

# Multivariate Density Estimation for Massive Datasets via Sequential Convex Hull Peeling

James P. McDermott

Bristol-Myers Squibb, Wallingford, CT 06492

Dennis K. J. Lin

Department of Supply Chain and Information Systems,

The Pennsylvania State University, University Park, PA 16802

## Abstract

We propose a low-storage, single-pass, sequential method for the execution of multivariate density estimation via convex hull peeling for massive datasets. The method is shown to vastly reduce the computation time required for the existing convex hull peeling algorithm from  $O(n^2)$  to  $O(n)$ . Further, the proposed method uses very low-storage as compared to the existing method. We demonstrate the accuracy and reduced computation time required of the proposed method by comparing to the existing convex hull peeling method through simulation studies and a real life example.

## 1 Introduction

Massive datasets are becoming more and more common in modern society. They arise from sources as diverse as large call centers, internet traffic data, sales transactional records, or satellite feeds. This phenomenon presents a clear need to be able to process the data accurately and efficiently so that current analyses may be performed before becoming inundated by a continually growing store of data.

In this paper, we are specifically interested in multivariate density estimation for massive datasets. We believe that the density function is essence of many statistical analysis. Once the the distribution (density) of the response variables can be specified, many statistical inferences can be straightforwardly obtained. Many density estimation methods have been studied. It should be clear that the conventional wisdom on density estimation is infeasible for massive datasets, mainly due to the complexity of computer memory and computational costs. Specifically, the denisty function is formulated here via quantiles. We thus need to address the multivariate quantile estimation for massive datasets.

In one dimension, the concept of a *quantile* is well defined. We define the  $p^{\text{th}}$  *quantile* as  $\xi_p = F^{-1}(p) = \inf\{x : F(x) \leq p\}$ , for  $0 < p < 1$ . In more than one dimension, however, there is no universally accepted notion of a quantile. Indeed, there is no universal concept of ordering in more than one dimension. Hence alternate definitions are required. One approach is to assign each point in a dataset a measure of *depth*. That is, for each point we have a measure of how far it is from some concept of the *center* of the dataset. Liu, Parelius, and Singh (1999) give various examples of different definitions of data depth, such as *Mahalanobis depth* (Mahalanobis (1936)), *half-space depth* (Hodges (1955), Tukey (1975)), *convex hull peeling depth* (Barnett (1976)), *Oja depth* (Oja (1983)), *simplicial depth* (Liu (1990)), and *likelihood depth* (Meloche and Fraiman (1999)).

Of these methods, we will focus on the convex hull peeling depth because of its ability to be adapted to the sequential low-storage form that we require. Although there are different versions of the convex hull peeling exist (see Eddy (1982) and Green (1981) for a variation given by Tukey (1975)), we will focus on the simple convex hull peeling version given by Barnett (1976). We will follow the definitions and notation in Liu et al. (1999).

1. *The depth of a point  $x \in \mathbb{R}^d$  is denoted by  $D(x) = t$ .*
2. *The set  $\{x \in \mathbb{R}^d : D(x) = t\}$  is called the contour of depth  $t$ .*
3. *The set  $\{x \in \mathbb{R}^d : D(x) > t\}$  is referred to as the region enclosed by the contour of depth  $t$ , and is denoted by  $R(t)$ .*

4. The set  $C_p = \bigcap_t \{R(t) : P_F(R(t)) \geq p\}$  is referred to as the  $p^{\text{th}}$  central region and it is the smallest region enclosed by depth contours to amass to probability  $p$ .
5. The convex hull peeling depth of a point  $x$  is defined in terms of the level of the depth contour to which it belongs, and the level of a depth contour,  $C_p$ , is defined as the proportion,  $p$ , of the population that is contained within  $C_p$ . We will denote the sample depth contour of level  $p$  by  $\hat{C}_p$ , and we define  $\hat{C}_p$  as the convex hull that contains approximately the proportion  $p$  of the sample.

Consider the set of points that all have the same depth. We then define the *depth contour of level  $p$*  as the boundary of the central region that contains  $p * 100\%$  of the population. By definition, we assign the point center of the distribution a depth of 0, thereby giving the center a maximum depth of 1. We note here that the convex hull of a set of points is invariant to all affine transformations of the data.

In one dimension, this reduces to a central interval of quantiles. For example, the one-dimensional depth contour of level  $p = .5$  is just the interquartile range. That is, it is the central interval that contains 50% of the population. We define the *depth* of a point  $\mathbf{x}$  as  $\text{depth}(\mathbf{x}) = D(\mathbf{x}) = 1 - p_{\mathbf{x}}$ , where  $p_{\mathbf{x}}$  is the level of the depth contour to which  $\mathbf{x}$  belongs.

In two dimensions, we get a central region determined by the *shape* of the underlying distribution we are considering. For example, the standard bivariate normal distribution gives us a central circle, with center at the origin and with radius uniquely determined by the level  $p$  as

$$\text{radius } r = \sqrt{-2 \ln(1 - p)}. \quad (1)$$

This result is derived as follows. We begin by considering integrating the density of the standard bivariate normal in polar coordinates, setting this equal to  $p$ , the proportion of the population within the depth contour of level  $p$ , and solving for the radius, in this case denoted by  $a$ . Hence we have

$$\begin{aligned} \int_0^{2\pi} \int_0^a f(r, \theta) r dr d\theta &= \int_0^{2\pi} \int_0^a \frac{1}{2\pi} e^{-((r \cos \theta)^2 + (r \sin \theta)^2)/2} r dr d\theta \\ &= \frac{1}{2\pi} \int_0^{2\pi} \int_0^a r e^{-r^2/2} dr d\theta \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2\pi} \int_0^{2\pi} \left[ -e^{-r^2/2} \right]_0^a d\theta \\
&= \frac{1}{2\pi} \int_0^{2\pi} \left( 1 - e^{-a^2/2} \right) d\theta \\
&= 1 - e^{-a^2/2}.
\end{aligned} \tag{2}$$

Now setting the result in (2) equal to  $p$  and solving for  $a$  gives us the result in (1).

The convex hull of a set of points is the minimum convex set that contains the entire dataset. In other words, it is the set of points that make up the outermost perimeter of the dataset where the polygon formed by connecting these points contains the entire dataset. Convex hull peeling is achieved by systematically identifying and then deleting the set of points that make up the convex hull of the set. For example, if we had a set of 100 points and we identified the set of, say 10, points that make up the convex hull, we would then delete these 10 points from the dataset and have 90 remaining. We would now repeat the procedure by finding the set of points that make up the convex hull of the remaining 90 points and again deleting them.

To find the *convex hull peel  $p^{\text{th}}$  depth contour*, we would peel the dataset consisting of  $n$  observations until approximately  $np$  observations are remaining. We say approximately here because you will rarely have exactly  $np$  points remaining at any given stage and further,  $np$  may not even be an integer. Chazelle (1985) gives an optimal algorithm for determining the convex layers of a set of  $n$  planar points. However, it requires the entire dataset to be stored as this method finds the convex layer to which each point belongs. There are two main problems with these existing methods. The first problem is that we have to physically store the entire dataset to execute the algorithms. The second problem is one of computation time. The time required to execute the existing methods grows at an exponential rate as the sample size increases. This presents an obvious problem for application to massive datasets.

