Geosci. Model Dev. Discuss., 6, 1427–1466, 2013
www.geosci-model-dev-discuss.net/6/1427/2013/
doi:10.5194/gmdd-6-1427-2013
© Author(s) 2013. CC Attribution 3.0 License.

Geoscientific Model Development Discussions Open Access

Parallel algorithms for planar and spherical Delaunay construction with an application to centroidal Voronoi tessellations

D. W. Jacobsen[1,2], M. Gunzburger[1], T. Ringler[2], J. Burkardt[1], and J. Peterson[1]

[1]Department of Scientific Computing, Florida State University, Tallahassee, FL 32306, USA
[2]Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Received: 5 February 2013 – Accepted: 6 February 2013 – Published: 22 February 2013

Correspondence to: D. W. Jacobsen (jacobsen.douglas@gmail.com)

Published by Copernicus Publications on behalf of the European Geosciences Union.

Abstract

A new algorithm, featuring overlapping domain decompositions, for the parallel construction of Delaunay and Voronoi tessellations is developed. Overlapping allows for the seamless stitching of the partial Delaunay tessellations constructed by individual processors. The algorithm is then modified, by the addition of stereographic projections, to handle the parallel construction of spherical Delaunay and Voronoi tessellations. The algorithms are then embedded into algorithms for the parallel construction of planar and spherical centroidal Voronoi tessellations that require multiple constructions of Delaunay tessellations. Computational tests are used to demonstrate the efficiency and scalability of the algorithms for spherical Delaunay and centroidal Voronoi tessellations. Compared to serial versions of the algorithm and to the STRIPACK-based approaches, the new parallel algorithm results in significant speedups for the construction of spherical centroidal Voronoi tessellations.

1 Introduction

Voronoi diagrams and their dual Delaunay tessellations have become, in many settings, natural choices for spatial gridding due to their ability to handle arbitrary boundaries and refinement well. Such grids are used in a wide range of applications and can, in principle, be created for almost any geometry in two and higher dimensions. The recent trend towards the exascale in most aspects of high-performance computing further demands fast algorithms for the generation of high-resolution spatial meshes that are also of high quality, e.g. featuring variable resolution with smooth transition regions; otherwise, the meshing part can dominate the rest of the discretization and solution processes. However, creating such meshes can be time consuming, especially for high-quality, high-resolution meshing. Attempts to speed up the generation of Delaunay tessellations via parallel divide-and-conquer algorithms were made in, e.g. Cignoni et al. (1998); Chernikov and Chrisochoides (2008). The few algorithms that do exist for the parallel construction of Delaunay triangulations are limited to two-dimensional planar surfaces. With the current need for high, variable-resolution grid generators in mind, we first develop a new algorithm that makes use of a novel approach to domain decomposition for the fast, parallel generation of Delaunay and Voronoi grids in Euclidean domains.

Centroidal Voronoi tessellations provide one approach for high-quality Delaunay and Voronoi grid generation (Du et al., 1999, 2003b, 2006b; Du and Gunzburger, 2002; Nguyen et al., 2008). The efficient construction of such grids involves an iterative process (Du et al., 1999) that calls for the determination of multiple Delaunay tessellations; we show how our new parallel Delaunay algorithm is especially useful in this context.

Climate modelling is a specific field which has recently begun adopting Voronoi tessellations as well as triangular meshes for the spatial discretization of partial differential equations (Pain et al., 2005; Weller et al., 2009; Ju et al., 2008, 2011; Duda et al., 2011). As this is a special interest of ours, we also develop a new parallel algorithm for the generation of Voronoi and Delaunay tessellations for the entire sphere or some subregion of interest. The algorithm uses stereographic projections to transform these tasks into planar Delaunay constructions for which we apply the new parallel algorithm we have developed for that purpose. Spherical centroidal Voronoi tessellation (SCVT) based grids are especially desirable in climate modelling because they not only provide for precise grid refinement, but also feature smooth transition regions (Ringler et al., 2011). We show how such grids can be generated using the new parallel algorithm for spherical Delaunay tessellations.

The paper is organized in the following fashion. In Sect. 2.1, background material about Delaunay and Voronoi tessellations is provided and then, in Sect. 2.2, we present our new parallel algorithm for the construction of such tessellations. In Sect. 2.3, we provide a brief discussion of centroidal Voronoi tessellations (CVTs) followed by showing how the new parallel Delaunay tessellation algorithm can be incorporated into an efficient method for the construction of CVTs. In Sect. 3.1, we provide a short review of stereographic projections followed, in Sect. 3.2, by a presentation of the new parallel algorithm for the construction of spherical Delaunay and Voronoi tessellations. In

Sect. 3.3, we consider the parallel construction of spherical CVTs. In Sect. 4, the results of numerical experiments and demonstrations of the new algorithms are given; for the sake of brevity, we only consider the algorithms for grid generation on the sphere. Finally, in Sect. 5, concluding remarks are provided.

2 Parallel Delaunay and Voronoi tessellation construction in $\mathbb{R}^k$

In this section, we present a new method for the construction of Delaunay and Voronoi tessellations. We begin with a brief review of the definitions and some of the properties of these tessellations.

2.1 Delaunay and Voronoi tessellations

Let $k+1$ points in $\mathbb{R}^k$ be in general position; this means that, for $s = 1, \ldots, k$, no subset of $s+1$ points lies on an $(s-1)$-dimensional hyperplane, i.e. no subset of $s+1 = 3$ points lies on a line, no subset of $s+1 = 4$ points lies on a two-dimensional plane, ..., the $k+1$ given points do not lie on an $(k-1)$-dimensional hyperplane. A k-*simplex* is the k-dimensional polytope which is the convex hull of the $k+1$ points; a k-simplex is the smallest convex set containing the $k+1$ points and the given points are the *vertices* of the simplex. A 1-simplex is a line segment, a 2-simplex is a triangle, a 3-simplex is a tetrahedron, and a 4-simplex is a pentachoron; by convention, a 0-simplex is a point. A k-simplex has s-*faces* ($s \le k$) corresponding to any $s+1$ distinct vertices of the k-simplex; s-faces are themselves s-simplices, e.g. a k-face is the simplex itself, ..., a 3-face is a tetrahedron, a 2-face is a triangle, a 1-face is an edge, and a 0-face is a vertex.

Given a set of points $P = \{\boldsymbol{x}_j\}_{j=1}^n$ in $\mathbb{R}^k$, the *Delaunay tessellation* $D(P)$ of the point set is the set of k-simplices such that:

- a point $\boldsymbol{p} \in \mathbb{R}^k$ is a vertex of a simplex in $D(P)$ if and only if $\boldsymbol{p} \in P$;

- the intersection of two simplices in $D(P)$ is either the empty set or a common face;

– the interior of the circumscribing k-sphere through the $k+1$ vertices of any simplex in $D(P)$ contains no other points from the set P; in the sequel, we refer this as the *circumsphere property*, or, in two dimensions, the *circumcircle property*.

If the circumscribing k-sphere of a simplex in $D(P)$ has more than $k+1$ points lying on its surface, the Delaunay tessellation is not unique. The Delaunay triangulation of a point set defined in $\mathbb{R}^k$ is related to the convex hull of the point set when projected onto a paraboloid in $\mathbb{R}^{k+1}$ (Cignoni et al., 1998).

Given a set of points $P = \{\boldsymbol{x}_j\}_{j=1}^n$ in $\mathbb{R}^k$, the *Voronoi region* or *Voronoi cell* associated with a particular point $\boldsymbol{x}_i$ is the subset $V_i \subset \mathbb{R}^k$ defined by

$$V_i = \{\boldsymbol{x} \in \mathbb{R}^k \quad : \quad \|\boldsymbol{x} - \boldsymbol{x}_i\| < \|\boldsymbol{x} - \boldsymbol{x}_j\| \quad \forall j \neq i\},$$

where here and throughout, $\|\cdot\|$ denotes the Euclidean norm. This property, referred to as the *Voronoi property*, states that the Voronoi cell V_i consists of all points in $\mathbb{R}^k$ that are closer to $\boldsymbol{x}_i$ than to any of the other points in P. The set $V(P) = \{V_j\}_{j=1}^n$ of Voronoi regions is referred to as a *Voronoi tessellation* or *Voronoi diagram* of the point set P and the points in P are referred to as the *generators* of the Voronoi tessellation. Some of the Voronoi regions are infinite in extent. However, we can also view a Voronoi tessellation of any bounded region Ω containing the point set $P = \{\boldsymbol{x}_j\}_{j=1}^n$ in which case all the Voronoi regions are finite in extent and are given by $V_i = \Omega \cap \{\boldsymbol{x} \in \mathbb{R}^k \ : \ |\boldsymbol{x} - \boldsymbol{x}_i| < |\boldsymbol{x} - \boldsymbol{x}_j| \quad \forall j \neq i\}$. In general, Voronoi cells are polytopes in $\mathbb{R}^k$, except that when they intersect the boundary of Ω, part of the boundary of the Voronoi cell is a portion of the boundary of Ω.

