Geophys. J. Int. (1995) **122,** 837–857

Geophysical parametrization and interpolation of irregular data using natural neighbours

Malcolm Sambridge,[1,2] Jean Braun[1] and Herbert McQueen[1]

[1] *Research School of Earth Sciences, Australian National University, Canberra,* ACT 0200, *Australia*
[2] *Centre for Information Science Research, Australian National University, Canberra,* ACT 0200, *Australia*

Accepted 1995 February 17. Received 1995 February 15; in original from 1994 November 1

SUMMARY

An approach is presented for interpolating a property of the Earth (for example temperature or seismic velocity) specified at a series of 'reference' points with arbitrary distribution in two or three dimensions. The method makes use of some powerful algorithms from the field of computational geometry to efficiently partition the medium into 'Delaunay' triangles (in 2-D) or tetrahedra (in 3-D) constructed around the irregularly spaced reference points. The field can then be smoothly interpolated anywhere in the medium using a method known as *natural-neighbour interpolation.* This method has the following useful properties: (1) the original function values are recovered exactly at the reference points; (2) the interpolation is entirely local (every point is only influenced by its natural-neighbour nodes); and (3) the derivatives of the interpolated function are continuous everywhere except at the reference points. In addition, the ability to handle highly irregular distributions of nodes means that large variations in the scale-lengths of the interpolated function can be represented easily. These properties make the procedure ideally suited for 'gridding' of irregularly spaced geophysical data, or as the basis of parametrization in inverse problems such as seismic tomography.

We have extended the theory to produce expressions for the derivatives of the interpolated function. These may be calculated efficiently by modifying an existing algorithm which calculates the interpolated function using only local information. Full details of the theory and numerical algorithms are given. The new theory for function and derivative interpolation has applications to a range of geophysical interpolation and parametrization problems. In addition, it shows much promise when used as the basis of a finite-element procedure for numerical solution of partial differential equations.

Key words: interpolation, natural neighbours, parametrization.

1 INTRODUCTION

Many areas of geophysics rely on methods of parametrization and interpolation in two and three dimensions. Examples include numerical modelling of mantle convection, crustal deformation and associated thermal conduction/advection, seismic tomography, and the interpolation of topographic, gravity, magnetic or other data fields. In the interpolation problem one has a distribution of reference points, or 'nodes', in two or three dimensions at which the value of some scalar quantity (for example seismic velocity, or temperature) is known, and the problem is to represent this variable everywhere using the values at the nodes. Often a 'local' method is preferred, because it is more physically realistic in data interpolation, and also well suited to numerical modelling applications. In this case, the values of the variable at any point depend only on the data in its neighbourhood. Local schemes usually involve a discretization of the region into cells (rectangular, tetrahedral etc.), and so methods of parametrization and interpolation are closely related.

An important property of interpolation methods is the degree of continuity or 'smoothness' they achieve in the interpolated function. Numerical schemes for the solution of partial differential equations (PDE) usually require continuity in first (and sometimes second) derivatives of variables, and some level of smoothness is often preferred when interpolating observational data. When the nodes have a regular distribution, for example on a rectangular grid, many local methods are available for smoothly interpolating

in two (2-D) and three dimensions (3-D) (see, for example, De Boor 1962). Smooth, local methods also exist for highly irregular distributions of nodes when special local constraints exist between the nodal positions, for example when groups of six nodes lie on either the vertices or sides of triangles (Powell & Sabin 1977). Modern applications of the finite-element method often used this type of 'locally constrained' nodal distribution to produce highly irregular parametrizations that concentrate accuracy and detail in the parts of a calculation where they are needed (Zienkiewicz 1977).

When the distribution of nodes is completely arbitrary the problem of smooth and local interpolation (in 2-D and 3-D) is more difficult, and has received less attenuation. Although various methods have been developed (comprehensive reviews appear in Schumaker 1976; Watson 1992) few of these have found applications in geophysical poblems. However, there appears to be considerable potential for local, smooth interpolation schemes that are able to handle arbitrary nodal distributions. One application is in Lagrangian finite-element techniques for the solution of partial differential equations. In this case the nodal distribution cannot be determined for the convenience of the underlying interpolation scheme, but is instead controlled by the calculation itself, and can become highly irregular. Examples include Lagrangian finite-element schemes for modelling crustal deformation (Braun & Beaumont 1987; Fullsack 1995). Another possible application is in the gridding of observed data (for example gravitational or magnetic fields) where the distribution is often determined by practical considerations, which can lead to highly anisotropic data sets with variable density. Existing methods are often based on simple spatial moving averages, for example Kriging (Matheron 1973; Delfiner & Delhomme 1975), which are non-local methods.