Massive datasets are becoming more and more pervasive in our society. Existing methods are not adequate for dealing with these large datasets now being generated. We are interested in multivariate density estimation via finding the convex hull (or contour) for massive datasets. Further, we require that the storage required be very low relative to the entire sample size and that it be computed sequentially so that if additional data arrives,

processing may be picked up where it left off. This paper is organized as follows. In Section 2, we propose new methods for the sequential computation of both the convex hull peeling median and  $p^{\text{th}}$  depth contour. We present the results of various simulation studies in Section 3. In Section 4, we present an application of our proposed convex hull peeling algorithms to bivariate density estimation. We discuss the use of thin plate splines in Section 5 as a method of fitting a multivariate surface to our bivariate density estimates and in Section 6 we present an application to a real dataset from astronomy. Concluding remarks are given in Section 7.

## 2 Proposed Method

### 2.1 Sequential Convex Hull Peeling - Multivariate Median

We have defined the *convex hull peeling multivariate median* as the centroid of the innermost hull obtained by successively peeling away the outermost convex hull layers. As noted above, the traditional method of obtaining this multivariate median is computationally intensive and imposes an extreme storage burden. The alternative we propose solves both of these problems by only storing a very small number of points at any one time and attacking the massive dataset with a “divide-and-conquer” strategy that yields a computational complexity that is linear in  $n$ , the sample size.

*Overview of Sequential Convex Hull Peeling Algorithm for Estimation of the Multivariate Median*

1. Take the first  $m$  points from the dataset and peel the layers until approximately  $md$  points are left, where  $d$  represents the *degree of intermediate peeling* and  $0 < d < 1$ ;
2. Add enough points from the remaining dataset to bring the number up to  $m$  again;
3. Repeat until the dataset is exhausted;
4. Peel the final set of points down to the innermost hull;

5. The “center” of this resulting hull is taken as an estimate of the multivariate median.

The *degree of intermediate peeling*,  $d$ , is the percentage of the  $m$  points that one must peel down to at each intermediate step 1 of the above algorithm. For example, if  $d = .1$  and  $m = 1000$ , then we must peel every set of 1000 points down as close as we can to 100 points at each step. In practice, we take  $d$  to be .5, although the effect of different values of  $d$  will also be considered in a later Section. By “center” we mean the centroid of the final hull. In practice, we take  $m$  to be between 1,000 and 10,000 as this range of values has proven to work well in all situations considered. It should be noted that one could take values of  $m$  larger than 10,000, but one must weigh the improvement in performance against the increase in computation time. We discuss the implications of taking different values of  $m$  in a later Section.

## 2.2 Sequential Convex Hull Peeling - Depth Contours

We have defined the *convex hull peeling  $p^{\text{th}}$  depth contour* as the convex hull obtained by successively peeling away the outermost convex hull layers until approximately  $np$  points are within the perimeter of the hull and approximately  $n(1 - p)$  points are outside of the perimeter. The algorithm given in Section 2 can be modified for estimation of these depth contours. We now present a sequential algorithm to sequentially compute and maintain the convex hull peeling depth contour for a given value of  $p$ , for  $0 < p < 1$ .

*Overview of Sequential Convex Hull Peeling Algorithm for Estimation of the  $p^{\text{th}}$  Depth Contour*

1. Take the first  $m$  points from the dataset and peel the layers until approximately  $mp$  points are left, where  $p$  represents the proportion associated with the desired depth contour.
2. Store the  $b_1$  points representing this convex hull.
3. Repeat  $k$  times, each time appending the  $b_i$  points from the new depth contour to the previous set of points.

4. Peel the resulting set of  $\sum_{i=1}^k b_i$  points until approximately  $\frac{1}{2} \sum_{i=1}^k b_i$  are remaining.
5. This final hull is the estimate of the  $p^{th}$  depth contour.

In Figure 1, we give an example of the output of the sequential depth contour algorithm. The black lines are the  $k$  contours obtained from steps 1 through 3 of the algorithm. We note that this collection of black lines is really just a set of points which will then be peeled in step 4. The white line in the center of the black lines is the depth contour obtained after completing step 4 of the algorithm.

[PUT FIGURE 1 ABOUT HERE]

In Figure 2, we give an example of a set of depth contours for a dataset of 1,000,000 observations drawn from a bivariate normal distribution with zero mean vector and with variances of 1 and a covariance of 3. We have used our proposed algorithms to compute the .01, .05, .1, .2, .3, .4, .5, .6, .7, .8, .9, .95, and .99 depth contours and the convex hull peeling median. In Figure 2, the centralmost point is the estimate of the median and the contours from the center moving out are the  $p = .99$  through  $p = .01$  depth contours. Again, a level of  $p$  for a given depth contour means that approximately  $(1 - p) * 100\%$  of the data is within it.

[PUT FIGURE 2 ABOUT HERE]

## 2.3 Theoretical Considerations

Efron (1965) gives integral formulae for the computation of the expected number of vertices for the convex hull of a finite set of points drawn at random from a normal or a uniform distribution in 2 or 3 dimensions. For example, the expected number of vertices comprising the convex hull of a random set of points from a standard bivariate normal distribution is

$$E(V_n) = 4\sqrt{\pi} \binom{n}{2} \int_{-\infty}^{\infty} \Phi^{n-2}(p) \phi^2(p) dp,$$

where  $\Phi$  and  $\phi$  are the cumulative distribution function and density respectively of the standard univariate normal. The expected number of vertices grows very slowly as a function of sample size. For example, at 1,000,000 the expected number of vertices is  $\approx 17$  and even

at 1,000,000,000 is only  $\approx 22$ . The result of this on the convex hull peeling algorithm is that, for very large datasets, it takes a very long time to peel away the outer layers because so few are peeled at each layer. For example, if we have a dataset of size 1,000,000, the expected number of vertices in the convex hull of this set is  $17.24 \approx 17$ . Hence after peeling away this layer we still have  $\approx 1,000,000 - 17 = 999,983$  points remaining.

Similarly, for the three dimensional normal distribution we see the same behavior (as given in Figure 3). Here the expected number of vertices for a random sample of size  $n$  is given by

$$2 + 2\sqrt{3\pi} \binom{n}{3} \int_{-\infty}^{\infty} \Phi^{n-3}(p) \phi^3(p) dp.$$

It follows that for a random sample of size 1,000,000 drawn from this distribution, the expected number of vertices in the convex hull is  $92.96 \approx 93$  and even when the size reaches 1,000,000,000 the expected number of vertices is still only  $147.77 \approx 148$ . Thus we see that the same problem exists in higher dimensions as well.

[PUT FIGURE 3 ABOUT HERE]

### 3 Simulation Studies

#### 3.1 Median Simulation Studies

For the median simulation studies, we will compare the sequential convex hull peeling median to the traditional convex hull peeling median. The data for the simulations will be drawn at random from the standard bivariate normal distribution. Our comparisons will include the mean distance of each method's estimates from the origin, the variances and the ratio of the variances of the two method's estimates of the distance from the origin, and a time comparison by looking at the average time required for each method. We look at different values of  $m$ ,  $d$ , and  $n$ , the total sample size to see if different values of these parameters will have an effect on the estimator's performance. We take  $m = 1000, 2000, 5000$ , and  $10000$ ,  $d = .1$  and  $.5$ , and  $n = 100000$  and  $1000000$ . The results of these studies are given in Table 1.