Given a set P of points in $\mathbb{R}^k$, the corresponding Delaunay and Voronoi tessellations are topological dual tessellations, e.g. if you connect all pairs of generators whose Voronoi cells share a common face, you obtain the Delaunay tessellation. Also, the generators of the Voronoi tessellation are the vertices of the Delaunay tessellation. See Okabe et al. (2000) for details about Delaunay and Voronoi tessellations.

2.2 Parallel algorithm for Delaunay tessellation construction

The construction of planar Delaunay triangulations in parallel has been of interest for several years; see, e.g. Amato and Preparata (1993); Cignoni et al. (1998); Zhou et al. (2001); Batista et al. (2010). Typically, such algorithms divide the point set up into several smaller subsets, each of which can then be triangulated independently from the others. The resulting triangulations need to be stitched together to form a global triangulation. This stitching, or merge step, is typically computed serially because one may need to modify significant portions of the individual triangulations. The merge step is the main difference between the different parallel algorithms. Here, we provide a new alternative merge step that, because no modifications of the individual triangulations are needed, can be performed in parallel.

We are given a set of points $P = \{\boldsymbol{x}_j\}_{j=1}^{n}$ in a given domain $\Omega \subset \mathbb{R}^k$. We begin by covering Ω by a set $S = \{S_k(\boldsymbol{c}_k, r_k)\}_{k=1}^{N}$ of N overlapping spheres S_k, each of which is defined by a centre point $\boldsymbol{c}_k$ and radius r_k. For each sphere S_k, a connectivity or neighbour list is defined which consists of the indices of all the spheres $S_i \in S$, $i \neq k$, that overlap with S_k. From the given point set P, we then create N smaller point sets P_k, $k = 1, \ldots, N$, each of which consists of the points in P which are in the sphere S_k. Due to the overlap of the spheres S_k, a point in P may belong to multiple points sets P_k.

The next step is to construct the N Delaunay tessellations T_k, $k = 1, \ldots, N$, of the N point sets P_k, $k = 1, \ldots, N$. For this purpose, one can use any Delaunay tessellation method at one's disposal; for example, in the plane, one can use the Delaunay triangulator that is part of the Triangle software of Shewchuk (1996). These Delaunay tessellations, of course, can be constructed completely in parallel after assigning one point set P_k to each of N processors.

At this point we are almost but not quite ready to merge the N local Delaunay tessellations into a single global one. Before doing so, we deal with the fact that although, by construction, the tessellation T_k of the point set P_k is a Delaunay tessellation of P_k, there are very likely to be simplices in the tessellation T_k that may not be globally Delaunay, i.e. that may not be Delaunay with respect to points in P that are not in P_k. This follows because the points in any T_k are unaware of the triangles and points outside of S_k so that there may be points in P that are not in P_k that lie within the circumsphere of a simplex in T_k. In fact, only simplices whose circumspheres are completely contained inside of the ball S_k are guaranteed to be globally Delaunay; those points satisfy the criteria

$$\|\boldsymbol{c}_k - \widehat{\boldsymbol{c}}_i\| + \widehat{r}_i < r_k, \tag{1}$$

where $\widehat{\boldsymbol{c}}_i$ and $\widehat{r}_i$ are the centre and radius of the circumsphere of the ith triangle in the local Delaunay tessellation T_k.

So, at this point, for each $k = 1,...,N$, we construct the triangulation $\widehat{T}_k$ by discarding all simplices in the local Delaunay tessellation T_k whose circumspheres are not completely contained in S_k, i.e. all simplices that do not satisfy Eq. (1). Figure 1 shows one of the local Delaunay triangulations T_k and the triangulation $\widehat{T}_k$ after the deletion of triangles that are not guaranteed to satisfy the Delaunay property globally.

The final step is to merge the N modified tessellations $\widehat{T}_k$, $k = 1,...,N$, that have been constructed completely in parallel into a single global tessellation. The key observation here is that *each regional tessellation* $\widehat{T}_k$ *is now exactly a portion of the global Delaunay tessellation* because if two local tessellations $\widehat{T}_i$ and $\widehat{T}_k$ overlap, they must coincide wherever they overlap. This follows from the uniqueness property of the Delaunay tessellations. Thus, *the union of the local Delaunay tessellations is the global Delaunay tessellation*; by using an overlapping domain decomposition, the stitching of the regional Delaunay tessellations into a global one is transparent. Of course, some bookkeeping chores have to be done such as rewriting simplex information in terms of global point indices and counting overlapping simplices only once.

A final note is that the radii $\{r_k\}_{k=1}^N$ should be chosen large enough so that there are no gaps between the regional Delaunay triangulations after the possibly non-globally Delaunay triangles are deleted, i.e. no gaps appear in the union of the $\widehat{T}_k$s.

For quasi-uniform grids, choosing r_k as the maximum distance from the centre $\boldsymbol{c}_k$ of its ball S_k to the centre of all its adjacent balls allows enough overlap; as discussed in Sect. 4.2.2, this heuristic may not be optimal for variable resolution grids.

In summary, the algorithm for the construction of a Delaunay tessellation in parallel consists of the following steps:

- define overlapping balls S_k, $k = 1,...,N$, that cover the given region Ω;
- sort the given point set P into the subsets P_k, $k = 1,...,N$, each containing points in P that are in S_k;
- for $k = 1,...,N$, construct in parallel the N Delaunay tessellations T_k of the points sets P_k;
- for $k = 1,...,N$, construct in parallel the tessellation $\widehat{T}_k$ by removing from T_k all simplices that are not guaranteed to satisfy the circumsphere property;
- construct the Delaunay tessellation of P as the union of the N modified Delaunay tessellations $\{\widehat{T}_k\}_{k=1}^N$.

Once a Delaunay tessellation is determined, it is an easy matter, at least for domains in $\mathbb{R}^2$ and $\mathbb{R}^3$, to construct the dual Voronoi tessellation; see, e.g. Okabe et al. (2000).

2.3 Application to the construction of centroidal Voronoi and Delaunay tessellations

Given a Voronoi tessellation $V = \{V_j\}_{j=1}^n$ of a bounded region $\Omega \subset \mathbb{R}^k$ corresponding to the set of generators $P = \{\boldsymbol{x}_j\}_{j=1}^n$ and given a nonnegative function $\rho(\boldsymbol{x})$ defined over Ω, referred to as the *point-density function*, we can define the *centre of mass* or *centroid*

$\boldsymbol{x}_j^*$ of each Voronoi region as

$$\boldsymbol{x}_j^* = \frac{\int_{V_j} \boldsymbol{x}\rho(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}{\int_{V_j} \rho(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}, \qquad j = 1,\ldots,n. \tag{2}$$

In general, $\boldsymbol{x}_j \neq \boldsymbol{x}_j^*$, i.e. the generators of the Voronoi cells do not coincide with the centres of mass of those cells. The special case for which $\boldsymbol{x}_j = \boldsymbol{x}_j^*$ for all $j = 1,\ldots,n$, i.e. for which all the generators coincide with the centres of mass of their Voronoi regions, is referred to as a *centroidal Voronoi tessellation* (CVT).

Given Ω, n, and $\rho(\boldsymbol{x})$, a CVT of Ω must be constructed. The simplest means for doing so is Lloyd's method (Lloyd, 1982) in which, starting with an initial set of n distinct generators, one constructs the corresponding Voronoi tessellation, then determines the centroid of each of the Voronoi cells, and then moves each generator to the centroid of its Voronoi cell. These steps are repeated until satisfactory convergence is achieved, e.g. until the movement of generators falls below a prescribed tolerance. The convergence properties of Lloyd's method are rigorously studied in Du et al. (2006a).

The point-density function ρ plays a crucial role in how the converged generators are distributed and the relative sizes of the corresponding Voronoi cells. If we arbitrarily select two Voronoi cells V_i and V_j from a CVT, their grid spacing and density are related as

$$\frac{h_i}{h_j} \approx \left(\frac{\rho(\boldsymbol{x}_j)}{\rho(\boldsymbol{x}_i)}\right)^{\frac{1}{d+2}}, \tag{3}$$

where h_i denotes a measure of the linear dimension, e.g. the diameter, of the cell V_i and $\boldsymbol{x}_i$ denotes a point, e.g. the generator, in V_i. Thus, the point-density function can be used to produce nonuniform grids in either a passive or adaptive manner by prescribing a ρ or by connecting ρ to an error indicator, respectively. Although the relation Eq. (3) is at the present time a conjecture, its validity has been demonstrated through many numerical studies; see, e.g. Du et al. (1999); Ringler et al. (2011). CVTs and their dual Delaunay tessellations have been successfully used for the generation of high-quality nonuniform grids; see, e.g. Du and Gunzburger (2002); Du et al. (2003b, 2006b); Ju et al. (2011); Nguyen et al. (2008).

As defined above, every iteration of Lloyd's method requires the construction of the Voronoi tessellation of the current set of generators followed by the determination of the centroids of the Voronoi cells. However, the construction of the Voronoi tessellation can be avoided until after the iteration has converged and instead, the iterative process can be carried out with only Delaunay tessellation constructions. Thus, the first task within every iteration of Lloyd's method is to construct the Delaunay tessellation of the current set of generators. The parallel algorithm for the generation of Delaunay tessellations given in Sect. 2.2 is thus especially useful in reducing the costs of the multiple tessellations needed in CVT construction.

