CENTROIDAL VORONOI DIAGRAMS: APPLICATIONS, ALGORITHMS, AND ANALYSIS

Color printers, mailboxes, and fish (and numerical PDE's too!)

Max Gunzburger
Iowa State University
Supported by the National Science Foundation

John Burkardt
Qiang Du
Vance Faber
Lili Ju
Janet Peterson

TESSELLATIONS

open set Ω

open subsets $V_i \subset \Omega, \; i = 1, \ldots, K$

$\{V_i\}_{i=1}^K$ is a *tessellation* of Ω if

- $V_i \cap V_j = \emptyset \quad$ if $i \neq j$
- $\bigcup_{i=1}^K \overline{V_i} = \overline{\Omega}$

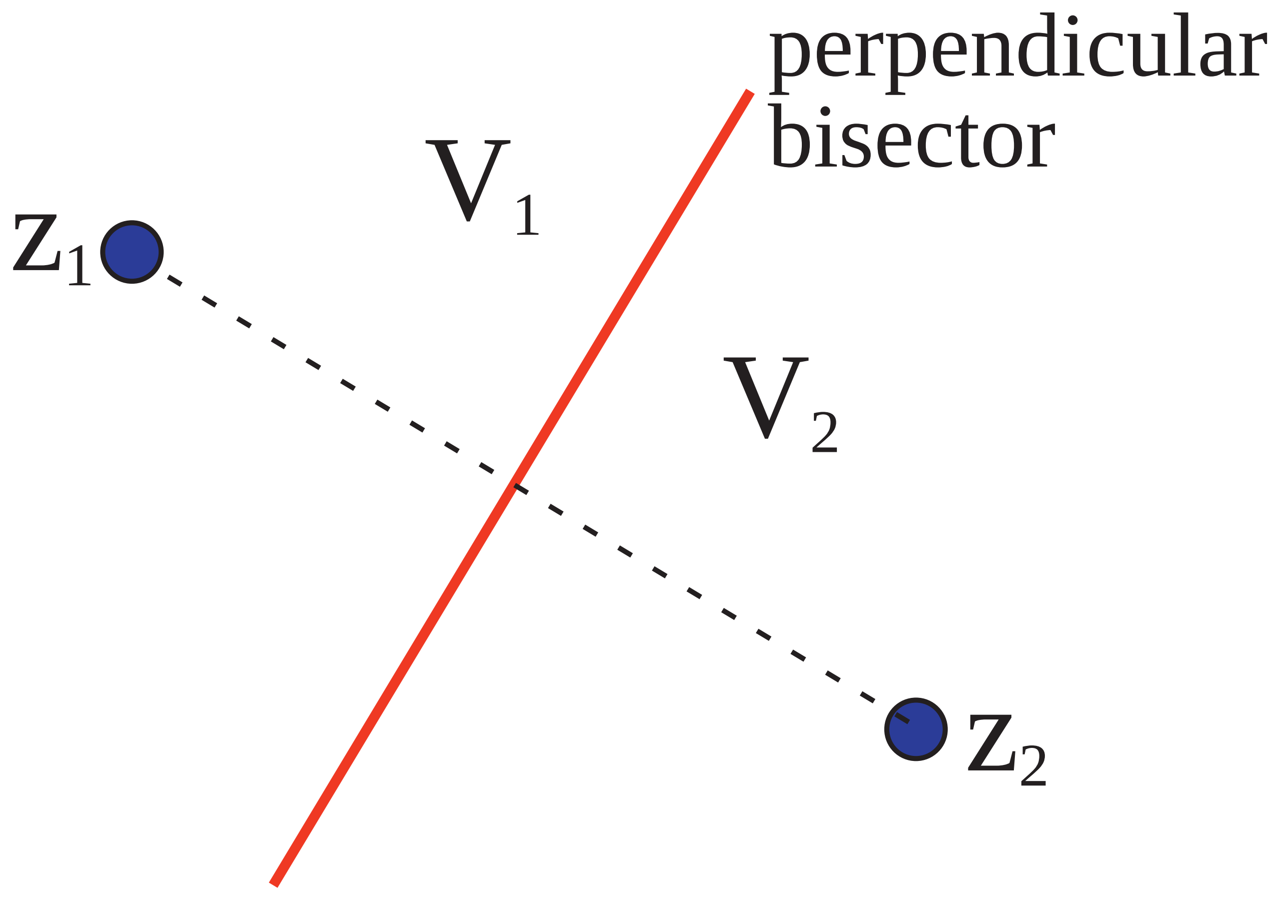
VORONOI SETS

- Given a set Ω
- Given K elements z_i, $i = 1, \ldots, K$, belonging to Ω
- Given a distance function $d(z, w)$ for points $z, w \in \Omega$

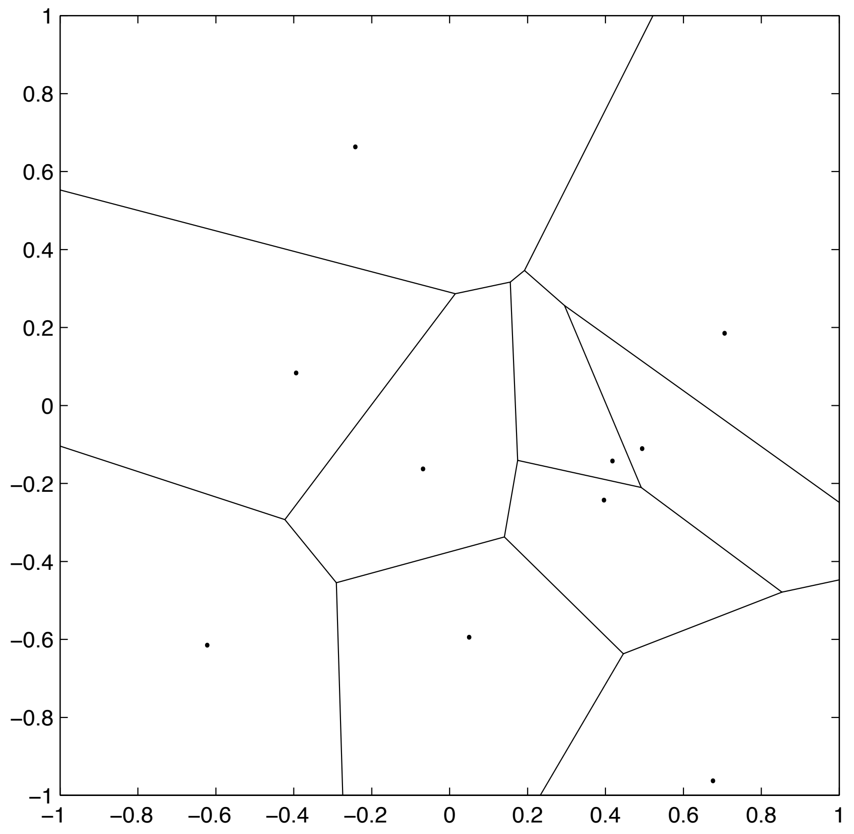
Then, the *Voronoi set* V_j is the set of all elements belonging to Ω that are closer to z_j than to any of the other elements z_i, $i = 1, \ldots, K$, $i \neq j$, i.e.,

$$V_j = \{w \in \Omega \,|\, d(w, z_j) < d(w, z_i),\ i = 1, \ldots, K,\ i \neq j\}$$

Voronoi sets =
Dirichlet regions =
Meijering cells =
S-mosaics =
Thiessen polygons =
area of influence polygons =
etc.



Voronoi regions for two points in the plane



Voronoi tessellation of 10 random points in a square

- *Voronoi polygons* and their **dual** tessellation, the *Delaunay triangulation*, are very useful in numerical computations, e.g., in interpolation, quadrature, partial differential equations, etc.

- For example, among all possible triangulations of a given set of points in the plane, the Delaunay triangulation is the one with *largest minimum angle*. This, in turn, is a good thing for, e.g., finite element approximations of partial differential equations.

- Another example is in "finite difference" discretizations of certain types of partial differential equations for which one must associate variables with *points*, and/or *regions*, and/or *edges*. On arbitrary grids, the best way to do this is to choose regions to be Voronoi polygons and/or Delaunay triangles, and to use the associated edges and vertices as needed.

MASS CENTROID

Given a region R and a density function $\rho(w)$, $w \in R$, the mass centroid z^* of R is given by

$$z^* = \frac{\displaystyle\int_R w\rho(w)\,dw}{\displaystyle\int_R \rho(w)\,dw}$$

— or —

Given a set of points $W = \{w_j\}_{j=1}^M$ and a density function $\rho(w_j)$, $j = 1, \ldots, M$, the mass centroid z^* of W is given by

$$z^* = \frac{\displaystyle\sum_{j=1}^{M} w_j \rho(w_j)}{\displaystyle\sum_{j=1}^{M} \rho(w_j)}$$

- Given K points z_i, $i = 1, \dots, K$,
  we can define the associated *Voronoi sets*

$$V_i\,, \quad i = 1, \dots, K$$

- Given K points z_i, $i = 1, \ldots, K$,
  we can define the associated *Voronoi sets*

$$V_i\,, \quad i = 1, \ldots, K$$

- Given the Voronoi sets V_i, $i = 1, \ldots, K$,
  we can define the associated *mass centroids*

$$z_i^*\,, \quad i = 1, \ldots, K$$

- Given K points z_i, $i = 1, \ldots, K$,
  we can define the associated *Voronoi sets*

$$V_i\,, \quad i = 1, \ldots, K$$

- Given the Voronoi sets V_i, $i = 1, \ldots, K$,
  we can define the associated *mass centroids*

$$z_i^*\,, \quad i = 1, \ldots, K$$

- We are interested in the special situation wherein

$$z_i = z_i^*\,, \quad i = 1, \ldots, K$$

- Given K points z_i, $i = 1, \ldots, K$,
  we can define the associated *Voronoi sets*

$$V_i\,, \quad i = 1, \ldots, K$$

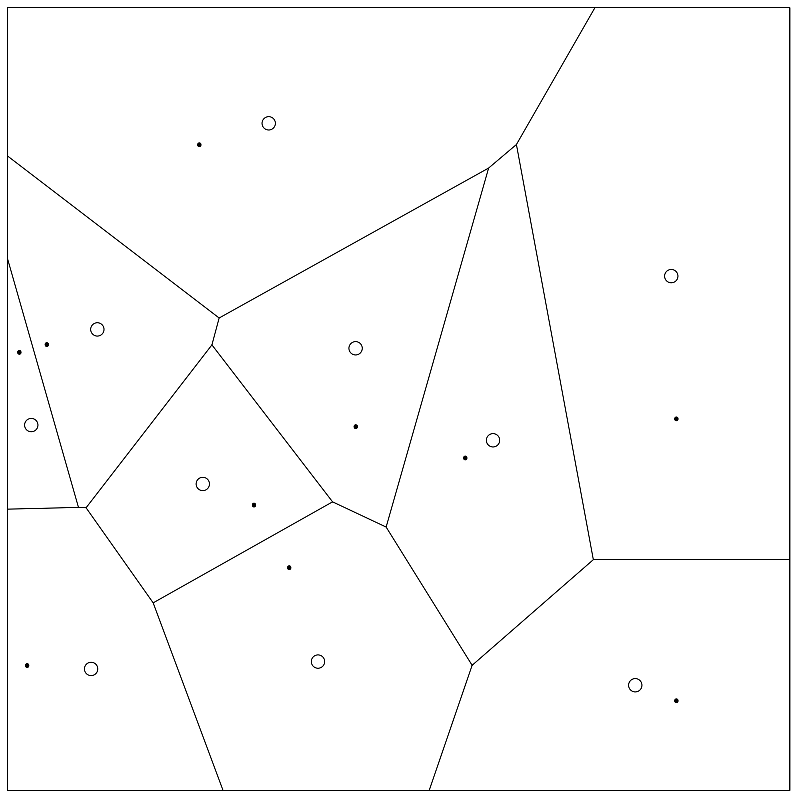
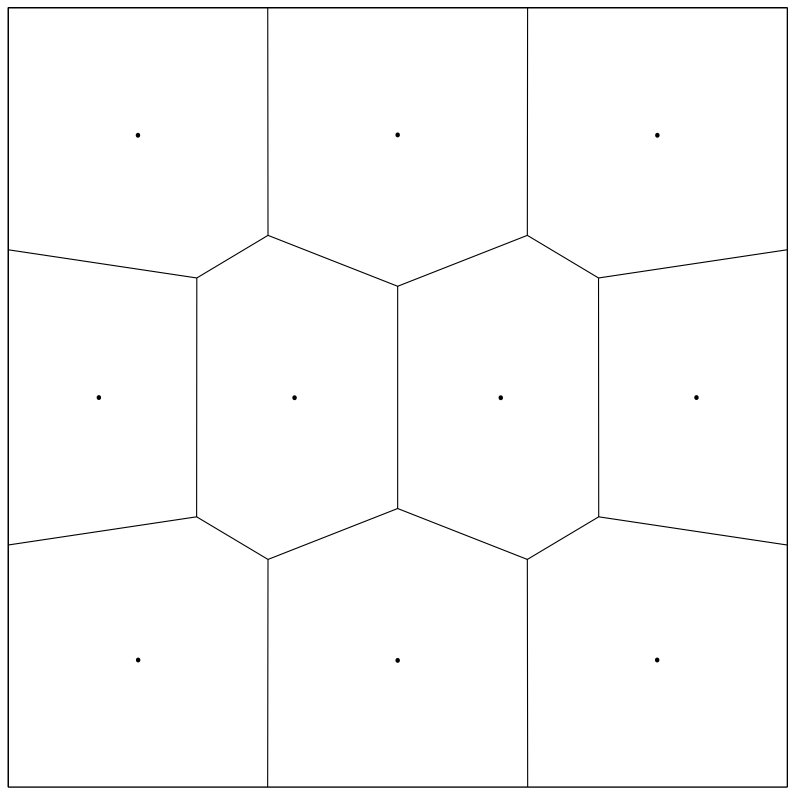
- Given the Voronoi sets V_i, $i = 1, \ldots, K$,
  we can define the associated *mass centroids*

$$z_i^*\,, \quad i = 1, \ldots, K$$

- We are interested in the special situation wherein

$$z_i = z_i^*\,, \quad i = 1, \ldots, K$$

$\Longrightarrow$ **CENTROIDAL VORONOI TESSELLATION**

Voronoi regions and their centroids for 10 randomly selected points (left) and a 10-point centroidal Voronoi tessellation (right) in a square

THE CONSTRUCTION PROBLEM FOR CENTROIDAL VORONOI TESSELLATIONS

- *Given*

  a region Ω, an integer $K > 1$, and a density function $\rho(w)$

- *we are interested in finding*

  K points $\{z_i\}_{i=1}^K$ and K regions $\{V_i\}_{i=1}^K$

- *such that*

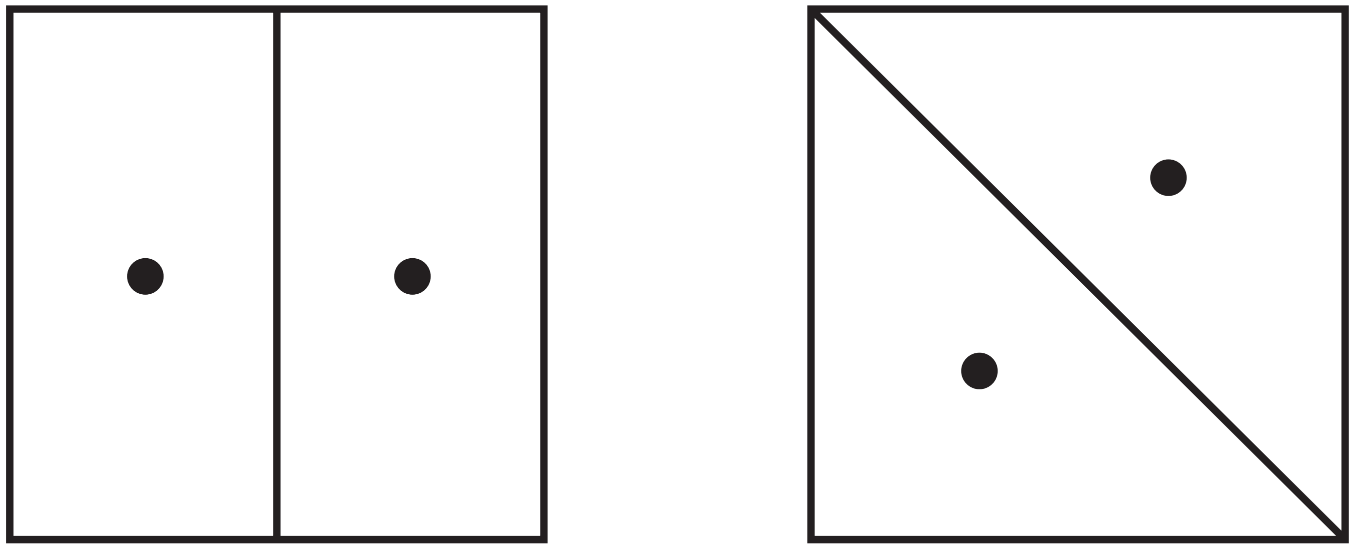
  $z_i \in \overline{\Omega}\,,\ i = 1, \ldots, K,\quad \{V_i\}_{i=1}^K$ tessellates Ω

- *and simultaneously*

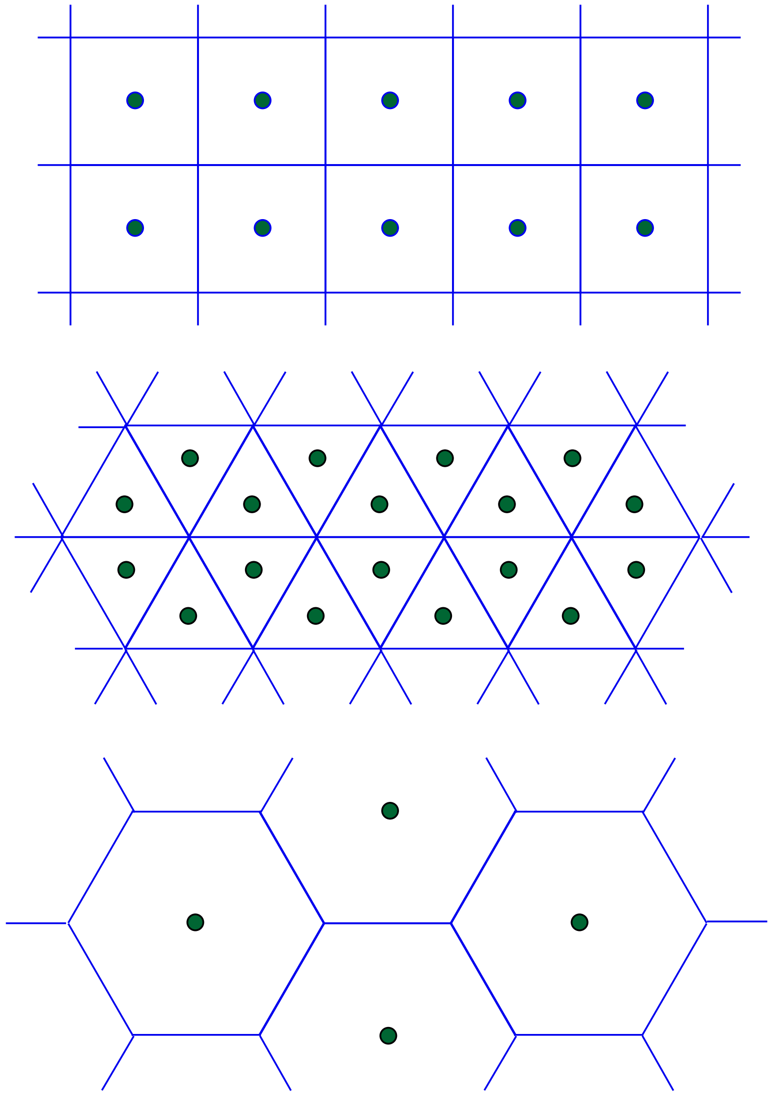
  the regions $\{V_i\}_{i=1}^K$ are Voronoi regions for the points $\{z_i\}_{i=1}^K$

  the points $\{z_i\}_{i=1}^K$ are the mass centroids of the regions $\{V_i\}_{i=1}^K$

Note that, in general, one *does not have uniqueness*



Two two-point centroidal Voronoi tessellations of a square



Three regular tessellations of the plane

— APPLICATION 1 —
OPTIMAL QUADRATURE RULES

- For a given region Ω and a given number K, consider the quadrature rule
$$\int_\Omega f(x)\,dx \approx \sum_{i=1}^{K} A_i f(z_i)$$
where

  $\{z_i\}_{i=1}^K$ are K points in $\overline{\Omega}$

  $\{A_i\}_{i=1}^K$ are the volumes of a set of regions $\{V_i\}_{i=1}^K$ that tessellate Ω

- We want to choose the z_i's and V_i's so that the quadrature error is minimized

- Quadrature error for Lipschitz continuous functions

$$\Big| \int_\Omega f(x)\,dx - \sum_{i=1}^K A_i f(z_i)\Big| = \Big|\sum_{i=1}^K \int_{V_i} f(x)\,dx - \sum_{i=1}^K \int_{V_i} f(z_i)\,dx\Big|$$
$$\le \sum_{i=1}^K \int_{V_i} \big|f(x) - f(z_i)\big|\,dx \le L \sum_{i=1}^K \int_{V_i} |x - z_i|\,dx$$

- *Optimal quadrature rule is one using a centroidal Voronoi tessellation of Ω*
  - the V_i's are the Voronoi regions for the z_i's
  - the z_i's are the centroids of the V_i's

- Can extend to higher-order quadrature rules, using function values and derivative values at the points z_i. Then, for functions having Lipschitz continuous $(m-1)$-st derivatives,

$$\text{Quadrature error} \le L_m \sum_{i=1}^K \int_{V_i} |x - z_i|^m\,dx$$

— APPLICATION 2 —
NUMERICAL PDE's

Covolume methods for the Poisson equation

(R. Nicolaides and N. Walkington)

- for Voronoi grids: L^2 error is $O(h)$
- for centroidal Voronoi grids: L^2 error is $O(h^2)$

Finite difference methods for the Poisson eq.

- for general grids: truncation error is $O(h)$
- for centroidal Voronoi grids: truncation error is $O(h^2)$

— APPLICATION 3 —
OPTIMAL DISTRIBUTION OF RESOURCES

What is the optimal placement of mailboxes in a given region?

- A user will use the mailbox nearest to their home
- The cost (to the user) of using a mailbox is proportional to the distance from the user's home to the mailbox
- The total cost to users as a whole is measured by the distance to the nearest mailbox averaged over all users in the region
- The optimal placement of mailboxes is defined to be the one that minimizes the total cost to the users

The optimal placement of the mail boxes is at the centroids of a centroidal Voronoi tessellation

A. Okabe, B. Boots, and K. Sugihara; *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley, Chichester, 1992

— APPLICATION 4 —
OPTIMAL REPRESENTATION, QUANTIZATION, AND CLUSTERING

Representation , e.g., interpolation, is replacement of observed data by a simpler set of data

Centroidal Voronoi tessellations are intimately related to optimal representation

Clustering is a tool to analyze similarities or dissimilarities between different objects

Centroidal Voronoi tessellations are intimately related to optimal k-means clustering

Quantization is the representation of a given quantity with less information

Centroidal Voronoi tessellations are intimately related to optimal vector quantization

These are central subjects of information theory and statistics

Clustering

- Let W contains m points in R^N

- Given a subset (cluster) V of W with ℓ points, the cluster is to be represented by the arithmetic mean

$$\overline{\mathbf{x}}_i = \frac{1}{\ell} \sum_{\mathbf{x}_j \in V} \mathbf{x}_j$$

  which corresponds to the mass centroid of V

- The variance is given by

$$Var(V) = \sum_{\mathbf{x}_j \in V} |\mathbf{x}_j - \overline{\mathbf{x}}|^2$$

- For a K-clustering $\{V_i\}_{i=1}^{K}$ (a tessellation of W into k disjoint subsets), the total variance is given by

$$Var(W) = \sum_{i=1}^{K} Var(V_i) = \sum_{i=1}^{K} \sum_{\mathbf{x}_j \in V_i} |\mathbf{x}_j - \overline{\mathbf{x}}_i|^2$$

- The optimal K-clustering having the minimum total variance occurs when $\{V_i\}_{i=1}^{K}$ is the Voronoi partition of W with $\{\overline{\mathbf{x}}_i\}_{i=1}^{K}$ as the generators, i.e., if we use the above variance-based criteria, the optimal k-clustering is a centroidal Voronoi diagram

— APPLICATION 5 —
CELL DIVISION

There are many examples of cells that are polygonal – often they can be identified with a Voronoi, indeed, a centroidal Voronoi tessellation

This is especially evident in monolayered or columnar cells, e.g., as in the early development of a starfish (*Asteria pectinifera*)

Cell division

- Start with a configuration of cells that, by observation, form a Voronoi tessellation
- The division of a cell can be modeled by the addition of a Voronoi generator, or more precisely, the splitting of one generator into two
- Having added generators, i.e., cells, what is the shape of the new cell arrangement?
- It is observed that the new cell arrangement is closely approximated by a centroidal Voronoi tessellation

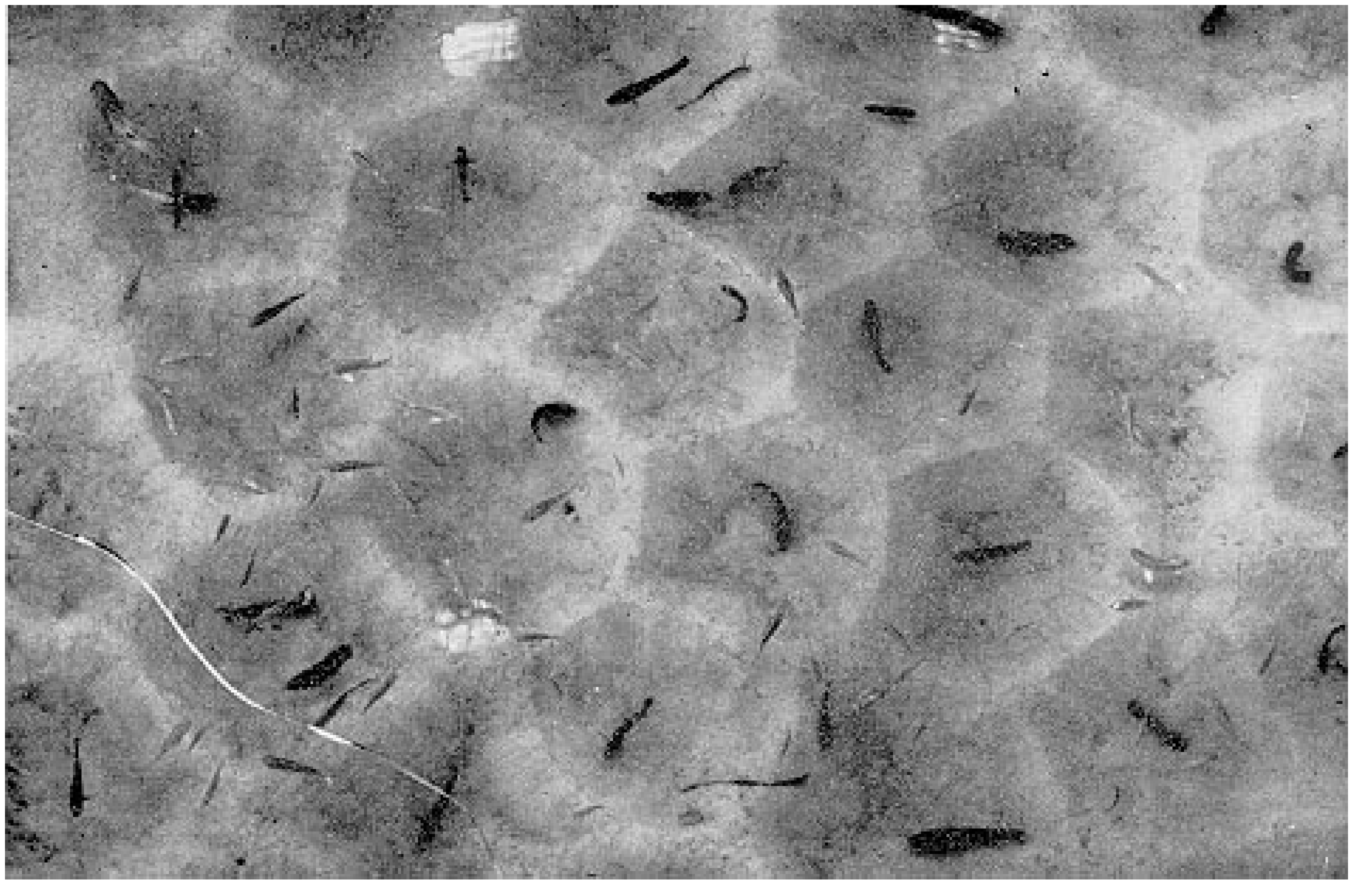
— APPLICATION 6 —
TERRITORIAL BEHAVIOR OF ANIMALS

Male moutbreeder fish – *Tilapia mossambica*

- Fishes dig nesting pits in sandy bottoms
- They adjust the centers and boundaries of the pits so that the final configuration of territories is a centroidal Voronoi tessellation

Actually,

fishes are observed to perform an iteration known as Lloyd's method !



A top view photograph, using a polarizing filter, of the territories of the male *Tilapia mossambica.*

Photograph from: George Barlow; Hexagonal territories, *Animal Behavior* **22** 1974, pp. 876–878.

— APPLICATION 7 —
DATA COMPRESSION (IMAGE PROCESSING)

- each point in a picture has a specific color
- each color is a combination of basic (primary, RGB, CMY) colors
- let the components of a vector w represent a possible combination of the basic colors
- let $\rho(w)$ denote the number of times the particular combination w occurs in the picture
- let Ω denote the set of admissible color combinations
- there are *zillions* of different colors in a given picture
- one would like to approximate the picture using just a *few* combinations of the basic colors

Questions:

1. How to choose the few colors that are to be used to represent the picture?

2. How to assign the colors in the picture to the few chosen colors?

Why data compression?

- Suppose the picture has 10^6 pixels
- Suppose the color at each pixel is determined by a 24 bit number, i.e., the set Ω of possible colors has cardinality 2^{24}

$$\text{amount of information in picture} = 10^6 \times 24 \text{ bits}$$

- Now suppose we approximate the picture, by some replacement algorithm, using only 8 bit numbers, i.e., $256 = 2^8$ different colors
- we still have 10^6 pixels, so data reduction does not come from reducing the number of points in the picture

$$\text{amount of information in approximate picture} = 10^6 \times 8 \text{ bits}$$

$$\frac{\text{number of bits in approximate picture}}{\text{number of bits in original picture}} = \frac{1}{3}$$

Algorithm for determining approximate picture

Given Ω, $\rho(w)$, and K, choose

- K color combinations $\{z_i\}_{i=1}^K$
- K subsets $\{V_i\}_{i=1}^K$ that tessellate Ω
- let w be any color appearing in the picture
- *Replacement method*

  If $w \in V_i$, **replace** w **with** z_i

Clearly, we want to choose $\{z_i\}_{i=1}^K$ and $\{V_i\}_{i=1}^K$ so that the approximate picture using only K different color combinations is a good approximation of the given picture that contains zillions of different color combinations.