For the median studies, the performance differences between the two methods are very small. As seen in Table 1, the traditional convex hull peeling method consistently outperforms the sequential version, however the differences are rather insignificant. We note that as the sample size,  $n$ , increases, the performance of both methods improves. Also, for a fixed sample size, as  $m$ , the maximum number of points stored, increases from 1000 to 10000 the performance of the sequential version remains relatively constant. We now consider the difference in computation time between the existing convex hull peeling median estimation method and our proposed sequential median estimation method. While changing the value of  $m$  has little or no effect on the performance of the sequential estimator, it does have some impact on the computation time. For example, in Table 1(a), when  $n = 50000$ , as  $m$  goes from 1000 to 2000 to 5000 to 10000, the average computation time goes from 2.05 to 2.27 to 2.46 to 3.74, respectively.

[PUT TABLE 1 ABOUT HERE]

In Figure 4, we present the results of a separate time comparison study. We plot the average time taken to execute the existing method and the proposed method for the convex hull peel multivariate median for samples of differing sizes drawn from the standard bivariate normal distribution. Note that the time for the existing method grows exponentially as the sample size  $n$  increases. In contrast, the computation time required for the execution of our proposed method is linear as a function of  $n$ . Additional time comparisons of the two median estimation methods are presented in Table 1 where again the same exponential versus linear computation time is observed. For example, in Table 1(a), when  $m = 1000$ , the average computation time for  $n = 5000$  is 0.29 and 0.21 for the two methods. There is not much difference at this point as the ratio of these two average times is 1.38. However, when the sample size reaches 100000, the average computation times are 40.7 and 5.18 with the ratio of these two average time being 7.88. Based upon experimentation and these simulation results, we make a recommendation of setting  $d$ , the depth of the intermediate peels, to .1 and of setting  $m$  to 10000. These settings should give slightly better accuracy and a reduction in computation time.

[PUT FIGURE 4 ABOUT HERE]

## 3.2 Depth Contour Simulation Studies

For the depth contour simulation studies, we will compare the sequential convex hull depth contour to the traditional convex hull depth contour. The data for the simulations will again be drawn at random from the bivariate standard normal distribution. Since for this study we will not have point estimates as output, but rather a collection of points in the form of a convex polygon, we will use different measures to compare the method's performance.

As mentioned in the introduction, for the standard bivariate normal the theoretical  $p^{\text{th}}$  depth contour,  $C_p$ , will be a circle with radius  $r = \sqrt{-2 \ln(1-p)}$ . Hence, to evaluate how closely the convex polygon,  $\hat{C}_p$ , determined by the set of points obtained from one of the two algorithms approximates this circle, we will convert the points output by the algorithm to polar coordinates. We will then compare how far each point is from the point on the circle with the same polar coordinate  $\theta$ . That is, for a given point  $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ , we calculate the angular polar coordinate for  $\mathbf{x} = (x_1, x_2)$  as  $\theta = \arctan\left(\frac{x_2}{x_1}\right)$ . We then calculate the straight-line distance between  $\mathbf{x} = (x_1, x_2)$  and the point on the circle,  $(r \cos \theta, r \sin \theta)$ . We do this for every point given as output for a given contour estimate and take the total of these errors. Since not all contours will have the same number of points, we will also take the average of these errors. Additionally, we will compare the area of the polygon,  $\hat{C}_p$ , with the area of the circle,  $C_p$ . This is a cross check against the total and average error measures because if there were very few points making up the polygon, then these error measures could potentially be very small but the area most likely wouldn't be close to that of the circle,  $C_p$ . As with the median simulation study, we will also conduct an average computation time comparison. The results from this study are given in Table 2.

[PUT TABLE 2 ABOUT HERE]

In these studies, we examine three different contours:  $p = .9, .5$ , and  $.1$ . We also vary the value of  $m$  between 5000 and 10000. A final variation is in the sample sizes. Table 2(a) sets  $n = 100000$  and Table 2(b) sets  $n = 1000000$ . Again we see similar results as in the median study. As  $m$  increases, we see improved performance with a commensurate increase in computation time. As  $n$  increases, we see the same behavior. As  $p$  goes from  $.9$  to  $.1$ , and hence from more outlying contours to more central ones, we see reduced variability in

the contours. Again, as with the median algorithm, we make a recommendation of setting  $m = 10000$  for the sequential depth contour algorithm.

## 4 Bivariate Density Estimation

We now consider estimation of the bivariate density based upon the computation of expected volumes under the density surface by utilizing the depth contours discussed above. We begin with the observation that, since the  $p^{th}$  depth contour contains  $p * 100\%$  of the population, if we could integrate the unknown density over the entire region of this depth contour it would integrate to  $p$ . That is, it would integrate to the total volume under the density and enclosed by the contour. Hence the total volume under the surface of the density outside of the contour would integrate to  $1 - p$ . Using this fact, we will then solve for the heights, or values of the density, of the points in each contour by beginning at the lowest contour and “building” our way up.

As we step through this algorithm, it will be helpful to refer to Figure 5 as a reference. This figure shows a cross-section of the lower part of a density and we use it to demonstrate our density estimation procedure. Our first volume computation begins at the bottom of the density. We take the difference of the areas for the convex hull of the entire dataset, which we denote by  $C.inf$ , and the most outlying depth contour, which in our example is the  $.999^{th}$  depth contour and is denoted by  $C.999$ . This difference gives us the total area between these two hulls, which we denote by  $A_1$ . Now, since we know the total volume under the density and *inside* of  $C.999$  is  $.999$ , we know that the volume under the density and *outside* of  $C.999$  is  $1 - .999 = .001$ . Hence we will set the volume of the section under the density and outside  $C.999$  equal to  $.001$  and solve for the height between the two contours. We denote this first volume by  $V_1$  and this first height by  $h_1$ . We approximate the curve of the outer surface of the density by a straight line. Note that in the cross-section in Figure 5 the volume  $V_1$  is approximated by a triangle. Hence we have,  $V_1 = \frac{1}{2} (\text{Area}(C.inf) - \text{Area}(C.999))$   $h_1 = \frac{1}{2} A_1 h_1 = .001$ , and so  $h_1 = \frac{2(.001)}{A_1}$ .