After the Delaunay tessellation of the current generators is computed, every Voronoi cell centre of mass must be computed by integration, so its generator can be replaced by the centre of mass. Superficially, it seems that we cannot avoid constructing the Voronoi tessellation to do this. However, it is easy to see that one does not actually need the Voronoi tessellation and instead one can determine the needed centroids from the Delaunay tessellation. For simplicity, we restrict this discussion to tessellations in $\mathbb{R}^2$. Each triangle in a Delaunay triangulation contributes to the integration over three different Voronoi cells. As seen in Fig. 2, the triangle is split into three kites, each made up of two edge midpoints, the triangle circumcentre, and a vertex of the triangle. Each kite is part of the Voronoi cell whose generator is located at the triangle vertex associated with the kite. Integrating over each kite and updating a portion of the integrals in the centroid formula Eq. (2) allows one to only use the Delaunay triangulation to determine the centroids of the Voronoi cells. Thus, because both steps within each Lloyd iteration can be performed using only the Delaunay triangulation, determining the generators of a CVT does not require construction of any Voronoi tessellations nor does it require any mesh connectivity information. If one is interested in the CVT, one need only construct a single Voronoi tessellation after the Lloyd iteration has converged; if one is interested in the Delaunay tessellation corresponding to the CVT, then no Voronoi construction is needed.

To make this algorithm parallel, one can compute the Voronoi cell centroids using the local Delaunay tessellations and not on the stitched-together global one. However, because the local Delaunay tessellations overlap, one has to ensure that each generator is only updated by one region, i.e. by only one processor. This can be done using one of a variety of domain decomposition methods. We use a coarse Voronoi diagram corresponding to the region centres $\boldsymbol{c}_k$. Each processor only updates the generators that are inside of its coarse Voronoi cell. Because Voronoi cells are non-overlapping, each generator will only get updated by one processor. After all the generators are updated, each coarse region needs to transfer its newly updated points only to its adjacent regions and not to all active processors. This limits each processor's communications to roughly six sends and receives, regardless of the total number of processors used.

We use two metrics to *check for the convergence* of the Lloyd iteration, namely the ℓ_2 and ℓ_∞ norms of generator movement given by

$$\ell_2\text{norm} = \left(\frac{1}{n}\sum_{j=1}^{n}(\boldsymbol{x}_j^{\text{old}} - \boldsymbol{x}_j^{\text{new}})^2\right)^{1/2} \qquad \text{and} \qquad \ell_\infty\text{norm} = \max_{j=1,\dots,n}(|\boldsymbol{x}_j^{\text{old}} - \boldsymbol{x}_j^{\text{new}}|),$$

respectively, are compared with a given tolerance; here, old and new refer to the previous and current Lloyd iterates. If either norm falls below the tolerance, the iterative process is deemed to have converged. The ℓ_∞ norm is more strict, but both norms follow similar convergence paths when plotted against the iteration number. Other metrics can be used such as the clustering energy (Du et al., 1999)

$$\text{CE} = \sum_{j=1}^{n}\int_{V_j}(\rho(\boldsymbol{x})\|\boldsymbol{x} - \boldsymbol{x}_j\|^2\,\mathrm{d}\boldsymbol{x}).$$

However, in practice, this choice tends to be less strict and more computationally expensive when compared with the use of generator movement.

A variety of point sets can be used to *initiate* Lloyd's method for CVT construction. The obvious one is Monte Carlo points (Metropolis and Ulam, 1949). These can either be sampled uniformly over Ω or sampled according to the point-density function $\rho(\boldsymbol{x})$. The latter approach usually reduces the number of iterations required for convergence. One can instead use a bisection method to build fine grids from a coarse grid (Heikes and Randall, 1995). To create a bisection grid, a coarse CVT is constructed using as few points as possible. After this coarse grid is converged, in $\mathbb{R}^2$, one would add the midpoint of every Voronoi cell edge or Delaunay triangle edge to the set of points. This causes the overall grid spacing to be reduced by roughly a factor of two in every cell so that the refined point set is roughly four times larger.

In summary, the algorithm for the construction of a CVT in parallel consists of the following steps; the steps in Roman font are the same as in the algorithm of Sect. 2.2 whereas the italicized steps, as well as the do loop, are additions for CVT construction:

```
– define overlapping balls that cover the given region;
– **while** not converged **do**
    – sort the current point set;
    – construct in parallel the local Delaunay tessellations;
    – remove from the Delaunay tessellations simplices which are not guaranteed
      to satisfy the circumsphere property;
    – *in parallel, determine the centroids of the Voronoi cells by integrating over
      simplices*;
    – *move each generator to the corresponding cell centroid*;
    – *test the convergence criteria*;
    – *communicate new generator positions to neighbouring balls*;
```

GMDD
6, 1427–1466, 2013

- **end**
- construct the Delaunay tessellation corresponding to the CVT as the union of the modified local Delaunay tessellations.

3 Parallel Delaunay and Voronoi tessellation construction on the sphere

Replacing the constructs defined in Sect. 2.1 with their analogous components on the sphere, i.e. on the surface of a ball in $\mathbb{R}^3$, creates the spherical versions of Delaunay triangulations and Voronoi tessellations. The Euclidean distance metric is replaced by the geodesic distance, i.e. the shortest of the two lengths along the great circle joining two points on the sphere. Triangles and Voronoi cells are replaced with spherical triangles and spherical Voronoi cells whose boundaries are geodesic arcs. To develop a parallel Delaunay and Voronoi tessellation construction algorithm on the sphere, we could try to parallelize the STRIPACK algorithm of Renka (1997). Here, however, we take a new approach that is based on a combination of stereographic projections (which we briefly discuss in Sect. 3.1) and the algorithm of Sect. 2.2 applied to planar domains. The serial version of the new Delaunay tessellation construction algorithm is therefore novel as well.

3.1 Stereographic projections

Stereographic projections are special mappings between the surface of a sphere and a plane tangent to the sphere. Not only are stereographic projections conformal mappings, meaning that angles are preserved, but they also preserve circularity, meaning that circles on the sphere are mapped to circles on the plane. Stereographic projections also map the interior of these circles to the interior of the mapped circles (Bowers et al., 1998; Saalfeld, 1999). For our purposes, the importance of circularity preservation is that it implies that stereographic projections preserve the Delaunay circumcircle property as defined in Sect. 2.1. This follows because triangle circumcircles (along with their interiors) are preserved. Therefore, Delaunay triangulations on the sphere are mapped to Delaunay triangulations on the plane and conversely. As a result, stereographic projections can be used to construct a Delaunay triangulation of a portion of the sphere by first constructing a Delaunay triangulation in the plane, which is a simpler and well-studied task.

Without loss of generality, we assume that we are given the unit sphere in $\mathbb{R}^3$ centred at the origin. We are also given a plane tangent to the sphere at the point $\boldsymbol{t}$. The focus point $\boldsymbol{f}$ is defined as the reflection of $\boldsymbol{t}$ about the centre of the sphere, i.e. $\boldsymbol{f}$ is the antipode of $\boldsymbol{t}$. Let $\boldsymbol{p}$ denote a point on the unit sphere. The stereographic projection of $\boldsymbol{p}$ onto the plane tangent at $\boldsymbol{t}$ is the point $\boldsymbol{q}$ on the plane defined by

$$\boldsymbol{q} = s\boldsymbol{p} + (1 - s)\boldsymbol{f}, \qquad \text{where} \qquad s = 2\frac{1}{\boldsymbol{f} \cdot (\boldsymbol{f} - \boldsymbol{p})}. \tag{4}$$

Figure 3 illustrates the stereographic projection. For our purposes, it is more convenient to define the projection relative to $\boldsymbol{t}$ rather than $\boldsymbol{f}$. The simple substitution of $\boldsymbol{f} = -\boldsymbol{t}$ into Eq. (4) results in

$$\boldsymbol{q} = s\boldsymbol{p} + (s - 1)\boldsymbol{t}, \qquad \text{where} \qquad s = 2\frac{1}{\boldsymbol{t} \cdot (\boldsymbol{p} + \boldsymbol{t})}. \tag{5}$$

The definitions Eqs. (4) and (5) can also be used to define stereographic projections in $\mathbb{R}^k$ for $k > 3$.

3.2 Parallel algorithm for spherical Delaunay triangulation construction

A spherical Delaunay triangulation (SDT) is the Delaunay triangulation of a point set P defined on the surface of the sphere. We adapt the algorithm developed in Sect. 2.2 for domains in $\mathbb{R}^k$ to develop a parallel algorithm for the construction of SDTs. The most important adaptation is to use stereographic projections so that the actual construction of Delaunay triangulations is done on planar domains.

