tri((2*i-3)*lp+2*j,2) = pt_index(i+1,j+1);
        tri((2*i-3)*lp+2*j,3) = pt_index(i,j+1);
    end
    j = lp;
    tri((2*i-3)*lp+2*j-1,1) = pt_index(i,j);
    tri((2*i-3)*lp+2*j-1,2) = pt_index(i+1,j);
    tri((2*i-3)*lp+2*j-1,3) = pt_index(i+1,1);
    tri((2*i-3)*lp+2*j,1) = pt_index(i,j);
    tri((2*i-3)*lp+2*j,2) = pt_index(i+1,1);
    tri((2*i-3)*lp+2*j,3) = pt_index(i,1);
end
i = lt-1;
for j = 1:lp-1
    tri((2*i-3)*lp+j,1) = pt_index(i,j);
    tri((2*i-3)*lp+j,2) = pt_index(i+1,1);
    tri((2*i-3)*lp+j,3) = pt_index(i,j+1);
end
j = lp;
tri((2*i-3)*lp+j,1) = pt_index(i,j);

```

```
tri((2*i-3)*1p+j,2) = pt_index(i+1,1);
tri((2*i-3)*1p+j,3) = pt_index(i,1);
hold on
trimesh(tri,x(1:pt_fin),y(1:pt_fin),z(1:pt_fin), 'EdgeColor','black');
axis equal
grid on
title('Sample Surface Triangulation');
xlabel('X');
ylabel('Y');
zlabel('Z');
hold off
```