[PUT FIGURE 5 ABOUT HERE]

We must next compute  $h_2$ , the height from  $C.999$  to  $C.995$ . We now use the volume  $V_2$  to again solve for  $h_2$ . The volume under the surface of the density and contained within  $C.995$  is  $.995$  and the volume outside  $C.995$  is  $1 - .995 = .005$ . We also know the volume  $V_1 = .001$ . A further known quantity is the volume of the “rectangle-donut” beneath  $V_2$  and beside  $V_1$ . We refer to this as a “rectangle-donut” because we sweep the rectangle around the circumference of  $C.999$ . The volume of this piece is just  $(C.inf - C.999) h_1$ , that is, it is the product of the difference in the areas of  $C.inf$  and  $C.999$  and the just computed height,  $h_1$ . Hence, we have

$$\begin{aligned} .005 &= .001 + V_2 + (\text{Area}(C.999) - \text{Area}(C.995)) h_1 \\ &= .001 + \frac{1}{2} (\text{Area}(C.999) - \text{Area}(C.995)) h_2 + (\text{Area}(C.999) - \text{Area}(C.995)) h_1 \\ &= .001 + \frac{1}{2} A_2 h_2 + A_2 h_1, \end{aligned}$$

and so  $h_2 = \frac{2(.004 - A_2 h_1)}{A_2}$ . Continuing in this manner, we may determine all of the heights,  $h_i$ . Then to obtain the density estimate for a given contour, we add up the heights until we reach the desired contour. For example, in Figure 5, to obtain the density estimate for points in the  $C.995$  contour, we add the first two heights. So,  $h_1 + h_2$  would be our density estimate for all points that make up this depth contour.

We now generalize this procedure. Assume we have  $0 < p_k < \dots < p_2 < p_1 < 1$  and corresponding depth contours,  $C_{p_k}, \dots, C_{p_2}, C_{p_1}$  and define  $C_{inf}$  to be the convex hull of the entire dataset. Let  $A_1 = \text{Area}(C_{inf}) - \text{Area}(C_{p_1})$  and  $A_i = \text{Area}(C_{p_1}) - \text{Area}(C_{p_2})$ , for  $i = 2, \dots, k$ . Then we have

$$\begin{aligned} h_1 &= \frac{2(1 - p_1)}{A_1}, \\ h_2 &= \frac{2((p_1 - p_2) - A_2 h_1)}{A_2}, \end{aligned}$$

and in general

$$h_i = \frac{2((p_{i-1} - p_i) - A_i (h_1 + \dots + h_{i-1}))}{A_i},$$

for  $i = 3, \dots, k$ .

In Table 3, we present the results of a simulation study to quantify the performance of our proposed bivariate density estimator. We have generated 1,000,000 observations from

the bivariate standard normal distribution and computed the .999, .995, .99, .95, .9, .8, .7, .6, .5, .4, .3, .2, .1, .05, .01, .005, and .001 depth contours. Recall that these contours as listed go from outermost to innermost since the .999<sup>th</sup> contour contains 99.9% of the data and the .001<sup>th</sup> depth contour contains 0.1% of the data. We ran the simulation for  $m = 10,000$  and averaged results over 100 iterations. We then compare the averaged estimates to the true density value for the points in each contour and we also examine the variance of the 100 estimates. One can see that the results are quite accurate with the .999<sup>th</sup> contour having the least accurate estimation. For example, with  $m = 10,000$ , the .001<sup>th</sup> contour averaged estimate was 0.00009 compared to the true density value of 0.00016, while the .999<sup>th</sup> contour averaged estimate was 0.16623 compared to a true density value of 0.15900.

[PUT TABLE 3 ABOUT HERE]

## 5 Multivariate Thin Plate Splines

Having thus obtained density estimates for all of our depth contours, we may then wish to fit a multivariate spline to our results. In Figure 6, we show an example of a set of contours obtained from a sample of 1,000,000 observations taken from the standard bivariate normal distribution. We then assign heights, obtained by the above bivariate density estimation method, to these two-dimensional contours. We plot the resulting three-dimensional points in Figure 7 and in Figure 8 we present the result of a multivariate spline fit to the three-dimensional set of points made up by the points in the contours and their associated estimated density values. The thin plate spline used here is essentially the one proposed in Green and Silverman (1994), as briefly summarized below.

[PUT FIGURES 6, 7 AND 8 ABOUT HERE]

Assume that we have points  $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n \in \mathbb{R}^2$ , where  $\mathbf{t}_i = (x_1, x_2)$ . We define a function  $\eta(r)$  as  $\eta(r) = \frac{1}{16\pi} r^2 \log r^2$ , for  $r > 0$ , and  $\eta(0) = 0$ ; and the functions,  $\phi_1(x_1, x_2) = 1$ ,  $\phi_2(x_1, x_2) = x_1$ ,  $\phi_3(x_1, x_2) = x_2$ , and finally define a  $3 \times n$  matrix  $T$  with elements  $T_{jk} = \phi_j(\mathbf{t}_k)$ , so that  $T = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_n \end{bmatrix}$ . The thin plate splines can then be defined as

follows (see Green and Silverman, 1994)).

**Definition 1** *A function  $g(\mathbf{t})$  is a thin plate spline on  $\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_n$  if and only if  $g$  is of the form*

$$g(\mathbf{t}) = \sum_{i=1}^n \delta_i \eta(\|\mathbf{t} - \mathbf{t}_i\|) + \sum_{j=1}^3 a_j \phi_j(\mathbf{t})$$

for suitable constants  $\delta_i$  and  $a_j$ .

Then if we have observed data,  $Y_i$  at the points  $\mathbf{t}_i$ , we define the penalized residual sum of squares of a surface  $g$  by  $S(g) = \sum_i \{Y_i - g(\mathbf{t}_i)\}^2 + \alpha J(g)$ , where  $\alpha$  is a smoothing parameter and  $J(g) = \iint_{\mathbb{R}^2} \left\{ \left( \frac{\partial^2 g}{\partial x^2} \right)^2 + 2 \left( \frac{\partial^2 g}{\partial x \partial y} \right)^2 + \left( \frac{\partial^2 g}{\partial y^2} \right)^2 \right\} dx dy$  is a roughness penalty. In our application, we have, say  $r$ , points that make up the depth contours. So we take the two-dimensional coordinates of these points as our  $\mathbf{t}_i$  and the density estimates at these points as our values of  $Y_i$ .

## 6 Application to Astronomy Dataset

We next apply our methods to an astronomy dataset as an illustration and comparison. The Digital Palomar Observatory Sky Survey (DPOSS) contains over 50 million galaxies and over 2 billion stars will be contained in the final sky catalog (Djorgovski et al. (2002)). We will be looking at a measure of the magnitude of these sky objects in the blue-green and near infra-red visible bands. A set of 11355 observations will be used here for comparison purposes. Note that our method can be applied to datasets of any size while the traditional methods cannot.

Figure 9 presents a scatterplot of the DPOSS dataset used in this example. In Figure 10, we give the results of executing both the traditional and sequential convex hull peeling algorithms on the DPOSS dataset as a side by side graphical comparison. We compute the multivariate medians, which are represented by a central point, and the sets of depth contours ranging from the centralmost contour with  $p = .1$  out to the farthest contour with  $p = .9$ . Despite utilizing only 11355 data points, visually there is not much difference between the two graphs. The outermost contour in Figure 10 is the convex hull of the entire dataset.