We are now given a set of points $P = \{\boldsymbol{x}_j\}_{j=1}^n$ on a subset Ω of the sphere; Ω may be the whole sphere. We begin by covering Ω by a set $U = \{U_k(\boldsymbol{t}_k, r_k)\}_{k=1}^N$ of N overlapping "umbrellas" or spherical caps U_k, each of which is defined by a centre point $\boldsymbol{t}_k$ and geodesic radius r_k. See the left plot in Fig. 4. For each spherical cap U_k, a connectivity (or neighbours) list is defined that consists of the indices of all the spherical caps $U_i \in U$, $i \neq k$, that overlap with U_k. From the given point set P, we then create N smaller point sets P_k, $k = 1, \ldots, N$, each of which consists of the points in P which are in the spherical cap U_k. Due to the overlap of the spherical caps U_k, a point in P may end up belonging to multiple points sets P_k.

At this point in Sect. 2.2, we assigned each of the point subsets P_k to a processor and constructed N Delaunay tessellations in parallel. Before doing so in the spherical case, we project each of the point sets P_k onto the plane tangent at the corresponding point $\boldsymbol{t}_k$. This additional step allows each processor to construct a planar Delaunay triangulation instead of spherical one. Specifically, for each k, we construct the planar point set $\widetilde{P}_k = \mathcal{S}(P_k; \boldsymbol{t}_k)$, where $\mathcal{S}(P_k; \boldsymbol{t}_k)$ denotes the stereographic projection of the points in P_k onto the plane tangent to the sphere at the point $\boldsymbol{t}_k$. We then assign each of the point sets $\widetilde{P}_k$ to a different processor and have each processor construct the planar Delaunay triangulation of its point set $\widetilde{P}_k$. Because stereographic projections preserve circularity and therefore preserve the Delaunay circumcircle property, it is then just a simple matter of keeping track of triangle vertex and edge indices to define the spherical Delaunay triangulations of the point sets P_k.

As in Sect. 2.2, we now proceed to remove spherical triangles that are not guaranteed to be Delaunay with respect to the global point set P. Instead of Eq. (1), in the spherical case, triangles must satisfy

$$\cos^{-1}\|\boldsymbol{t}_k - \widehat{\boldsymbol{c}}_i\| + \widehat{r}_i < r_k,$$

for that guarantee to be in effect, where $\widehat{\boldsymbol{t}}_i$ and $\widehat{r}_i$ are the centre and radius, respectively, of the circumsphere of the ith triangle in the local Delaunay tessellation of P_k. Once the possibly unwanted triangles are removed, we can, as in Sect. 2.2, transparently stitch together the modified Delaunay triangulations into a global one.

In summary, the algorithm for the construction of a spherical Delaunay tessellation in parallel consists of the following steps, where the italicized steps are the ones added to the algorithm of Sect. 2.2:

- define overlapping spherical caps U_k, $k = 1, \ldots, N$, that cover the given region Ω on the sphere;

- sort the given point set P into the subsets P_k, $k = 1, \ldots, N$, each containing the points in P that are in U_k;

- *for $k = 1, \ldots, N$, construct in parallel the point set $\widetilde{P}_k$ by stereographically projecting the points in P_k onto the plane tangent to the sphere at the point $\boldsymbol{t}_k$;*

- for $k = 1, \ldots, N$, construct in parallel the planar Delaunay triangulation $\widetilde{T}_k$ of the points set $\widetilde{P}_k$;

- *for $k = 1, \ldots, N$, construct in parallel the spherical Delaunay triangulation T_k by mapping the planar Delaunay triangulation $\widetilde{T}_k$ onto the sphere*;

- for $k = 1, \ldots, N$, construct in parallel the spherical triangulation $\widehat{T}_k$ by removing from T_k all simplices that are not guaranteed to satisfy the circumsphere property;

- construct the Delaunay tessellation of P as the union of the N modified spherical Delaunay tessellations $\{\widehat{T}_k\}_{k=1}^{N}$.

See the right plot in Fig. 4 for an illustration of a spherical Delaunay triangulation determined by this algorithm.

Because of the singularity in Eqs. (4) or (5) for $\boldsymbol{p} = \boldsymbol{f}$, the serial version of this algorithm, i.e. if $N = 1$, can run into trouble whenever the antipode $\boldsymbol{f}$ of the tangency point $\boldsymbol{t}$ is in the spherical domain Ω. Of course, this is always the case if Ω is the whole sphere.

In such cases, the number of subdomains used cannot be less than two, even if one is running the above algorithm in serial mode.

In setting the extent of overlap, i.e. the radii of the spherical caps, as the maximum geodesic distance from the a cap centre $\boldsymbol{t}_j$ to neighbouring cap centres $\boldsymbol{t}_i$, the geodesic distance is given by $\cos^{-1}(\boldsymbol{t}_j \cdot \boldsymbol{t}_i)$.

3.3 Application to the construction of spherical centroidal Voronoi tessellations (SCVTs)

The parallel CVT construction algorithm of Sect. 2.3 is easily adapted for the construction of centroidal Voronoi tessellations on the sphere (SCVTs). Obviously, one uses the spherical version of the Delaunay triangulation algorithm as given in Sect. 3.2 instead of the version of Sect. 2.2. One also has to deal with the fact that if one computes the centroid of a spherical Voronoi cell using Eq. (2), then that centroid does not lie on the sphere; a different definition of a centroid has to be used (Du et al., 2003a). Fortunately, it is shown in Du et al. (2003a) that the correct centroid can be determined by using Eq. (2), which yields a point inside the sphere, and then projecting that point onto the sphere (actually, this was shown to be true for general surfaces) so that the correct spherical centroid is simply the intersection of the radial line going through the point determined by Eq. (2) and the sphere.

4 Results

All results are created using a high performance computing (HPC) cluster with 24 AMD Opteron 6176 cores per node and 64GB of RAM per node.

4.1 Delaunay triangulations of the full sphere

We use STRIPACK (Renka, 1997), a serial Fortran 77 Association of Computing Machinery (ACM) Transactions on Mathematical Software (TOMS) algorithm that constructs Delaunay triangulations on a sphere, as a baseline for comparison with our approach. It is currently one of the few well-known spherical triangulation libraries available.

Table 1 compares the algorithm described in this paper, referred to as MPI-SCVT, with STRIPACK for the construction of spherical Delaunay triangulations. The results given compare the cost to compute a single triangulation of a 163 842 generator global grid that corresponds to a roughly 60 km grid on the surface of the Earth. The results show, for the computation of a full spherical triangulation MPI-SCVT is roughly a factor of 10 slower than STRIPACK in serial mode, and roughly 60 % slower when using 96 processors. Although this result shows STRIPACK out-preforming MPI-SCVT, this is only with respect to a single full triangulation of the sphere.

4.2 Initial generator placement and sorting heuristics for SCVT construction

We now examine the effects of initial generator placement and sorting heuristics with regards to SCVT generation. The results of this section are intended to guide decisions about which initial condition or sorting method we use later to generate SCVTs.

4.2.1 Initial generator placement

Because most climate models are shifting towards global high-resolution simulations, our target quasi-uniform grid is a global 15 km resolution grid, which corresponds to 2 621 442 grid points or Voronoi cells. We compare SCVT grids constructing starting with uniform Monte Carlo and bisection initial generator placement. The times for these grids to converge to a tolerance 10^{-6} in the ℓ_2 norm are presented. The 10^{-6} threshold is the strictest convergence level that the Monte Carlo grid can attain in a reasonable amount of time, and is therefore chosen as the convergence threshold for this study. However, the bisection grid can converge well beyond this threshold in a similar amount of time. Table 2 shows timing results for the parallel algorithm for the two different options for initial generator placement. Although the time spent converging a mesh with Monte Carlo initial placements is highly dependent on the initial point set, it is clear from this table that bisection initial conditions provide a significant speedup in the overall cost to generate an SCVT grid. Based on the results presented in Table 2 and unless otherwise specified, only bisection initial generator placements are used in subsequent experiments. Bisection initial conditions are not specific to this algorithm, and can be used to accelerate any SCVT grid generation method.

4.2.2 Sorting of points for assignment to processors

The heuristic mentioned in Sect. 2.2 determines regions associated with processors based on the maximum distance from region centres to their neighbouring centres; in the variable resolution grid setting, this approach does not provide good load balancing because in transition regions (where grid spacings can vary significantly), one ends up with larger region radii than required. To resolve this issue, we develop a new algorithm for determining regional point sets. We begin by sorting points into the cells of a coarse Voronoi tessellation of the region centres. For each region centre, the union of the points in its Voronoi cell and in its neighbouring Voronoi cells gives the final point set used for that region centre. This sorting method is more expensive when compared the method described in Sect. 2.2, but the resulting better load balancing reduces idle computing time from processors that have small loads. Timings using both approaches are given in Table 3. Figure 5 shows the number of points that each processor has to triangulate on a per iteration basis. Timings presented are for 163 842 generators, 42 regions, 42 processors, and are averages over 3000 iterations. Three sets of timings are given. Two use the approach of Sect. 2.2, one for a uniform grid and the other for a grid which, after convergence, has a sixteen to one ratio in its maximum and minimum grid sizes. Table 3 and Fig. 5 show that there is a significant advantage to the Voronoi-based decomposition in that it not only speeds up the overall cost per iteration, but also provides a more balanced loads across the processors. Note that, in Table 3, timings are taken relative to processor number 0, and as can be seen in Fig. 5a, processor 0 has a very small load so the majority of its iteration time is spent waiting for the processors with large loads to finish and catch up; this idling time is included in the Communication column of the table.