“Obvious” method

- using $\rho(w)$ (suitably normalized) as a probability density, use a Monte Carlo method to find $\{z_i\}_{i=1}^K$
- once $\{z_i\}_{i=1}^K$ is the determined, choose $\{V_i\}_{i=1}^K$ to be the associated Voronoi sets

With $N = 256$, “experts” can easily see differences between the original and approximate pictures

RESULTS ARE NOT GREAT

Better method

- Determine $\{z_i\}_{i=1}^K$ and $\{V_i\}_{i=1}^K$ so that

$$\mathcal{E} = \sum_{i=1}^K \int_{V_i} \rho(w)|w - z_i|^2 \, dw$$

  is minimized over all possible sets of K points belonging to Ω and all possible tessellations of Ω into K regions

- For $\Omega =$ a discrete set of M points $\{w_j\}_{j=1}^M$, we instead have

$$\mathcal{E} = \sum_{i=1}^K \sum_{w_j \in V_i} \rho(w_j)|w_j - z_i|^2$$

- Note that $\mathcal{E} = \mathcal{E}(z_i, V_i)$ and that we assume no *a priori* relation between the z_i's and V_i's

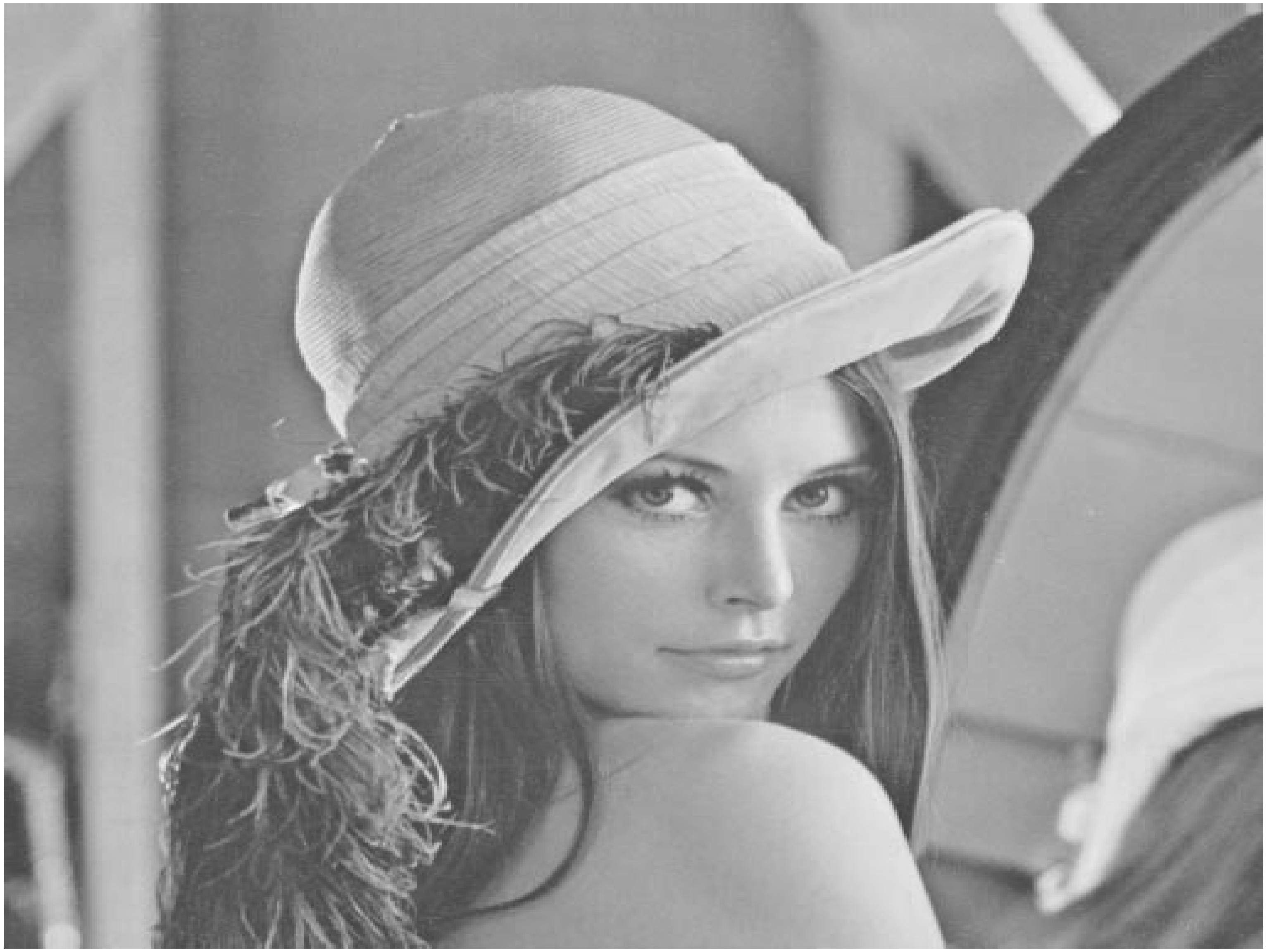
$\mathcal{E}$ is minimized when

- the V_i's are the Voronoi sets for the z_i's

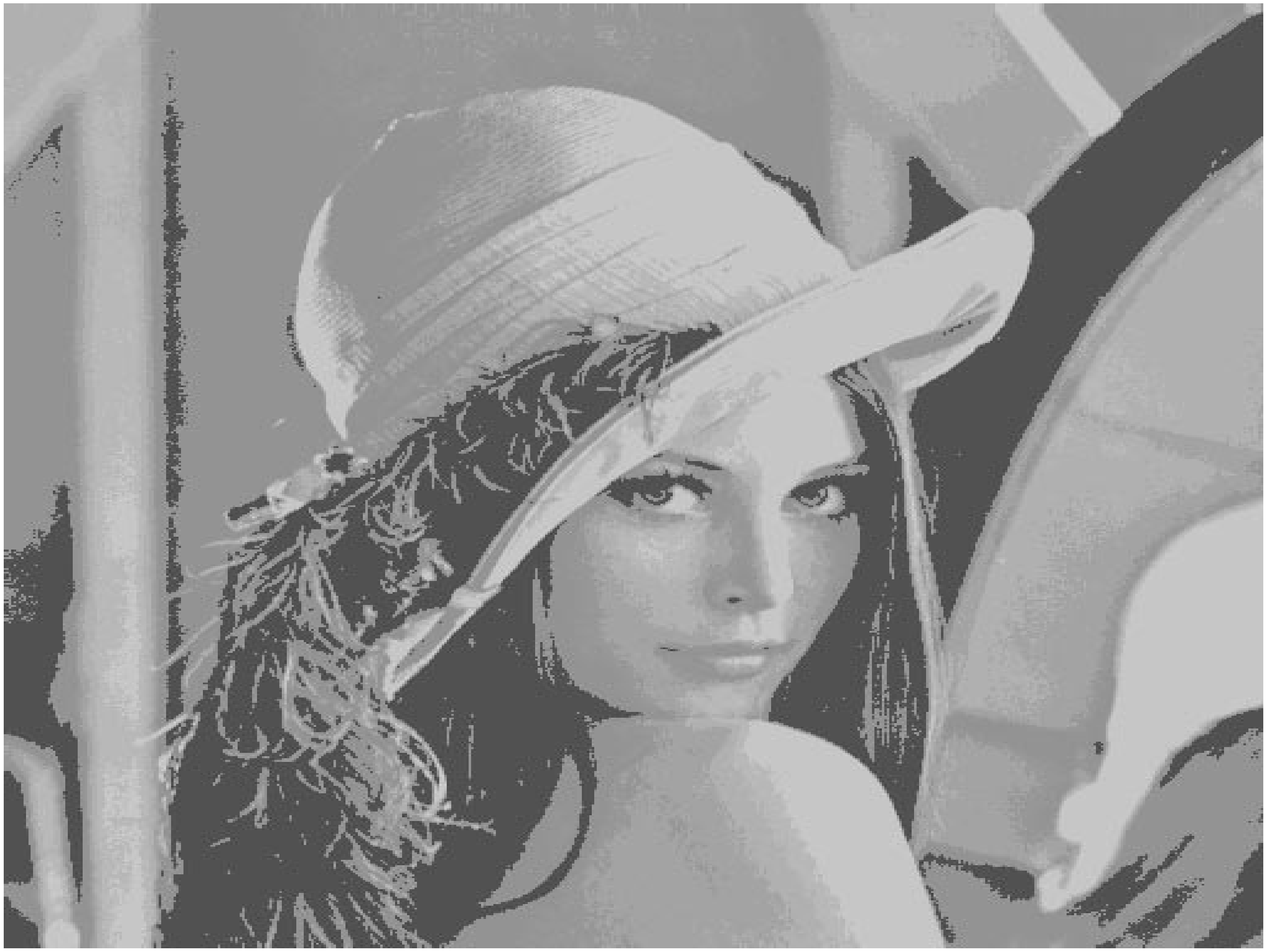
and

- the z_i's are the centroids of the V_i's

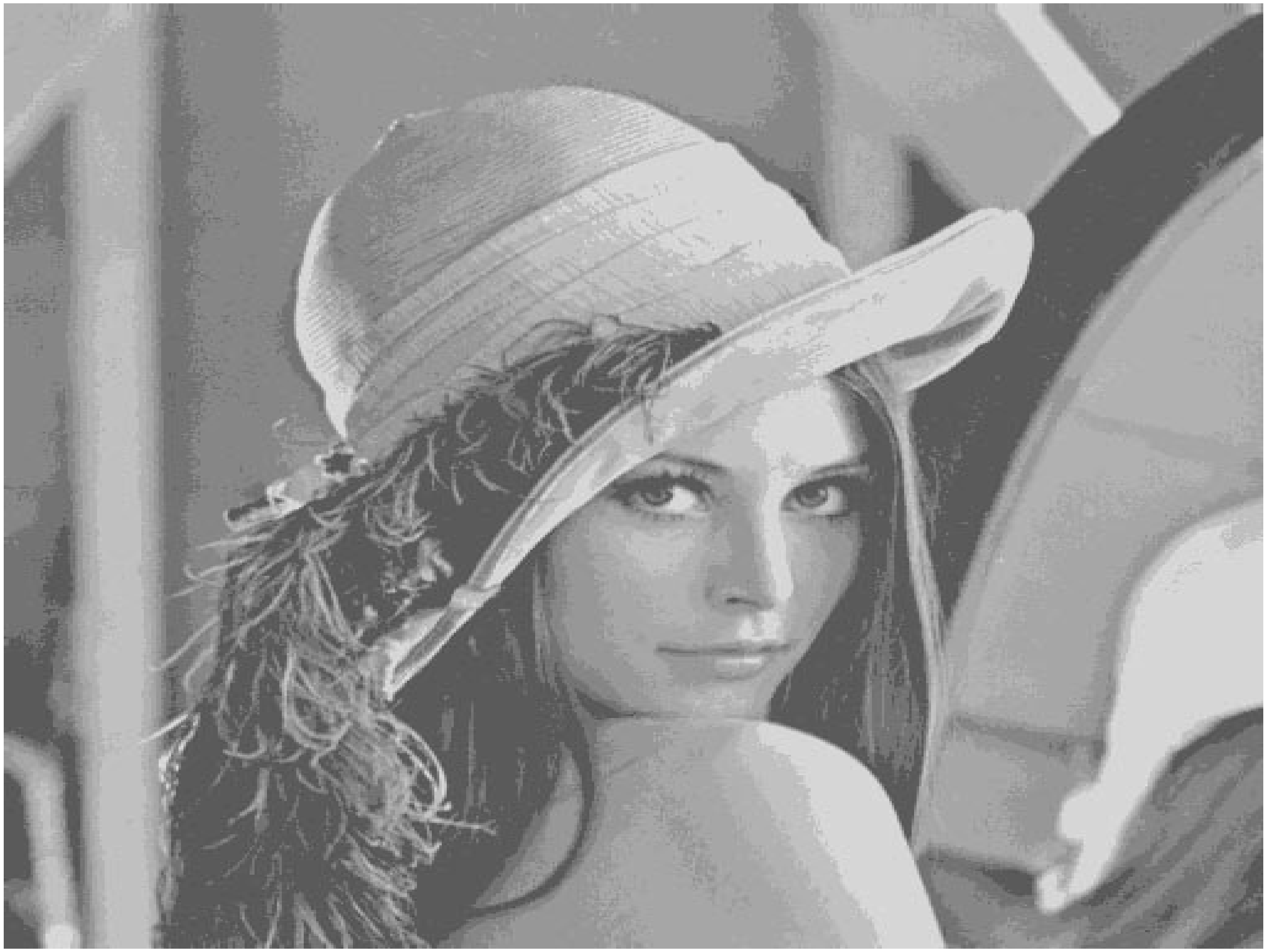
PRODUCES GREAT APPROXIMATE PICTURES!!!



Original 8-bit grayscale image of Lena



Monte Carlo 3-bit approximate picture



Centroidal Voronoi 3-bit approximate picture

Contouring

Contouring is an effect that results from discontinuous approximations to "continuous" pictures

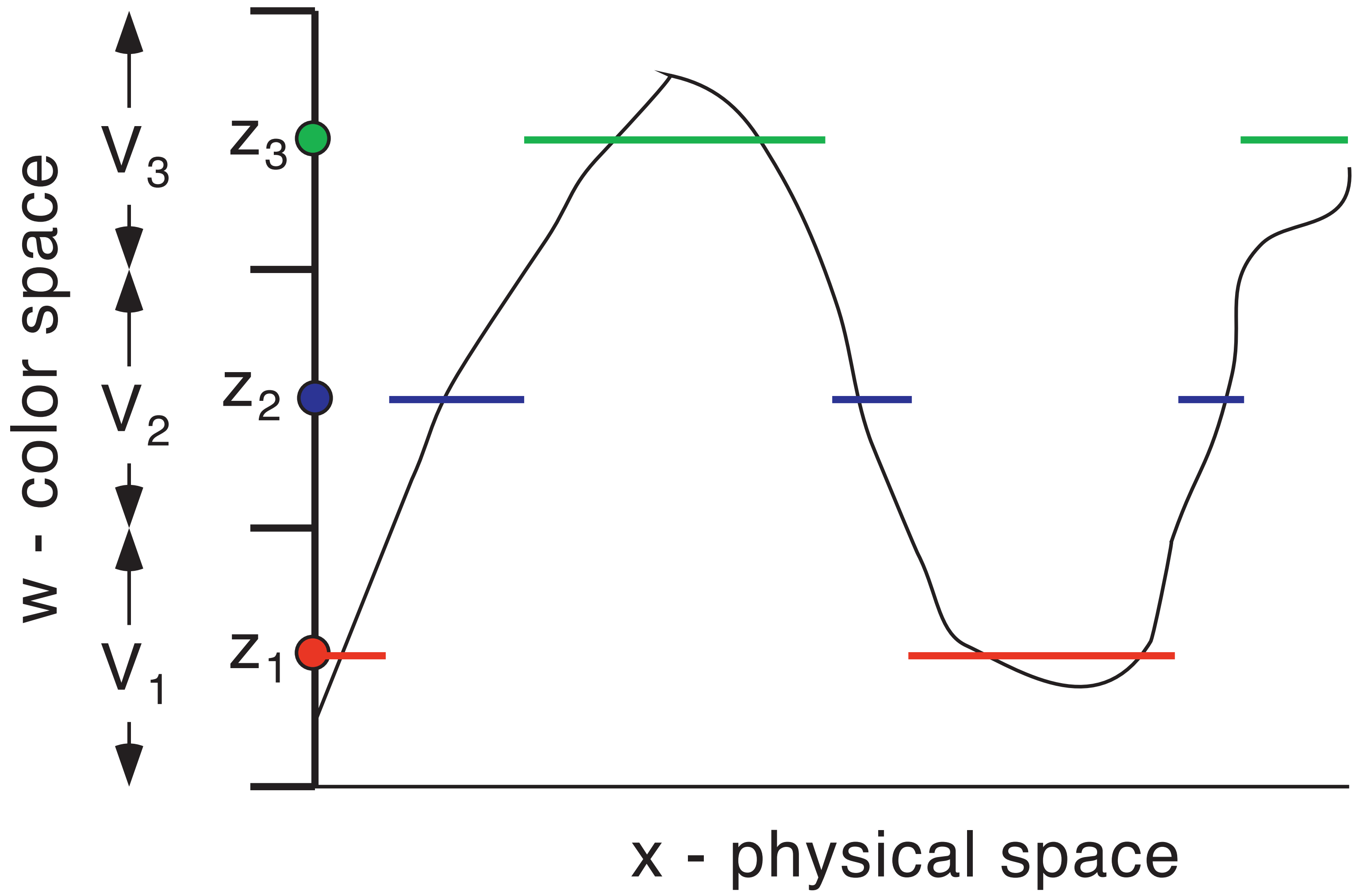
- 1-D picture $w(x)$ where w is the color and x is a physical length coordinate

- Suppose $w(x)$ is continuous

- Suppose we have K colors $\{z_i\}_{i=1}^{K}$ to which the colors in the picture are assigned

- *Result: a piecewise constant picture* - **contouring**

- Contouring is a problem with the *assignment* part of the algorithm, and not with the choice of the colors z_i used in the assignment



Piecewise constant (contoured) 3-color approximation to a continuous 1-D picture

Dithering

- *Dithering* is a class of solutions of the contouring problem such that one does not always assign to the nearest color in the set $\{z_i\}_{i=1}^{K}$

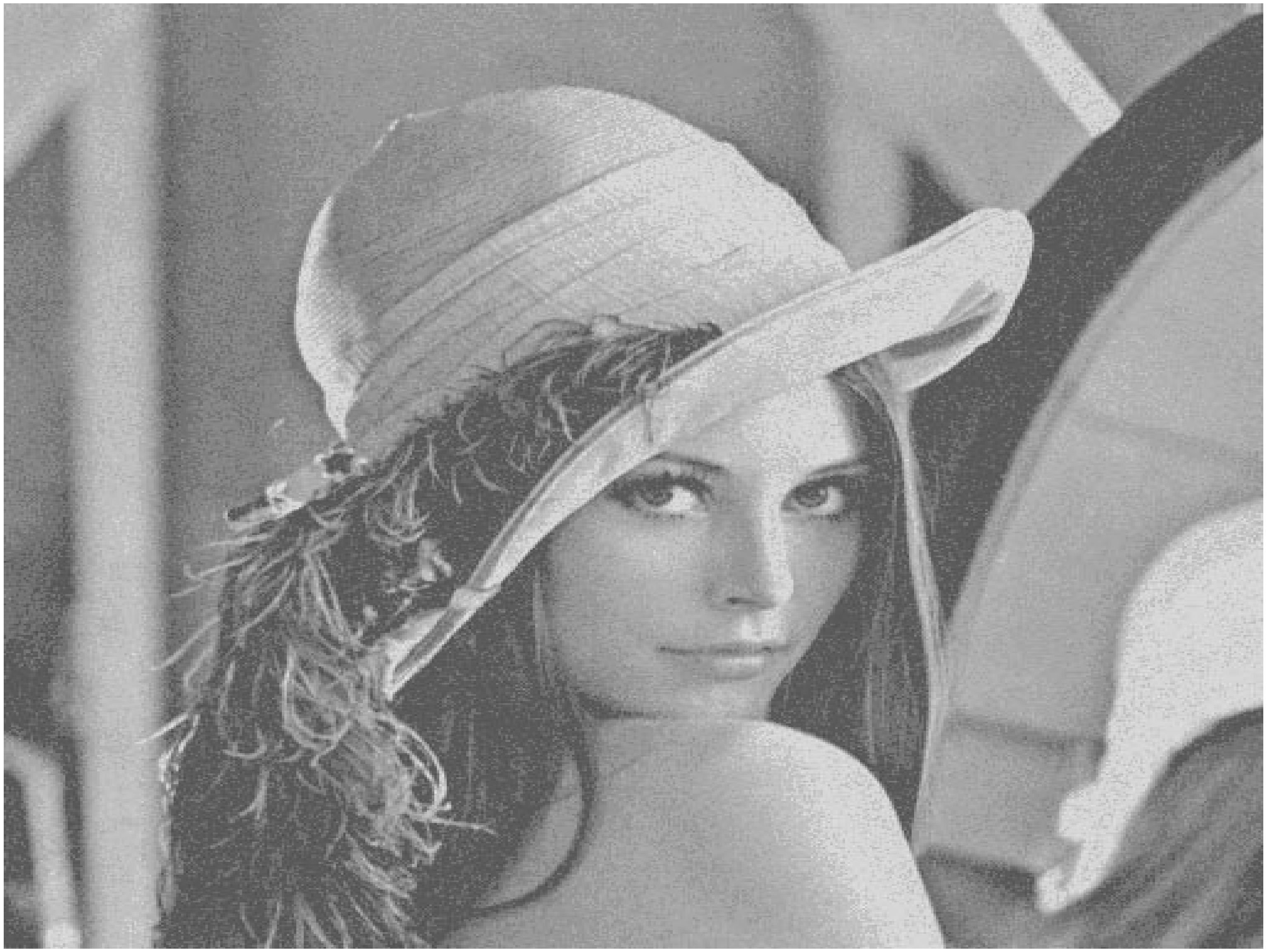
- A simple dithering algorithm:

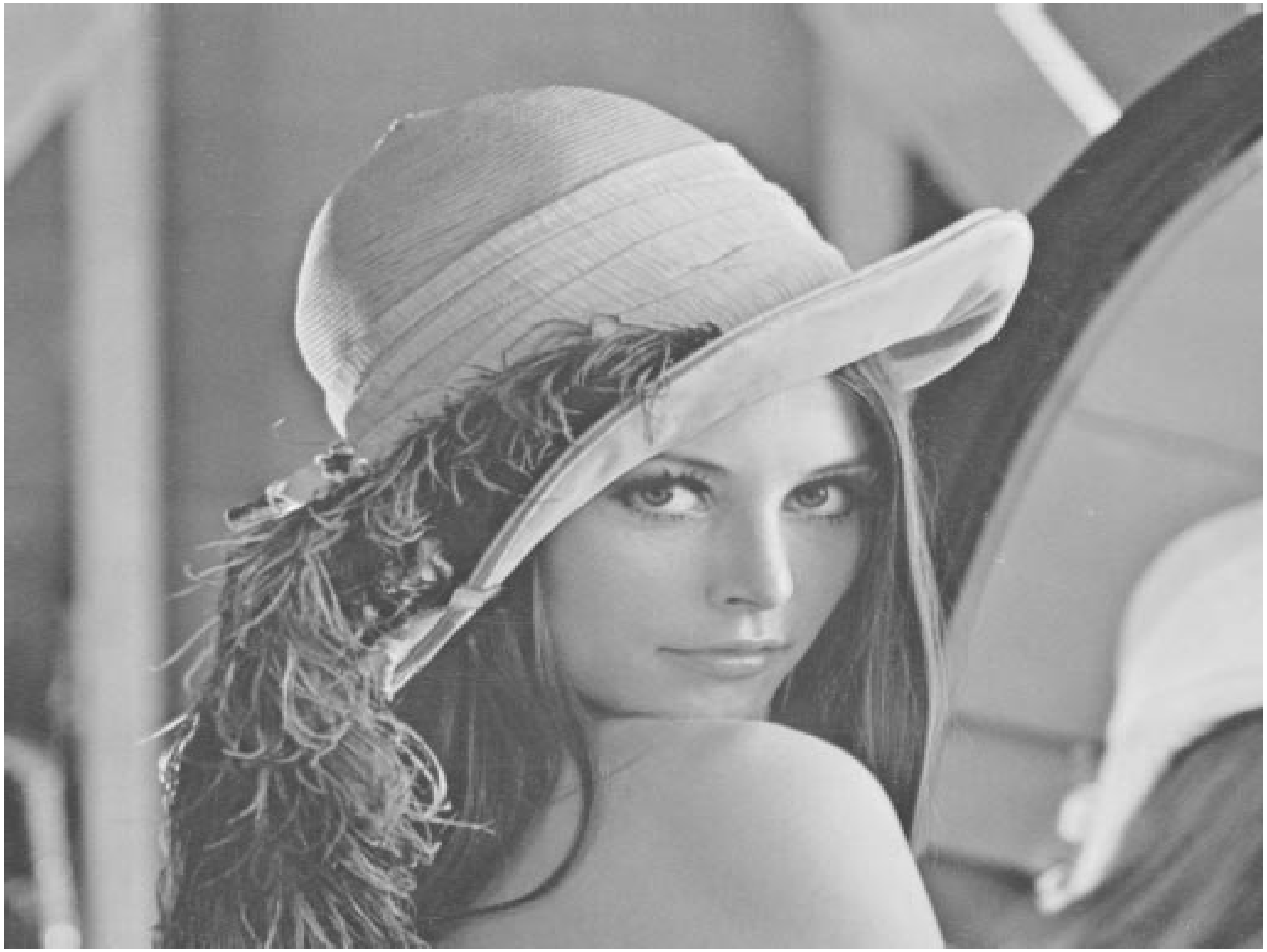
  a. Let w be the color to be assigned to one of the z_i , $i = 1, \ldots, K$

  b. Find the *two* closest z_i's to w; call them z_a and z_b

  c. Replace w with *either* z_a or z_b, according to a probability based on the inverse of the distances from w to z_a and z_b

- Results in a picture that is no longer piecewise constant


Dithered centroidal Voronoi 3-bit approximate picture

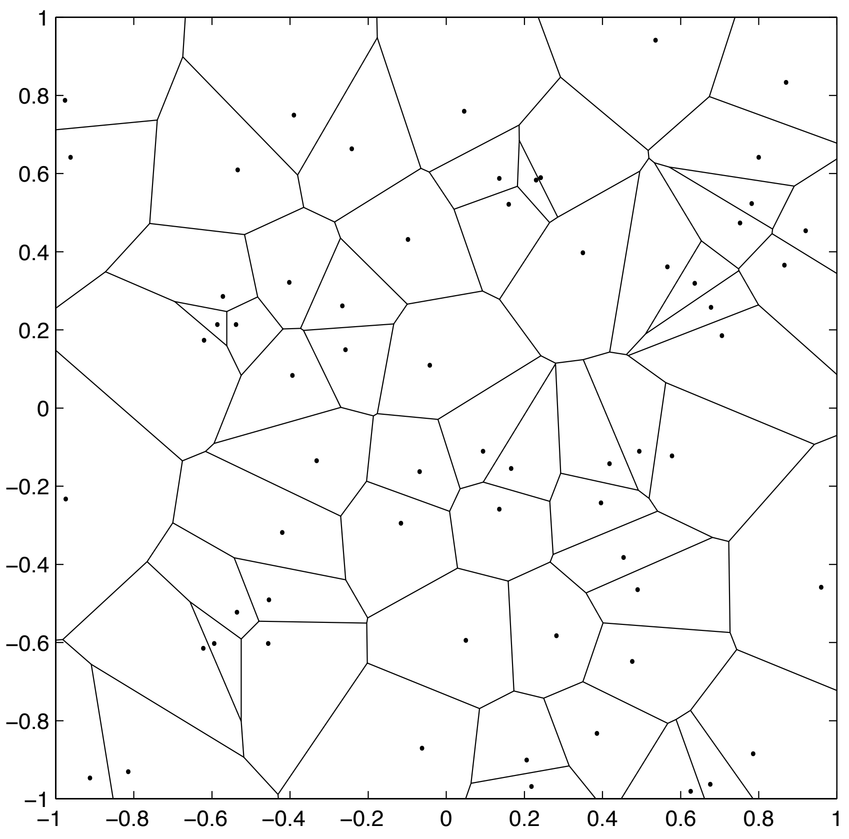

Original 8-bit grayscale image of Lena

CENTROIDAL VORONOI TESSELLATIONS IN THE SQUARE $[-1,1]^2$

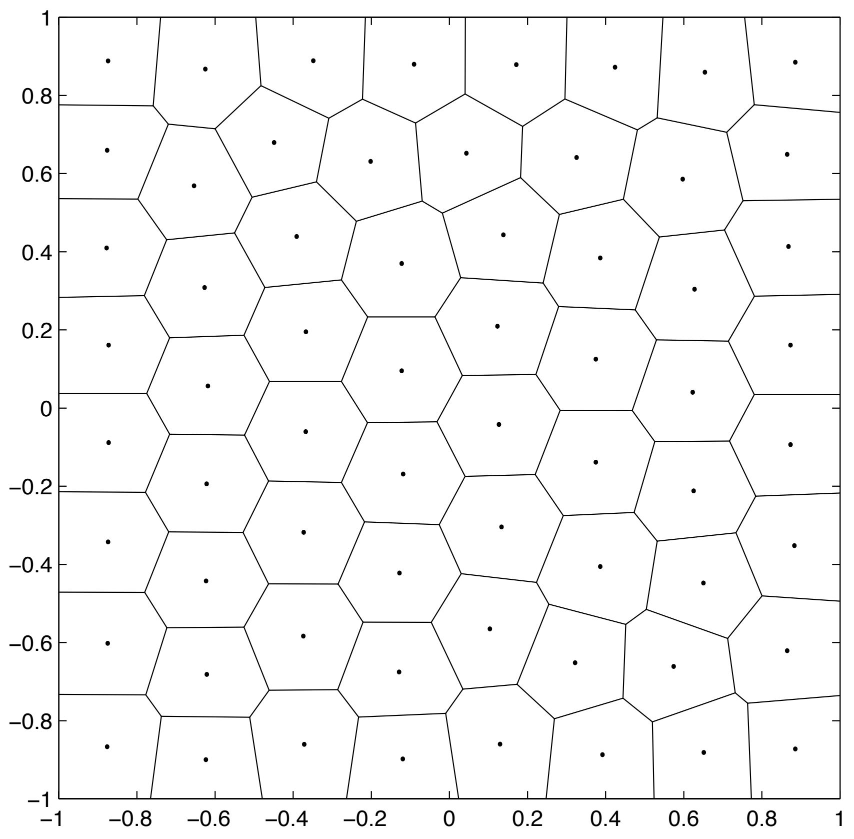
Effect of the density function

- Constant $\rho = 1$
- Peaked at the center $\rho = e^{-x^2-y^2}$
- A more pronounced peak at the center $\rho = e^{-10x^2-10y^2}$
- A very pronounced peak at the center plus a small variation throughout the picture
  $$\rho = e^{-20x^2-20y^2} + 0.05\sin^2(\pi x)\sin^2(\pi y)$$
- An even more pronounced peak at the center plus a smaller variation throughout the picture
  $$\rho = e^{-40x^2-40y^2} + 0.001\sin^2(\pi x)\sin^2(\pi y)$$
- Peaked at the lower left corner $\rho = e^{-2x-2y}$