This is given as a visual reference. We note here that the outermost convex hull of the entire dataset may easily be computed sequentially by taking  $m$  points at a time, taking the convex hull of these points, and then taking the convex hull of the union of this hull and the next  $m$  points. Continuing in this manner we may keep a sequential version of the outermost hull and this will be exactly the same as if we had stored the entire dataset and taken the convex hull. We quantify the difference between the areas of depth contours for the two methods as follows from the outermost contour to the innermost: 0.6603, 0.3690, 0.3364, 0.2541, 0.0791, 0.0736, 0.0488, 0.0128, 0.0593. We note that the largest error occurred with the outermost contour with  $p = .9$ . This agrees with the simulation results which show greater variability in the more outlying contours. The distance between the traditional and sequential multivariate medians was 0.0258. Figure 11 gives the thin plate spline fit to the bivariate density estimates as discussed in Section 5.

[PUT FIGURES 9, 10 AND 11 ABOUT HERE]

## 7 Concluding Remarks

One drawback to the sequential depth contour algorithm is that, although it uses much less storage than the traditional method, it still requires perhaps more storage than we would like. Experiments have shown that approximately 1%-2% of the dataset must be stored to execute the algorithm. There is room for improvement in this area. Another thing to note is that at this time the depth contours must be computed separately. Hence, to avoid having to make more than one pass through the data, one would have to program the algorithm in parallel with each individual contour being computed by a separate processor.

By utilizing the contours and recognizing their relationship to expected volumes under the surface of the unknown density, we are able to obtain bivariate density estimates for all points on each contour without making any choice of bandwidth as we must do with kernel density estimation. We may then fit a multivariate spline to the resulting three dimensional set of points to obtain the functional form of the entire density.

The convex hull of a random set of points in two dimensions is a convex polygon. In

three dimensions, the convex hull will be a *polyhedron*, a region in three-dimensional space whose boundary is made up of a finite number of two-dimensional polygon *faces*. Any two of these faces will be disjoint and meet at *edges* and *vertices*. O'Rourke (1998) gives algorithms for the computation of the convex hull in three dimensions.

The code for these methods was written exclusively with the R programming language as the basic algorithm for finding the convex hull of a planar set of points (Eddy (1977)) is part of the base R code.

### Acknowledgements

Dennis Lin's research was partially supported by *National Security Agency*, via Grant MDA904-02-1-0054.

## References

- [1] Barnett, V. (1976). "The Ordering of Multivariate Data," *Journal of the Royal Statistical Society A*, **139**, 3, 318-344.
- [2] Chazelle, B. (1985). "On the Convex Layers of a Planar Set," *IEEE Transactions on Information Theory*, IT31, 509-517.
- [3] Djorgovski, S.G., Gal, R.R., de Carvalho, R.R., Odewahn, S.C., Mahabal, A.A., Brunner, R., Lopes, P., and the DPOSS Team (2003). "The Digital Palomar Observatory Sky Survey (DPOSS): General Description and the Public Data Release," *Bulletin of the American Astronomical Society*, **34**, 743.
- [4] Eddy, W.F. (1977a). "A New Convex Hull Algorithm for Planar Sets," *ACM Transactions on Mathematical Software*, **3**, 4, 398-403.
- [5] Eddy W.F. (1977b). "Algorithm 523: CONVEX, A New Convex Hull Algorithm for Planar Sets," *ACM Transactions on Mathematical Software*, **3**, 4, 411-412.
- [6] Eddy, W.F. (1982). "Convex Hull Peeling," in COMPSTAT (H. Caussinus et al., eds.), 42-47, Physica, Veinna.

- [7] Efron, B. (1965). "The Convex Hull of a Random Set of Points," *Biometrika*, **52**, 3 and 4, 331-343.
- [8] Green, P.J. (1981). "Peeling Bivariate Data," In *Interpreting Multivariate Data*, Ed. V. Barnett, pp. 3-20. New York, Wiley.
- [9] Hodges, J. (1955). "A Bivariate Sign Test," *Annals of Mathematical Statistics*, **26**, 523-527.
- [10] Liu, R.Y., Parelius, J.M., and Singh, K. (1999). "Multivariate Analysis by Data Depth: Descriptive Statistics, Graphics and Inference," *Annals of Statistics*, **27**, 3, 783-858.
- [11] Mahalanobis, P.C. (1936). "On the Generalized Distance in Statistics," *Proceedings of the National Academy of Science of India*, **12**, 49-55.
- [12] Meloche, J., Fraiman, R. (1999). "Multivariate  $L$ -estimation (with discussion)," *Test*, Vol 8, 255-317.
- [13] O'Rourke, J. (1998). *Computational Geometry in C*. Cambridge, Cambridge University Press.
- [14] Small, C.G. (1990). "A Survey of Multidimensional Medians," *International Statistical Review*, **58**, 3, 263-277.
- [15] Tukey, J. (1975). "Mathematics and Picturing Data," In *Proceedings of the 1975 International Congress of Mathematics*, **2**, 523-531.

Figure 1: Example of quantile contour algorithm output

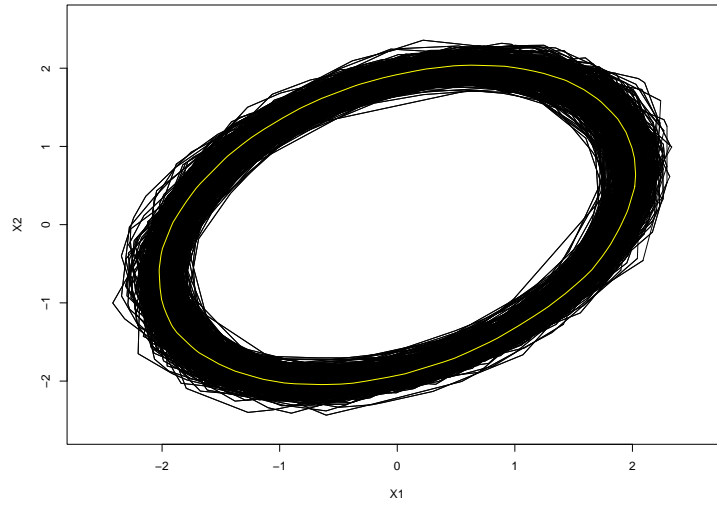


Figure 2: Example of a set of depth contours

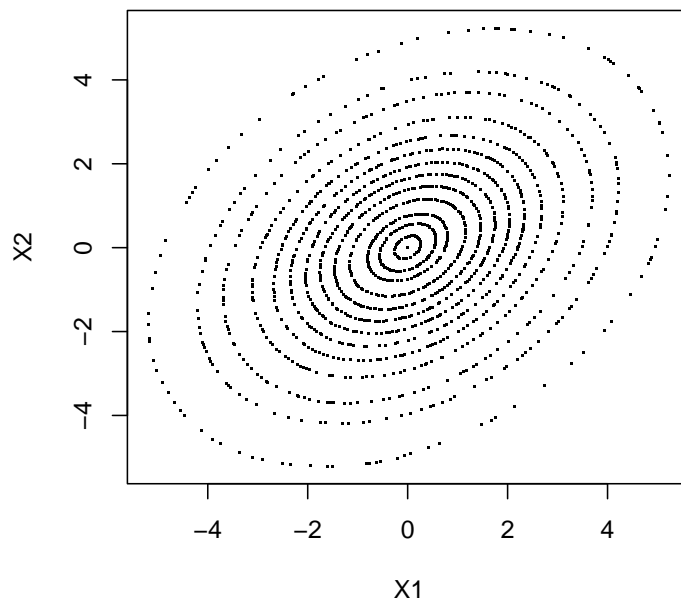


Figure 3: Expected number of points in the convex hull of a set of points in three dimensions