Based on the results in Table 3, the new Voronoi-based sorting approach is used for all subsequent results. This should provide appropriate load balancing when generating SCVT meshes. Using this sorting method should result in comparable quasi-uniform and variable resolution SCVT generator performance.

4.3 SCVT generation

We now provide both quasi-uniform and variable resolution SCVT generation results. The major contributor to the differences in computational performance arises as a result of load balancing differences. Results in this section make use of the density function

$$\rho(\boldsymbol{x}_i) = \frac{1}{2(1-\gamma)}\left[\tanh\left(\frac{\beta - |\boldsymbol{x}_c - \boldsymbol{x}_i|}{\alpha}\right) + 1\right] + \gamma \qquad (6)$$

which is visualized in Fig. 6, where $\boldsymbol{x}_i$ is constrained to lie on the surface of the unit sphere. This function results in relatively large value of ρ within a distance β of the point $\boldsymbol{x}_c$, where β is measured in radians and $\boldsymbol{x}_c$ is also constrained to lie on the surface of the sphere. The function transitions to relatively small values of ρ across a radian distance of α. The distance between $\boldsymbol{x}_c$ and $\boldsymbol{x}_i$ is computed as $|\boldsymbol{x}_c - \boldsymbol{x}_i| = \cos^{-1}(\boldsymbol{x}_c \cdot \boldsymbol{x}_i)$ with a range from 0 to π.

Figure 7 shows an example grid created using this density function, with $\boldsymbol{x}_c$ set to be $\phi_c = 3\pi/2$, $\lambda_c = \pi/6$, where ϕ denotes longitude and λ latitude, $\gamma = (1/8)^4$, $\beta = \pi/6$, and $\alpha = 0.20$ with 10 242 generators. This set of parameters used in Eq. (6) is referred to as x8. The quasi-uniform version is referred to as x1.

In Sect. 4.1, we showed that MPI-SCVT performs comparably to STRIPACK when computing a single full triangulation. However, computing a full triangulation is only part of the story. For SCVT generation, a triangulation needs to be computed at every iteration of Lloyd's algorithm as described in Sect. 2.3. When using STRIPACK, the full triangulation needs to be computed at every iteration, but with MPI-SCVT only each regional triangulation needs to be computed at each iteration. This means the merge step can be skipped resulting in significantly cheaper triangulations.

Figure 8 shows the performance of a STRIPACK-based SCVT construction as the number of generators is increased through bisection as mentioned in Sect. 2.3. Values are averages over 2000 iterations. The green dashed line represents the portion of the code that computes the centroids of the Voronoi regions whereas the red solid line represent the portion of the code that computes the Delaunay triangulation.

Table 4 compares STRIPACK with the triangulation routine in MPI-SCVT that is called on every iteration. The results presented relative to MPI-SCVT are averages over 2000 iterations.

As a comparison with Fig. 8, in Figs. 9 and 10 we present timings made for MPI-SCVT for two and 96 regions and processors, respectively, as the problem size, i.e. the number of generators, increases. A minimum of two processors are used because the stereographic projection used in MPI-SCVT has a singularity at the focus point. Eventually, at around 2 621 442 generators, the triangulation becomes more expensive than the integration step.

Whereas Fig. 9 shows performance similar to that of STRIPACK (see Fig. 8), Fig. 10 shows roughly two orders of magnitude faster performance relative to STRIPACK. As mentioned previously, this is only the case when creating SCVTs as the full triangulation is no longer required when computing a SCVT in parallel.

4.4 General algorithm performance

This section is intended to showcase some general performance results of MPI-SCVT. Figure 11a–c shows the timings for a 40 962 generator grid (roughly 120 km global resolution), a 163 842 generator grid (60 km resolution), and a 2 621 442 generator grid (15 km resolution), respectively.

To assess the overall performance of the MPI-SCVT algorithm, scalability results are presented in Fig. 12. Figure 12a shows that this algorithm can easily under-saturate processors; when this happens, communication ends up dominating the overall runtime for the algorithm which is seen in Fig. 11a; as a result, scalability ends up being sub-linear. As the number of generators increases (as seen in Fig. 12b, c), the limit for being under-saturated is higher. Currently in the algorithm, communications are done asynchronously using non-blocking sends and receives. Also, overall communications are reduced by only communicating with a region's neighbours. This is possible because points can only move within a region radius on any two subsequent iterations, and because of this can only move into another region which is overlapping the current region. More efficiency gains could be realized through improvements in the communication and integration algorithms and could result in linear scaling. In principle, since all of the computation is local this algorithm should scale linearly very well up to hundreds if not thousands of processors.

5 Summary

A novel technique for the parallel construction of Delaunay triangulations is presented. This parallel algorithm can be applied to the generation of planar and spherical centroidal Voronoi tessellations. Results were presented for the generation of spherical centroidal Voronoi tessellations, with comparisons to STRIPACK, a well-known algorithm for the computation of spherical Delaunay triangulations. The algorithm presented in the paper (MPI-SCVT) shows slower performance than STRIPACK when computing a single triangulation in serial and comparable performance when using roughly 100 processors. When paired with a SCVT generator, the algorithm shows significant speed up relative to a STRIPACK based SCVT generator. The implementation of MPI-SCVT described and explored in this paper can be freely downloaded at (http://sourceforge.net/projects/mpi-scvt/).

Acknowledgements. We would like to thank Geoff Womeldorff and Michael Duda for many useful discussions. The work of Doug Jacobsen, Max Gunzburger, John Burkardt, and Janet Peterson was supported by the US Department of Energy under grants number DE-SC0002624 and DE-FG02-07ER64432.

References

Amato, N. and Preparata, F.: An NC parallel 3D convex hull algorithm, in: Proceedings of the ninth annual symposium on Computational geometry – SCG '93, 289–297, May 1993. 1432

Batista, V., Millman, D., Pion, S., and Singler, J.: Parallel geometric algorithms for multi-core computers, Comp. Geom.-Theor. Appl., 43, 663–677, 2010. 1432

Bowers, P., Diets, W., and Keeling, S.: Fast algorithms for generating Delaunay interpolation elements for domain decomposition, available at: http://www.math.fsu.edu/~aluffi/archive/paper77.ps.gz, (last access: May 2011), 1998. 1439

Chernikov, A. and Chrisochoides, N.: Algorithm 872: parallel 2D constrained Delaunay mesh generation, ACM T. Math. Software, 34, 6:1–6:20, 2008. 1428

Cignoni, P., Montani, C., and Scopigno, R.: DeWall: a fast divide and conquer Delaunay triangulation algorithm in E-d, Comput. Aided Design, 30, 333–341, 1998. 1428, 1431, 1432

Du, Q. and Gunzburger, M.: Grid generation and optimization based on centroidal Voronoi tessellationss, Appl. Math. Comput., 133, 591–607, 2002. 1429, 1436

Du, Q., Faber, V., and Gunzburger, M.: Centroidal Voronoi tessellations: applications and algorithms, SIAM Rev., 41, 637–676, 1999. 1429, 1436, 1437

Du, Q., Ju, L., and Gunzburger, M.: Constrained centroidal Voronoi tessellations for surfaces, SIAM J. Sci. Comput., 24, 1488–1506, 2003a. 1443

Du, Q., Ju, L., and Gunzburger, M.: Voronoi-based finite volume methods, optimal Voronoi meshes, and PDEs on the sphere, Comput. Method. Appl. M., 192, 3933–3957, 2003b. 1429, 1436

Du, Q., Emelianenko, M., and Ju, L.: Convergence of the Lloyd Algorithm for computing centroidal Voronoi tessellations, SIAM J. Numer. Anal., 44, 102–119, 2006a. 1435

Du, Q., Ju, L., and Gunzburger, M.: Adaptive finite element methods for elliptic PDE's based on conforming centroidal Voronoi Delaunay triangulations, SIAM J. Sci. Comput., 28, 2023–2053, 2006b. 1429, 1436

Duda, M., Jacobsen, D., Gunzburger, M., Ju, L., Ringler, T., and Skamarock, W.: Exploring a multi-resolution modeling approach within the shallow-water equations, Mon. Weather Rev., 139, 3348–3368, 2011. 1429

Heikes, R. and Randall, D.: Numerical integration of the shallow-water equations on a twisted icosahedral grid. Part I: Basic design and results of tests, Mon. Weather Rev., 123, 1862–1880, 1995. 1438

Ju, L., Ringler, T., and Gunzburger, M.: A multi-resolution method for climate system modeling: application of spherical centroidal Voronoi tessellations, Ocean Dynam., 58, 475–498, 2008. 1429

Ju, L., Ringler, T., and Gunzburger, M.: Voronoi tessellations and their application to climate and global modeling, Ocean Dynam., 58, 313–342, 2011. 1429, 1436