$\rho = 1$ 64 generators

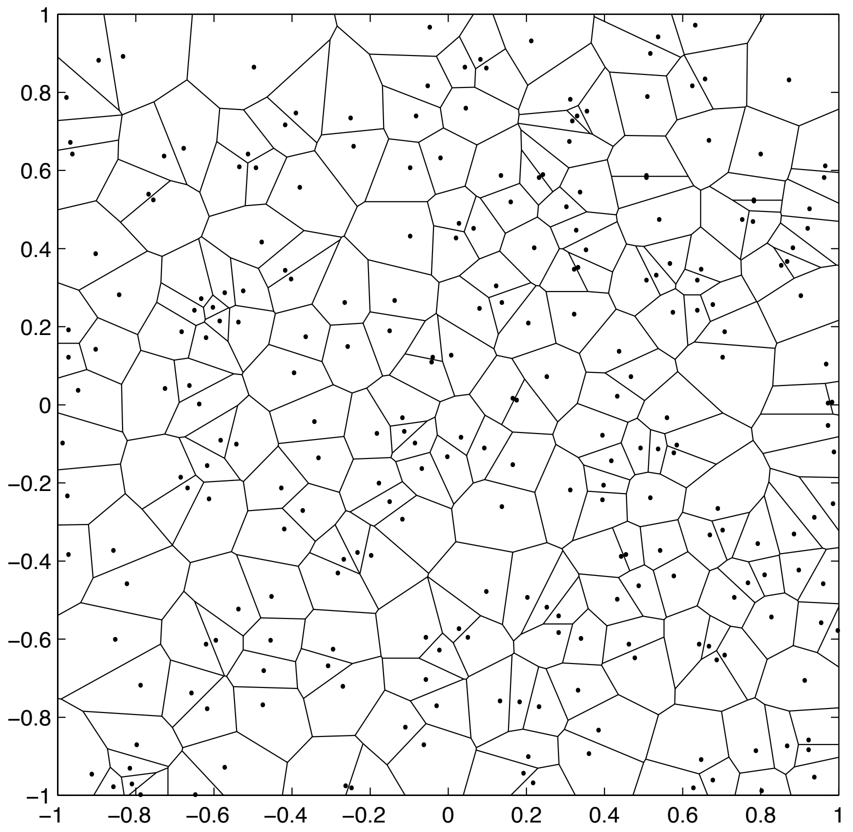


Monte Carlo

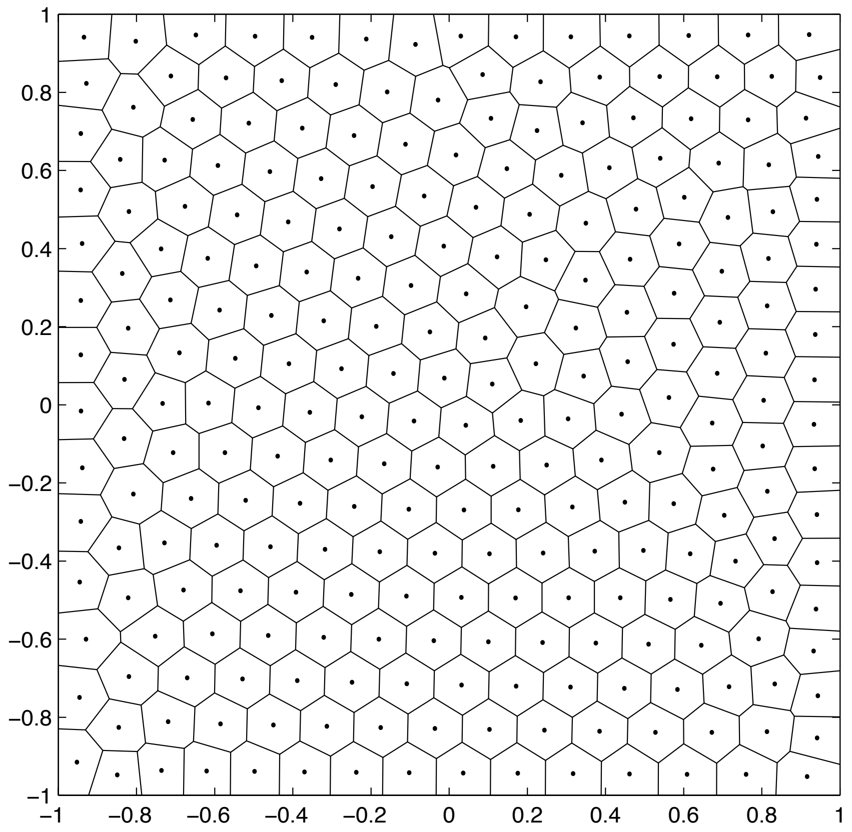


Centroidal Voronoi

$\rho = 1$ 256 generators

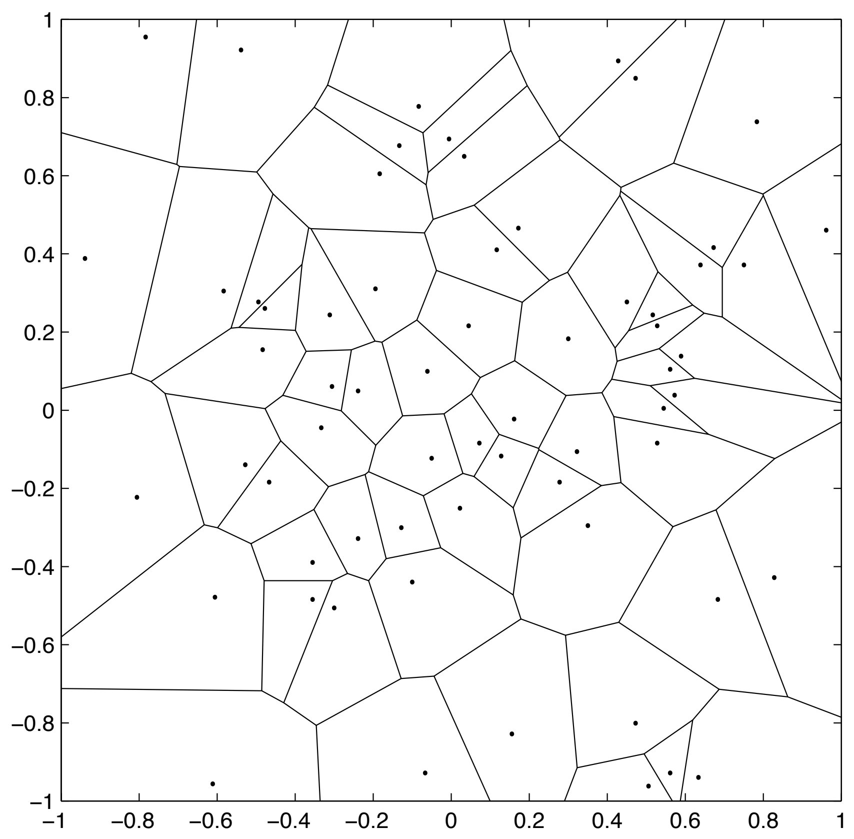


Monte Carlo

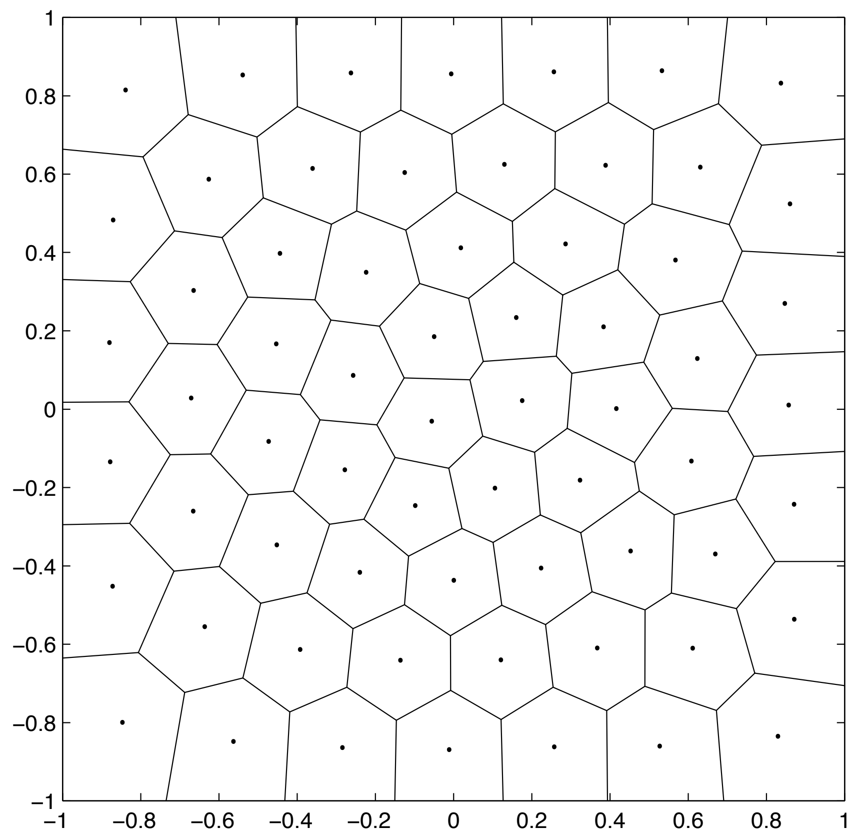


Centroidal Voronoi

$\rho = e^{-x^2-y^2}$ 64 generators

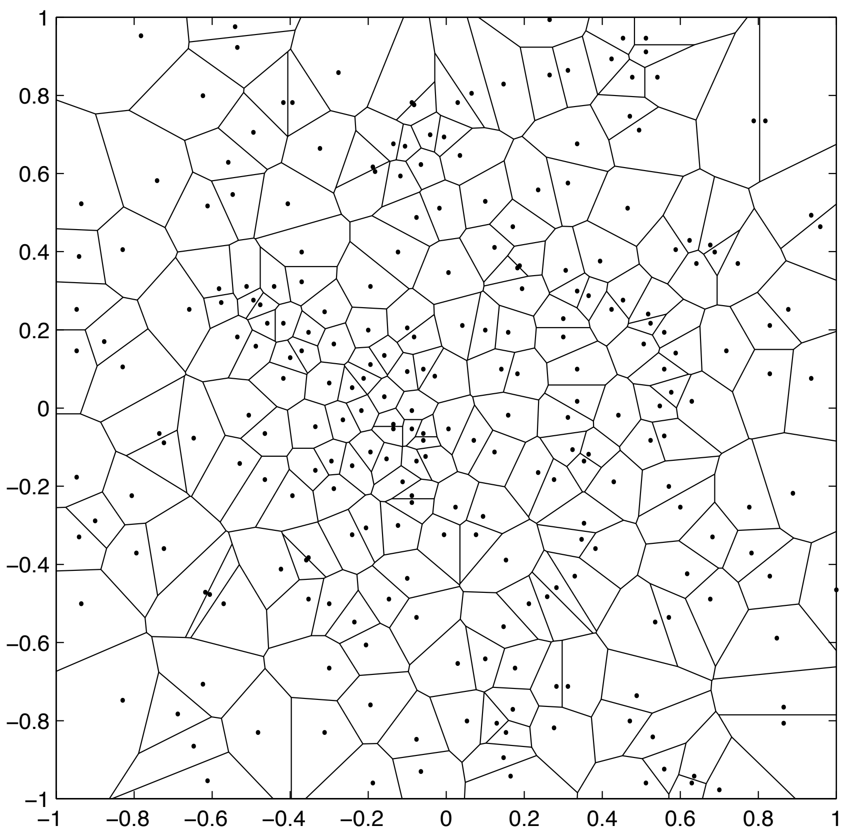


Monte Carlo

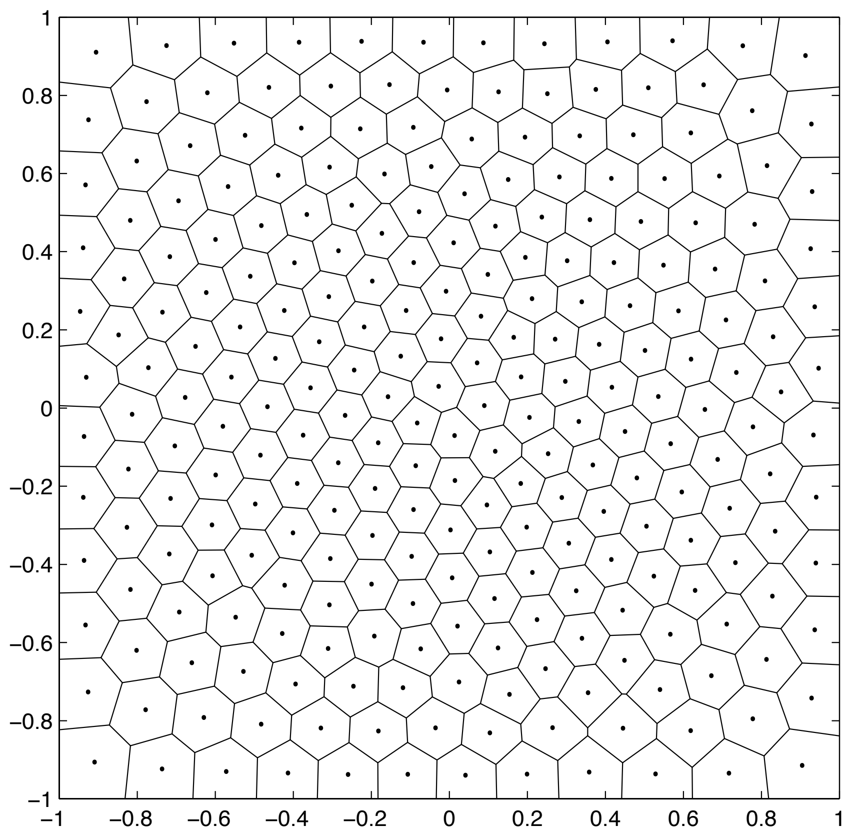


Centroidal Voronoi

$\rho = e^{-x^2-y^2}$ 256 generators

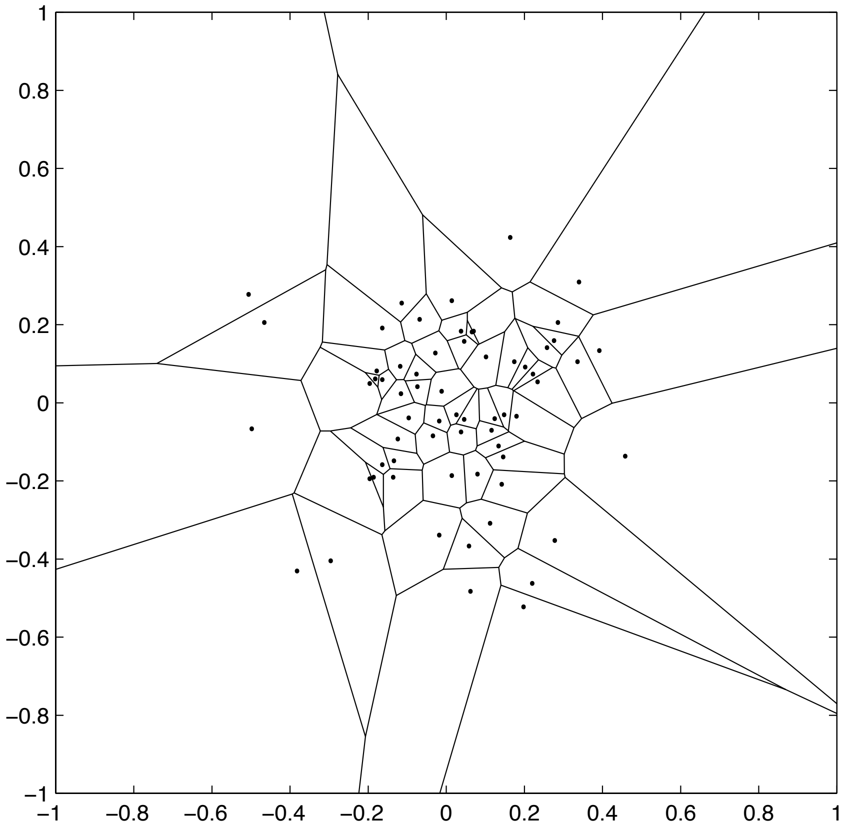


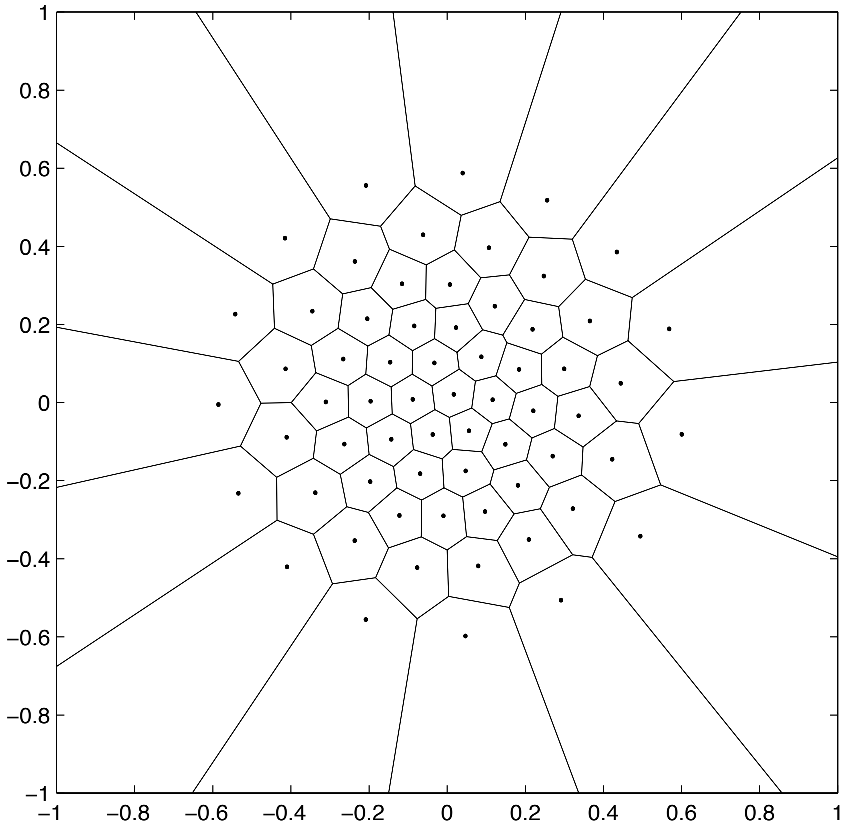
Monte Carlo



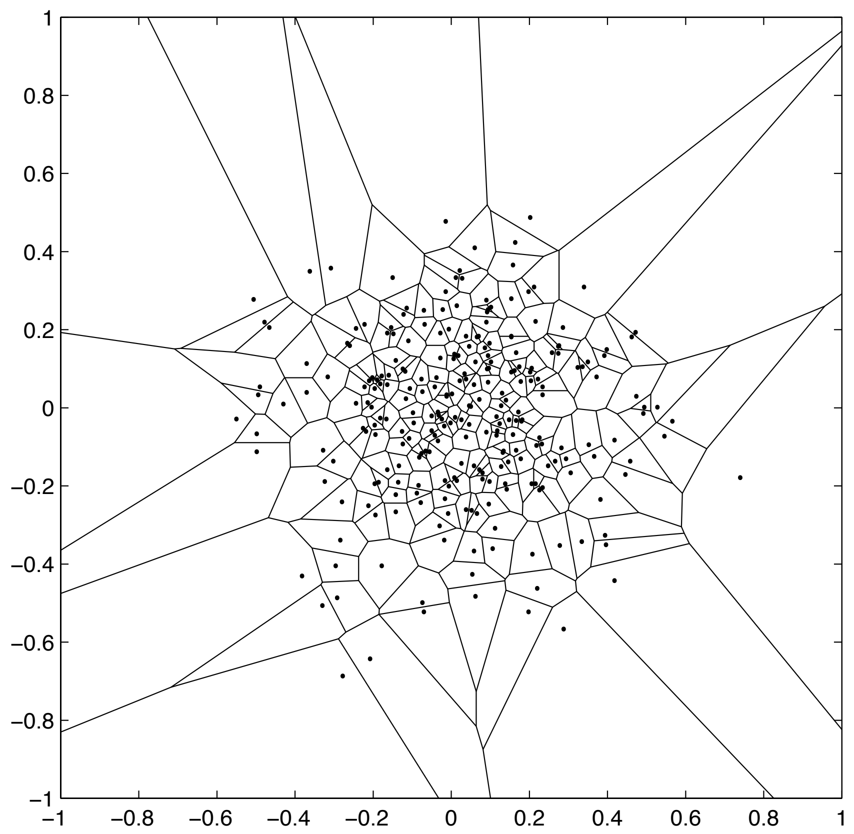
Centroidal Voronoi

$\rho = e^{-10(x^2+y^2)}$ 64 generators

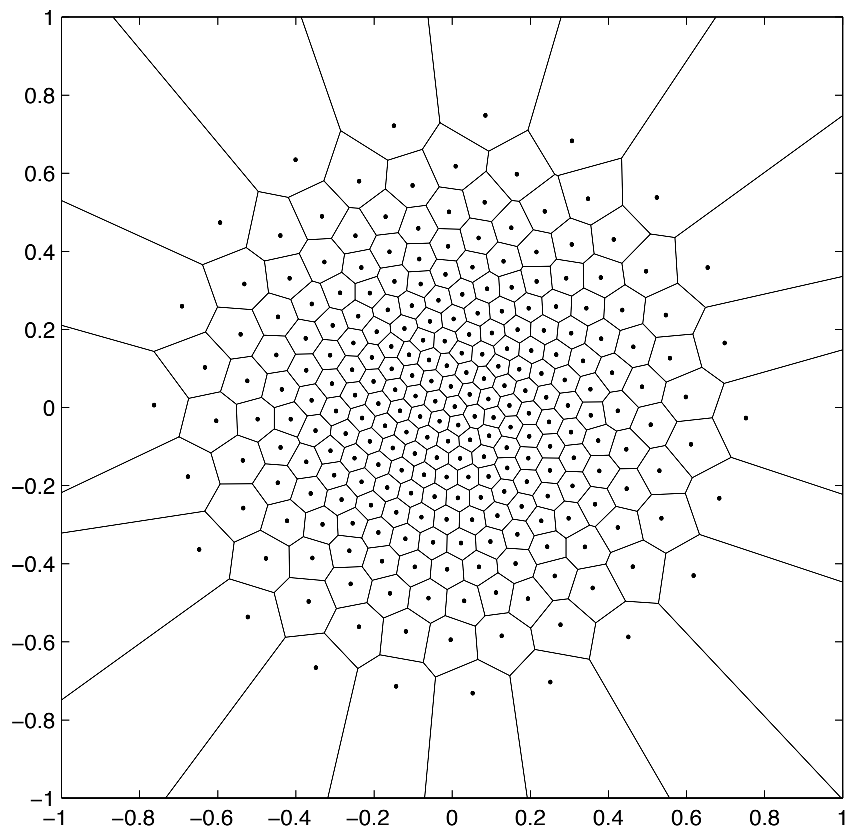

Monte Carlo


Centroidal Voronoi

$\rho = e^{-10(x^2+y^2)}$ 256 generators

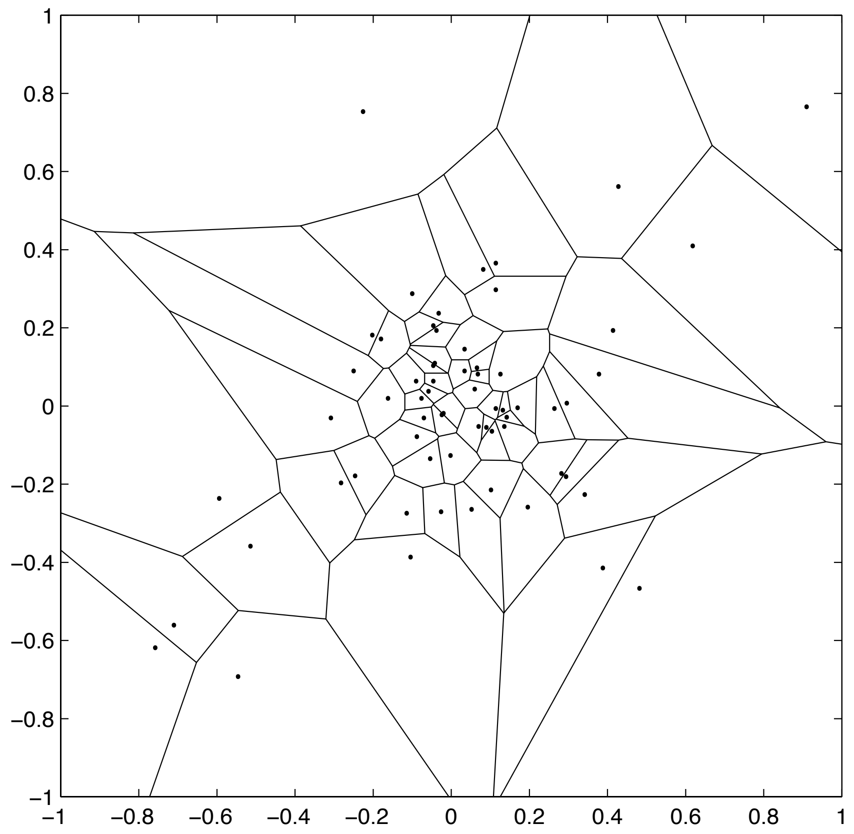


Monte Carlo

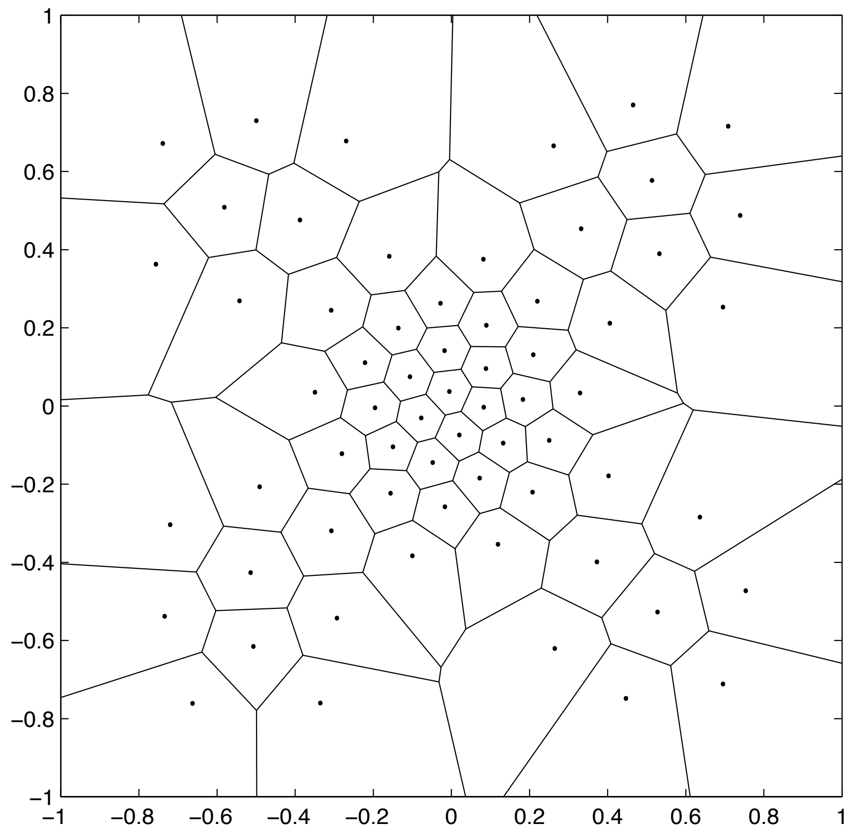


Centroidal Voronoi

$$\rho = e^{-20(x^2+y^2)} + 0.05\sin^2(\pi x)\sin^2(\pi y) \qquad \text{64 generators}$$

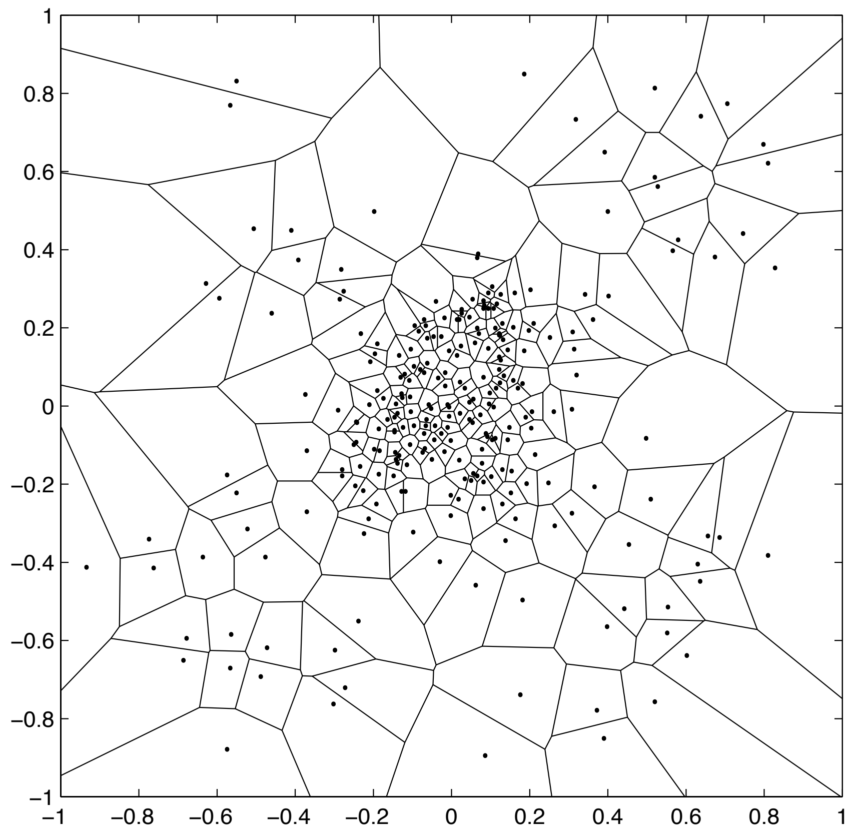


Monte Carlo

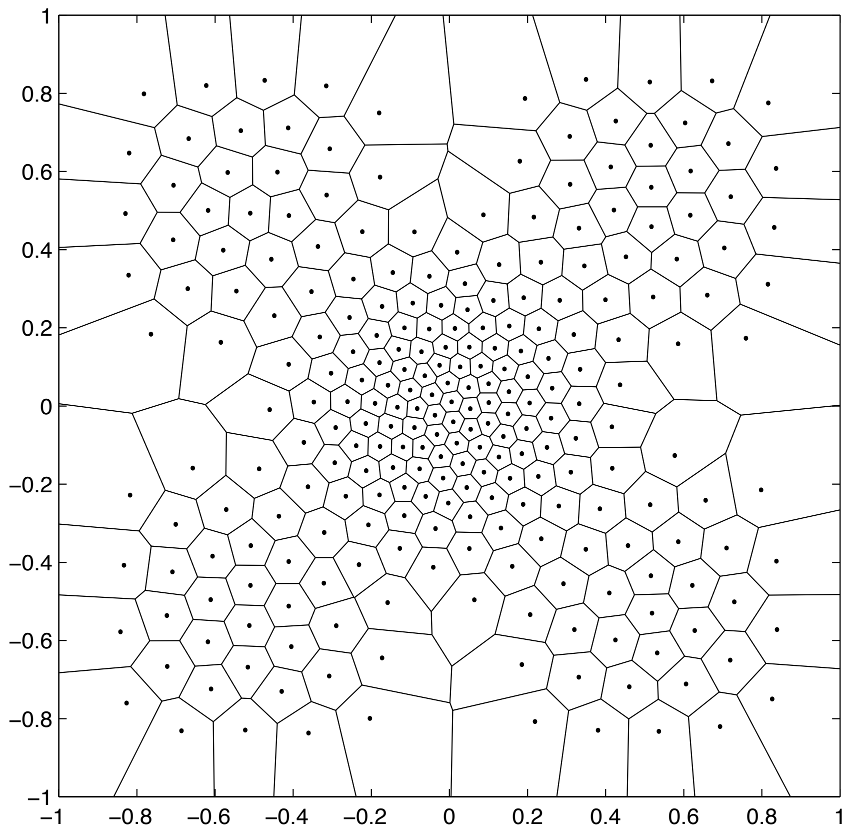


Centroidal Voronoi

$\rho = e^{-20(x^2+y^2)} + 0.05 \sin^2(\pi x) \sin^2(\pi y)$ 256 generators

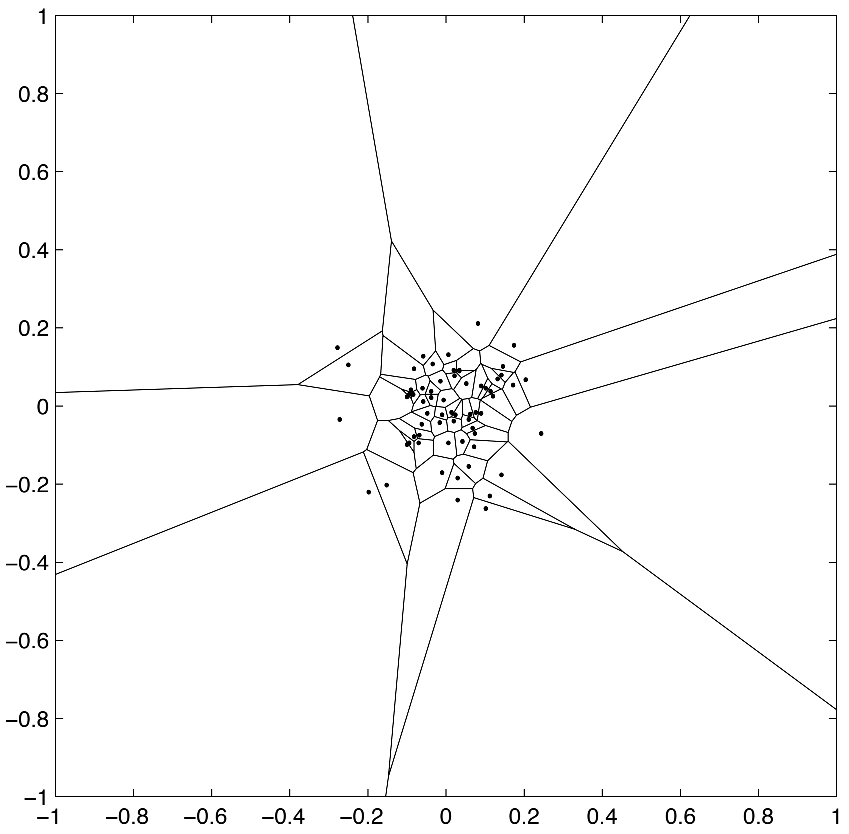


Monte Carlo

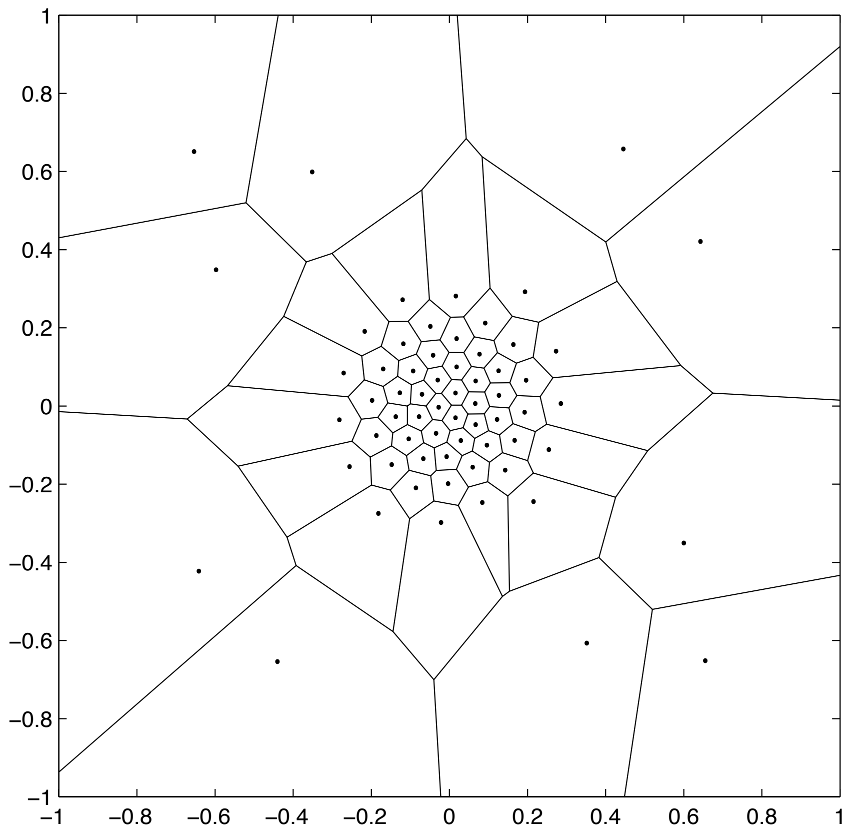


Centroidal Voronoi

$$\rho = e^{-40(x^2+y^2)} + 0.001 \sin^2(\pi x) \sin^2(\pi y) \qquad \text{64 generators}$$

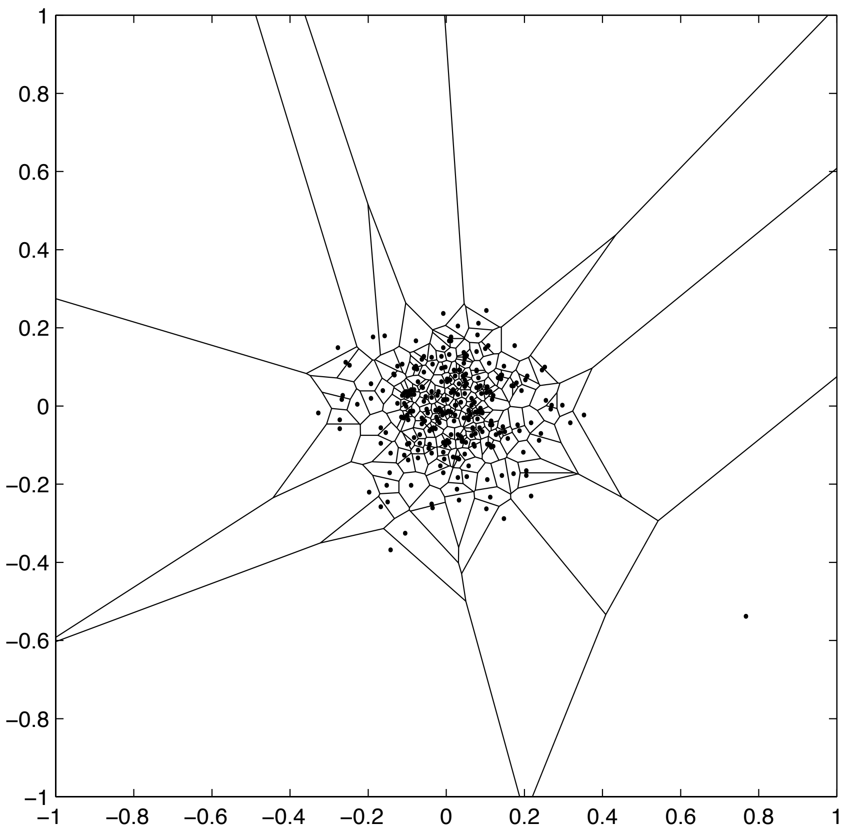


Monte Carlo

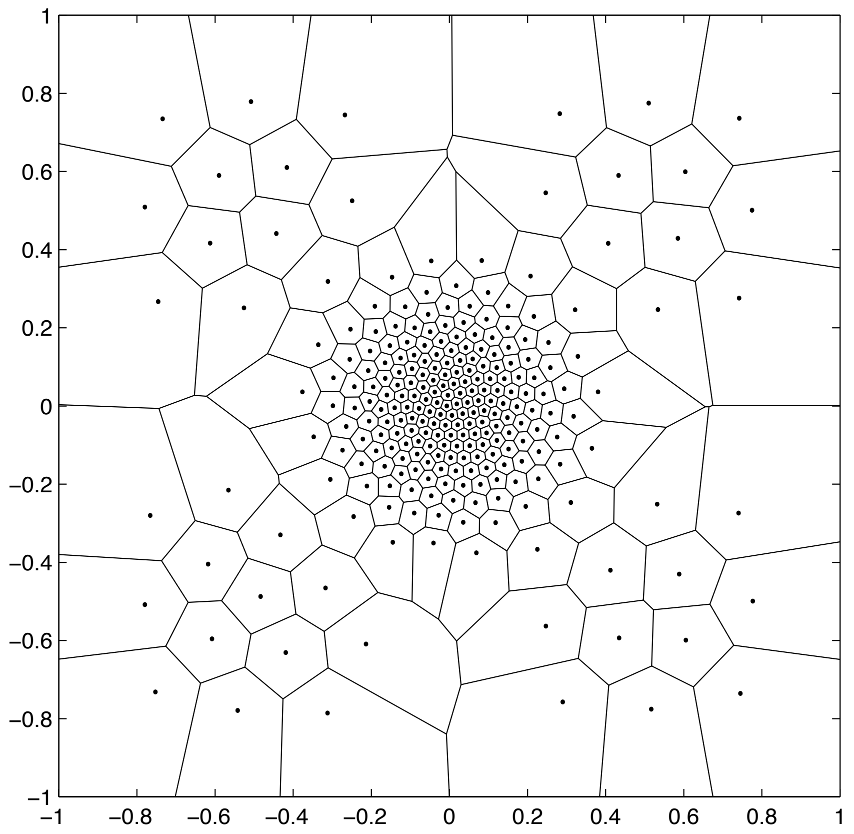


Centroidal Voronoi

$$\rho = e^{-40(x^2+y^2)} + 0.001\sin^2(\pi x)\sin^2(\pi y) \qquad \text{256 generators}$$