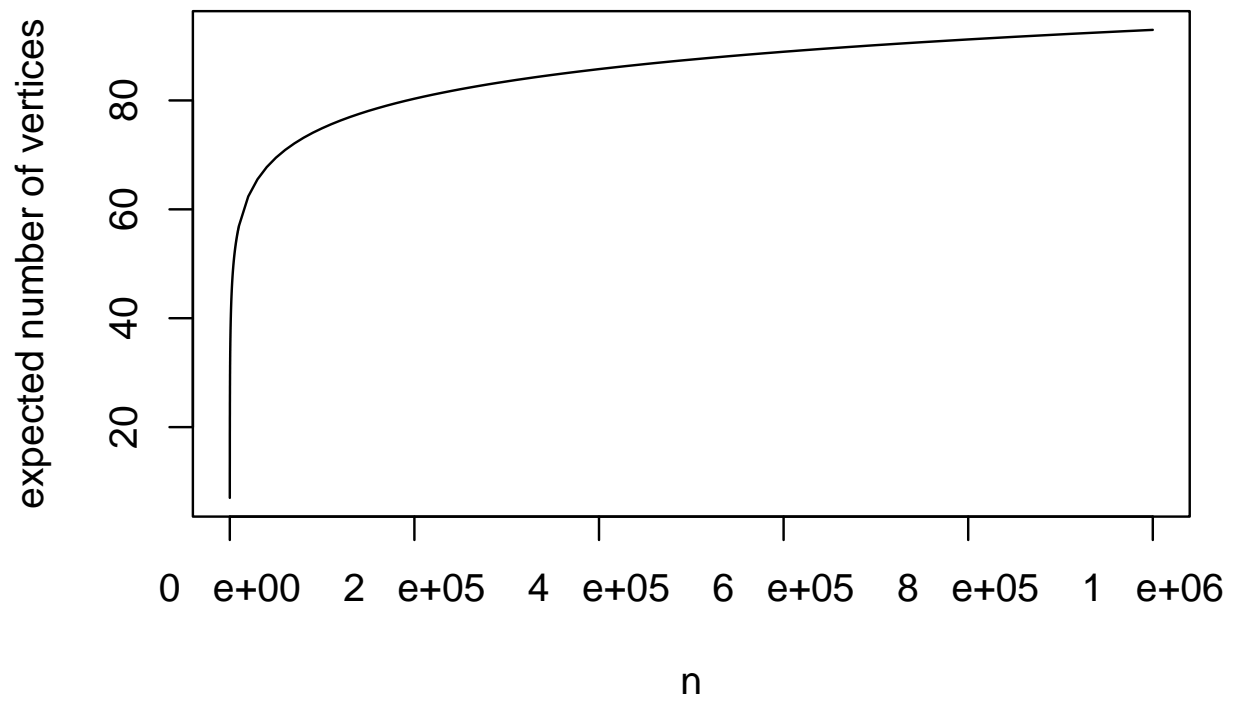


Figure 4: Average time comparison of usual convex hull algorithm with proposed method for median estimation