Lloyd, S.: Least squares quantization in PCM, IEEE T. Inform. Theory, 28, 129–137, 1982. 1435

Metropolis, N. and Ulam, S.: The Monte Carlo method, J. Am. Stat. Assoc., 44, 335–341, 1949. 1438

Nguyen, H., Burkardt, J., Gunzburger, M., Ju, L., and Saka, Y.: Constrained CVT meshes and a comparison of triangular mesh generators, Ocean Dynam., 42, 1–19, 2008. 1429, 1436

Okabe, A., Boots, B., Sugihara, K., and Chiu, S.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, John Wiley, Baffins Lane, Chichester, West Sussex, PO19 1UD, England, 2000. 1431, 1434

Pain, C. C., Piggot, M. D., Goddard, A. J. H., Fang, F., Gorman, G. J., Marshall, D. P., Eaton, M. D., Power, P. W., and de Oliveira, C. R. E.: Three-dimensional unstructured mesh ocean modelling, Ocean Model., 10, 5–33, 2005. 1429

Renka, R.: Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere, ACM T. Math. Software, 23, 416–434, 1997. 1439, 1443

Ringler, T. D., Jacobsen, D., Gunzburger, M., Ju, L., Duda, M., and Skamarock, W.: Exploring a multi-resolution modeling approach within the shallow-water equations, Mon. Weather Rev., accepted, 2011. 1429, 1436

Saalfeld, A.: Delaunay triangulations and stereographic projections, Cartogr. Geogr. Inform., 26, 289–296, 1999. 1439

Shewchuk, J.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator, Applied Computational Geometry: Towards Geometric Engineering, 1148, 203–222, 1996. 1432

Weller, H., Weller, H., and Fournier, A.: Voronoi, Delaunay, and block-structured mesh tefinement for solution of the shallow-water equations on the sphere, Mon. Weather Rev., 137, 4208–4224, 2009. 1429

Zhou, J., Deng, X., and Dymond, P.: A 2-D parallel convex hull algorithm with optimal communication phases, in: Proceedings 11th International Parallel Processing Symposium, April 1997, University of Geneva, Geneva, Switzerland, 596–602, 2001. 1432

Table 1. Comparison of STRIPACK with serial and parallel versions of MPI-SCVT for computation of a full spherical triangulation. Computed using the same initial set of 163 842 generators.

Algorithm	Processors	Regions	Time (ms)	Speedup (STRIPACK/MPI-SCVT)
STRIPACK	1	1	410.94	Baseline
MPI-SCVT	1	2	4276.95	10.41
MPI-SCVT	2	2	2216.92	5.39
MPI-SCVT	96	96	679.114	1.65

Table 2. Timing results for MPI-SCVT with bisection and Monte Carlo initial generator placements and the speedup of bisection relative to Monte Carlo. Final mesh contains 262 1442 generators.

Timed Portion	Bisection (B)	Monte Carlo (MC)	Speedup $\frac{MC}{B}$
Total Time (ms)	112 890	358 003 000	3171.25
Triangulation Time (ms)	10 887	48 034 600	4412.11
Integration Time (ms)	30 893	66 374 400	2148.52
Communication Time (ms)	70 878	110 958 000	1565.47

Table 3. Timings based on sorting approach used. Uniform uses a coarse quasi-uniform SCVT to define region centres and their associated radii and sorts using maximum distance between centres and neighbouring centres. x16 uses a coarse SCVT with a 16 to 1 ratio in grid sizes to effect the same type of sorting. Voronoi uses the same x16 coarse SCVT grid to define regions centres and sorts using the neighbouring Voronoi cell-based sort.

Sorting	Costs Of Different Algorithm Steps			Cost Per	
Approach	Triangulation	Integration	Communication	Iteration	Speedup
Uniform	14.5264	37.6193	1314.22	1396.53	Baseline
x16	35.8437	92.9066	865.323	995.039	1.40
Voronoi	39.8263	84.9515	200.721	325.832	4.28

Table 4. Comparisons of SCVT generators using STRIPACK, serial MPI-SCVT, and parallel MPI-SCVT. Cost per triangulation and iteration are presented. Speedup is compared using the cost per iteration. Computed using 163 842 generators. Variable resolution results are presented for MPI-SCVT only.

Algorithm	Procs	Regions	Triangulation Time (ms)	Iteration Time (ms)	Speedup Per Iteration (STRIPACK/MPI-SCVT)
STRIPACK	1	1	410.94	91185.43	Baseline
MPI-SCVT x1	1	2	3732.96	9741.2	9.36
MPI-SCVT x1	2	2	2216.92	5529.16	16.49
MPI-SCVT x1	96	96	37.6262	233.873	389.89
MPI-SCVT x8	96	96	39.8263	325.832	279.85

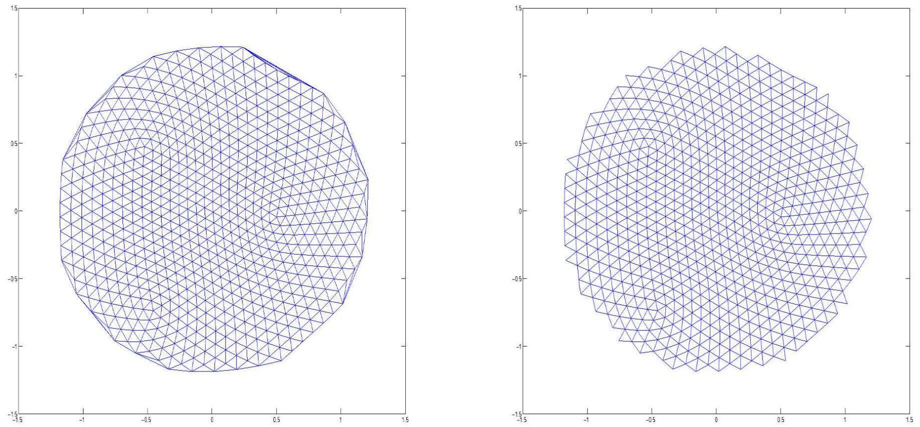


Fig. 1. A local Delaunay triangulation T_k (left) and the triangulation $\widehat{T}_k$ after the deletion of triangles that are not guaranteed to satisfy the Delaunay property globally.

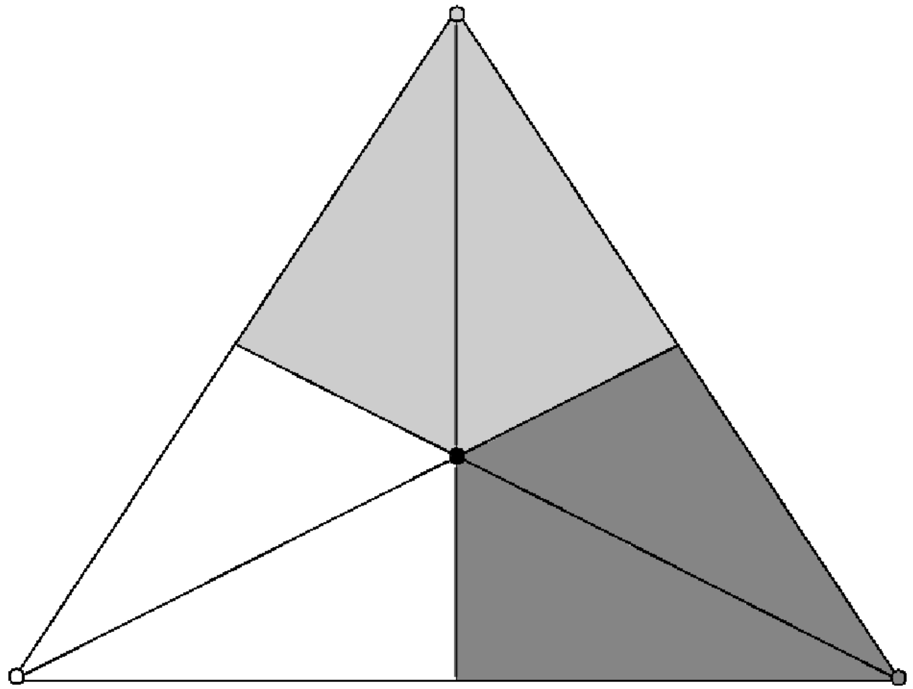


Fig. 2. Triangle subdivision used for integrating over Voronoi cells using only the Delaunay triangulation without any adjacency information. Kite sections contribute to the Voronoi cell associated with its vertex. The triangle vertices denote generators in the point set. Triangular regions that are shaded similarly form the kites.