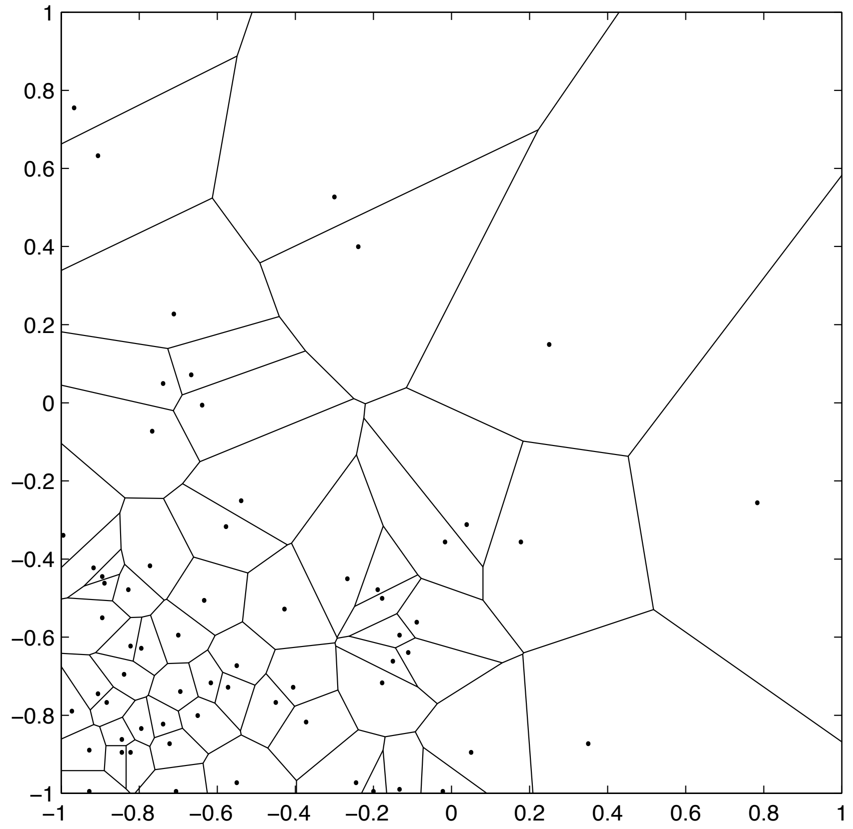


Monte Carlo

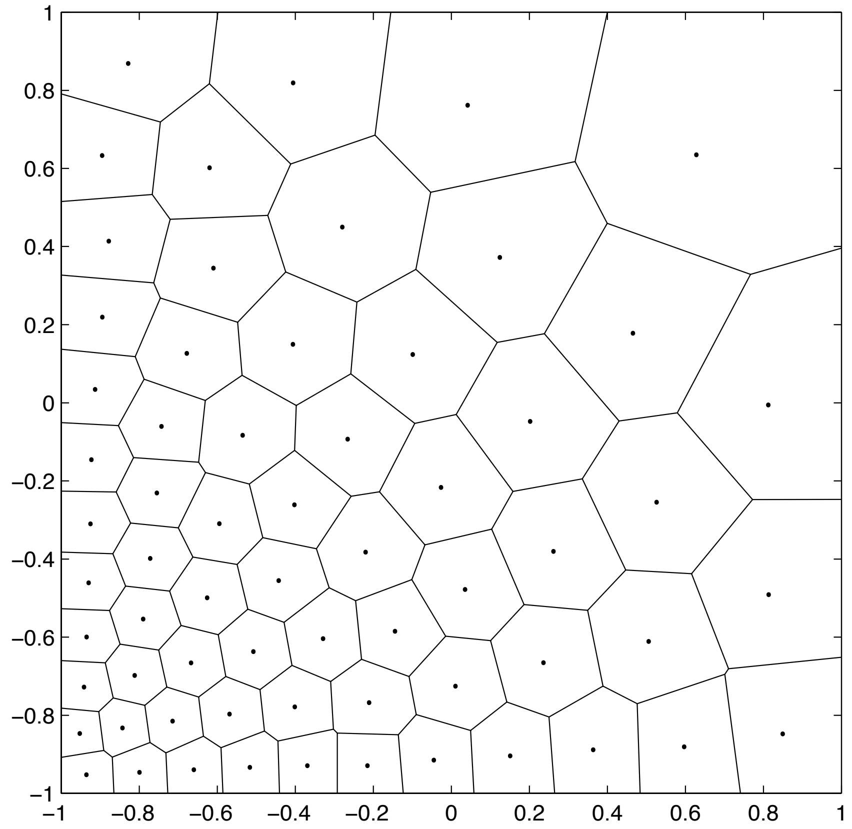


Centroidal Voronoi

$\rho = e^{-2x-2y}$ 64 generators

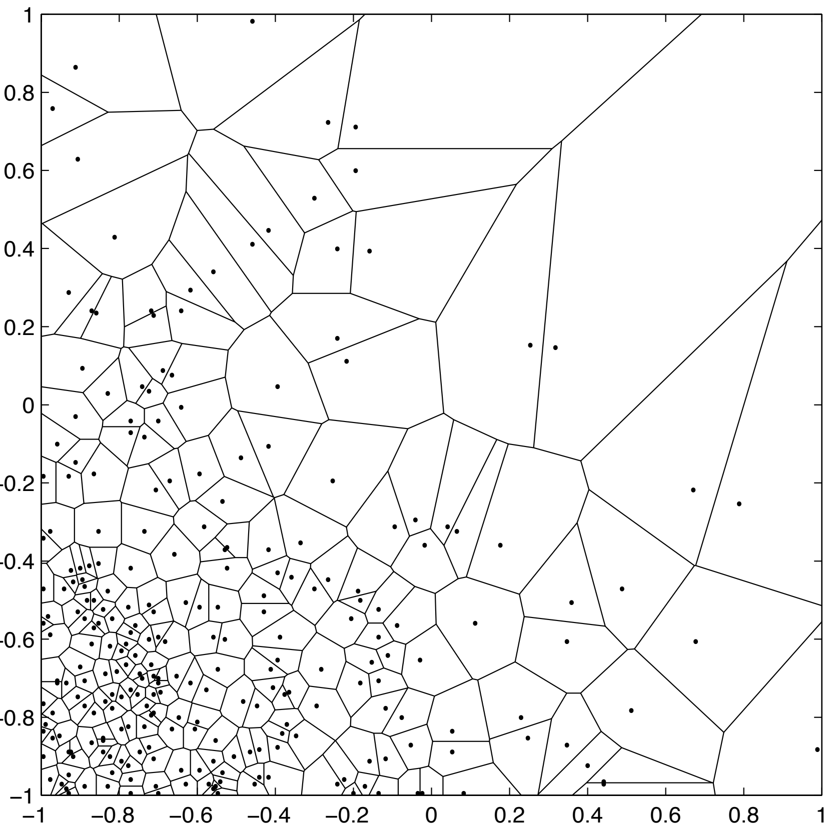


Monte Carlo

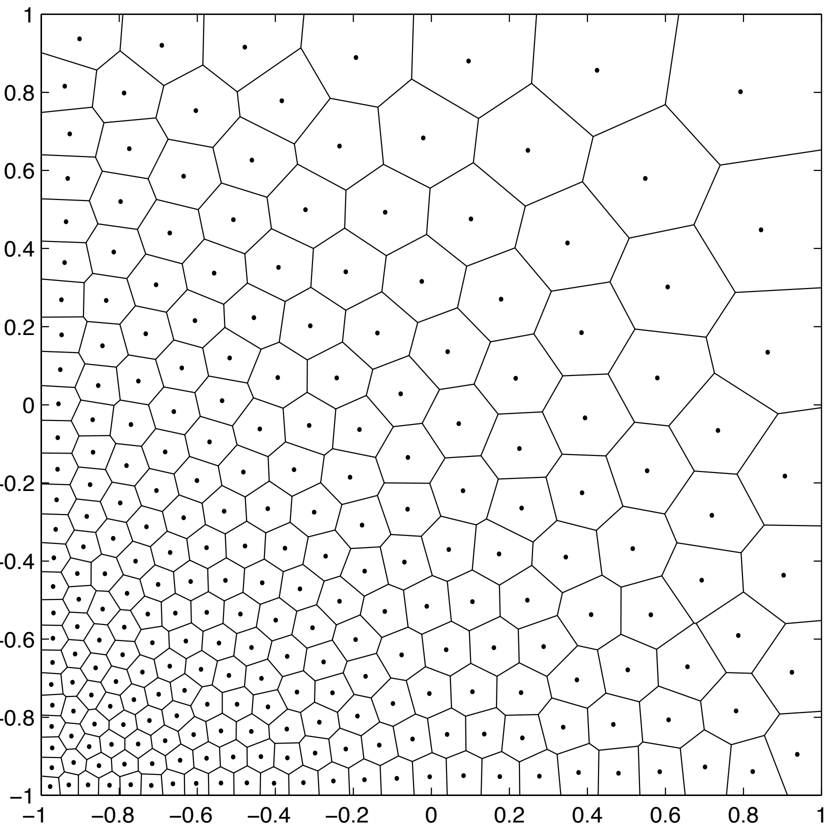


Centroidal Voronoi

$\rho = e^{-2x-2y}$ 256 generators



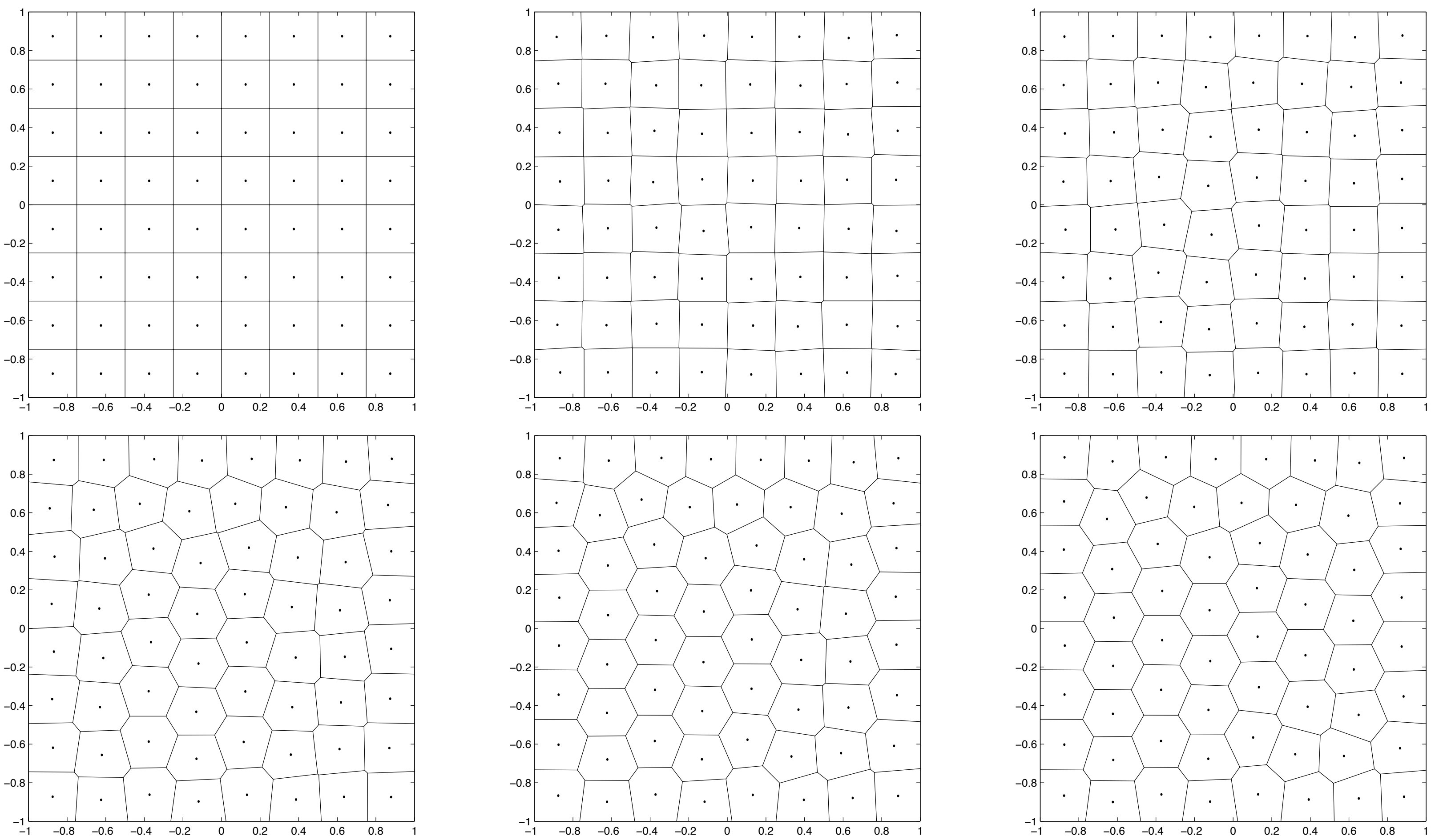
Monte Carlo



Centroidal Voronoi

Stability of centroidal Voronoi tessellations

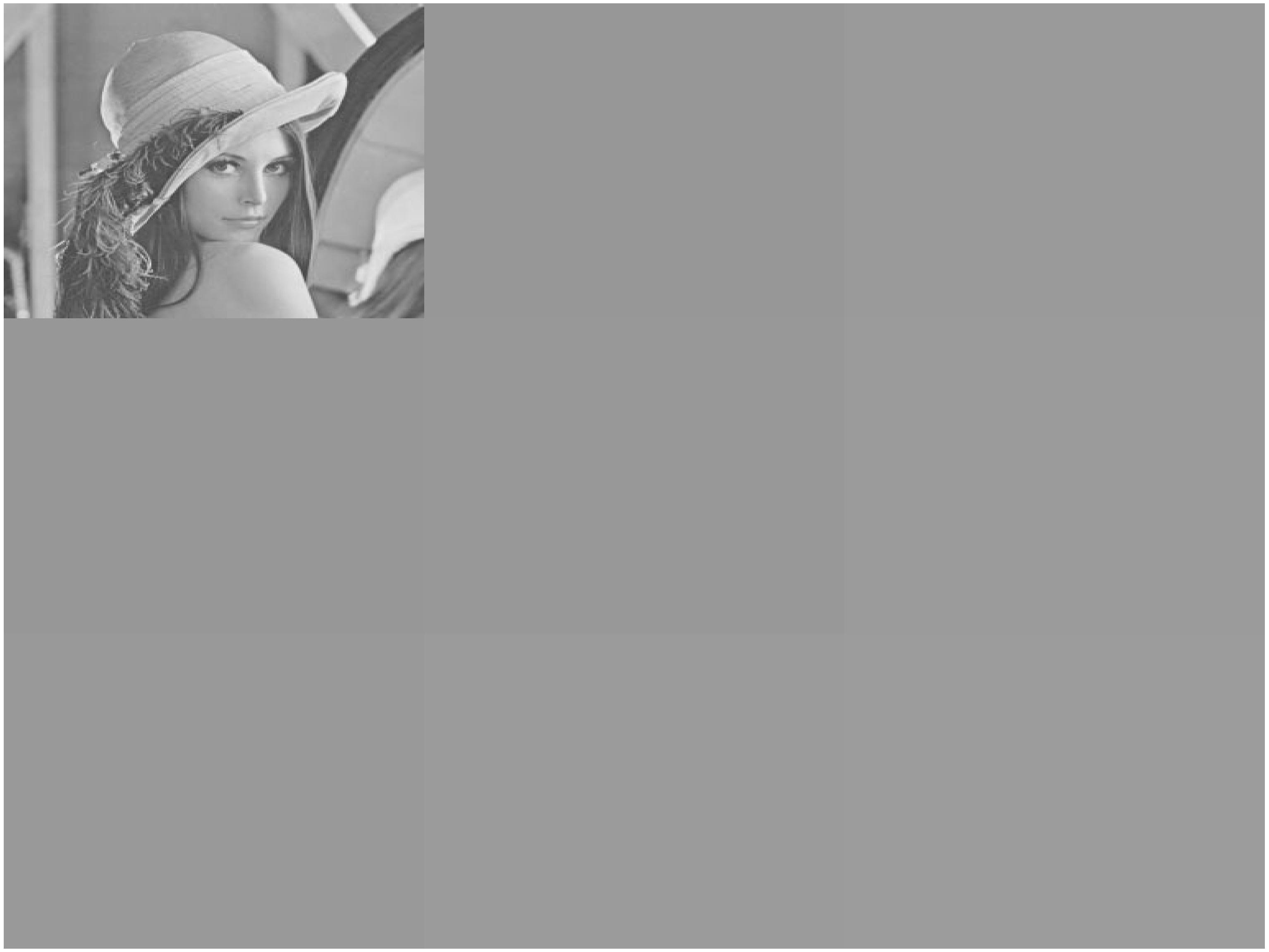
- A small perturbation of the square lattice (64 generators) leads to the hexagonal lattice

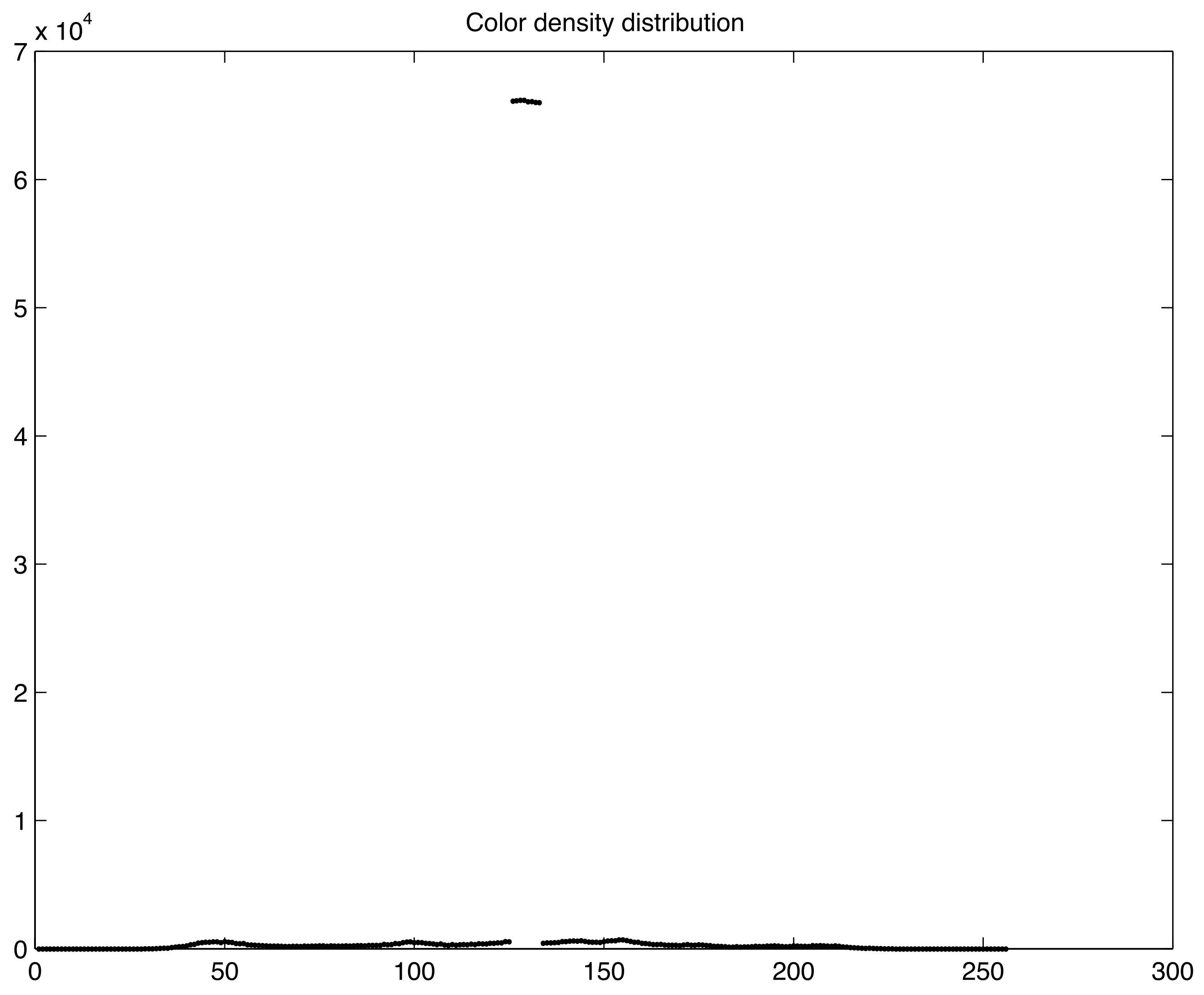


Perturbation (top-right) of a uniform square Voronoi diagram (top-left) with 64 generators leads to a hexagonal-like lattice (bottom-right) with Lloyd's iteration. The pictures are generated at iteration numbers 0, 15, 30, 60, and 120.

Density functions with large peaks

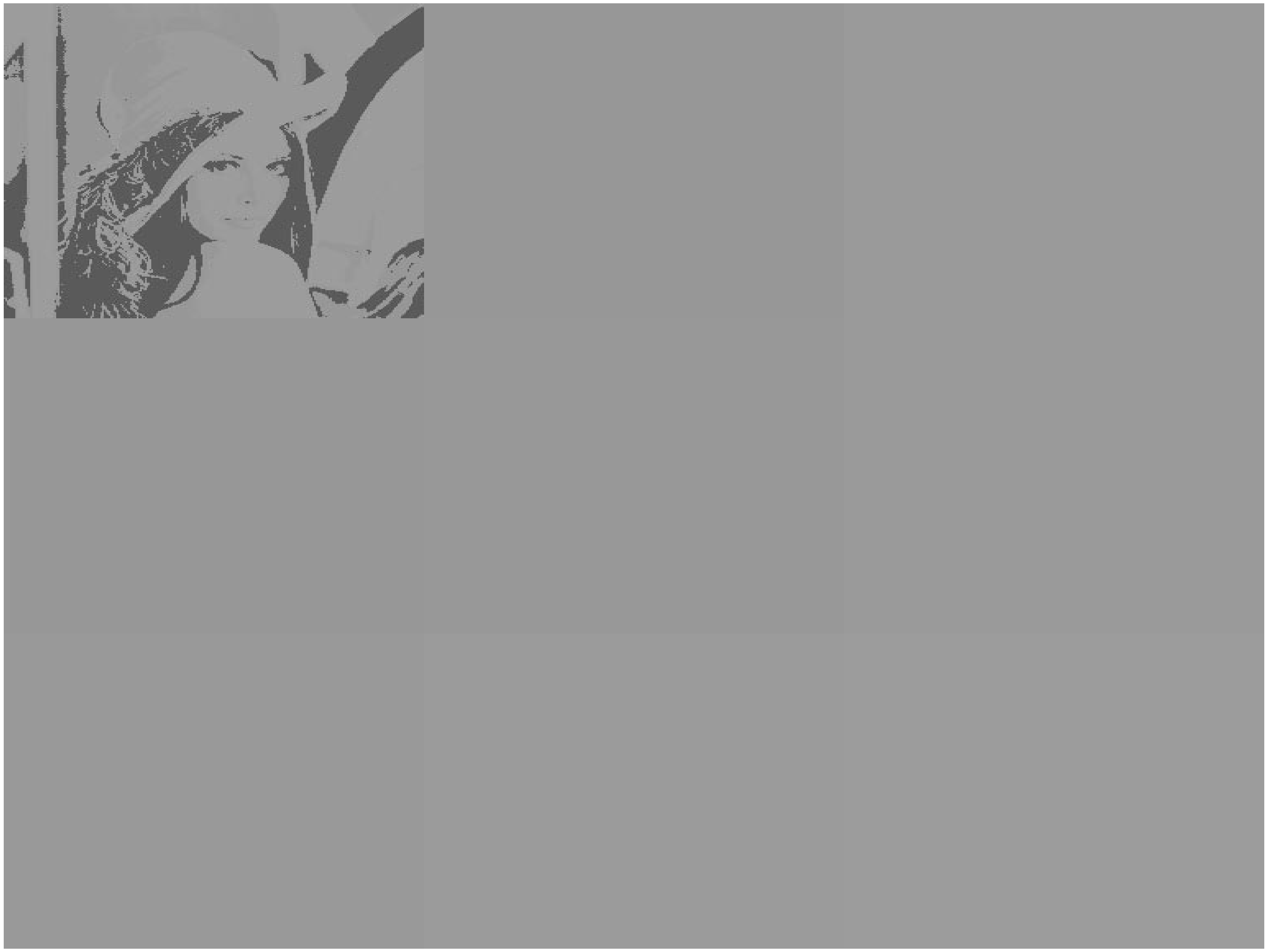
- Let’s return to Lena to examine the effects of density functions having large peaks
- We embed Lena in a picture having a nearly uniform background


The original 8-bit image embedded in a nearly uniform background



The grayscale density function

- Example has 256 gray scales (represented by the integers 0 to 255)
- The background has the 8 contiguous shades (126 to 133)
  - clearly, the background could be well approximated by just one or two shades
- Generators chosen at random tend to cluster near large peaks in the density function; this may not be a good
- A random choice for generators picks 7 shades in the background (126-129 and 131-133) and only 1 shade (50) not in the background
  - results in a very bad approximate picture
- The centroidal Voronoi generators contain only two shades in the background (127 and 132) and 6 shades not in the background (49, 76, 101, 154, 177, 205)
  - there are now more shades available to approximate the interesting part of the picture
  - results in a much better approximate picture



The Monte Carlo compressed 3-bit image

The centroidal Voronoi compressed 3-bit image

The original 8-bit image embedded in a nearly uniform background

ALGORITHMS FOR CONSTRUCTING CENTROIDAL VORONOI TESSELLATIONS

Lloyd's method

0. Start with some initial set of K points $\{z_i\}_{i=1}^K$, e.g., using a Monte Carlo method

1. Construct the Voronoi tessellation $\{V_i\}_{i=1}^K$ of Ω associated with the points $\{z_i\}_{i=1}^K$

2. Compute the mass centroids of the Voronoi regions $\{V_i\}_{i=1}^K$ found in Step 1; these centroids are the new set of points $\{z_i\}_{i=1}^K$

3. Go back to Step 1, or, if happy with convergence, quit

McQueen's method (a random sampling algorithm)

(Doesn't require the calculation of Voronoi sets)

0. Start with some initial set of K points $\{z_i\}_{i=1}^{K}$, e.g., using a Monte Carlo method; set the integer array $J_i = 1$ for $i = 1, \ldots, K$

1. Pick a point $w \in \Omega$ at random according to the probability distribution $\rho(w)$ (normalized)

2. Find the z_i closest to w; denote the index of that z_i by i^*

3. Set

$$z_{i^*} \leftarrow \frac{J_{i^*} z_{i^*} + w}{J_{i^*} + 1} \qquad \text{and} \qquad J_{i^*} \leftarrow J_{i^*} + 1$$

(Note that J_i keeps track of how many times the point z_i has been updated)

4. Go back to Step 1, or, if happy with convergence, quit

A random 2-clustering algorithm for the discrete case

Given a finite set of points $W = \{y_\ell\}_{\ell=1}^{M}$

0. Sample an initial subset T of m points from W, e.g., by using a Monte Carlo method.

1. For every linearly separable 2-clustering (T_1, T_2) of T,
   - compute the centroids t_1 and t_2 of T_1 and T_2, respectively;
   - find a 2-clustering (W_1, W_2) of W divided by the perpendicular bisector of the line segment connecting t_1 and t_2;
   - compute the total variance of the 2-clustering (W_1, W_2) and maintain the minimum among these values.

2. Output the 2-clustering of W with minimum value above.

Many other algorithms are known

- Gradient based deterministic algorithms
- Newton and quasi-Newton methods
- For the discrete case, one can change McQueen's method so that one goes through all the elements of the set W sequentially; get a deterministic algorithm
- Other probabilistic and deterministic algorithms

CENTROIDAL VORONOI TESSELLATIONS AND THEIR MINIMIZATION PROPERTIES

Centroidal Voronoi tessellations as minimizers

Proposition – Given $\Omega \subset R^N$, a positive integer K, and a density function $\rho(\cdot)$ defined on $\overline{\Omega}$. Let $\{\mathbf{z}_i\}_{i=1}^K$ denote any set of K points belonging to $\overline{\Omega}$ and let $\{V_i\}_{i=1}^K$ denote any tessellation of Ω into K regions. Let

$$\mathcal{F}\big((\mathbf{z}_i, V_i), i = 1, \ldots, K\big) = \sum_{i=1}^{K} \int_{\mathbf{y} \in V_i} \rho(\mathbf{y}) |\mathbf{y} - \mathbf{z}_i|^2 \, d\mathbf{y} \, .$$

A necessary condition for $\mathcal{F}$ to be minimized is that the V_i's are the Voronoi regions corresponding to the $\mathbf{z}_i$'s and, *simultaneously*, the $\mathbf{z}_i$'s are the centroids of the corresponding V_i's.

Proposition – Given $\Omega \subset R^N$, a positive integer K, and a density function $\rho(\cdot)$ defined on $\overline{\Omega}$. Given a set of points $\{\mathbf{z}_i\}_{i=1}^K$, let $\{\widehat{V}_i\}_{i=1}^K$ denote the corresponding Voronoi regions. Let

$$\mathcal{K}(\mathbf{z}_i, i = 1, \ldots, K) = \sum_{i=1}^{K} \int_{\mathbf{y} \in \widehat{V}_i} \rho(\mathbf{y}) |\mathbf{y} - \mathbf{z}_i|^2 \, d\mathbf{y} \, .$$

Then, $\mathcal{F}$ and $\mathcal{K}$ have the same minimizer.

Existence of a minimizer

Theorem – If $\Omega \in R^N$ is bounded, then $\mathcal{K}$ has a global minimizer.

Proposition – Assume that $\rho(\cdot)$ is positive except on a set of measure zero in Ω. Then $\mathbf{z}_i \neq \mathbf{z}_j$ for $i \neq j$.

Remark – For general metrics, existence is provided by the compactness of the Voronoi regions. Uniqueness can also be attained under some assumptions, e.g., convexity, on the Voronoi regions and the metric.

Results for the discrete case

There are many results available for the discrete case. Many of these are in the nature of limiting results as the sample size increases.

LLOYD'S METHOD AS A FIXED POINT ITERATION

- Let $\Omega \subset R^N$
- Let the mappings $G_i : R^{KN} \to R^K$, $i = 1, \ldots, K$ be given by

$$G_i(\mathbf{z}) = \frac{\displaystyle\int_{V_i(\mathbf{z})} w\rho(w)\,dw}{\displaystyle\int_{V_i(\mathbf{z})} \rho(w)\,dw}$$

where

$$\mathbf{z} = (z_1, z_2, \ldots, z_K)^T$$

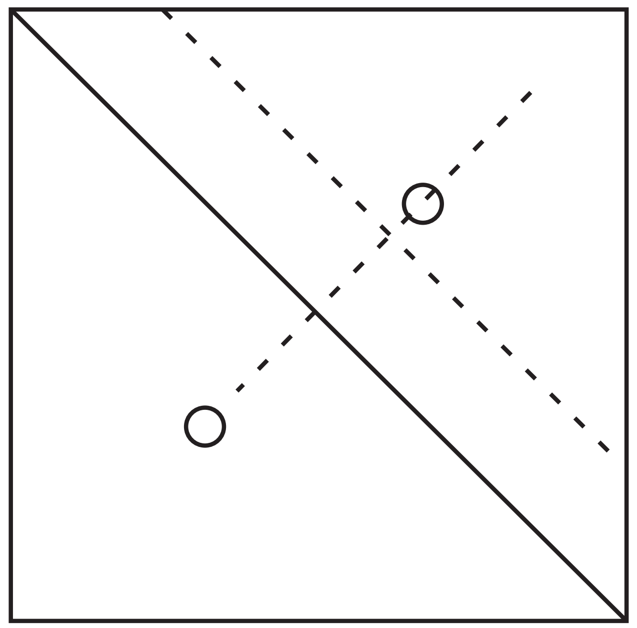
and

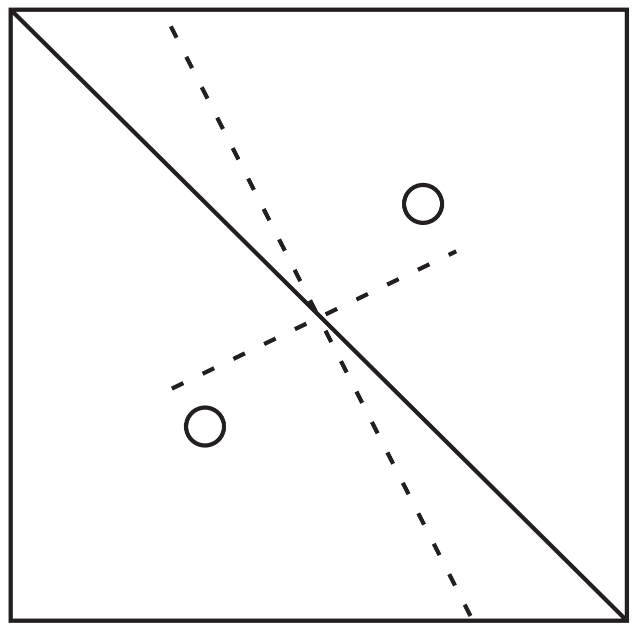
$$V_i(\mathbf{z}) = \text{Voronoi regions for } z_i$$

- Let the mapping $\mathbf{G}(\cdot) : R^{KN} \to R^{KN}$ be defined by

$$\mathbf{G} = (G_1, G_2, \ldots, G_K)^T$$

- A centroidal Voronoi tessellation is a fixed point of the Lloyd map $\mathbf{G}$
- Not all fixed points of the Lloyd map correspond to minimizers





Two possible moves from the saddle point of the Voronoi tessellations of a square. The left move increases the energy while the right move decreases the energy.

Consider the fixed point iteration

$$\mathbf{z}^{(n+1)} = \mathbf{G}(\mathbf{z}^{(n)})$$

i.e., for $i = 1, \ldots, K$,

$$z_i^{(n+1)} = \frac{\displaystyle\int_{V_i(\mathbf{z}^{(n)})} w\rho(w)\,dw}{\displaystyle\int_{V_i(\mathbf{z}^{(n)})} \rho(w)\,dw}$$

- *A fixed point exists for any density function*

$$\mathbf{G}(\mathbf{z}) \,:\, \mathcal{D} \subset R^{KN} \to \mathcal{D} \quad \text{and is continuous}$$

- *The iteration converges for any density function*

$$\mathcal{E}(\mathbf{z}^{(n+1)}) < \mathcal{E}(\mathbf{z}^{(n)}) \quad \text{and} \quad \mathcal{E}(\mathbf{z}^{(n)}) \geq 0$$

- *The convergence rate is, in general, no better than linear, i.e.*

$$|\mathbf{z}^{(n+1)} - \mathbf{z}| \le \alpha |\mathbf{z}^{(n)} - \mathbf{z}|\,, \quad 0 < \alpha < 1$$

$$\text{look at } N = 1 \text{ and } \rho(w) = 1$$

- *With certain restrictions on the density function, convergence is linear*

$$\text{spectral radius}(\mathbf{G}') < 1 \text{ at the fixed point}$$

- *Results hold for functionals of the form*

$$\sum_{i=1}^{N} \int_{V_i} \rho(w) |w - z_i|^p \, dw$$

and

$$\sum_{i=1}^{N} \sum_{w_j \in V_i} \rho(w_j) |w_j - z_i|^p$$

Details may be found in:

Qiang Du, Vance Faber, and Max Gunzburger; Centroidal Voronoi tessellations: Applications and algorithms; *SIAM Review* **41** 1999, 637–676.

Some of these results about Lloyd's method were previously obtained by Gray, Kieffer, and Linde (1980) (and others)

Convergence results for McQueen's random sampling algorithm were obtained by MacQueen (1967)

NEW ALGORITHMS FOR DETERMINING CENTROIDAL VORONOI TESSELLATIONS

Algorithm 1 McQueen's method (of course, this is not new)

0. Choose an initial set of k points $\{z_i\}_{i=1}^k$, e.g., by using a Monte Carlo method; set $j_i = 1$ for $i = 1, \ldots, k$;
1. determine a point y in Ω at random, according to the probability density function $\rho(x)$;
2. find a z_{i^*} among $\{z_i\}_{i=1}^k$ that is the closest to y;
3. set
$$z_{i^*} \leftarrow \frac{j_{i^*} z_{i^*} + y}{j_{i^*} + 1} \qquad \text{and} \qquad j_{i^*} \leftarrow j_{i^*} + 1$$
the new z_{i^*}, along with the unchanged $\{z_j\}, j \neq i^*$, form the new set of points $\{z_i\}_{i=1}^k$;
4. if this new set of points meets some convergence criterion, terminate; otherwise, return to step 1.

Generating sampling points for nonuniform densitities

Before continuing, a few words about how we generate sampling points from a given density function $\rho(x)$

- Classical procedure in 1D

  Given the interval $[a,b]$ and the density function $\rho(x)$ defined on $[a,b]$, a random point x in $[a,b]$ is determined as follows:

  1. sample a random point X with constant density in [0,1];
  2. solve for x which satisfies $\dfrac{\displaystyle\int_a^x \rho(s)\,d\,s}{\displaystyle\int_a^b \rho(s)\,d\,s} = X$

Very expensive due to the need to do repeated numerical integrations

- The rejection method in 1D (a completely probabilistic procedure)

  Given the interval $[a, b]$ and the density function $\rho(x)$ defined on $[a, b]$, set $\widehat{\rho} = \max_{x\in[a,b]} \rho(x)$.

Then, a random point x in $[a, b]$ is determined as follows:

1. sample a random point X' with constant density in [0,1];
2. set $X = a + (b - a)X'$;
3. sample a random point U with constant density in [0,1];
4. if $U < \rho(X)/\widehat{\rho}$, let $x = X$; otherwise, return to step 1.

Although we may need to call the random number generator many times (the number of times depend on the density function $\rho(x)$), the total computation time is in general trivial compared to the classical procedure using numerical integrations.

The two procedures are theorticaly equivalent when applied to a constant density function.

- The rejection method in 2D

  Given the domain Ω and the density function $\rho(x, y)$ defined on Ω, set $\widehat{\rho} = \max_{(x,y)\in\Omega} \rho(x, y)$. Choose a, b, c, and d such that $\overline{\Omega} \subset [a, b] \times [c, d]$. Set $\rho(x, y) = 0$ in $([a, b] \times [c, d])\backslash\Omega$.

Then, a random point $(x, y) \in \Omega$ is determined as follows:

1. sample a random point X' with constant density in [0,1] and set $X = a + (b - a)X'$;
2. sample a random point Y' with constant density in [0,1] and set $Y = c + (d - c)Y'$;
3. sample a random point U with constant density in [0,1];
4. if $U < \rho(X, Y)/\widehat{\rho}$, set $(x, y) = (X, Y)$; otherwise, return to step 1.

TABLE 1

TABLE 2

TABLE 3

Algorithm 2 Lloyd's method (also not new)

```
0. Select an initial set of $k$ points $\{z_i\}_{i=1}^k$, e.g., by using a Monte Carlo method;
1. construct the Voronoi sets $\{V_i\}_{i=1}^k$ associated with $\{z_i\}_{i=1}^k$;
2. determine the mass centroids of the Voronoi sets $\{V_i\}_{i=1}^k$; these centroids form the new set of points $\{z_i\}_{i=1}^k$;
3. if this new set of points meets some convergence criterion, terminate; otherwise, return to step 1.
```

- Each McQueen iteration is cheap, but lots of them are needed.
- Lloyd's method requires fewer iterations, but each iteration is expensive

Algorithm 3 A probabilistic Lloyd's/generalized McQueen's method

0. Choose a positive integer q and positive constants $\{\alpha_i, \beta_i\}_{i=1}^2$ such that $\alpha_1 + \alpha_2 = 1$, $\beta_1 + \beta_2 = 1$; choose an initial set of k points $\{z_i\}_{i=1}^k$, e.g., by using a Monte Carlo method; set $j_i = 1$ for $i = 1, \dots, k$;
1. choose q points $\{y_r\}_{r=1}^q$ in Ω at random, according to the probability density function $\rho(x)$;
2. for $r = 1, \dots, q$, determine a z_{r^*} among $\{z_i\}_{i=1}^k$ that is closest to y_r;
3. for $i = 1, \dots, k$, gather together in the set W_i all sampling points y_r closest to z_i (i.e., in the Voronoi region of z_i); if the set W_i is empty, do nothing; otherwise, compute the average y_i^* of the set W_i and set

$$z_i^* \leftarrow \frac{(\alpha_1 j_i + \beta_1) z_i + (\alpha_2 j_i + \beta_2) y_i^*}{j_i + 1} \qquad \text{and} \qquad j_i \leftarrow j_i + 1;$$

   the new set of $\{z_i^*\}$, along with the unchanged $\{z_j\}, j \neq i$, form the new set of points $\{z_i\}_{i=1}^k$;
4. if this new set of points meets some convergence criterion, terminate; otherwise, return to step 1.