There is also much potential for using highly irregular cellular parametrizations to represent Earth properties, such as seismic velocity, where the length-scales of variation can vary considerably in different parts of the Earth. An irregular parametrization would allow large variations in scale-length to be represented by concentrating small cells (e.g. tetrahedra) in the parts of the model that required them. For example, in seismic tomography one might wish to parametrize finely a region around a subduction zone but use a larger cell size in the lower mantle. The main difficulties in using highly irregular parametrizations lie in 'book-keeping' problems, i.e. how to build the parametrization and how to keep track of which cell any given point is in.

In this paper we show how some powerful new methods from the field of computational geometry can be used to solve these problems. The result is a procedure to generate a unique set of triangles (or tetrahedra) from arbitrarily distributed nodes in 2-D or 3-D that provides an ideal basis for local interpolation methods. A theory of local and smooth interpolation is also described. This theory has largely been developed by practitioners in the field of computational geometry (Sibson 1980, 1981; Watson 1992). We have extended the theory to derive expressions for the first derivatives of the interpolated function and an algorithm for their efficient calculation. This has useful applications when the new approach is used as the basis of PDE finite-element solvers. For clarity, most of the mathematical details are confined to the appendices.

At present the interpolation procedure does not appear to be widely used, although the theory of Delaunay tessellations, upon which it is based, is a much studied area of computational geometry and has already found applications in a number of fields. In geophysics, the earliest use of a Delaunay triangulation, that we are aware of, was by Parker, Shure & Hildebrand (1987), who used it to represent the surface of seamounts in an application of inverse theory to seamount magnetism. In that and subsequent uses of the technique (e.g. Hildebrand & Parker 1987) linear interpolation was performed over Delaunay triangles that were derived from an irregular distribution of nodes. More recently, Constable, Parker & Stark (1993) used a Delaunay tessellation to represent the radial component of the geomagnetic field at the core–mantle boundary, by distributing regularly positioned nodes on a sphere. They also performed a linear interpolation of function values (specified at the vertices) after a gnomonic projection of each triangle onto a tangent plane.

In this paper we discuss the more general theory of natural-neighbour interpolation and irregularly distributed nodes in 2-D and 3-D. It is our objective to bring this tessellation and interpolation theory to the attention of the geophysical community, describe the algorithms needed to use it (together with some improvements of our own), and to demonstrate its potential for a range of geophysical problems.

2 VORONOI CELLS AND DELAUNAY TESSELLATIONS

Computational geometry deals with the design and analysis of algorithms for geometric problems. Many areas of the natural sciences, mathematics, engineering and computer science give rise to problems that are inherently geometrical. Applications of geometric algorithms and theory are rapidly growing (for survey papers see O'Rouke 1988; Dobkin 1988), although to date geophysicists have not made use of the advances in this expanding field.

Over the past decade, fundamental geometric constructs known as *Voronoi diagrams* and *Delaunay tessellations* have received a large amount of attention. (A 'tessellation' is a covering of a surface with tiles; here we use the term to represent the filling of space with triangles in 2-D, or tetrahedra in 3-D.) In 2-D, the Voronoi diagram of an irregular set of nodes divides the plane into a set of regions, one for each node, such that any point in a particular region is closer to that region's node than to any other node. Fig. 1(a) shows the Voronoi diagram for a set of 16 nodes. Each region, or cell, consists of the part of the plane nearest to that node. The cells are unique, space-filling, and can be defined similarly in any dimension. The Voronoi diagram is regarded as 'one of the most fundamental and important geometrical constructs determined by an irregular set of points' (Avis & Bhattacharya 1983).