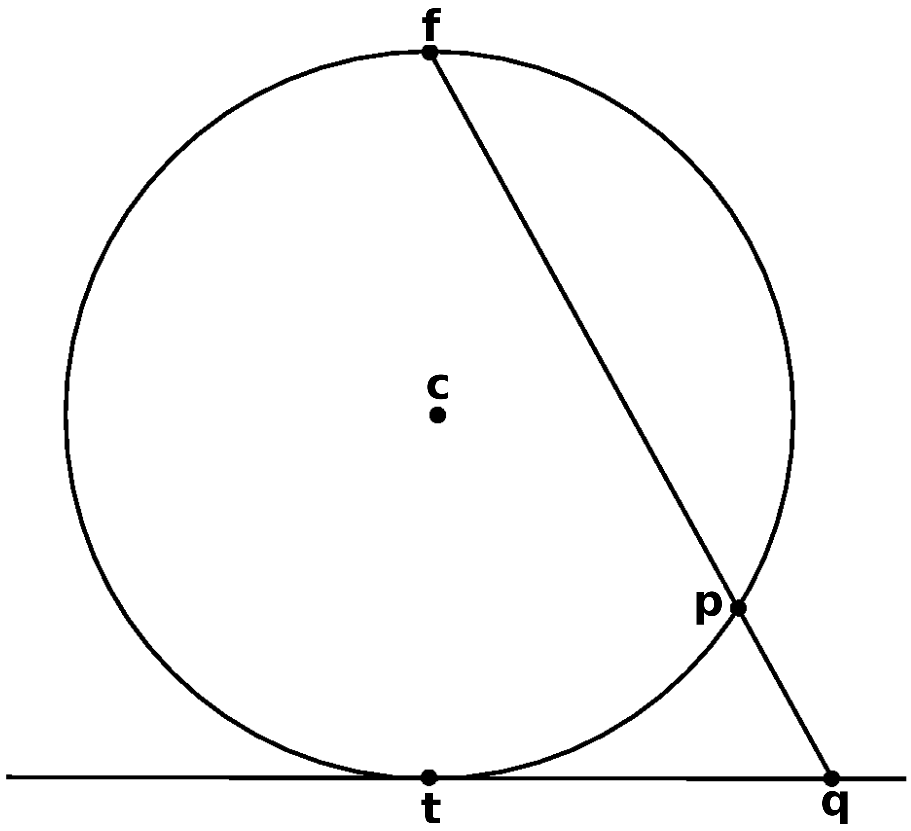


Fig. 3. Cross-sectional illustration of the stereographic projection ***q*** of a point ***p*** on the sphere onto the plane tangent to the sphere at the point ***t***.

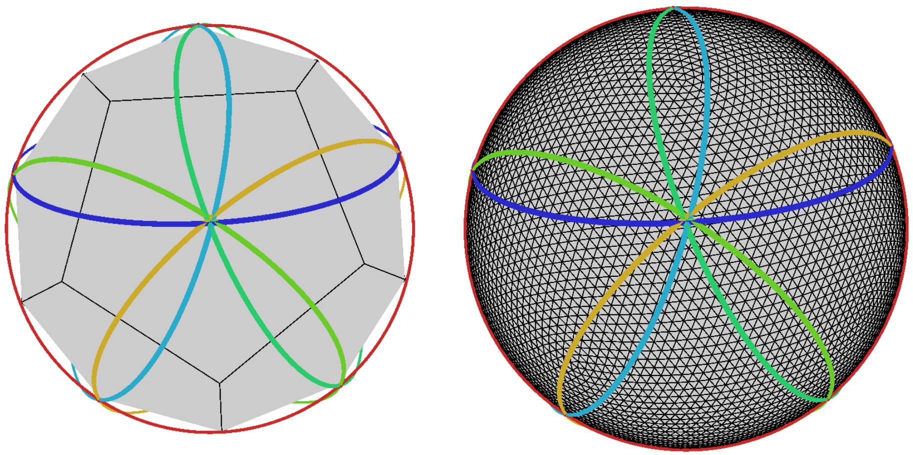
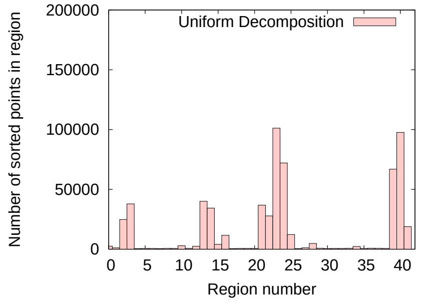
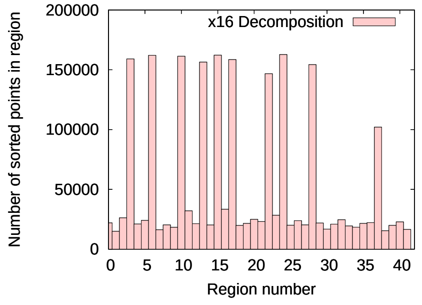


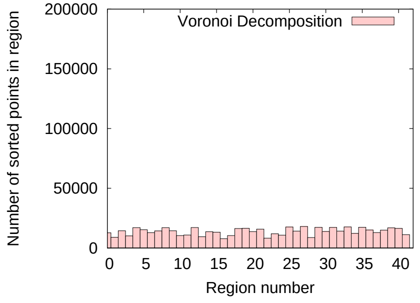
Fig. 4. Left: an overlapping subdivision of the sphere into $N = 12$ spherical caps. The cap centres $\boldsymbol{t}_k$ are the projections onto the sphere of the centroids of the 12 pentagons of the inscribed regular icosahedron. Each coloured ring represents the edge of cap of radius r_k. Right: a 10 242 generator Delaunay triangulation by the parallel algorithm.



(a) Uniform



(b) x16



(c) Voronoi

Fig. 5. Number of points each processor has to triangulate. **(a)**, **(b)**, and **(c)** use the sorting approaches corresponding to the three rows of Table 3. All plots were created with the same set of 163 842 generators.

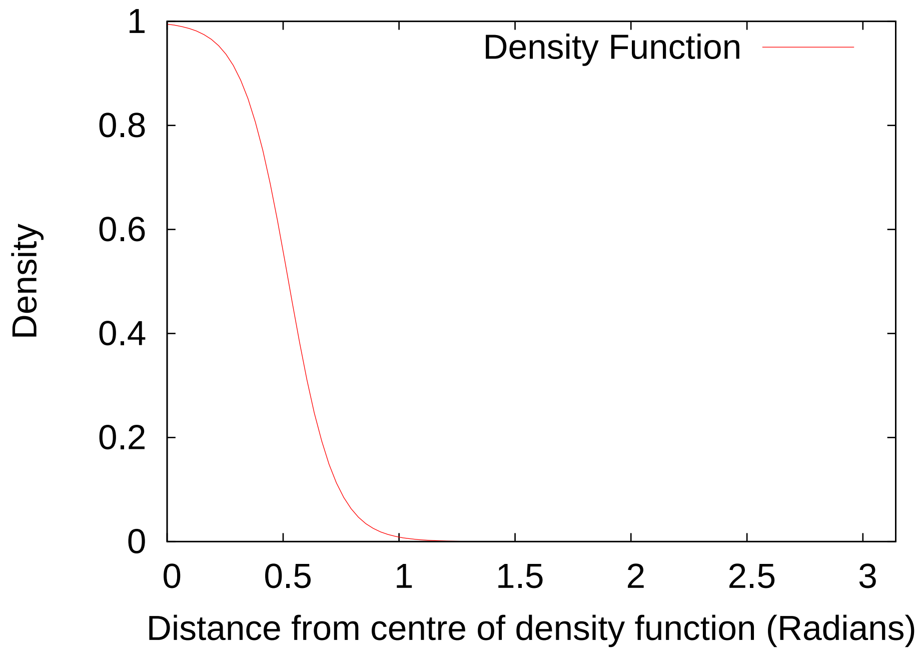


Fig. 6. Density function that creates a grid with resolutions that differ by a factor of 8 between the coarse and the fine regions. The maximum value of the density function is 1 whereas the minimum value is $(1/8)^4$.

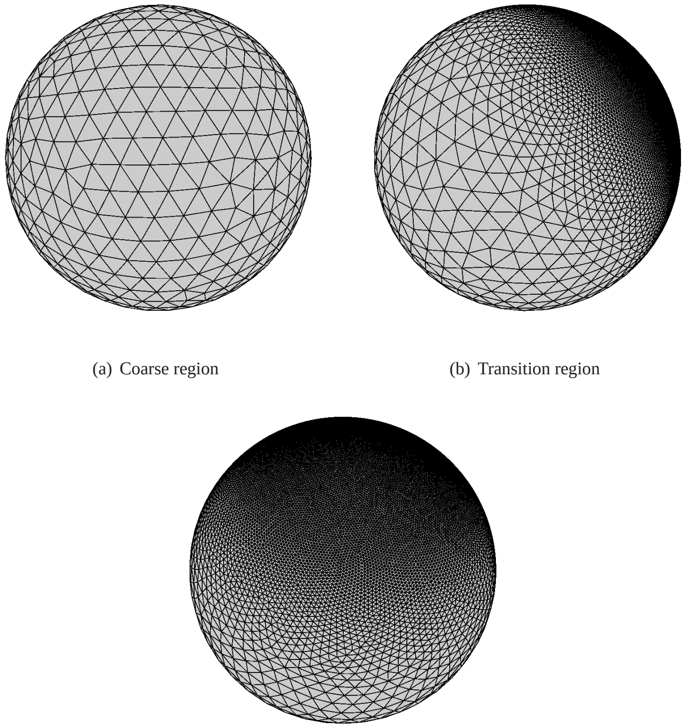


Fig. 7. Different views of the same variable resolution grid created using the density function Eq. (6). **(a)** shows the coarse region of the grid, **(b)** shows the transition region, and **(c)** shows the fine region.