- For $q = 1$, $\alpha_1 = \beta_2 = 1$, and $\alpha_2 = \beta_1 = 0$, this method reduces to the McQueen's method.

- If $\alpha_1 = \beta_1 = 0$, $\alpha_2 = \beta_2 = 1$, then $z_i^* = y_i^*$, the average of the points in the set W_i;
  - then, since the points in W_i are randomly selected points in the Voronoi region V_i correspoding to z_i, we one may view z_i^* as an probabilistic approximation to the centroid of V_i
  - thus, for $\alpha_1 = \beta_1 = 0$, $\alpha_2 = \beta_2 = 1$, this method is *a probabilistic version of Lloyd's method*;
  - the larger q is, the better the centroid approximations.

- Other choices for $\{\alpha_i, \beta_i\}_{i=1}^2$ define other methods.

- For all three density functions, Algorithm 3 performs much better than McQueen's method; specifically, for the same number of total random sampling points generated or for the same CPU time, Algorithm 3 gives a larger reduction in the energy

TABLE 4

TABLE 5

TABLE 6

TABLE 7

TABLE 8

TABLE 9

Algorithm 4 A parallel McQueen's method

0. Suppose we have p processors with $rank = 0, 1, \ldots, p-1$ respectively, let $m = [\frac{k}{p}]$, $s = [\frac{q}{p}]$; then, each processor independently chooses its own initial set of m points $\{z_i^{rank}\}_{i=1}^m$, e.g., by using a Monte Carlo method; set $j_i^{rank} = 1$ for $i = 1, \ldots, m$;

1. processor 0 selects a $y \in \Omega$ at random, according to the probability density function $\rho(x)$ and then broadcasts it to all other processors;

2. each processor finds that z^{rank} which is the closest to y in $\{z_i^{rank}\}_{i=1}^m$ respectively; determine the corresponding distance d^{rank};

3. compare them by communication to get the set of d^{rank^*}, where d^{rank^*} is the minimum among the d^{rank} of all processors;

4. on each processor, if $d^{rank} \neq d^{rank^*}$, do nothing; otherwise, set

$$z^{rank^*} \leftarrow \frac{j^{rank^*} z^{rank^*} + y}{j^{rank^*} + 1} \qquad \text{and} \qquad j^{rank^*} \leftarrow j^{rank^*} + 1;$$

this new z^{rank^*}, along with the unchanged z_i^{rank}, form the new set of points $\{z_i^{rank}\}_{i=1}^m$;

5. if this new set of points meets some convergence criterion, terminate; otherwise, return to step 1.

This parallel version of McQueen's method is not efficient with respect to communications between procesors

Algorithm 5 A modified parallel McQueen's method

0. Suppose we have p processors with $rank = 0, 1, \ldots, p-1$, respectively; choose a positive integer s; let $m = \left[\frac{k}{p}\right]$; then, each processor independently chooses its own initial set of m points $\{z_i^{rank}\}_{i=1}^m$, e.g., by using a Monte Carlo method; set $j_i^{rank} = 1$ for $i = 1, \ldots, m$;

1. each processor independently selects s points in Ω at random, according to the probability density function $\rho(x)$; combine them together by communication to form a set of sampling points $\{y_r\}_{r=1}^q$ $(q = sp)$, keeping a copy on each processor;

2. on each processor, for $r = 1, \ldots, q$, find a $z_{i^*,r}^{rank}$ among $\{z_i^{rank}\}_{i=1}^m$ that is closest to y_r; determine the corresponding distance $d_{i^*,r}^{rank}$;

3. compare them by communication to obtain the set $\{d_{i^*,r}^{rank^*}\}_{r=1}^q$ where $d_{i^*,r}^{rank^*}$ is the minimum among $d_{i^*,r}^{rank}$ on all processors.

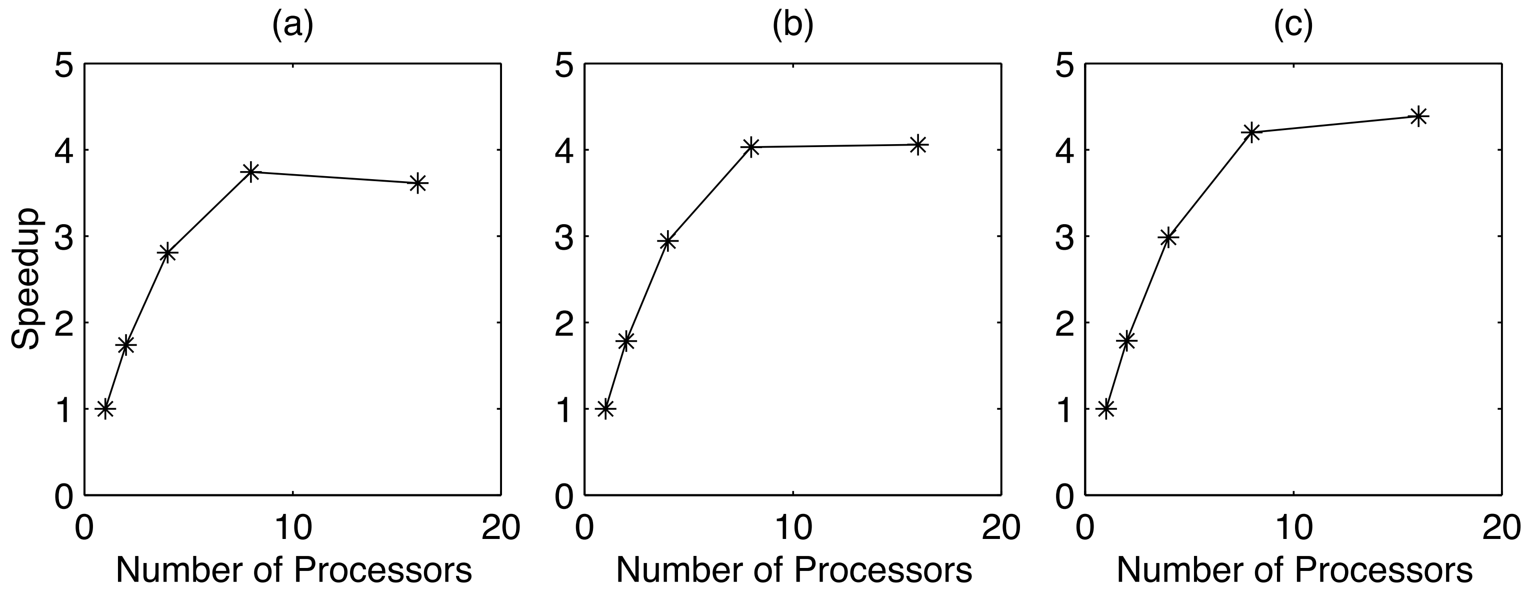
4. for $r = 1, \ldots, q$, on each processor, if $d_{i^*,r}^{rank} \neq d_{i^*,r}^{rank^*}$, do nothing; otherwise, set

$$z_{r^*}^{rank^*} \leftarrow \frac{j_{r^*}^{rank^*} z_{r^*}^{rank^*} + y_r}{j_{r^*}^{rank^*} + 1} \qquad \text{and} \qquad j_{r^*}^{rank^*} \leftarrow j_{r^*}^{rank^*} + 1;$$

   this new set of $\{z_{r^*}^{rank^*}\}_{r=1}^{q}$, along with the unchanged z_i^{rank}, form the new set of points $\{z_i^{rank}\}_{i=1}^{m}$;

5. If this new set of points meets some convergence criterion, terminate; otherwise, return to step 1.

TABLE 10



Speedup of Algorithm 5 with $q = ps = 160$.

(a) $\rho(x) = 1$

(b) $\rho(x) = e^{-10x^2}$

(c) $\rho(x) = e^{-20x^2} + 0.05\sin^2(\pi x)$

Algorithm 6 Parallel version of of algorithm 3

0. Suppose we have p processors with $rank = 0, 1, \ldots, p-1$, respectively; choose a positive integer s and positive constants $\{\alpha_i, \beta_i\}_{i=1}^2$ such that $\alpha_1 + \alpha_2 = 1$,$\beta_1 + \beta_2 = 1$; let $m = \left[\frac{k}{p}\right]$; then, each processor independently chooses its own initial set of m points $\{z_i^{rank}\}_{i=1}^m$, e.g., by using a Monte Carlo method;

1. combine them together by communication to form a complete initial set of k points $\{z_i\}_{i=1}^k$, keeping a copy on each processor; set $j_i = 1$ for $i = 1, \ldots, k$;

2. each processor selects its own set of$s$ points $\{y_r^{rank}\}_{r=1}^s$ in Ω at random according to the probability density function $\rho(x)$;

3. on each processor, for $r = 1, \ldots, s$, find a $z_{i^*,r}^{rank}$ among $\{z_i\}_{i=1}^k$ that is closest to y_r^{rank};

4. on each processor, for each generator z_i, gather together all sampling points y_r^{rank} closest to z_i, (i.e., in the Voronoi region of z_i) in the set W_i^{rank}, and denote the size of W_i^{rank} by w_i^{rank};

5. if the set W_i^{rank} is empty, set $u_i^{rank} = 0$; otherwise, determine the sum u_i^{rank} of the members of the set W_i^{rank};

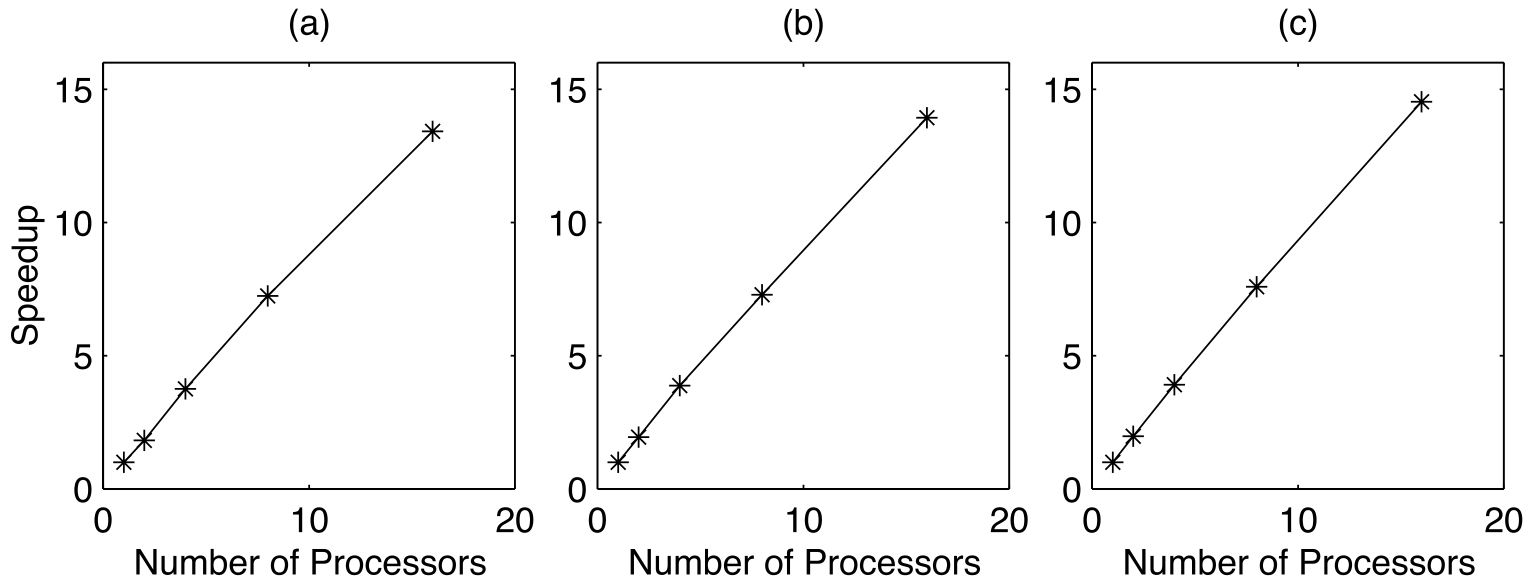
6. by communication, obtain the set $\{u_i\}_{i=1}^k$, where u_i is the sum of u_i^{rank} over all processors, and the corresponding set $\{w_i\}_{i=1}^k$, where w_i is the sum of w_i^{rank} over all processors;

7. on each processor, for $i = 1, \ldots, k$, if $(w_i \neq 0)$ and $(rank * m < i < (rank + 1) * m + 1)$, set
$$z_i^* \leftarrow \frac{(\alpha_1 j_i + \beta_1) z_i + (\alpha_2 j_i + \beta_2) y_i^*}{j_i + 1} \qquad \text{and} \qquad j_i \leftarrow j_i + 1;$$
otherwise, do nothing; by communication, this new set of z_i^*, along with the unchanged z_i, form the new set of points $\{z_i\}_{i=1}^k$;

8. if this new set of points meets some convergence criterion, terminate; otherwise, return to step 1.

TABLE 11

TABLE 12



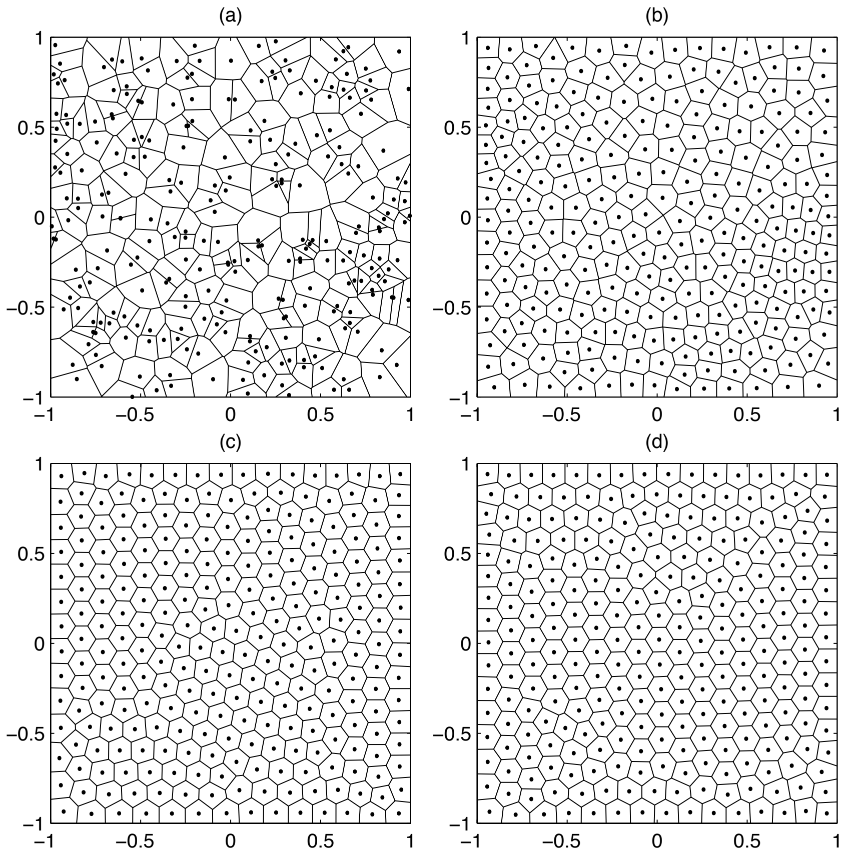
Speedup of Algorithm 6 with $q = ps = 8000$.

(a) $\rho(x) = 1$

(b) $\rho(x) = e^{-10x^2}$

(c) $\rho(x) = e^{-20x^2} + 0.05\sin^2(\pi x)$

TABLE 13



2D Voronoi diagrams for 256 generators in $[-1,1]^2$ with the constant density function

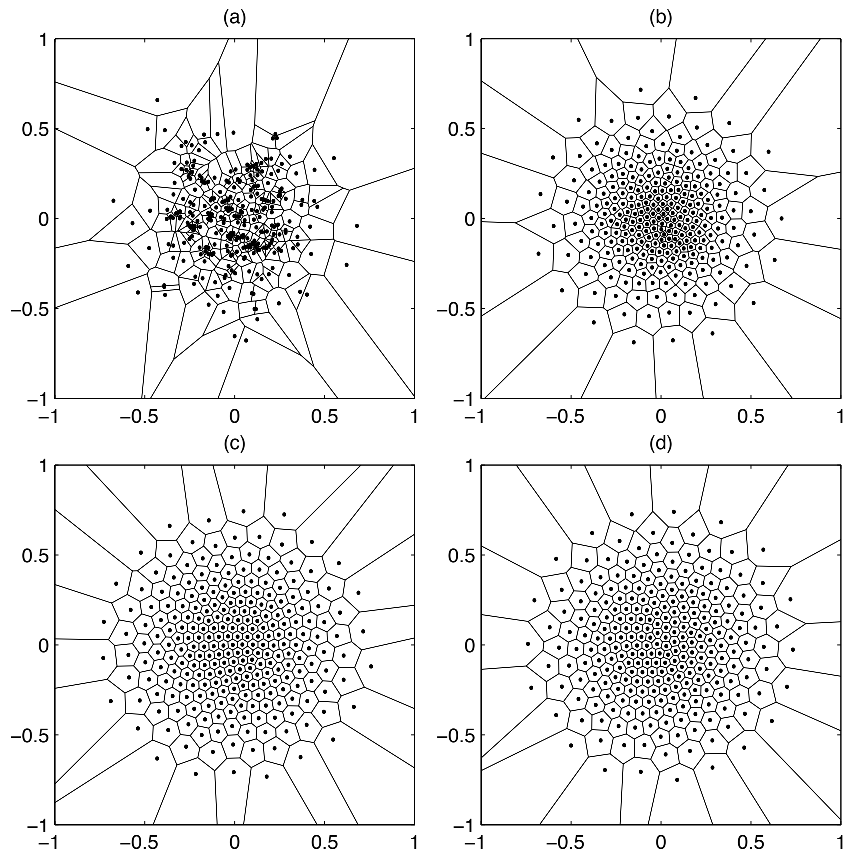
$\rho(x) = 1$

(a) Monte Carlo simulation;

(b) CV using Algorithm 5;

(c) CV using Algorithm 6 with $\alpha_1 = 0$, $\alpha_2 = 1$, $\beta_1 = 0$, $\beta_2 = 1$;

(d) CV using Algorithm 6 with $\alpha_1 = 0.5$, $\alpha_2 = 0.5$, $\beta_1 = 0.5$, $\beta_2 = 0.5$.



2D Voronoi diagrams for 256 generators in $[-1,1]^2$ with the density function

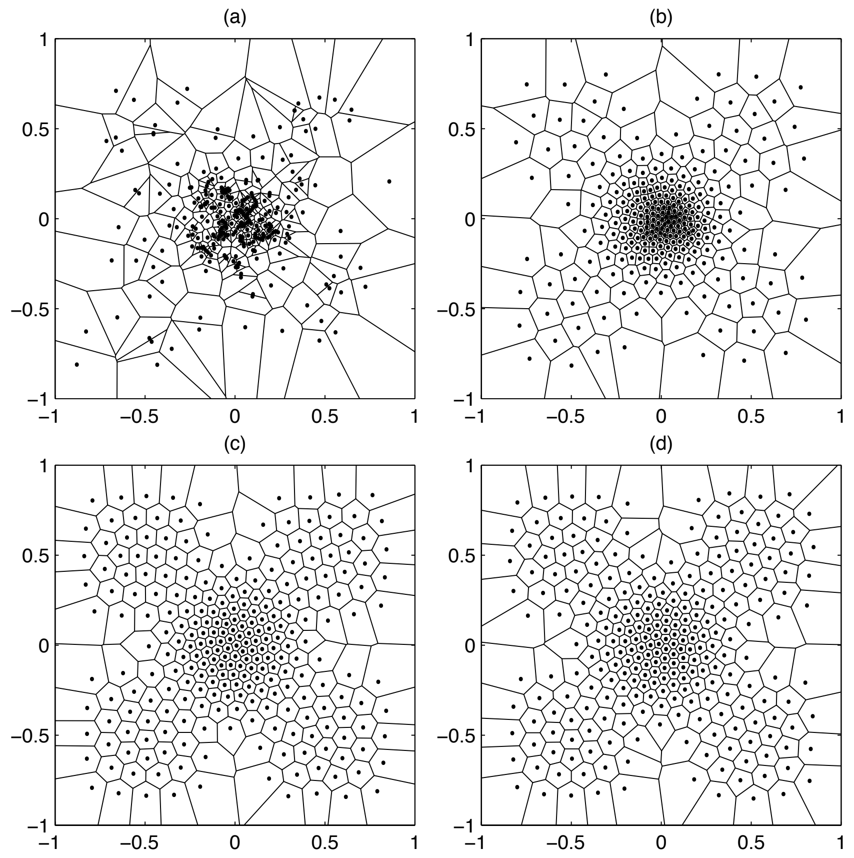
$$\rho(x) = e^{-10x^2-10y^2}$$

(a) Monte Carlo simulation;

(b) CV using Algorithm 5;

(c) CV using Algorithm 6 with $\alpha_1 = 0$, $\alpha_2 = 1$, $\beta_1 = 0$, $\beta_2 = 1$;

(d) CV using Algorithm 6 with $\alpha_1 = 0.5$, $\alpha_2 = 0.5$, $\beta_1 = 0.5$, $\beta_2 = 0.5$.



2D Voronoi diagrams for 256 generators in $[-1,1]^2$ with the density function

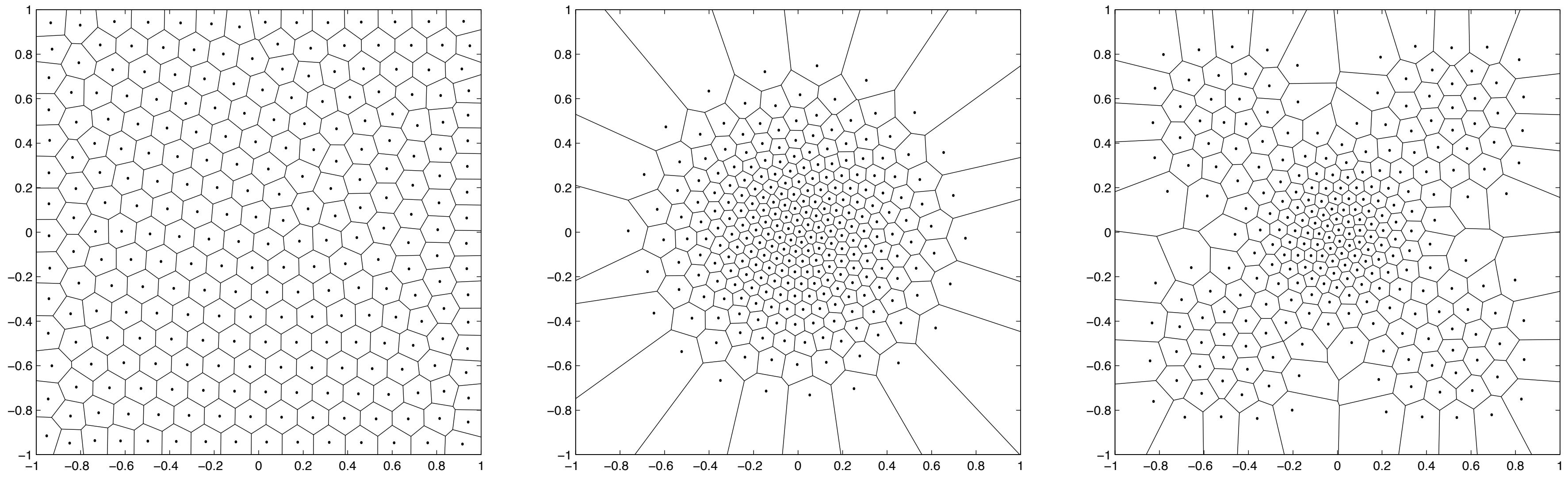
$$\rho(x) = e^{-20x^2-20y^2} + 0.05\sin^2(\pi x)\sin^2(\pi y)$$

(a) Monte Carlo simulation;

(b) CV using Algorithm 5;

(c) CV using Algorithm 6 with $\alpha_1 = 0$, $\alpha_2 = 1$, $\beta_1 = 0$, $\beta_2 = 1$;

(d) CV using Algorithm 6 with $\alpha_1 = 0.5$, $\alpha_2 = 0.5$, $\beta_1 = 0.5$, $\beta_2 = 0.5$.



Two-dimensional Voronoi diagrams for 256 generators in $[-1,1]^2$ found by the deterministic Lloyd's method; left: $\rho(x,y) = 1$; middle: $\rho(x,y) = e^{-10(x^2+y^2)}$; right: $\rho(x,y) = e^{-20(x^2+y^2)} + 0.05\sin^2(\pi x)\sin^2(\pi y)$.

Details may be found in:

> Qiang Du, Max Gunzburger, and Lili Ju; Probablistic methods for centroidal Voronoi tessellations or k-means clustering and their parallel implementation; to appear.

APPLICATION TO GRID GENERATION

- We are currently exploring the use of centroidal Voronoi tessellations (more precisely, their associated dual Delaunay grids) for the numerical solution of partial differential equations

- The density function used to determine the distribution of the Voronoi generators is to be related to a posteriori error estimators

- Anisotropic and/or tensor valued density functions could possibly be used to obtain grids with special features, e.g., large aspect ratio triangles in boundary layers

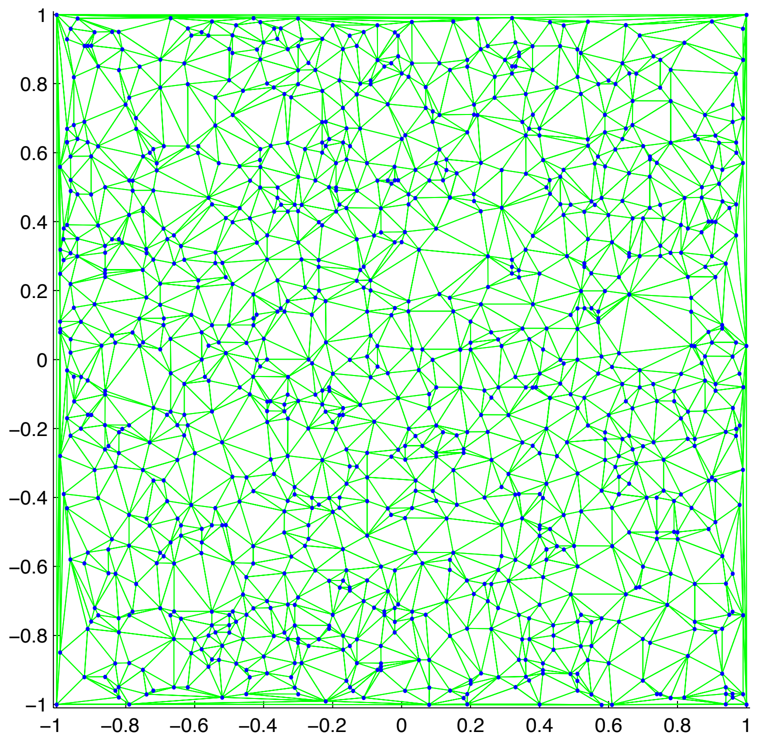
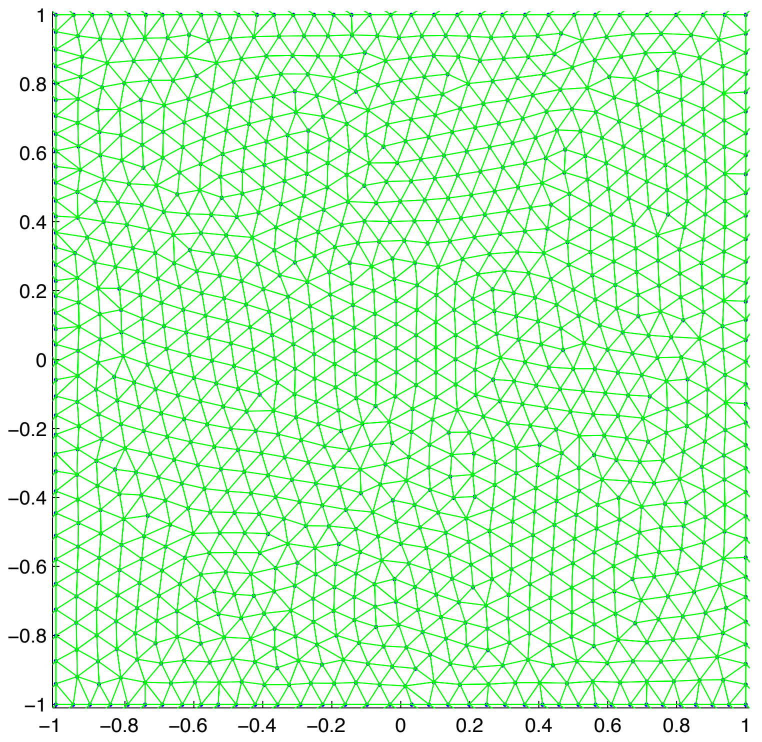
- There is reason to believe that grids generated from centroidal Voronoi tessellations have desirable features and avoid undesirable features such as slivers in three dimensions

Centroidal Voronoi-Delaunay triangulations (CVDT)

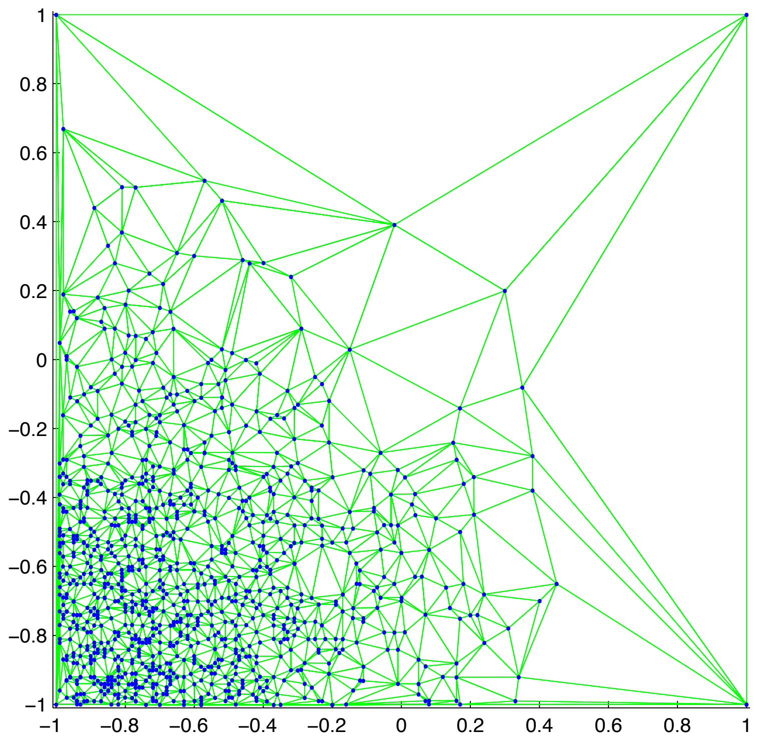
- We first illustrate the Delaunay triangulations corresponding to Voronoi tessellations determined by random selection of the generating points and to centroidal Voronoi tessellations

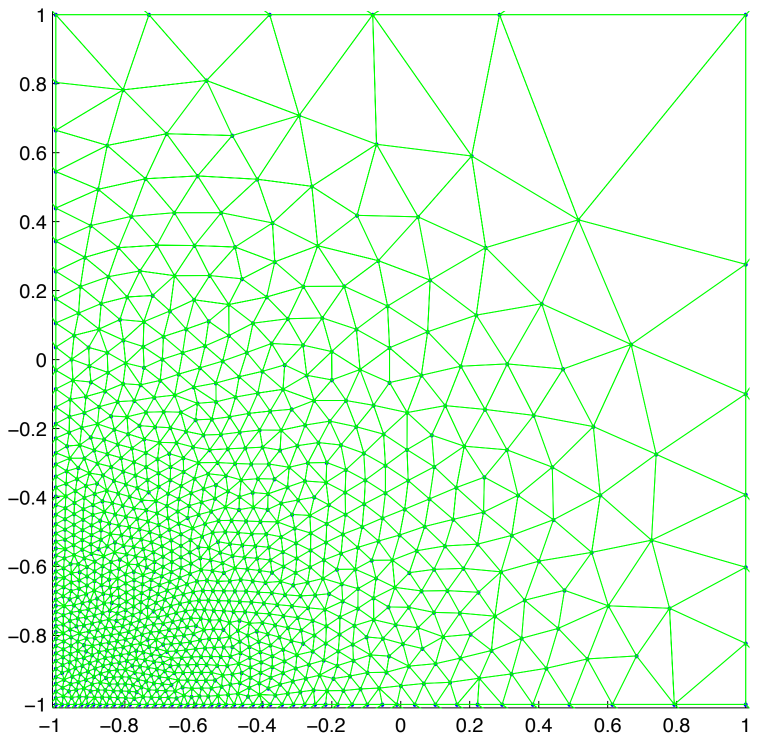
- We use a constant density function and the density function

$$\exp(-2(x+1)^2 - 2(y+1)^2))$$

2-D Delaunay triangulations for random Voronoi diagrams (left) and CVT's (right) for a constant density





2-D Delaunay triangulations for random Voronoi diagrams (left) and CVT's (right) for the density $\exp(-2(x+1)^2 - 2(y+1)^2))$

Numerical solution of PDE's

- We next illustrate the use of CVDT's for the solution of partial differential equations by considering the Poisson equation with Dirichlet boundary conditions in a unit square domain

- Discretization is effected using standard continuous, piecewise quadratic finite element spaces

- A "uniform" CV grid is one generated using a constant density function

- The uniform Cartesian grid illustrated has 98 triangles, 225 nodes, and 169 unknowns; the "uniform" CVDT has 96 triangles, 223 nodes, and 163 unknowns

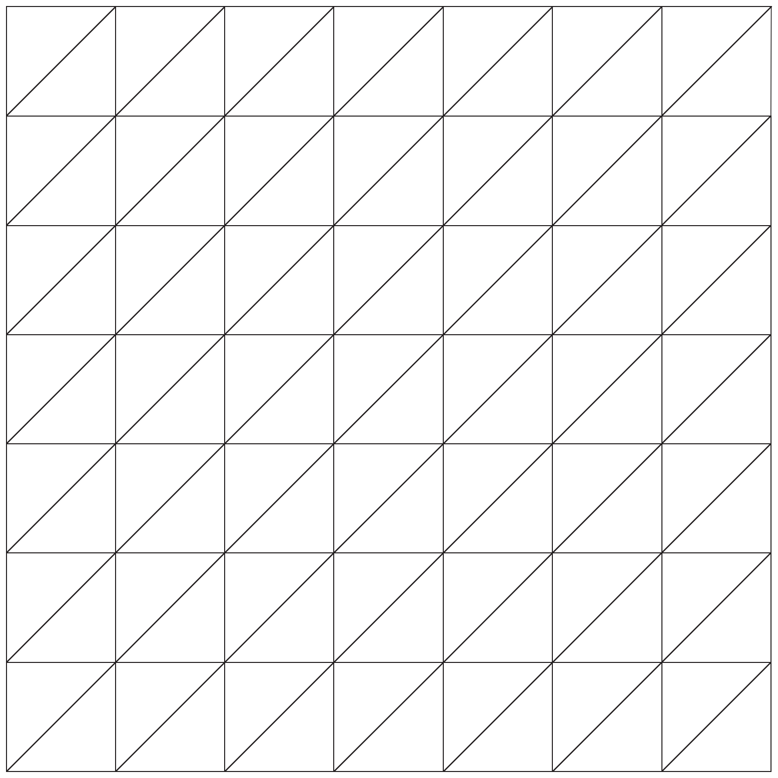
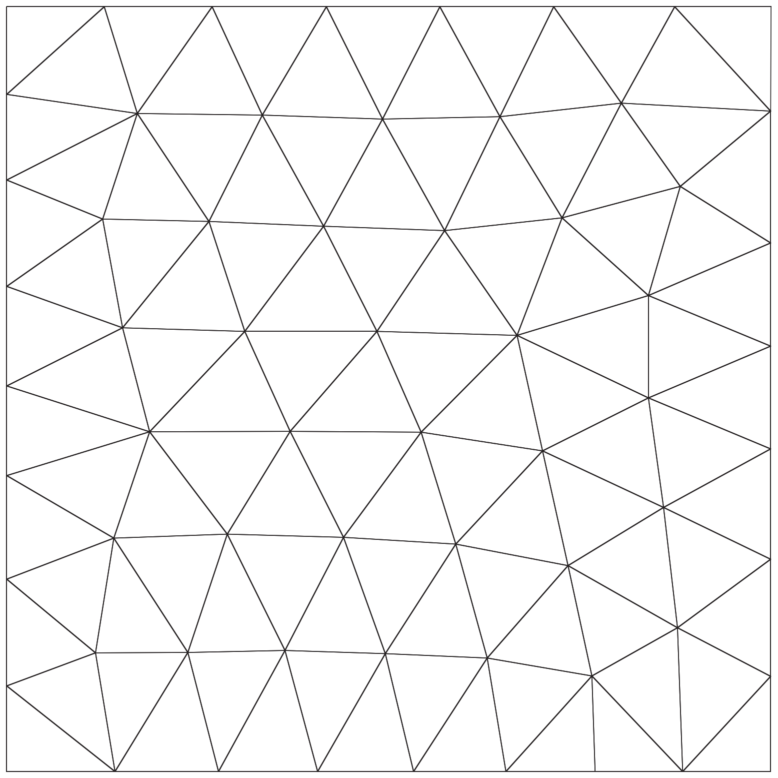