The concept of Voronoi diagrams was originally developed by mathematicians (Dirichlet 1850; Voronoi 1908) and subsequently redeveloped in many fields, including meteorology (Thiessen 1911), crystallography (Niggli 1927), metallurgy (Frank & Kasper 1958; Wigner &

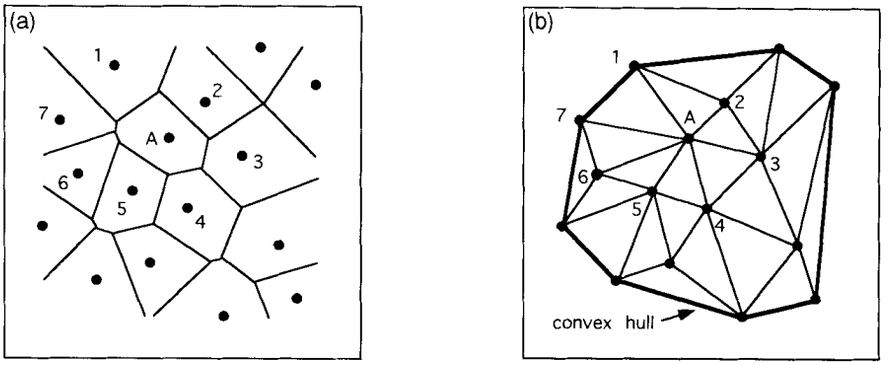


Figure 1. (a) The Voronoi diagram for a set of 16 nodes in a plane. (b) The corresponding Delaunay triangulation. The thick 'perimeter' line connects the nodes in the convex hull.

Seitz 1933), and pattern recognition (Blum 1967). In each case, different names have been used. The other closely related geometric construct which is of interest here is the Delaunay tessellation (or triangulation in 2-D), which was introduced by Voronoi (1908) and extended by Delaunay (1934). The Delaunay triangulation of the 16 nodes in Fig. 1(a) is shown in Fig. 1(b). The Delaunay triangles are formed by simply connecting the nodes whose Voronoi cells have common boundaries. Again these concepts generalize easily to higher dimensions, for example in 3-D we have Delaunay tetrahedra. (In this paper we tend to use terminology appropriate to the 2-D case, although the discussion is equally appropriate for three or more dimensions.) Recently, several comprehensive reviews have appeared on the subject and readers are referred to these for a detailed discussion of the history, theory, properties, methods of construction and applications of these fundamental geometric structures (Aurenhammer 1991; Fortune 1992; Okabe, Boots & Sugihara 1992).

2.1 Useful properties of Delaunay tessellations

The Voronoi cells and Delaunay triangles are said to be 'dual' to one another, and once one is known the other is completely defined. Delaunay triangles are of interest to us because of their useful properties. In a sense they provide the 'best-looking' set of triangles. By this we mean that they are the set of least 'long and thin' triangles that can be generated among the many triangulations that are possible with irregularly distributed points. This is often referred to as the *maximum–minimum angle property* and can be used as the basis for calculating the set of triangles (Fortune 1992).

Another useful property is that the size of the Delaunay triangles is strongly determined by the density of the nodal distribution. This property is derived directly from the Voronoi cells, whose areas can be used as an inverse measure of the nodal density. Therefore the size of Delaunay triangles will vary enormously when the nodal distribution is highly irregular. This property is very useful for parametrizing geophysical models where large variations in length-scale exist. If Delaunay triangles, or Voronoi cells, are used to parametrize a medium (for some numerical calculation, or perhaps in seismic tomography) then extremely large variations in cell size and hence in scale-length can readily be represented in the model. A numerical example of this is presented in Section 4.

The combined properties of the maximum–minimum internal angle and density-dependent size make Delaunay triangles ideal for use as the basis of an interpolation procedure. The simplest interpolation procedure is linear, i.e. when the value of a function at any point in a triangle is found by linear interpolation of the three 'data' values at the vertices (see Section 3.1). In this case the interpolated value is only dependent on the function values at the three vertices, and for Delaunay triangles these are the best three 'nearby surrounding nodes'. This property holds regardless of the complexity of the nodal distribution. If the density and angular distribution of the nodes varies enormously across a data set then the Delaunay triangles will adapt accordingly, and the linear interpolation will always be between three nearby surrounding nodes.