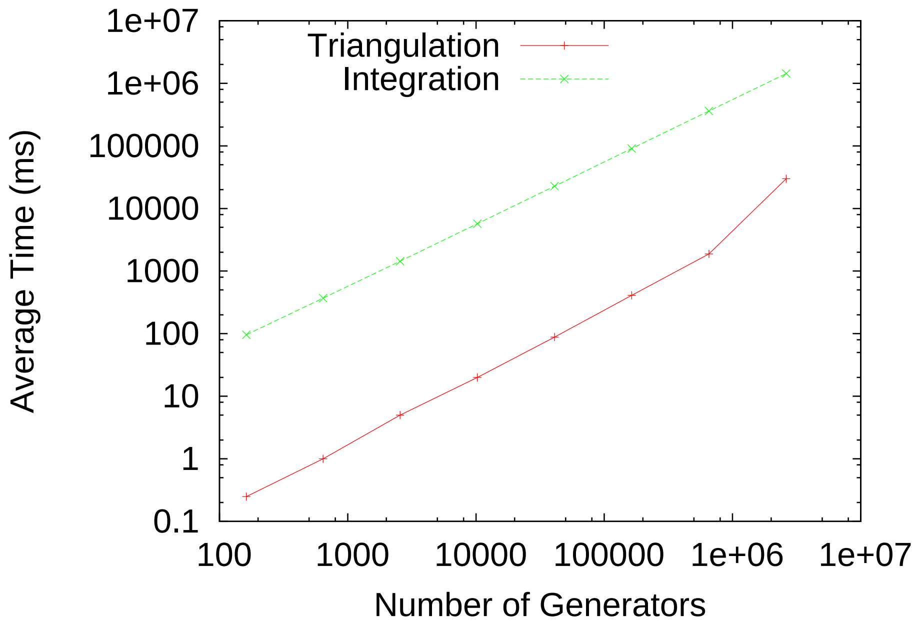


Fig. 8. Timings for STRIPACK-based SCVT construction for 162, 642, 10 242, 40 962, 163 842, 655 362, and 2 621 442 generators. Red solid lines represent the time spent in STRIPACK computing a triangulation whereas green dashed lines represent the time spent integrating the Voronoi cells outside of STRIPACK in one iteration of Lloyd's algorithm.

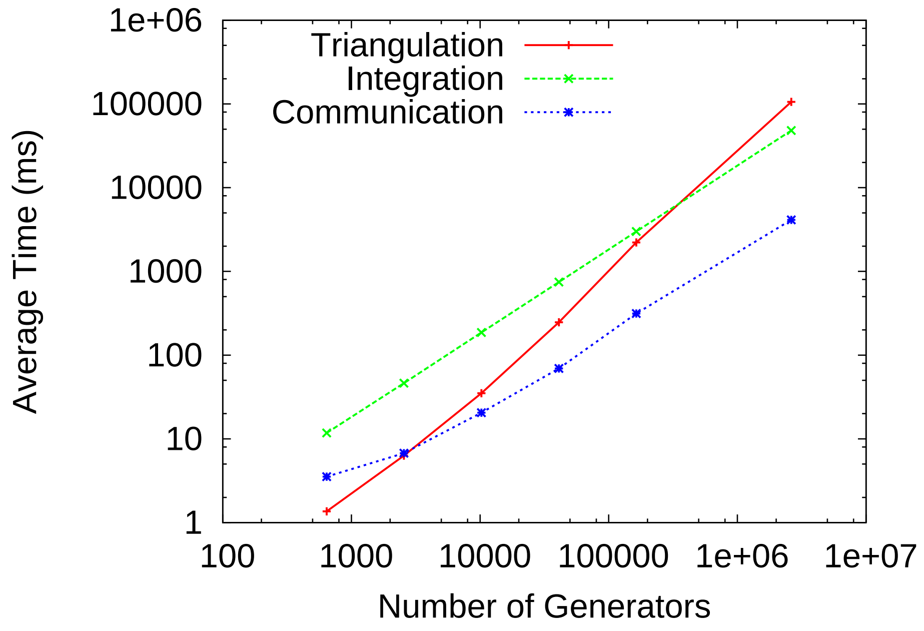


Fig. 9. Timings for various portions of MPI-SCVT using 2 processors and 2 regions. As the problem size increases the slope for both triangulation (red-solid) and integration (green-dashed) remain roughly constant. The triangulation does not become more expensive than the integration until after roughly 2 621 442 generators.

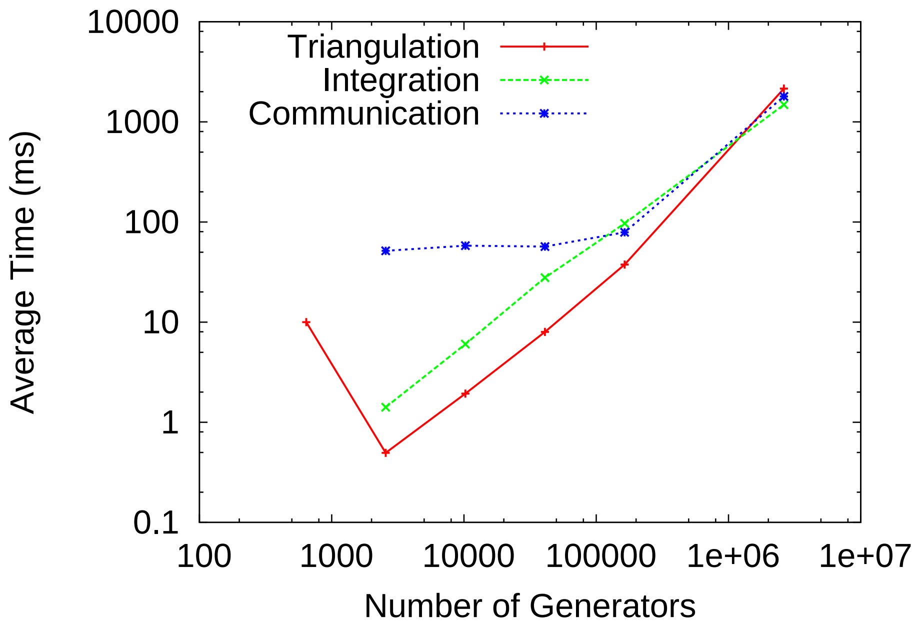
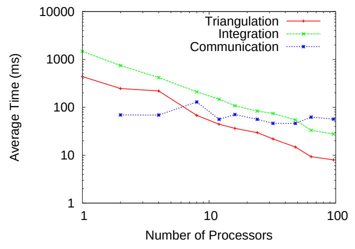
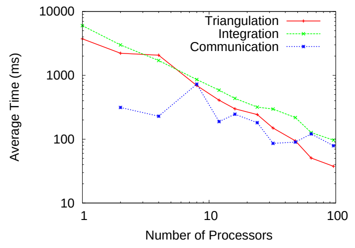


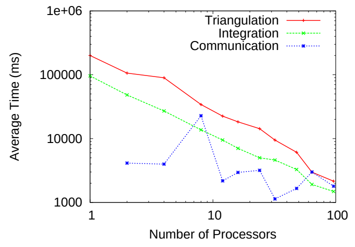
Fig. 10. Same information as in Fig. 9 but for 96 processors and 96 regions.



(a) 40,962 generators

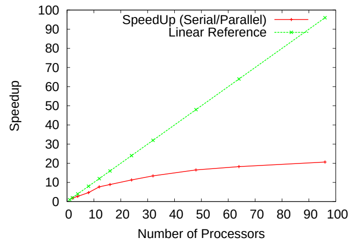


(b) 163,842 generators

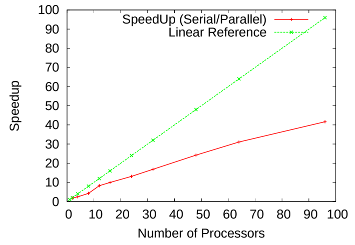


(c) 2,621,442 generators

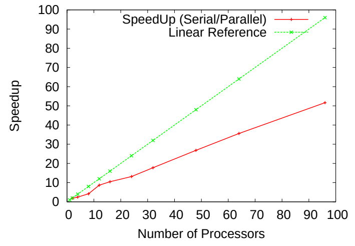
Fig. 11. Timing results for MPI-SCVT vs. number of processors for three different problem sizes. Red solid-lines represent the cost of computing a triangulation, whereas green-dashed lines represent the cost of integrating all Voronoi cells, and blue-dotted lines represent the cost of communicating each region's updated point set to its neighbours.



(a) 40,962 generators



(b) 163,842 generators



(c) 2,621,442 generators

Fig. 12. Scalability results based on number of generators. Green is a linear reference whereas red is the speedup computed using the parallel version of MPI-SCVT compared to a serial version.