Uniform Cartesian (on the left) and centroidal Voronoi (on the right) grids

TABLE 14

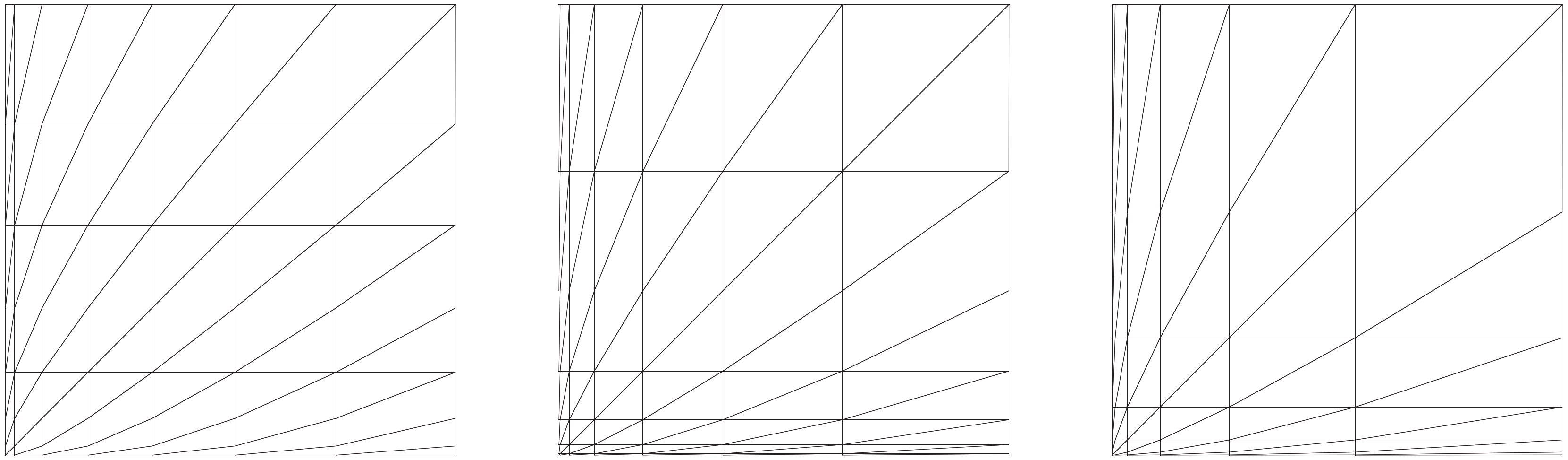
- We next use the exact solution

$$u(x,y) = \frac{1}{1 + 40x^2 + 40y^2}$$

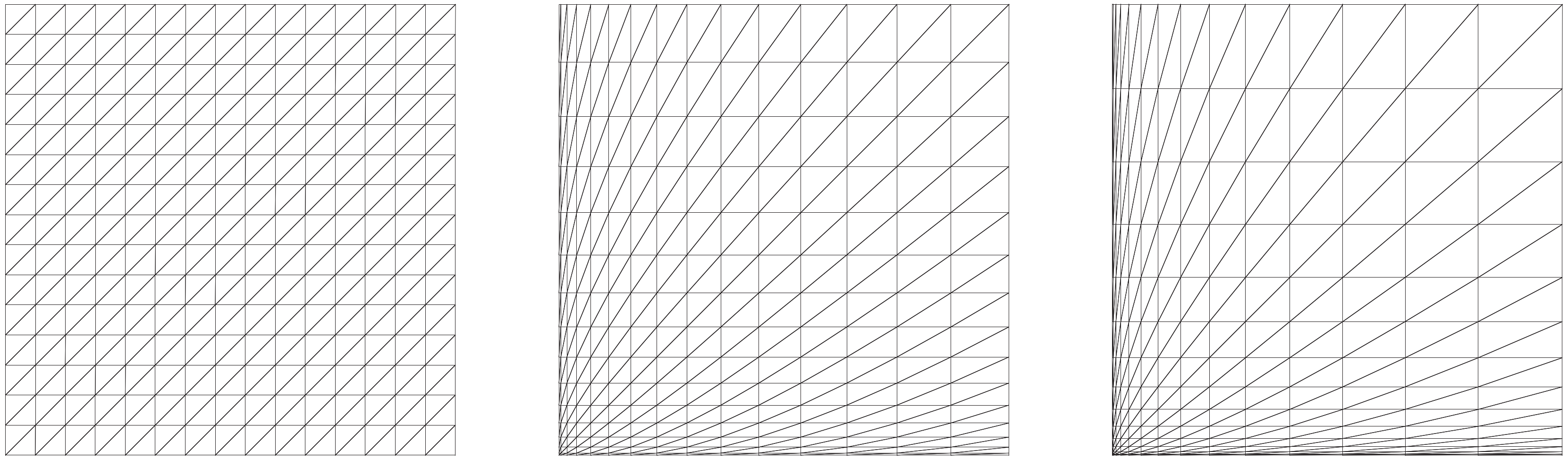
  which decays quickly away from the origin

- For Cartesian grids, refinement in each coordinate direction is effected by monomial mappings, i.e., the interval $[0,1]$ is mapped into itself using the mapping x^s which, for $s > 1$, has the effect of piling up points near the left end of the interval
  - we use the values $s = 1$, 2, 3, and 4

- The 8×8 Cartesian grids have 225 nodes, 98 triangles, and 169 unknowns; the 16×16 grids have 961 nodes, 450 triangles, and 841 unknowns



8×8 Cartesian grids increasingly refined near the origin: $s = 2, 3, 4$ (left to right)



16 $\times$ 16 Cartesian grids increasingly refined near the origin: $s = 1, 2, 3$ (left to right)

TABLE 15

- If the number of grid points remains fixed, then refinement near the origin (with the corresponding coarsening away from the origin) at first yields better approximations
  - compare the $s = 1$ and $s = 2$ cases

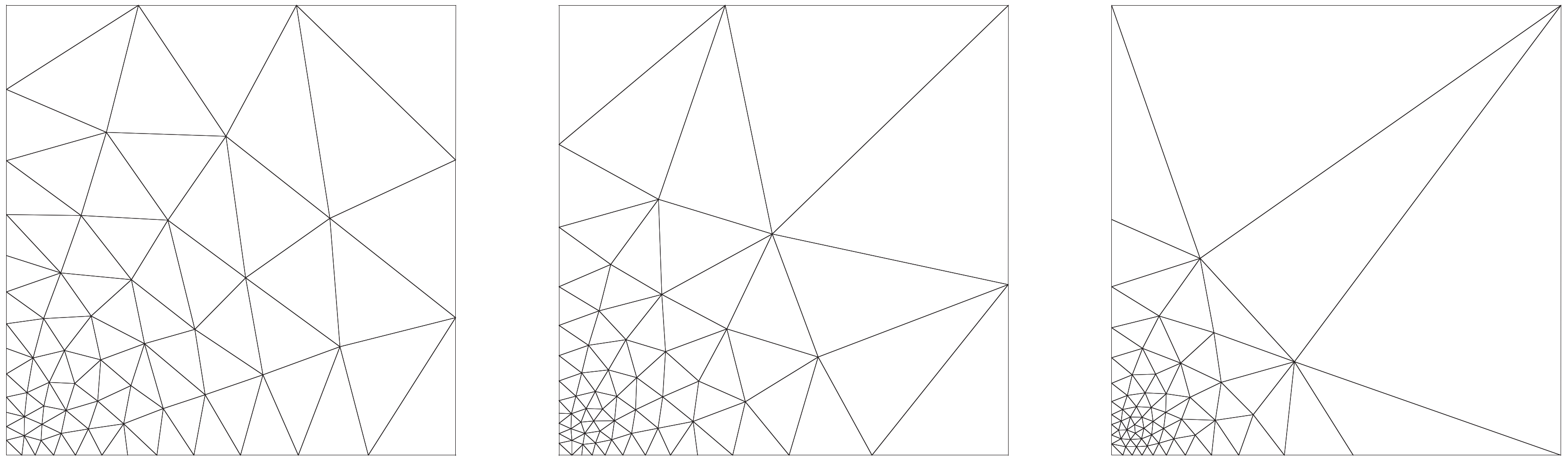
- However, more refinement, e.g., the $s = 3$ and $s = 4$ cases, near the origin causes sufficient coarsening away form the origin so that the overall error increases

- Of course, adding grid points for the same refinement exponent reduces the error
  - the 8×8 and 16×16 certainly illustrate the quadratic and cubic convergence of the H^1-seminorm and L^2-norm of the error, respectively

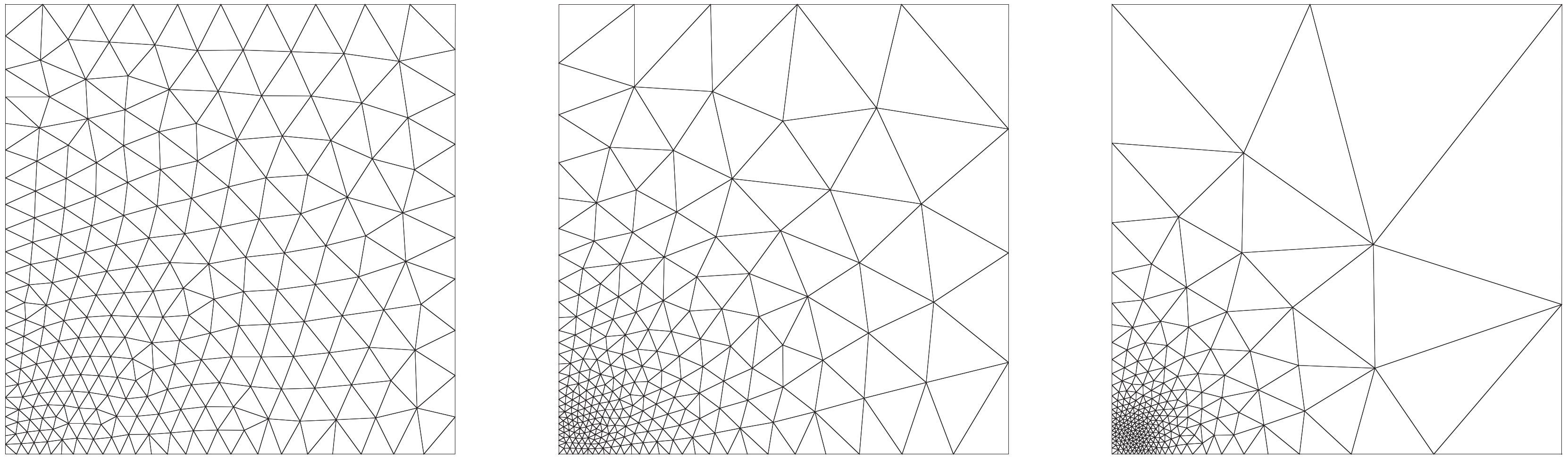
- For CVDT grids, refinement is effected by appropriately choosing the density function
- Ultimately, we would like to relate the density function to an a posteriori error estimate for the solution
- Here, we merely choose a series of density functions to see their effect on the accuracy of the solution
- Since a priori error estimates for quadratic finite element approximations involve local norms of the third derivatives of the exact solution, we choose the density functions to be proportional to powers of these local third derivative,

$$\rho(x,y) = \left\{ \sum_{i=0}^{3} \left| \partial_x^i \partial_y^{3-i} u(x,y) \right|^2 \right\}^{k/2} .$$

- The number of triangles for grids corresponding to 64 Voronoi generators is roughly the same as that for an 8×8 Cartesian grid; the number of triangles for grids having 256 Voronoi generators is roughly the same as for a 16×16 Cartesian grid



CVDT grids having roughly 100 triangles and increasing refinement near the origin; $k = 0.5, 1.0, 1.5$ (left to right).



CVDT grids having roughly 450 triangles and increasing refinement near the origin; $k = 0.25, 0.50, 0.75$ (left to right.)

TABLE 16

- Refinement near the origin reduces the error from that of a uniform grid
- However, with a fixed number of Voronoi generators, too much refinement near the origin causes excessive coarsening away from the origin and results in increases in the error
- To achieve further reduction in the error, refinement near the origin should be accompanied by the addition of Voronoi generators, i.e., more triangles.
- We certainly see that having more points greatly reduces the error
  - as was the case for Cartesian grids, we see, for CVDT grids, the expected quadratic and cubic convergence of the H^1-seminorm and L^2-norm of the error, respectively

---

Details may be found in:

Qiang Du and Max Gunzburger; Grid generation and optimization based on centroidal Voronoi tessellations; to appear.

APPLICATION TO MESHLESS COMPUTING

- Meshless computing refers to numerical methods which do not involve meshes
- They can be used, e.g., for approximating
  - multivariate functions
  - multiple integrals
  - solutions of partial differential equations
- The hope is that since meshes are not required, problems posed on complicated regions can be treated more easily
- Of course, meshless computing is nothing new, e.g., recall
  - Monte Carlo methods for numerical integration and the numerical solution of PDEs
  - particle methods for PDEs

- A typical efficient meshless computing method requires
  1. the selection of a set of points
  2. the selection of a support region associated with each point
  3. the selection of a basis function associated with each point having support over the region selected in step 2

- The implementation of a typical meshless method also requires
  4. knowledge about the overlap of the support regions associated with distinct basis functions
  5. the application of a discretization method, e.g., in the PDE setting, one can use any of
     - Galerkin method
     - collocation method
     - mixed method
     - least squares method

     while in the function approximation setting, one can use, among many choices, one of
     - interpolation
     - least squares approximation

- One can also analyze meshless methods
  6. derive error estimates

- A great deal of attention, especially in the mathematical literature, has been devoted to steps 3 and 6
  - lots of papers on deriving error estimates for specific choices of basis functions, assuming the point destribution is given and the support radii and the overlap information are known or at least are easily determined

- In fact, many papers consider one-dimensional problems or uniform point distributions in higher dimensions

- Other papers determine support radii and overlap information by using graph theoretic ideas
  - since graph = mesh, these papers do not address truly meshless methods

- Much less effort has been devoted to efficient determination, in a truly meshless way, of optimal point distributions, support radii, and overlap information

Centroidal Voronoi point selection

- We select the points to be the generators of a centroidal Voronoi tessellation corresponding to a density function $\rho(x)$
- We use Algorithm 6 to determine the points in a meshless manner

Halton sequences point selection

- We will compare centroidal Voronoi point sets to a popular way of generating uniformly distributed points based on *Halton sequences*
  - Given a prime number q, any $n \in \mathbb{N}$ can be represented as $n = \sum_i n_i q^i$
  - We can then define a mapping H_q from $\mathbb{N}$ to $[0, 1]$ by $H_q(n) = \sum_i n_i / q^{j+1}$
  - Then, the (q, r) Halton-sequence of k points in two dimensions is defined as $\{H_q(n), H_r(n)\}_{n=1}^{k}$
- Halton-sequences are pseudo Monte Carlo sequences

Support radii

- Having chosen a set of k points $\{x_i\}_{i=1}^k$ in the domain $\overline{\Omega} \subset \mathbb{R}^N$, we want to now choose, for each point x_i, an associated radius h_i

- For each point x_i we then define the sphere centered at x_i and having radius h_i
$$S_i = \{ \; y \in \mathbb{R}^N \quad : \quad \|y - x_i\| \leq h_i \; \}$$

- Other patches associated with points can be used, e.g., ellipsoids, rectangles, or bricks

- The spheres (or other patches) associated with each point will determine the support regions for basis functions associated with the points

- The selection of the radii $\{h_i\}_{i=1}^k$ should meet the two requirements:
  - the union of the spheres $\{S_i\}_{i=1}^k$ covers $\overline{\Omega}$
$$\Omega \in \cup_{i=1}^k S_i$$
  - if the intersection $S_i \cap S_j$ of two distinct spheres is not empty, it should not be "too small" nor "too large"

Algorithm 7– (adapted from C. Duarte and J. Oden and also M. Griebel and M. Schweitzer)

Given a region Ω, a density function $\rho(x)$ defined for all $x \in \overline{\Omega}$, and a set of points $\{x_i\}_{i=1}^k$ in Ω;

0. choose a positive integer m and a constant $\gamma > 1$ and set $h_i = 0$ for all $i = 1, 2, \ldots, k$;

1. select a set of points $\{\xi_i\}_{i=1}^m$ which are the nodes of a coarse meshing of Ω; let $P = \{\xi_i\}_{i=1}^m \cup \{x_i\}_{i=1}^k$;

2. for all $y \in P$:

- evaluate the set $S_{y,R}$ of all points x_i that fall within a searching sphere B_R centered at y and having radius is equal to R; if $S_{y,R} = \emptyset$ or, in case of $y \in \{x_i\}_{i=1}^k$, if $S_{y,R} = \{y\}$, then increase R and try again;
- compute the distances $d_i^y = ||y - x_i||$ for all $x_i \in S_{y,R}$ with $x_i \neq y$
- determine the point x_{i^*} with $d_{i^*}^y = \min_i d_i^y$
- if $h_{i^*} < d_{i^*}^y$, then increase h_{i^*} such that $y \in S_i$ holds;

3. for all $i = 1, 2, \ldots, k$, set $h_i = \gamma h_i$.

The ideas behind Algorithm 7 areas follows.

- The additional mesh nodes $\{\xi_i\}_{i=1}^m$ are used as pseudo-points to guarantee that the patches $\{S_i\}_{i=1}^k$ cover the entire domain $\overline{\Omega}$
- The parameter γ is motivated by the requirement of having not too small nor too large intersection of patches
  - γ controls the density of nonzeros in the stiffness matrix; if γ is "too large," then the stiffness matrix will have lots of nozero entries; if γ is "too small," then the approximation capabilities of the associated functions could be compromised
- An efficient way to evaluate the sets $S_{y,R}$ (which is a crucial step in Algorithm 7) has been given by J. Swegle, S. Attaway, F. Mello, and D. Hicks

- Algorithm 7 has some defficiencies and is not well suited for point sets $\{x_i\}_{i=1}^k$ that are generators of a centroidal Voronoi tessellation
  - it employs a coarse *mesh*
  - the points $\{x_i\}_{i=1}^k$ are determined according to a density function and generally they are not uniformly distributed; as a result, the coarse set of points $\{\xi_i\}_{i=1}^m$ often has to be refined so that the sets S_i are not too often trivial; this may not be easy and may, in Algorithm 7, involve mesh generation
  - Algorithm 7 does not take advantage of any of the many special properties of centroidal Voronoi tessellations, e.g., each point in the set $\{x_i\}_{i=1}^k$ has a associated Voronoi region that is generally a polyhedra

- We thus propose a new (totally meshless) algorithm for determining support radii

Algorithm 8

Given a region Ω, a density function $\rho(x)$ defined for all $x \in \overline{\Omega}$, and a set of points $\{x_i\}_{i=1}^k$ in Ω;

0. choose positive integers m_1 and m_2 and a constant $\gamma > 1$ and set $h_i = 0$ for all $i = 1, 2, \ldots, k$;

1. select a set of m_1 points $\{\xi_i\}_{i=1}^{m_1}$ uniformily distributed over Ω by, e.g., a Monte Carlo method; select another set of m_2 points $\{\eta_j\}_{j=1}^{m_2}$ distibuted over Ω according to the density function $\rho(x)$, e.g., again by a Monte Carlo method; let $\mathcal{P} = \{\xi_i\}_{i=1}^{m_1} \cup \{\eta_j\}_{j=1}^{m_2}$;

2. for all $y \in \mathcal{P}$:
- find a x_{i^*} in $\{x_i\}_{i=1}^k$ which is closest to y;
- compute the distance $d_{i^*} = \|y - x_{i^*}\|$;
- if $h_{i^*} < d_{i^*}$, then set $h_{i^*} = d_{i^*}$;

3. set $h_i = \gamma h_i$ for all $i = 1, 2, \ldots, k$.

There are a number of ideas behind Algorithm 8.

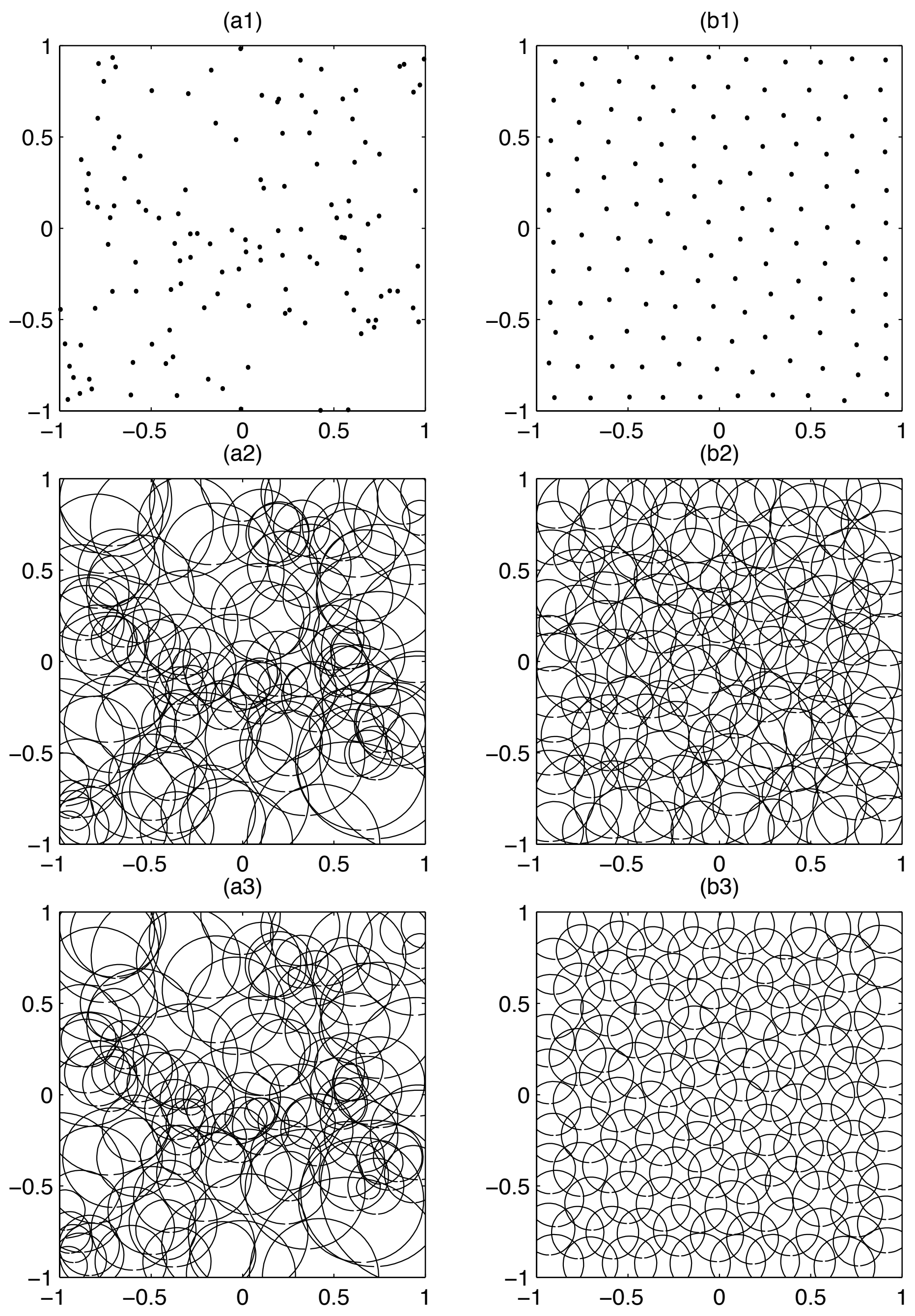
- The coarse mesh nodes are replaced by a set $\{\xi_i\}_{i=1}^{m_1}$ of uniformly distributed random points

- The set of points $\{\eta_i\}_{i=1}^{m_2}$ generated by a Monte Carlo method associated with the density fuction $\rho(x)$ is used to
  - to refine the pseudo-points set $\{\xi_i\}_{i=1}^{m_1}$
  - to find approximations, in the form of discrete point sets, of the Voronoi regions corresponding to $\{x_i\}_{i=1}^{k}$
  - for all x_i, to find an approximation to the maximum distance between the generator x_i and the farthest pseudo-point in its associated region

- If the number of the pseudo-points is large enough, then the sphere with the maximum distance should cover the corresponding Voronoi region for each x_i
  - thus, the union of all spheres should cover the domain $\overline{\Omega}$

  - since the Voronoi regions are in general polyhedra, the covering is optimal in some sense

- The parameter γ is used for the same reasons as for Algorithm 7 and, since we only have a probabilistic approximation to the maximum distances within patches, to guarantee the complete covering of the domain $\overline{\Omega}$



Sets of 128 uniformly distributed points in the square and the corresponding sphere coverings;

left - Monte Carlo point distribution;

right - centroidal Voronoi point distribution;

middle - Algorithm 7;

bottom - Algorithm 8.

TABLE 17

- Some meshless methods for PDE's require that each point in Ω be covered several times by the support spheres associated with the point set.
- The following algorithm guarantees (in a probabilistic sense) that each point in Ω is covered p-times, where p is an input integer.

Algorithm 9

Given a region Ω, a density function $\rho(x)$ defined for all $x \in \overline{\Omega}$, an integer $p > 1$, and a set of points $\{x_i\}_{i=1}^{k}$ in Ω;

0. choose positive integers m_1 and m_2 and a constant $\gamma > 1$ and set $h_i = 0$ for all $i = 1, 2, \ldots, k$;

1. select a set of m_1 points $\{\xi_i\}_{i=1}^{m_1}$ uniformily distributed over Ω by, e.g., a Monte Carlo method; select another set of m_2 points $\{\eta_j\}_{j=1}^{m_2}$ distibuted over Ω according to the density function $\rho(x)$, e.g., again by a Monte Carlo method; let $\mathcal{Q} = \{\xi_i\}_{i=1}^{m_1} \cup \{\eta_j\}_{j=1}^{m_2} \cup \{x_i\}_{i=1}^{k}$;

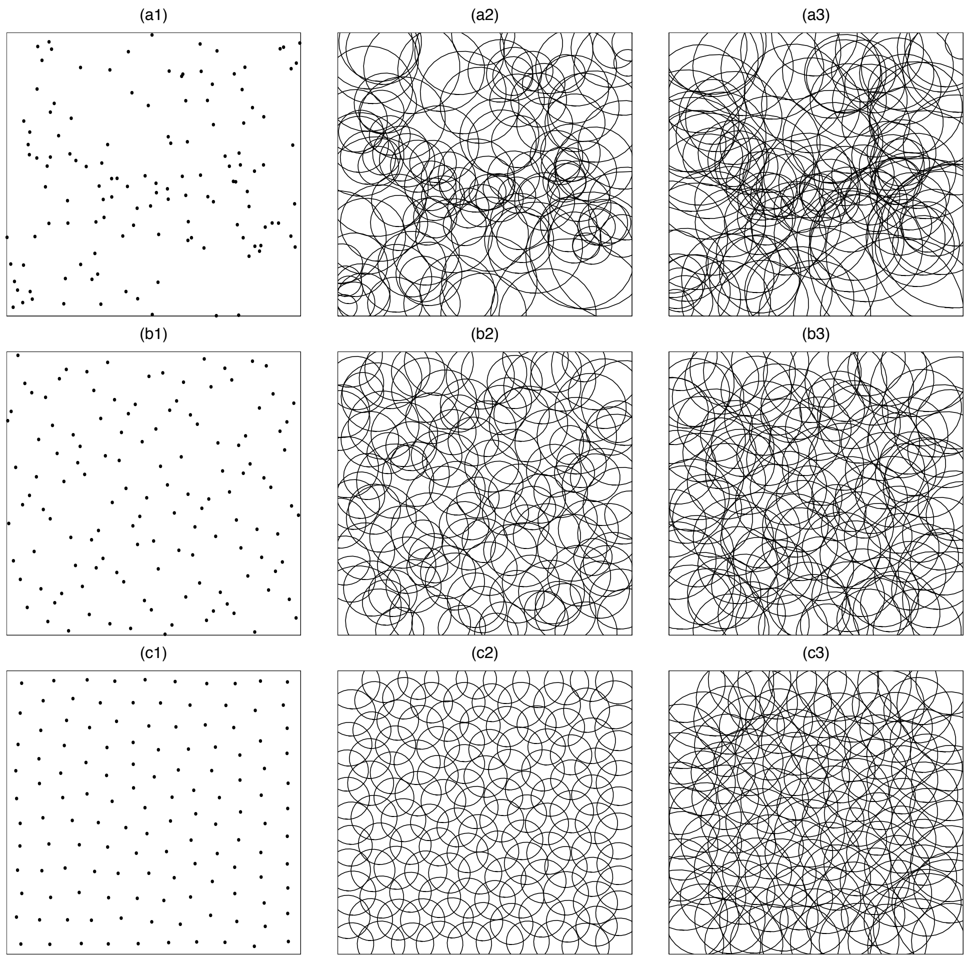
2. For all $y \in \mathcal{Q}$:

- determine the set $S_{y,R}$ of all points x_i that fall within a searching square B_R which is centered at y and whose side length is equal to $2R$; if $\text{card}(S_{y,R}) < p$, then increase R and try again; denote the points in the set $S_{y,R}$ by $x_{k(j)}$, $j = 1, 2, \ldots, \text{card}(S_{y,R})$;
- determine the distance $d_{k(j)} = \|y - x_{k(j)}\|$ for all $x_{k(j)} \in S_{y,R}$, i.e., for $j = 1, 2, \ldots, \text{card}(S_{y,R})$;

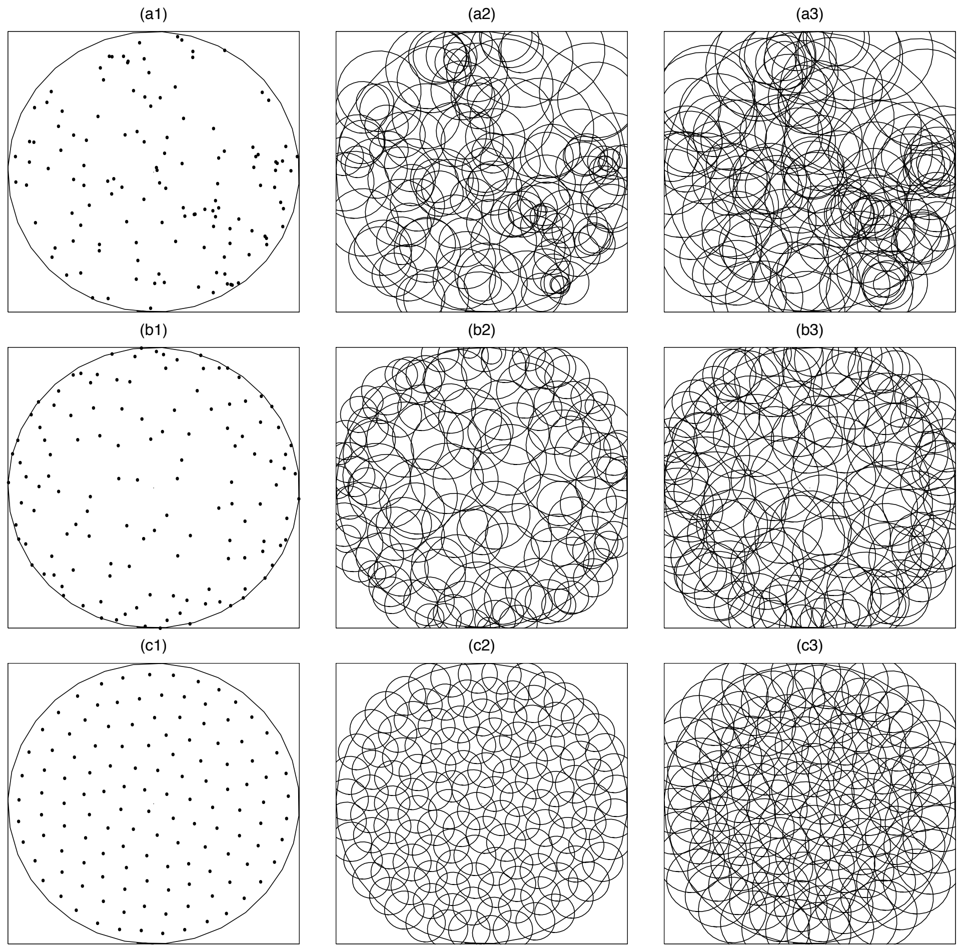
- sort the set $\{d_{k(j)}\}_{j=1}^{\mathrm{card}(S_{y,R})}$ in increasing order;
- for $j = 1, 2, \ldots, p$, if $h_{k(j)} < d_{k(j)}$, then set $h_{k(j)} = d_{k(j)}$;

3. set $h_i = \gamma h_i$ for all $i = 1, 2, \ldots, k$.

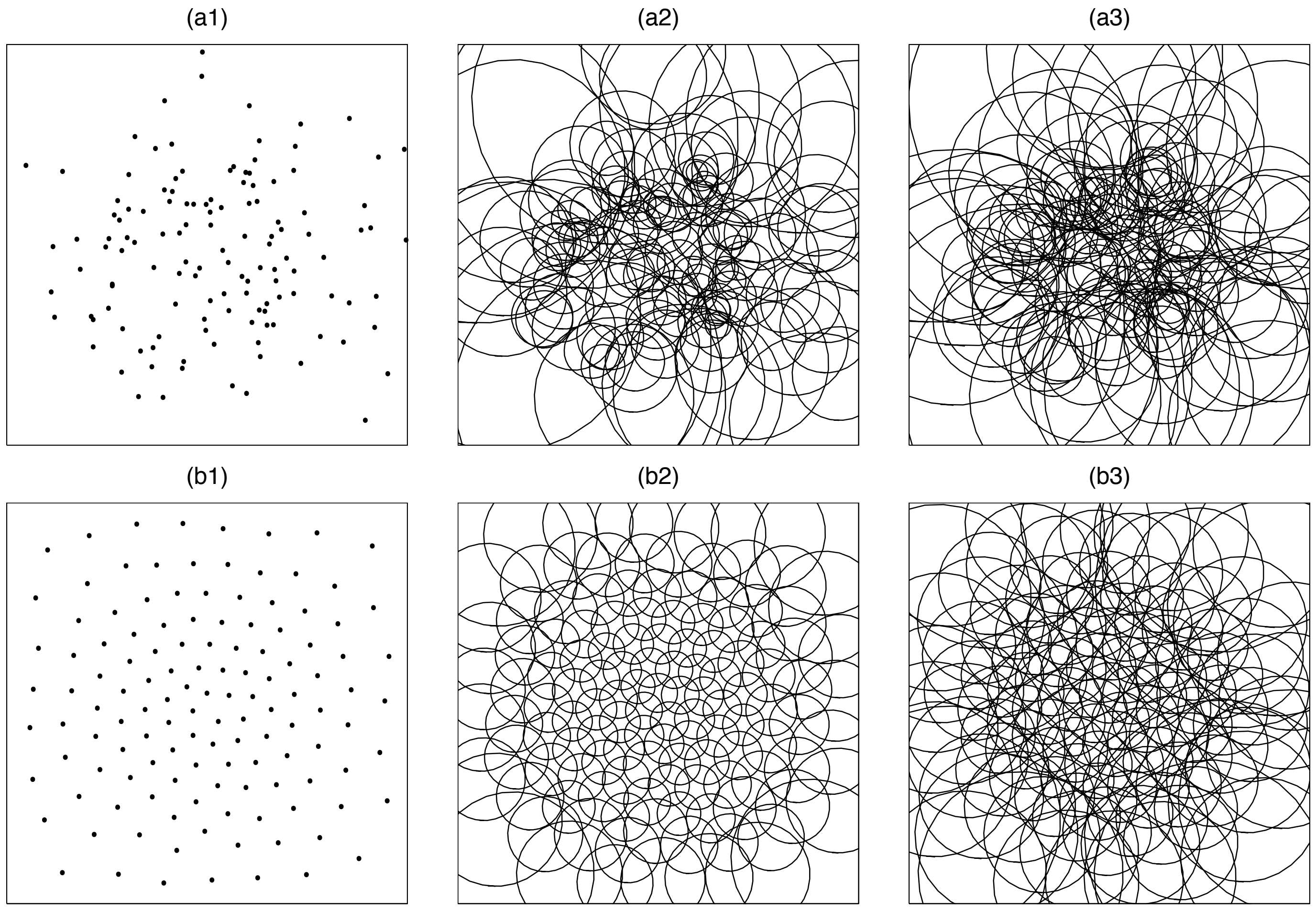
- Each x_i and all of Ω is probabilistically guaranteed to be covered p times.
- An efficient implementation of the evaluation of the set $S_{y,R}$ is given by J. Swegle, S. Attaway, F. Mello, and D. Hicks.
- The overall complexity of Algorithm 9 is $O(\mathrm{card}(\mathcal{Q}) \log(n))$.
- Note that $\mathcal{P} = \{\xi_i\}_{i=1}^{m_1} \cup \{\eta_j\}_{j=1}^{m_2}$ in Algorithm 8 because x_i is the center of the patch, i.e., we already have x_i in its associated patch for each i, but $\{x_i\}_{i=1}^{n}$ is added to $\mathcal{Q}$ in Algorithm 9 since now $p > 1$ and each x_i also has to be covered by at least $p - 1$ other support regions.