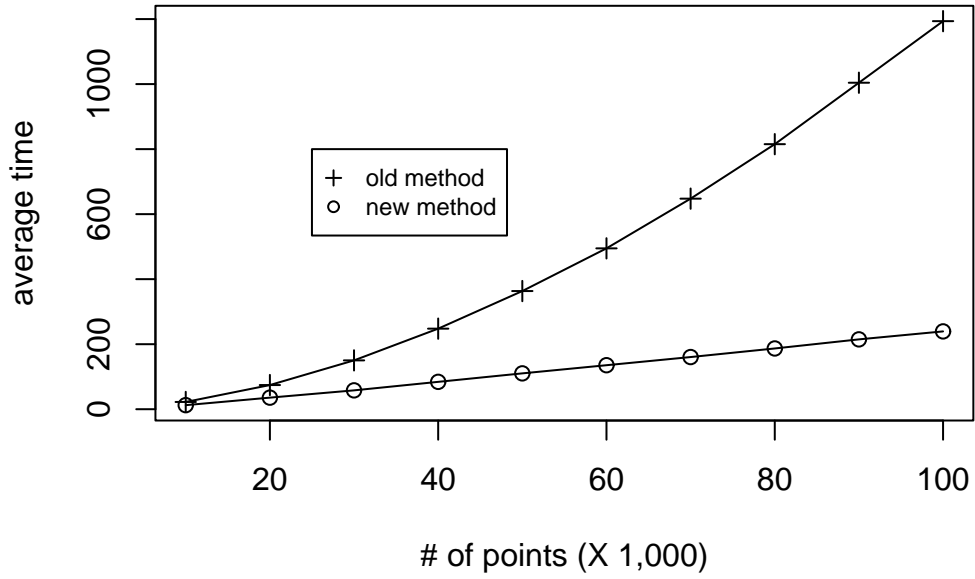


Figure 5: Schematic for computation of the bivariate density

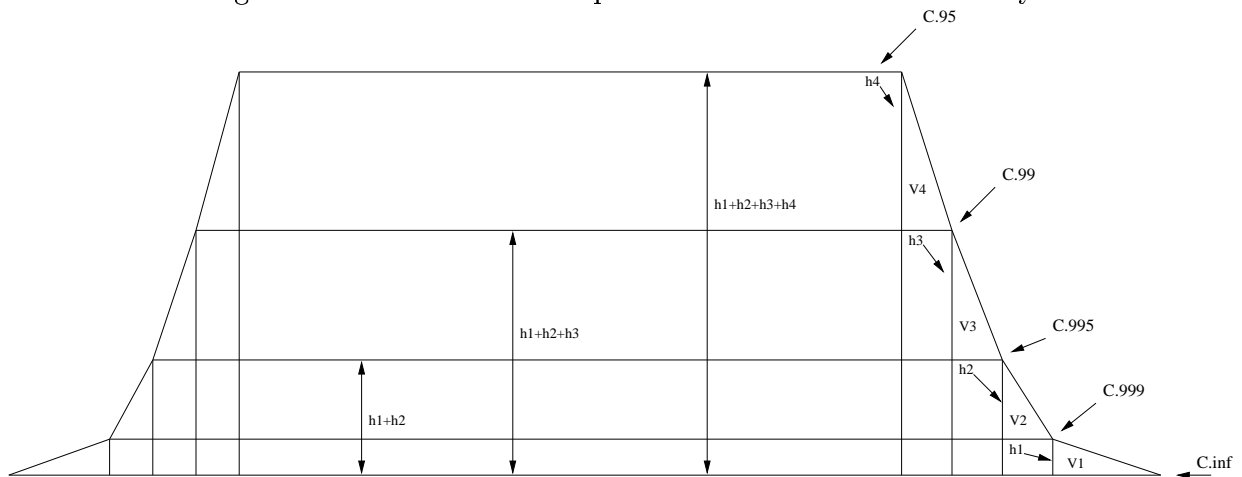


Figure 6: Two-Dimensional contours of a standard bivariate normal density

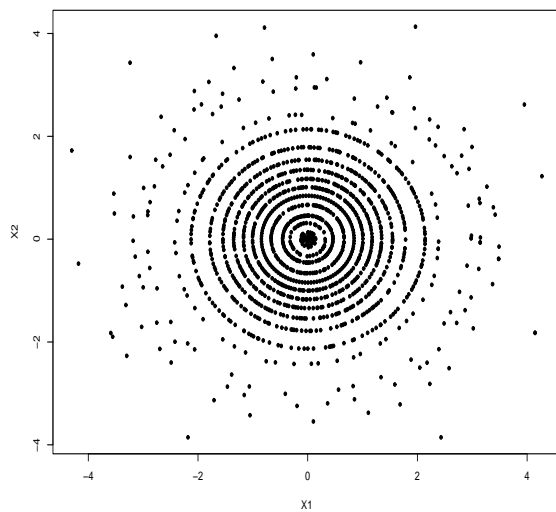


Figure 7: Three-Dimensional contours of a standard bivariate normal density

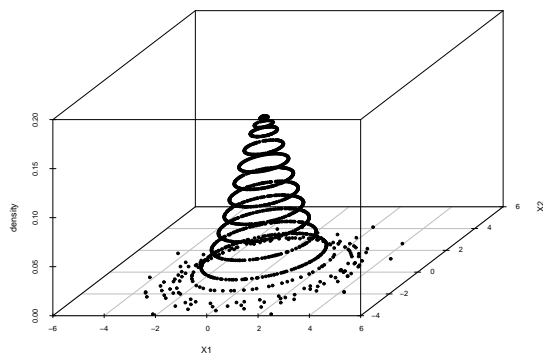


Figure 8: Example of multivariate spline fit to contours

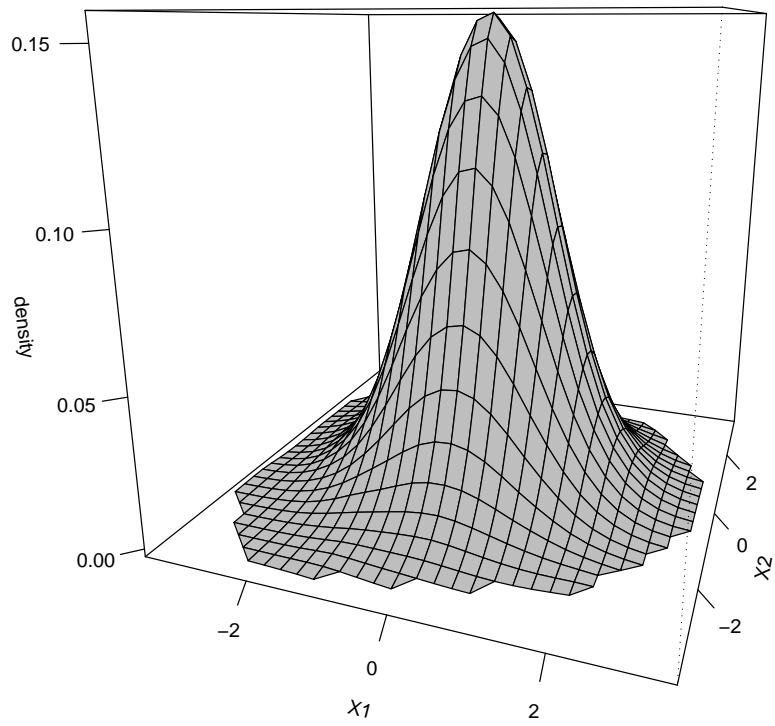


Figure 9: DPOSS dataset used in example -  $n = 11,355$

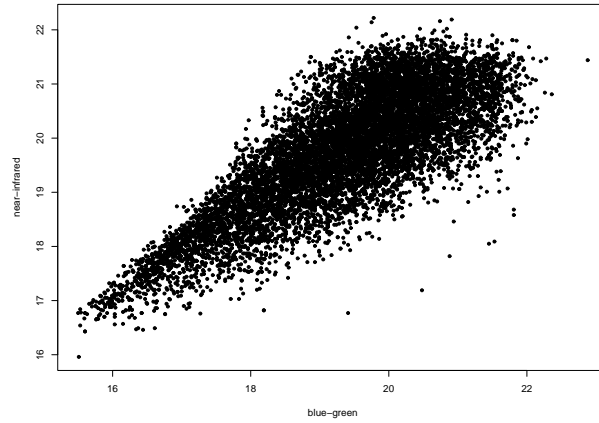


Figure 10: Convex hull peeling depth contours and medians using (a) traditional methods (b) sequential methods

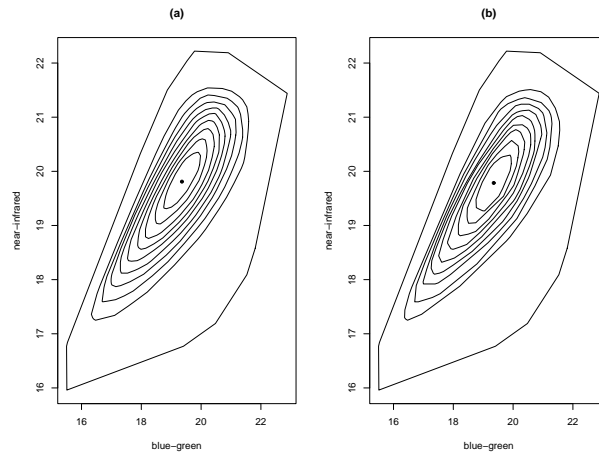


Figure 11: Thin plate spline fit to the depth contours with density estimates from the DPOSS dataset

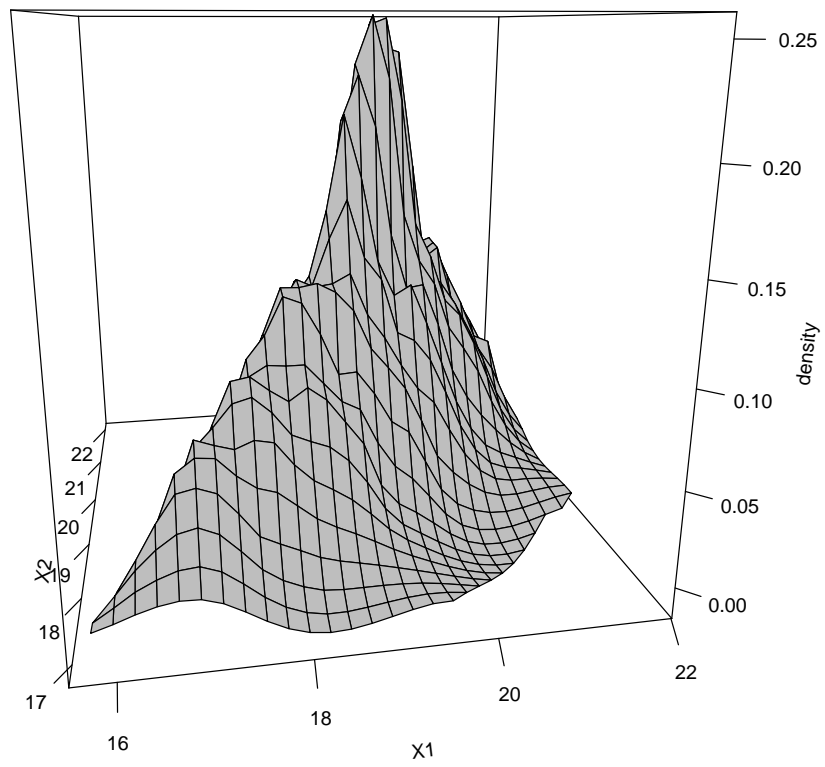


Table 1: Median study on Bivariate standard normal distribution with results averaged over 100 replications (trad=traditional exhausted method; seq=proposed sequential methods).