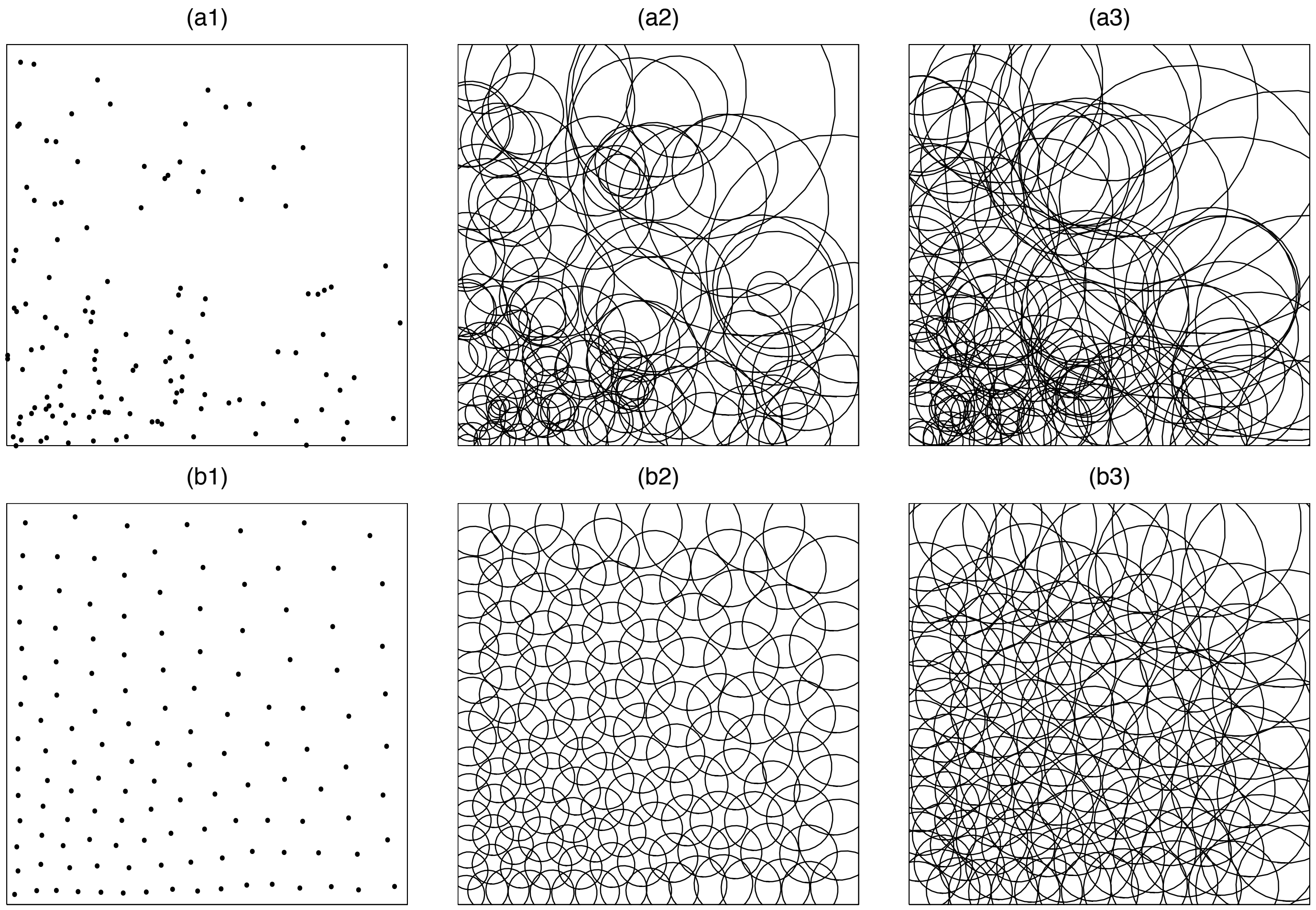
The sets of 128 points in a square (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (with $p = 2$) (right) for the Monte Carlo (top), (2,3) Halton sequence (middle), and centroidal Voronoi tessellation (bottom) point selection methods for a uniform density function.



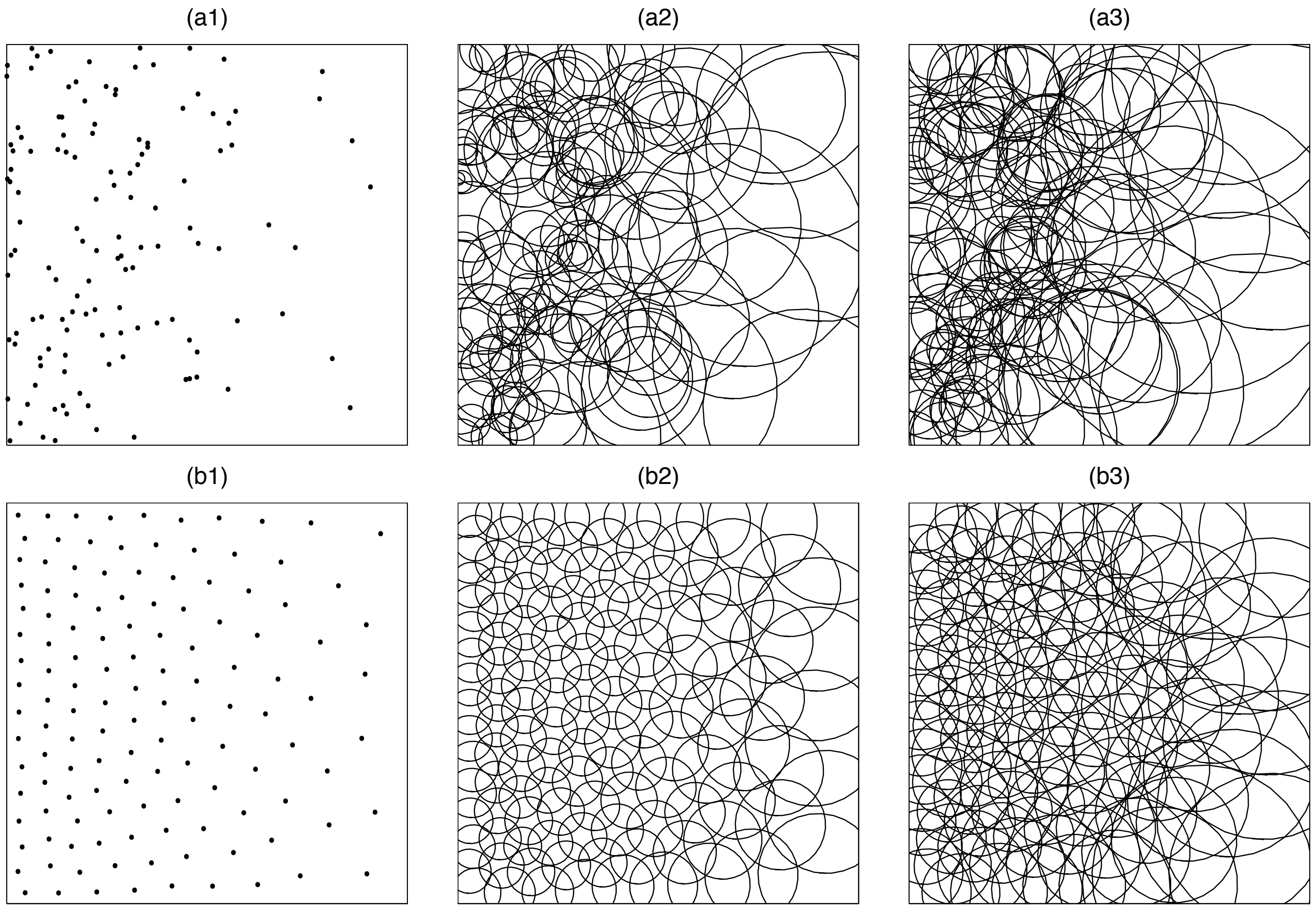
The sets of 128 points in a circle (left) and the associated spherical patches determined by Algorithms 8(middle) and 9 (with $p = 2$) (right) for the Monte Carlo (top), (2,3) Halton sequence (middle), and centroidal Voronoi tessellation (bottom) point selection methods for a uniform density function.



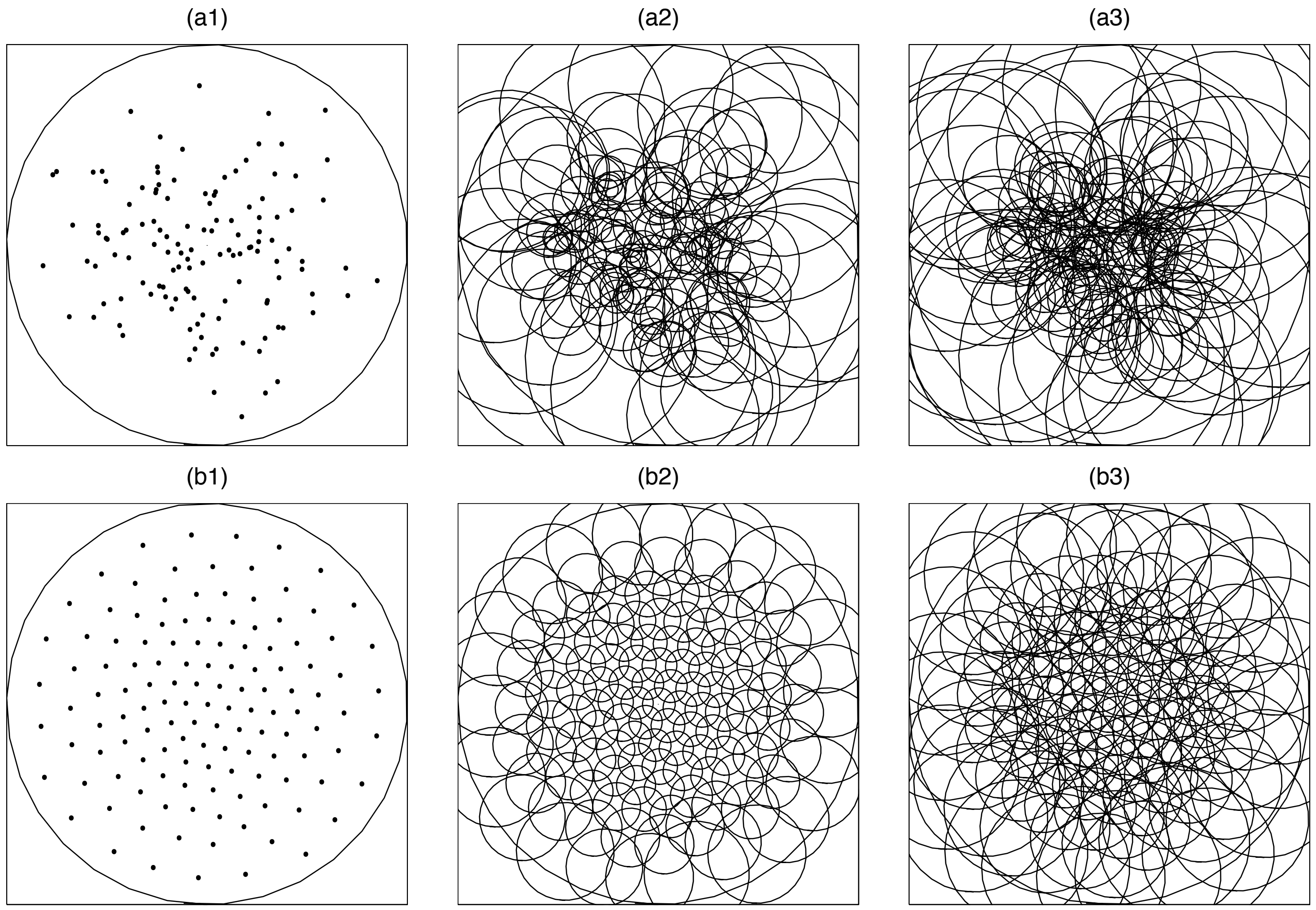
The sets of 128 points in a square (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for the density function $e^{-3(x^2+y^2)}$.



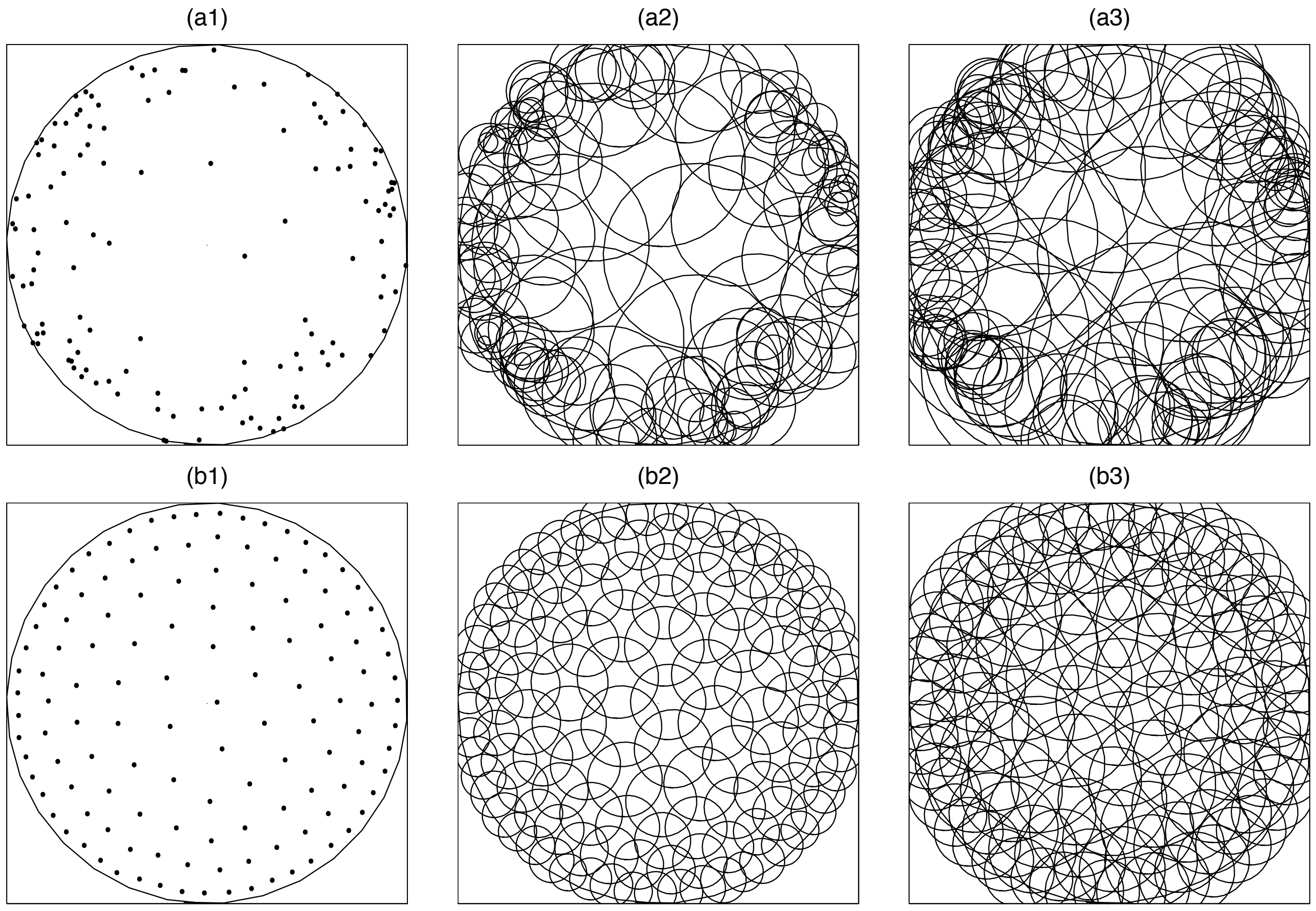
The sets of 128 points in a square (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for the density function $e^{-(2+x+y)}$.



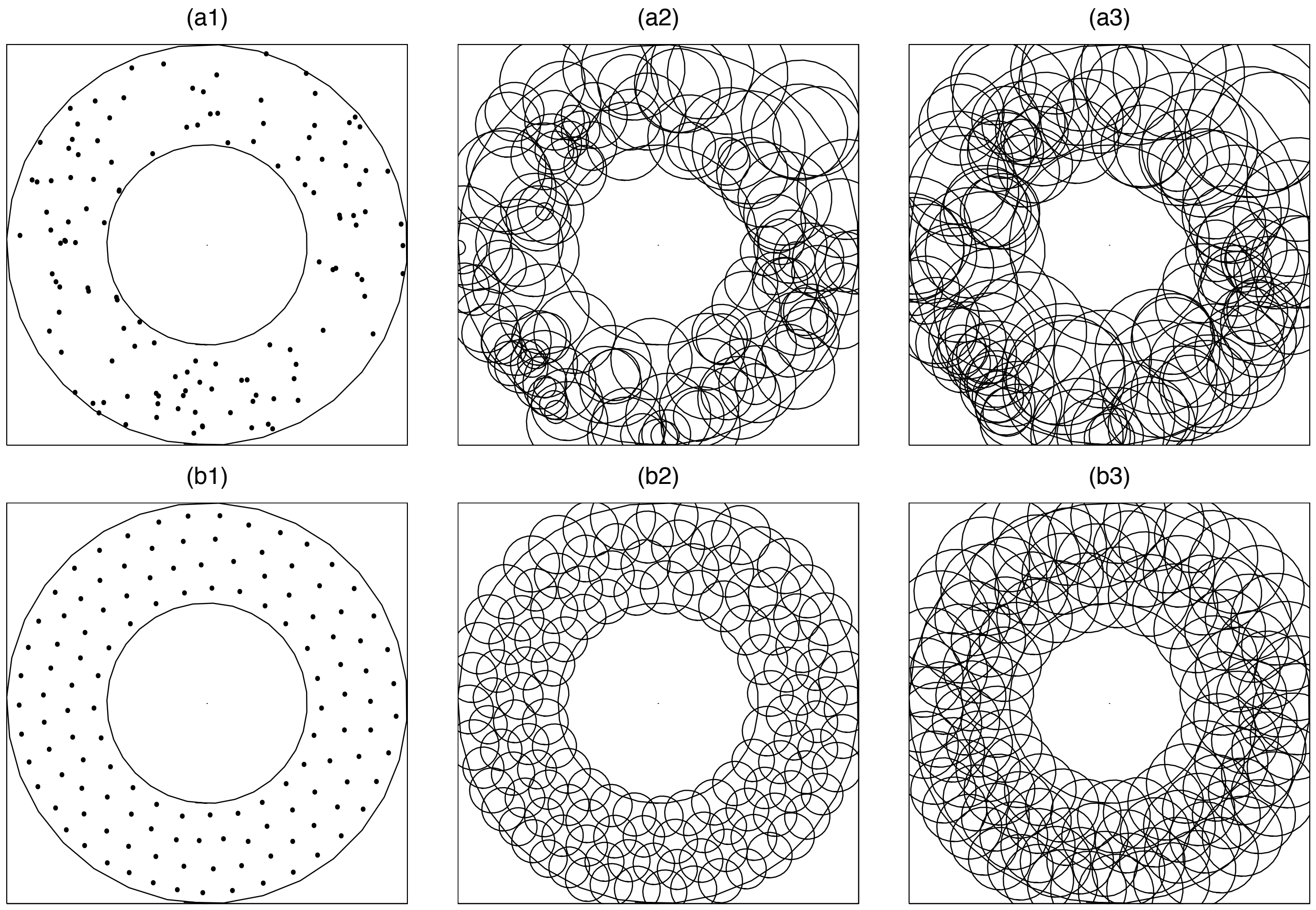
The sets of 128 points in a square (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for the density function $e^{-(1+x)}$.



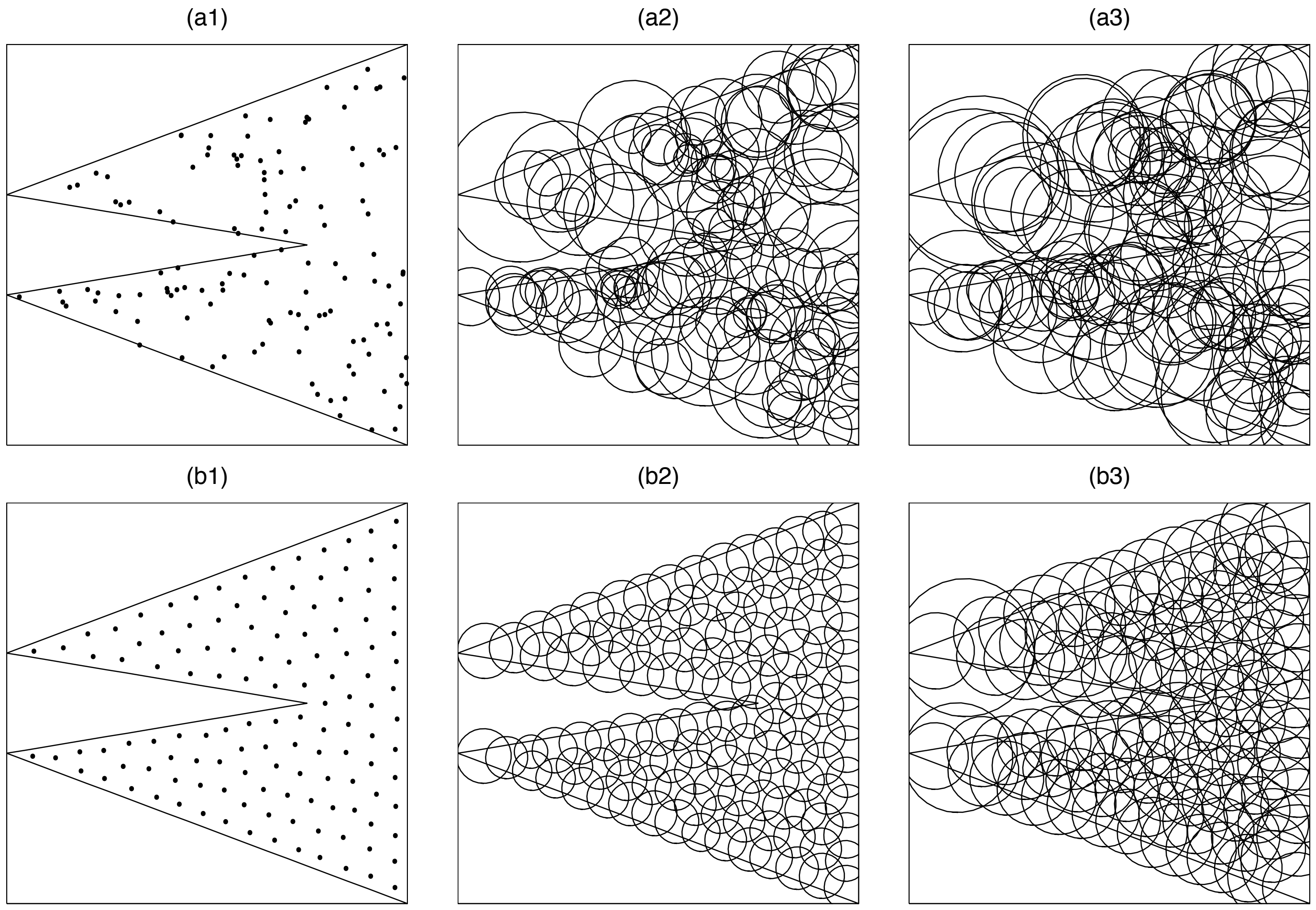
The sets of 128 points in a circle (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for the density function $e^{-4(x^2+y^2)}$.



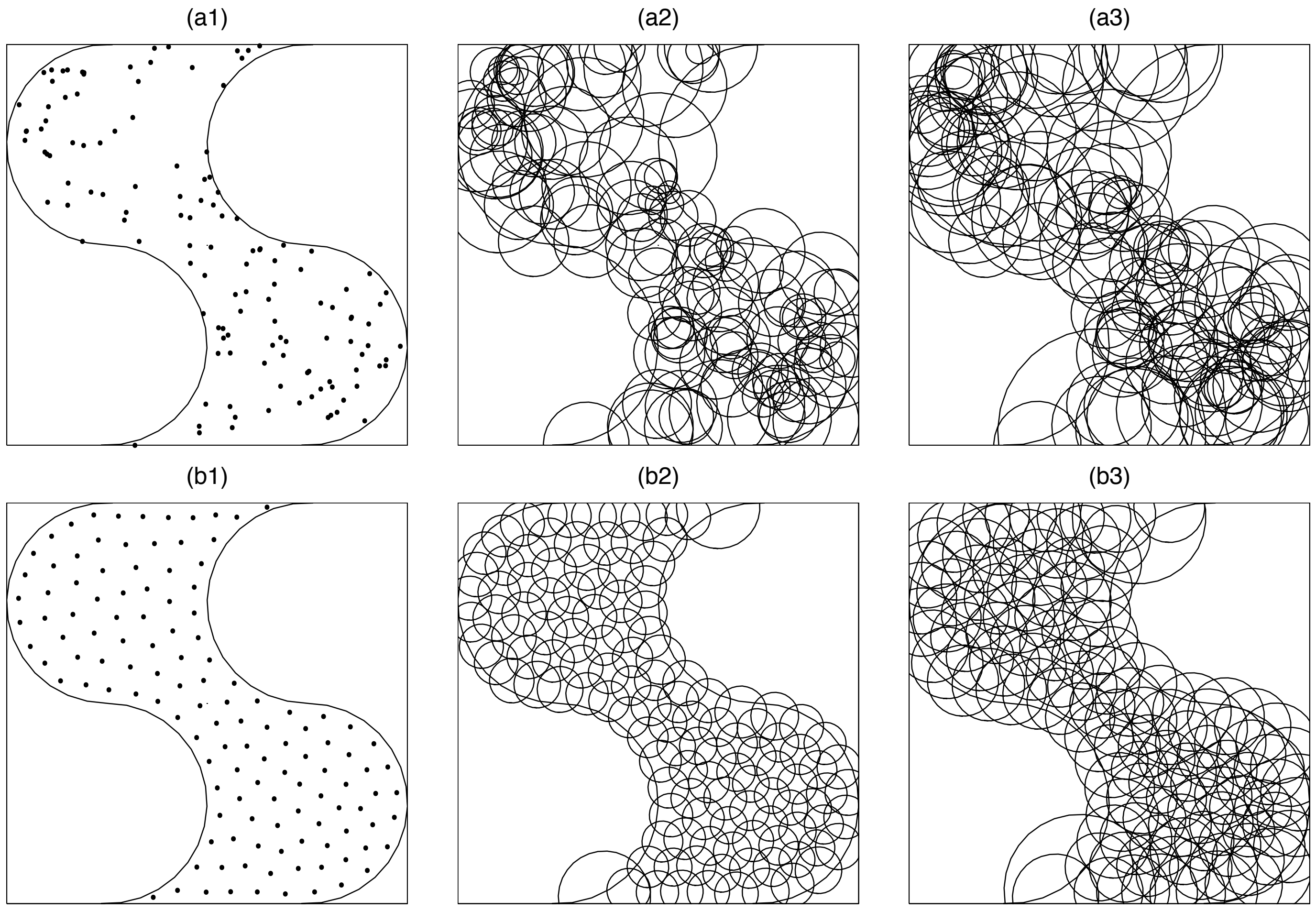
The sets of 128 points in a circle (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for the density function $e^{-3(1-x^2-y^2)}$.



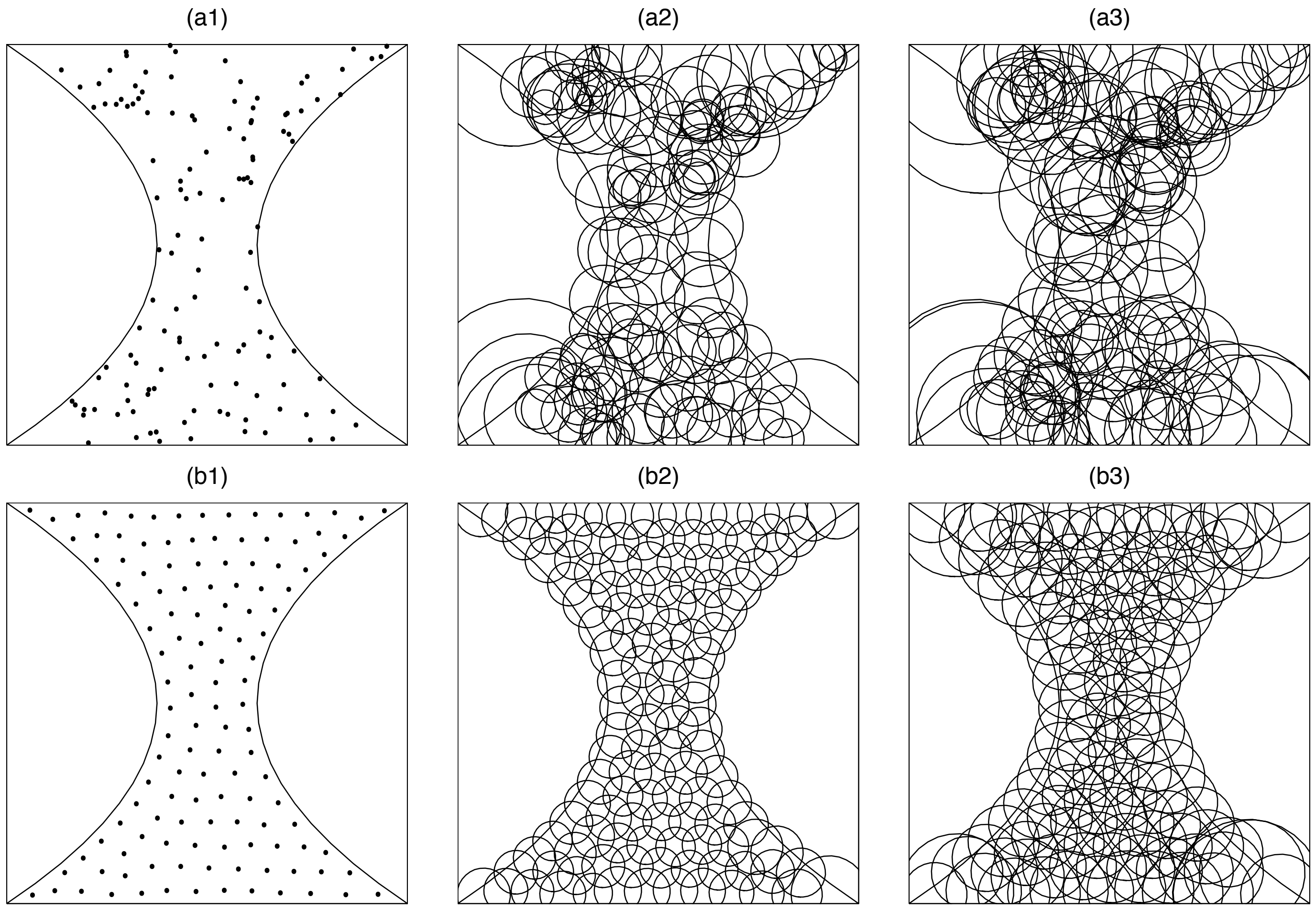
The sets of 128 points in a non-convex domain (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for a uniform density function.



The sets of 128 points in a non-convex domain (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for a uniform density function.



The sets of 128 points in a non-convex domain (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for a uniform density function.



The sets of 128 points in a non-convex domain (left) and the associated spherical patches determined by Algorithms 8 (middle) and 9 (right) for the Monte Carlo (top) and centroidal Voronoi tessellation (bottom) point selection methods for a uniform density function.

TABLE 18

TABLE 19

TABLE 20

Anisotropic point distributions

- One very important and useful feature of centroidal Voronoi tessellations is that for any smooth density function, the the centroidal Voronoi point distribution is locally uniform and isotropic, e.g., in two dimensions, as the number of generators tends to infinity, locally the centroidal Voronoi regions become congruent regular hexagons.

- In grid generation, this feature may be useful for avoiding mishaped regions, e.g., slivers, as one refines a grid

- On the other hand, one often needs anisotropic grid distribution, e.g., in boundary layers or near interfaces

- We illustrate with some examples how the methodologies we have been looking at can be adapted to generate anisotropic point distributions and thus anisotropic grids

- A central task in our probabilistic point generation algorithms is to determine which element of the set of current generators $\{\mathbf{z}_i\}_{i=1}^K$ is closest to a sampled point $\mathbf{y}$, i.e., to find and index j such that

$$\|\mathbf{z}_j - \mathbf{y}\| \le \|\mathbf{z}_j - \mathbf{y}\| \quad \forall\, j = 1, \ldots, K\,,$$

  where $\|\cdot\|$ denotes the Euclidean distance, e.g., in two dimensions

$$\|\mathbf{z} - \mathbf{y}\|^2 = (z_1 - y_1)^2 + (z_2 - y_2)^2$$

- If $\mathbf{y}$ is closest to the generator $\mathbf{z}_j$, then $\mathbf{y}$ is assigned to that generator

- Anistropic point distributions are generated by using instead

$$\mu_1(y_1, y_2)(z_1 - y_1)^2 + \mu_2(y_1, y_2)(z_2 - y_2)^2\,,$$

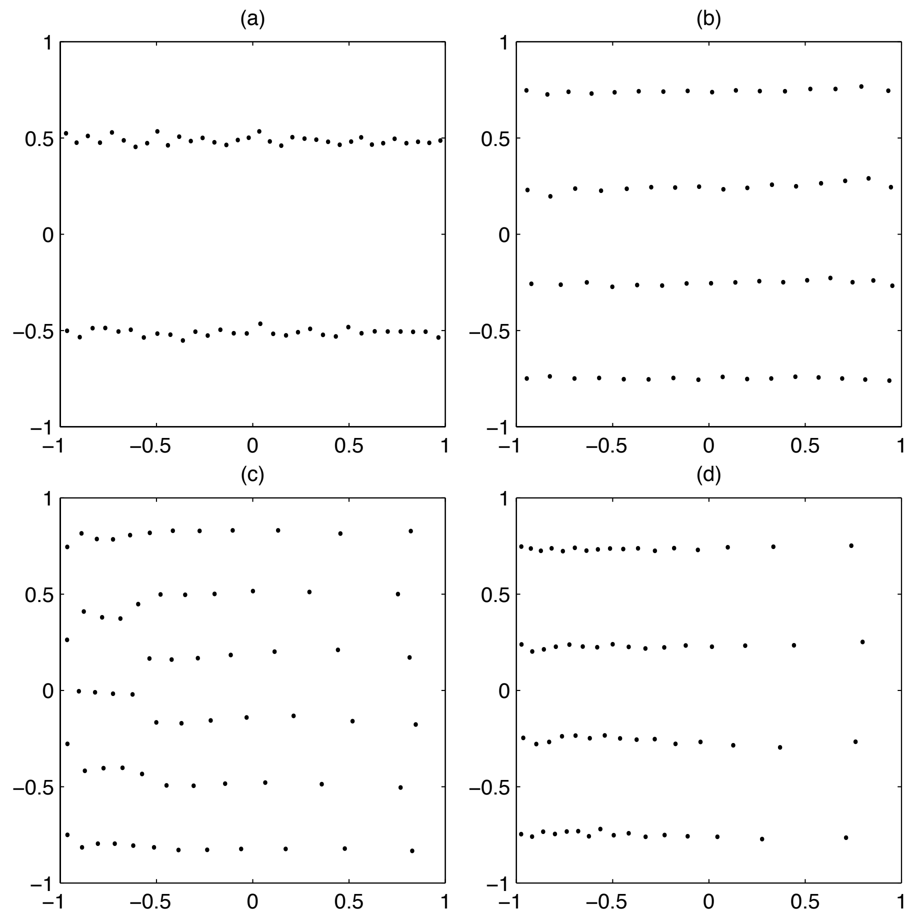
  where $\mu_1(\cdot,\cdot)$ and $\mu_2(\cdot,\cdot)$ are postive functions, when we determine the assigment of a sample point $\mathbf{y}$ to a generator

- We illustrate with some examples for which we assume that we want a point distribution that is more closely packed normal to a boundary than tangential to it

- We look at two cases:

  – the point distribution is anisotropic and uniform in the sense that the anisotropy is independent of point location

  – the point distribution is anisotropic and nouniform; in particular, near one side of a square the points are more closely packed normally than they are tangentially, but away from that side the point distribution is uniform

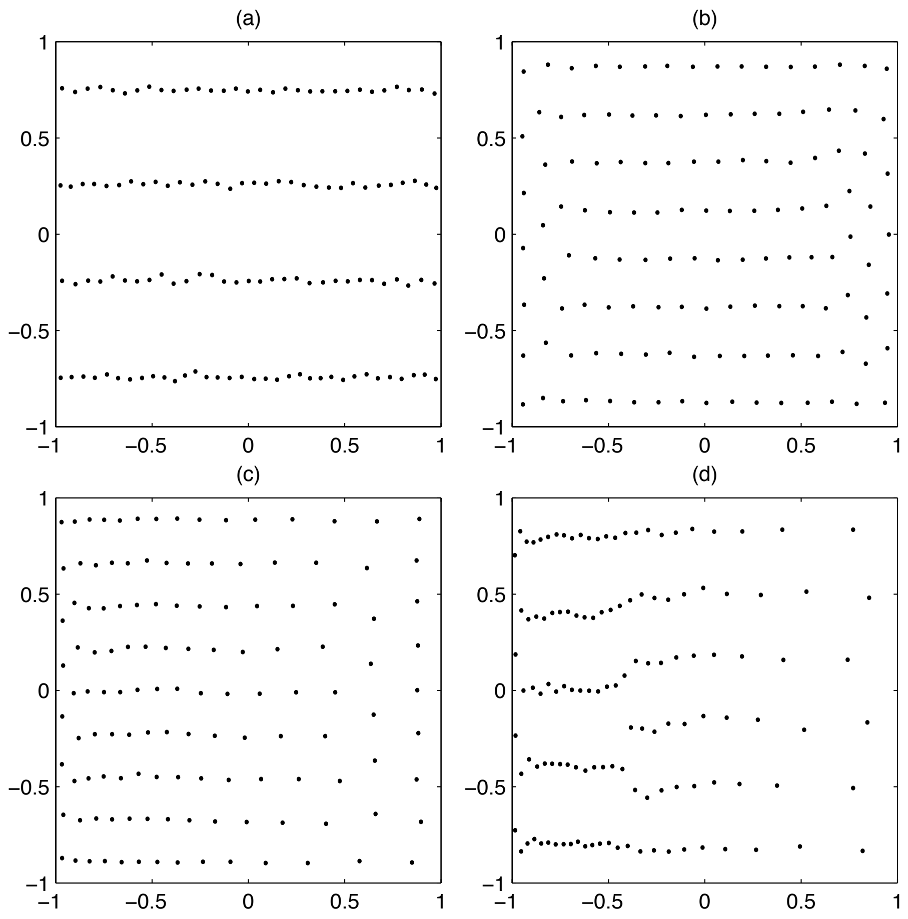


Anisotropic distributions of 64 points in the square $[-1,1]^2$

(a) $256(z_1-y_1)^2+(z_2-y_2)^2$ (b) $20(z_1-y_1)^2+(z_2-y_2)^2$

(c) $\left(1+20e^{-2(y_1+1)^2}\right)(z_1-y_1)^2+(z_2-y_2)^2$

(d) $\left(1+100e^{-2(y_1+1)^2}\right)(z_1-y_1)^2+(z_2-y_2)^2$



Anisotropic distributions of 128 points in the square $[-1,1]^2$

(a) $80(z_1-y_1)^2+(z_2-y_2)^2$ (b) $5(z_1-y_1)^2+(z_2-y_2)^2$

(c) $\left(1+10e^{-2(y_1+1)^2}\right)(z_1-y_1)^2+(z_2-y_2)^2$

(d) $\left(1+150e^{-2(y_1+1)^2}\right)(z_1-y_1)^2+(z_2-y_2)^2$

Centroidal Voronoi tessellations on surfaces

- In many applications, point distributions on surfaces are needed
- In order to generalize CVT's to surfaces, two main ingredients are needed
  - the generalization of the concept of Voronoi regions to surfaces
  - the generalization of the concept of mass centroids to surfaces
- There are a number of ways to do each of these
  - we choose generalizations which are "easy" to use
- We consider a compact and continuous surface $\mathbf{S} \subset \mathbb{R}^N$ defined by

$$\mathbf{S} = \{\mathbf{x} \in \mathbb{R}^N \quad | \quad g_0(\mathbf{x}) = 0 \;\text{ and }\; g_j(\mathbf{x}) \le 0 \;\text{ for } j = 1, \ldots, m\}$$

- Given a set of points $\{\mathbf{z}_i\}_{i=1}^k \in \mathbf{S}$, we define their corresponding *Voronoi regions* on $\mathbf{S}$ by

$$V_i = \{\, \mathbf{x} \in \mathbf{S} \quad | \quad |\mathbf{x} - \mathbf{z}_i| < |\mathbf{x} - \mathbf{z}_j| \quad \text{for } j = 1, \ldots, k,\ j \neq i \,\}$$

  for $i = 1, \ldots, k$

- For each Voronoi region V_i, we call $\mathbf{z}_i^c$ the *constrained mass centroid* of V_i on $\mathbf{S}$ if $\mathbf{z}_i^c$ is a solution of the following problem:

$$\min_{\mathbf{z} \in \mathbf{S}} F_i(\mathbf{z}) \,, \qquad \text{where} \qquad F_i(\mathbf{z}) = \int_{V_i} \rho(\mathbf{x}) |\mathbf{x} - \mathbf{z}|^2 \, d\mathbf{x}$$

- We call a Voronoi tessellation a *constrained centroidal Voronoi tessellation* (CCVT) if and only if the points $\{\mathbf{z}_i\}_{i=1}^k$ which serve as the generators of the Voronoi regions $\{V_i\}_{i=1}^k$ are also the constrained mass centroids of those regions

- Note that the definition of CCVT implies that
  - the generators are constrained to the surfaces
  - but distances are still the standard Euclidean distances, not the more general geodesic distances

- Due to the following result, the constrained mass centroid is "easy" to construct by normal projection

  > *For each $i = 1, \cdots, k$, the constrained mass centroid of V_i exists. Furthermore, if $\mathbf{z}_i^c$ is a constrained centroid of V_i, then $\mathbf{z}_i^* - \mathbf{z}_i^c$ is a vector normal to the surface $\mathbf{S}$ at $\mathbf{z}_i^c$, where $\mathbf{z}_i^*$ is the ordinary mass centroid of V_i, i.e. $\mathbf{z}_i^c$ is the projection of $\mathbf{z}_i^*$ onto $\mathbf{S}$ along the normal direction at $\mathbf{z}_i^c$.*

- Constrained CVT's defined in the manner above enjoy, in much the same way as do CVT's, an optimization property

  > *Given a compact surface $\mathbf{S} \subset \mathbb{R}^N$, a positive integer k, and a positive and measurable density function $\rho(\cdot)$ defined on $\mathbf{S}$. Let $\{\mathbf{z}_i\}_{i=1}^k$ denote any set of k points belonging to $\mathbf{S}$ and let $\{V_i\}_{i=1}^k$ denote any tessellation of $\mathbf{S}$ into k regions. Define the* energy functional *or the* distortion value *for $\{(\mathbf{z}_i, V_i)\}_{i=1}^k$ by*
  >
  > $$\mathcal{F}(\{(\mathbf{z}_i, V_i)\}_{i=1}^k) = \sum_{i=1}^{k} \int_{\mathbf{x}\in V_i} \rho(\mathbf{x})|\mathbf{x} - \mathbf{z}_i|^2 \, d\mathbf{x}\,.$$
  >
  > *A necessary condition for $\mathcal{F}$ to be minimized is that the V_i's are the Voronoi regions corresponding to the $\mathbf{z}_i$'s and, simultaneously,*

the $\mathbf{z}_i$'s are the constrained centroids of the corresponding V_i's, i.e. $\{(\mathbf{z}_i, V_i)\}_{i=1}^k$ is a constrained centroidal Voronoi tessellation of $\mathbf{S}$.

- Algorithms (deterministic or probabilistic, serial or parallel) for CVT's may be then easily generalized to the case of CCVT's

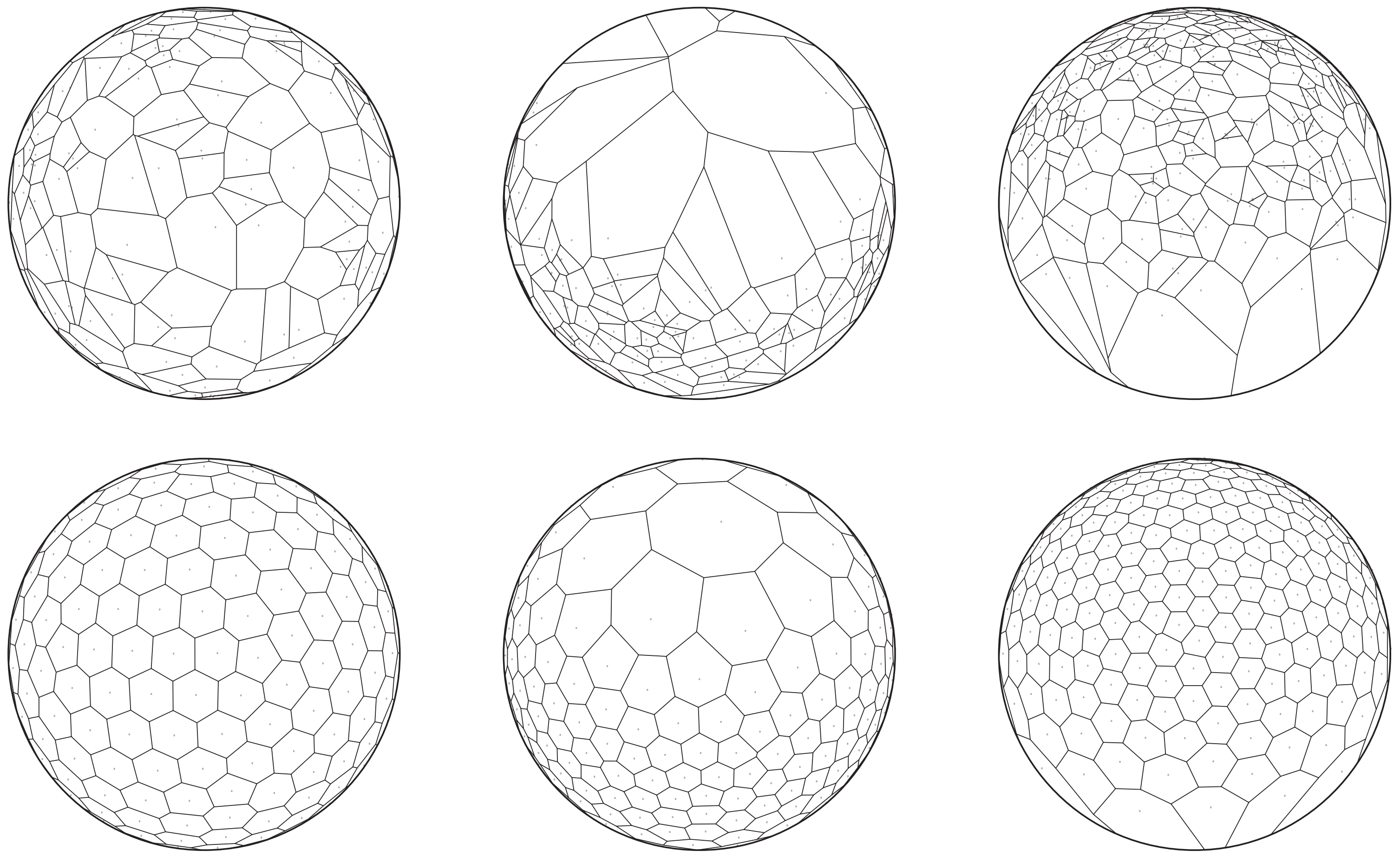
- We illustrate the use of CCVT's on three surfaces
  - the sphere
  - the developable surface

$$\mathbf{S} = \left\{(x,y,z) \mid z = -x^2,\ |x| \le \frac{1}{2},\ |y| \le \frac{1}{2}\right\}$$

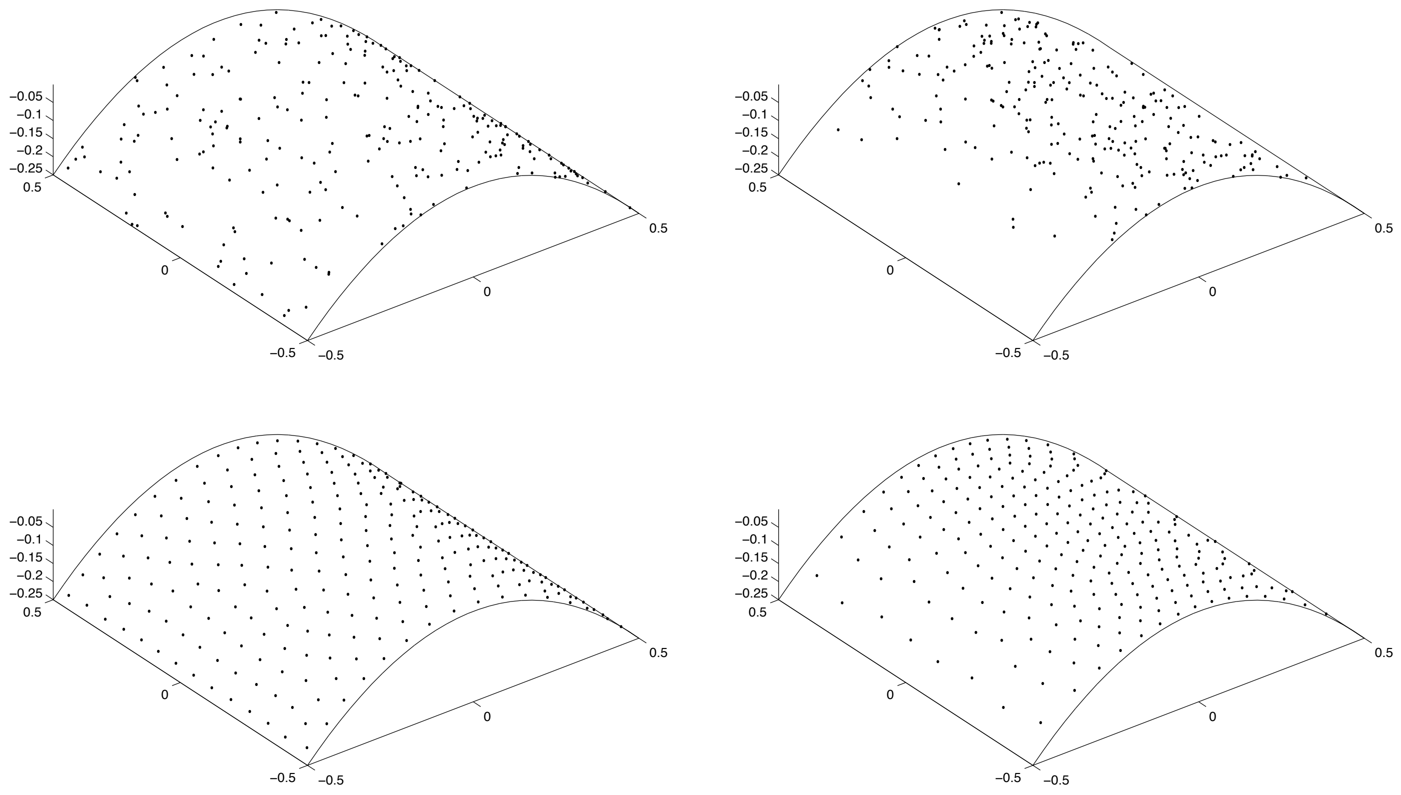
  - the torus

$$\mathbf{S} = \left\{(x,y,z) \mid (x - \frac{x}{r})^2 + (y - \frac{y}{r})^2 + z^2 = 0.3^2,\ r = \sqrt{x^2+y^2}\right\}$$

- We illustrate the quality of CCVT's by examining their use for interpolation and quadrature on the sphere



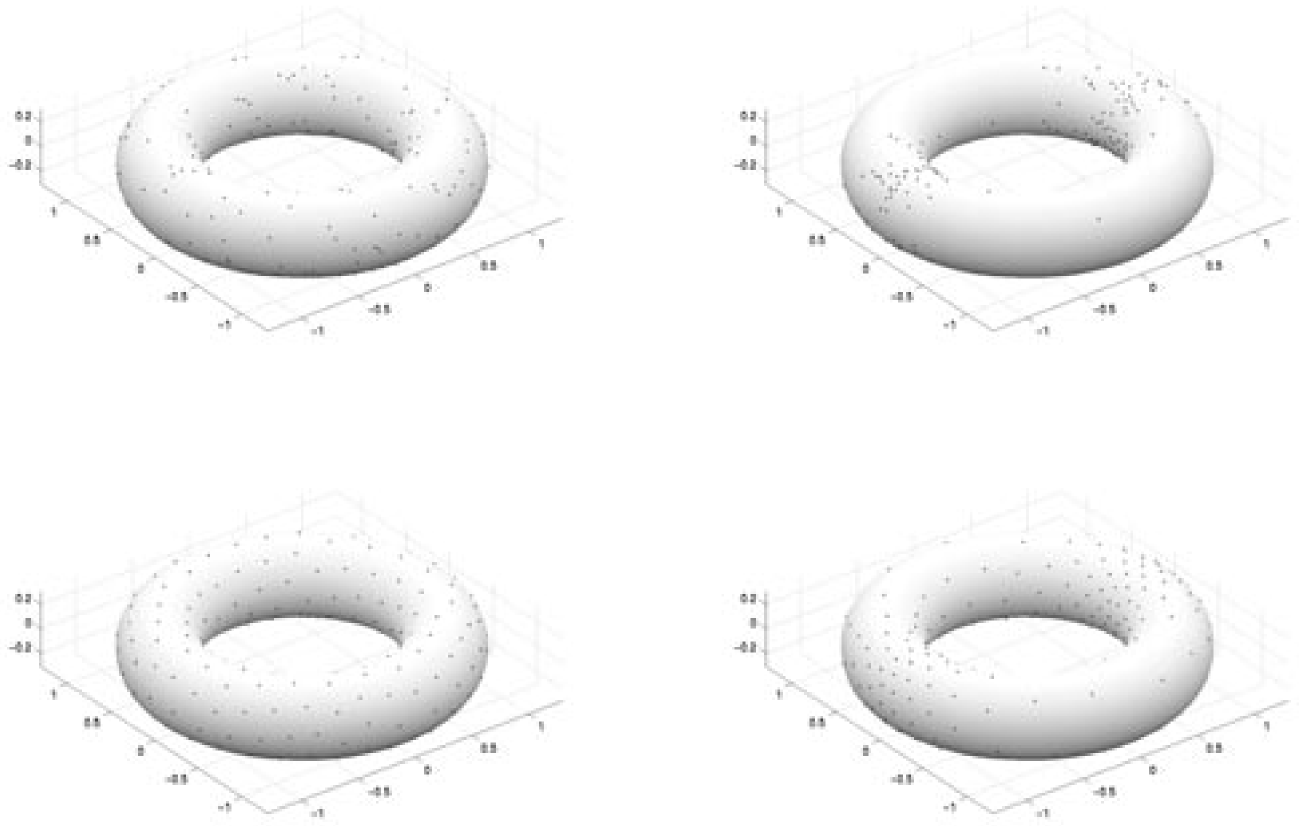
Voronoi diagrams for 256 generators on the surface of the unit sphere. top: Monte Carlo (random sampling); bottom: constrained CVT; left: $\rho(x, y, z) = 1$; middle: $\rho(x, y, z) = e^{-6.0z^2}$; right: $\rho(x, y, z) = e^{-3.0(1-z)^2}$



Voronoi diagrams for 256 generators on a developable surface.

top: Monte Carlo (random sampling); bottom: constrained CVT;

left: $\rho(x, y, z) = 1$; right: $\rho(x, y, z) = e^{-20.0x^2}$



Voronoi diagrams for 256 generators on a torus.

top: Monte Carlo (random sampling); bottom: constrained CVT;

left: $\rho(x, y, z) = 1$; right: $\rho(x, y, z) = e^{-5.0|y|}$

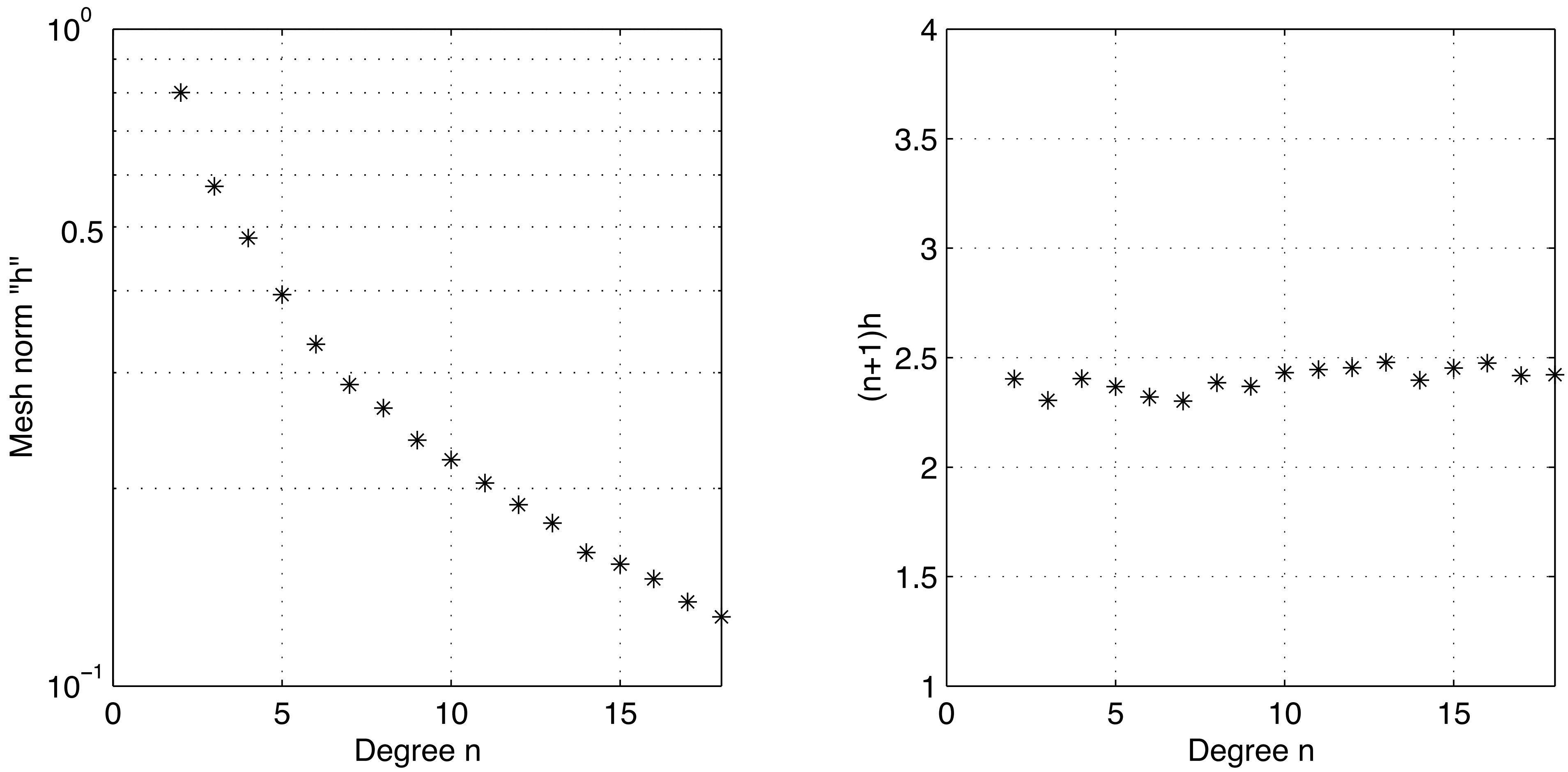
- The *mesh norm* h of a set of points $\{\mathbf{x}_i\}_{i=1}^{k}$ on the unit sphere S^2 is defined by
$$h = \max_{\mathbf{x}\in S^2} \min_{i=1,\dots,k} \cos^{-1}(\mathbf{x}^T\mathbf{x}_i)$$

- Of course, it is topologically impossible to tessellate the surface of a sphere exactly uniformly
  - however, that it is clear that a "uniform" tessellation of the surface of a sphere into hexagonal-like regions would result in $h\sqrt{k} \approx \sqrt{8\sqrt{3}\pi/9} \approx 2.2$ when k is large

- Thus, we can use h as an indicator of the "uniformity" of point distributions on the sphere

- CCVT's on the sphere achieve very good uniformity

- This implies that CCVT point distributions are useful for piecewise polynomial interpolation on the sphere and for finite element discretizations of partial differential equations posed on a sphere

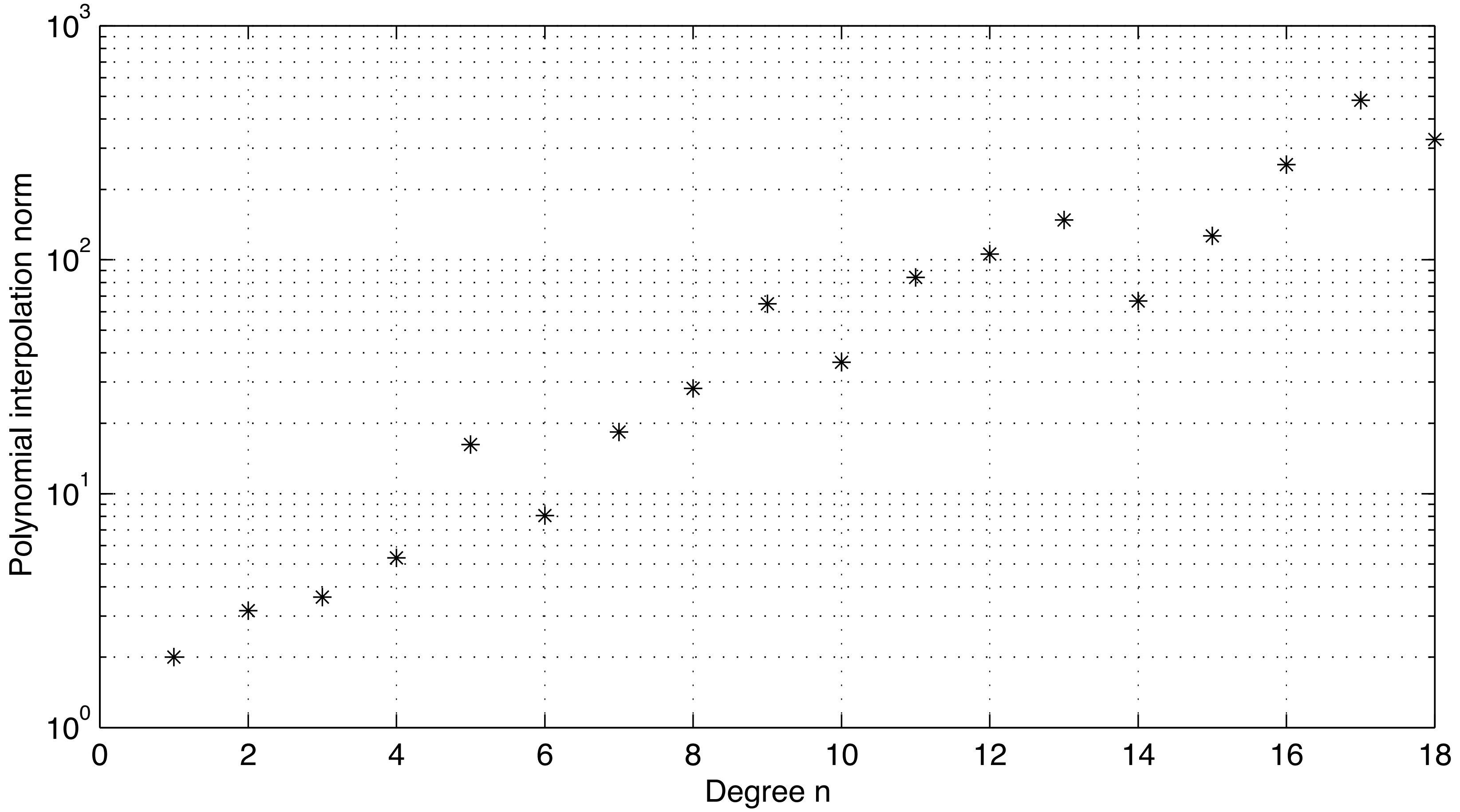


Mesh norm of constrained CVT generators vs. degree of the polynomial n; the number of generators $k_n = (n + 1)^2$

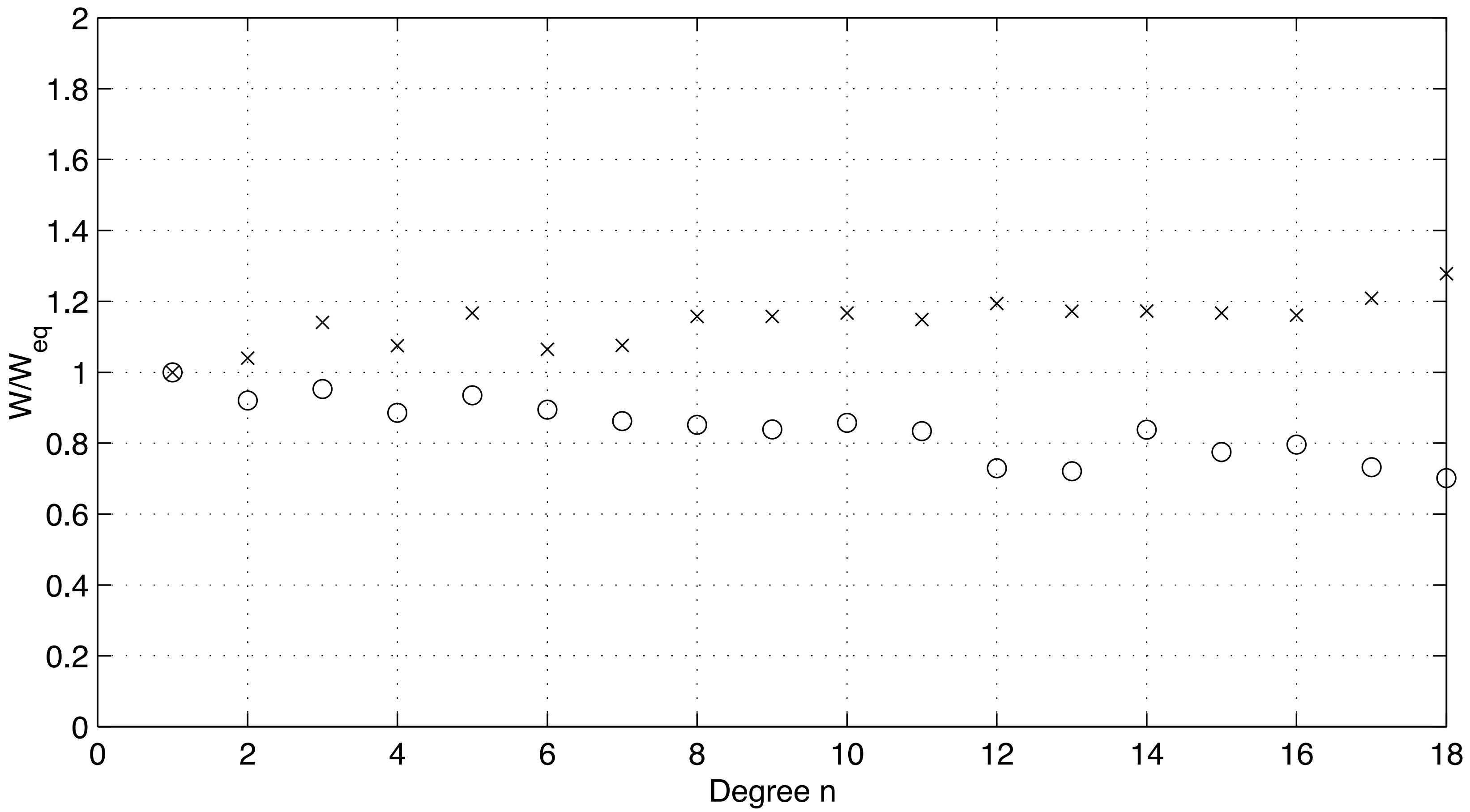
- Consider global polynomial interpolation on the sphere S^2
  - note that if the degree of interpolating polynomial is n, then $k_n = (n+1)^2$ interpolating points $\{\mathbf{x}_i\}_{i=1}^{k_n}$ are needed
  - the “goodness” of a set of interpolation points can be characterized by the uniform norm of the interpolation operator

- Also consider interpolatory quadrature on the surface of the sphere S^2 based on the interpolating polynomial of degree n
  - one would like all the quadrature weights to be positive and to be as uniform as posssible

- CCVT point sets yield high-quality quadrature rules, but there are better choices of interpolation points known
  - however, the better choices are much more expensive to compute
  - CCVT interpolation points, although not the best, are perfectly adequate for applications



Global polynomial interpolation norms using constrained CVT generators vs. degree of the polynomial n; the number of generators $k_n = (n+1)^2$



Maximum and minimum value of the quadrature weights (scaled by $4\pi/k_n$) when the quadrature points are chosen to be constrained CVT generators vs. degree of the polynomial n; the number of generators is $k_n = (n+1)^2$

- One important observation is that all algorithms discussed locate *local minimizers* of the energy functional
  - this may account for the lack of monotonicity in the plots
  - moreover, it is possible that if global minimizers were located, that the performance of CCVT point sets for global interpolation on the sphere would be as good as that of the best point sets known

Current work

- Connecting the density function used for generating the points with a priori and a posteriori error estimates
  - then adaptive point generation algorithms are easily defined

- Further explorations of the generation of anisotropic point distributions

- Further explorations of centroidal Voronoi tessellations under constraints

- Developing algorithms for determining global minimizers of the energy functional associated with centroidal Voronoi tessellations

- Using our point distributions to solve PDE's
  - grid generation
  - meshless methods
  - adaptive point placement

- Using centroidal Voronoi tessellations for model reduction

- More analyses