(a)  $d = .1$

m	n	mean		variance		ratio	time	
		trad	seq	trad	seq		trad	seq
1000	5000	0.0410	0.0444	0.000452	0.000527	1.166	0.31	0.20
	10000	0.0306	0.0345	0.000262	0.000314	1.197	0.85	0.37
	20000	0.0219	0.0246	0.000130	0.000158	1.209	2.62	0.75
	50000	0.0144	0.0169	0.000057	0.000073	1.274	12.7	2.05
	100000	0.0110	0.0125	0.000032	0.000042	1.328	46.1	4.37
2000	5000	0.0415	0.0467	0.000458	0.000632	1.381	0.41	0.25
	10000	0.0315	0.0340	0.000262	0.000310	1.182	0.85	0.42
	20000	0.0218	0.0246	0.000126	0.000165	1.305	3.46	1.18
	50000	0.0148	0.0165	0.000060	0.000073	1.210	12.5	2.27
	100000	0.0107	0.0121	0.000033	0.000039	1.198	37.2	4.27
5000	10000	0.0302	0.0313	0.000249	0.000264	1.060	0.74	0.52
	20000	0.0221	0.0231	0.000141	0.000156	1.106	2.76	1.29
	50000	0.0148	0.0168	0.000061	0.000077	1.271	10.4	2.46
	100000	0.0107	0.0121	0.000030	0.000039	1.292	37.4	5.52
10000	20000	0.0218	0.0226	0.000126	0.000137	1.087	2.24	1.42
	50000	0.0148	0.0159	0.000060	0.000070	1.172	11.2	3.74
	100000	0.0107	0.0115	0.000030	0.000036	1.220	38.0	7.11

(b)  $d = .5$

m	n	mean		variance		ratio	time	
		trad	seq	trad	seq		trad	seq
1000	5000	0.0411	0.0446	0.000467	0.000537	1.150	0.29	0.21
	10000	0.0302	0.0330	0.000256	0.000302	1.177	0.81	0.46
	20000	0.0220	0.0247	0.000143	0.000172	1.201	2.89	1.22
	50000	0.0150	0.0173	0.000063	0.000082	1.297	11.7	2.55
	100000	0.0105	0.0123	0.000032	0.000043	1.348	40.7	5.18
2000	5000	0.0413	0.0462	0.000480	0.000567	1.182	0.28	0.19
	10000	0.0297	0.0324	0.000287	0.000310	1.078	0.78	0.47
	20000	0.0220	0.0249	0.000126	0.000161	1.280	2.33	1.06
	50000	0.0144	0.0165	0.000065	0.000072	1.108	10.9	2.81
	100000	0.0106	0.0122	0.000030	0.000042	1.381	47.4	6.63
5000	10000	0.0301	0.0346	0.000261	0.000309	1.186	1.00	0.58
	20000	0.0224	0.0246	0.000143	0.000171	1.191	3.58	1.92
	50000	0.0147	0.0164	0.000060	0.000071	1.192	16.8	5.47
	100000	0.0105	0.0118	0.000031	0.000040	1.304	48.6	9.42
10000	20000	0.0219	0.0251	0.000136	0.000167	1.223	2.40	1.33
	50000	0.0151	0.0164	0.000060	0.000067	1.124	13.2	5.50
	100000	0.0105	0.0116	0.000031	0.000037	1.196	52.1	14.2

Table 2: Contour study on Bivariate standard normal distribution with results averaged over 100 replications (trad=traditional exhausted method; seq=proposed sequential method).

(a) n = 100,000									
p	m	total error		average error		area error		average time	
		trad	seq	trad	seq	trad	seq	trad	seq
.9	5000	0.9667	1.6247	0.0101	0.0217	7.3e-03	1.1e-01	18.00	1.31
.9	10000	0.9784	1.1311	0.0101	0.0158	7.0e-03	5.3e-02	17.54	1.91
.5	5000	0.5811	0.7033	0.0053	0.0090	5.1e-04	5.0e-03	61.49	3.90
.5	10000	0.6188	0.6504	0.0056	0.0085	6.7e-04	4.0e-03	60.59	5.85
.1	5000	0.3882	0.3730	0.0054	0.0073	6.8e-05	4.6e-04	80.99	5.36
.1	10000	0.3757	0.3376	0.0052	0.0067	5.4e-05	2.6e-04	80.45	7.90

(b) n = 1,000,000									
p	m	total error		avg error		area error		avg time	
		trad	seq	trad	seq	trad	seq	trad	seq
.9	5000	0.6890	2.9041	0.0034	0.0184	7.1e-04	6.6e-02	727.1	12.71
.9	10000	0.6904	0.9395	0.0034	0.0068	4.7e-04	9.6e-03	702.3	17.42
.5	5000	0.3746	0.5713	0.0019	0.0038	3.9e-05	9.4e-04	2443.8	36.67
.5	10000	0.3966	0.4279	0.0017	0.0030	3.9e-05	4.7e-04	2548.4	53.27
.1	5000	0.3631	0.2999	0.0024	0.0029	3.4e-05	6.9e-05	3231.0	52.03
.1	10000	0.3517	0.2280	0.0023	0.0023	3.4e-05	2.6e-05	3251.4	73.47

Table 3: Bivariate density estimation. Bivariate standard normal density with  $n = 1,000,000$  and results averaged over 100 replications.

p	true	m = 10000	
	density	mean	variance
.999	0.00016	0.00009	5.7e-11
.995	0.00080	0.00064	6.3e-10
.99	0.00159	0.00163	6.1e-09
.95	0.00796	0.00643	1.5e-08
.90	0.01592	0.01680	5.8e-08
.80	0.03183	0.02932	1.3e-07
.70	0.04775	0.04943	4.0e-07
.60	0.06366	0.06147	1.2e-06
.50	0.07958	0.08160	2.6e-06
.40	0.09549	0.09320	4.9e-06
.30	0.11141	0.11404	7.2e-06
.20	0.12732	0.12495	8.5e-06
.10	0.14324	0.14720	1.1e-05
.05	0.15120	0.14755	1.8e-05
.01	0.15756	0.15696	2.4e-05
.005	0.15836	0.15935	1.3e-04
.001	0.15900	0.16623	2.7e-04