The main limitation of linear interpolation is that it produces discontinuities in the first derivatives of the function across the sides of the triangles and all second derivatives are zero inside each triangle. In many applications, higher order smoothness is required and in the gridding of irregular data smooth contours are often desirable. In these cases a higher order interpolation procedure is required.

2.1.1 Natural neighbours

The idea of a set of 'nearby surrounding nodes' is generalized by the definition of *natural-neighbour* nodes. The natural neighbours of any node are those in the neighbouring Voronoi cells, or equivalently, those to which the node is connected by the sides of Delaunay triangles. For example, in Fig. 1(a), node A has natural neighbours numbered 1 to 7. (For a thorough discussion of natural neighbours see Watson 1985 1992). Although natural neighbours usually refer to the nodes, one can equally well define a set of natural neighbours to any (x, y) point in the plane as that set of nodes which would be connected to the point if it were added to the Delaunay triangulation. Again these definitions generalize in a straightforward way to higher dimensions. The importance of natural neighbours is that they represent a set of 'closest surrounding nodes' whose number and positions are well defined and vary according to the local nodal distribution. (Some theoretical upper bounds on the number of possible neighbours to any node have been derived; see Okabe *et al.* 1992 for details.)

One can think of the natural neighbours about any point as a unique set of nodes that defines the 'neighbourhood' of the point in the plane. If the distance between nodes is relatively large in some places, or the distribution is highly anisotropic, then the set of natural neighbours will reflect these features, but nevertheless still represent the best set of nearby surrounding nodes. They are therefore ideal candidates for the basis of a local interpolation scheme; i.e. we have

$$f(x, y) = \sum_{i=1}^{n} \phi_i(x, y) f_i, \tag{2.1}$$

where $f(x, y)$ is the interpolated function value, f_i $(i = 1, \ldots, n)$ are the data vaues at the n natural-neighbour nodes to the point (x, y) and the ϕ_i $(i = 1, \ldots, n)$ are weights associated with each node. The way in which the weights, ϕ_i, are determined will control the smoothness properties of the interpolation. However, since the summation in (2.1) is only over the natural-neighbour nodes then, regardless of how the ϕ_i are determined, the interpolation is guaranteed to be local. Furthermore, the size and shape of the region that can influence any point will adapt naturally to the local variation in node density. In Section 3 a method known as 'natural-neighbour interpolation' is described which uses the areas of Voronoi cells to determine the weights. This method results in continuous first and second derivatives in the interpolated function at all points except the nodes themselves, and is ideally suited to highly irregular nodal distributions.

2.2 Methods of calculating Delaunay triangulations

During the past decade a number of algorithms have been devised to calculate Delaunay tessellations in 2-D and 3-D. Early 2-D algorithms, suitable for computer programming are ones by Lawson (1977), and Green & Sibson (1978). A comprehensive list can be found in the review articles cited above. There are several types of algorithm, each of which may be suitable for particular applications. The 'edge flipping' algorithm (Fortune 1992) starts with an existing non-Delaunay triangulation and uses the maximum–minimum angle property to update iteratively all connections between nodes until they are in a Delaunay configuration. This method can be very useful when an existing Delaunay triangulation in 2-D is known and one wishes to re-calculate the Delaunay connections after perturbing the position of all of the nodes. The previous set of nodal connections usually provides an excellent starting point for the flipping algorithms. This approach was used by Braun & Sambridge (1994) to update a Delaunay mesh at each time-step of a Lagrangian finite-element scheme. This resulted in a new method for solving large-strain rock deformation problems.

Other types of algorithm include 'incremental' methods, which add nodes sequentially and update the Delaunay triangulation after each addition. Often methods are developed specifically for the 2-D case and will not work, or will quickly become inefficient, in higher dimensions. In 2-D, the method generally accepted as the most efficient is the 'sweepline' algorithm of Fortune (1987). For three or more dimensions, the early methods are ones by Watson (1981), Bowyer (1981) and Devijver & Dekesel (1983). Many of the higher dimensional methods use the 'empty circle' property, which says that the circles (spheres in 3-D) passing through the vertices of Delaunay triangles contain no other nodes. In our experience, the most efficient higher dimensional method at present is the 'quickhull' algorithm of Barber, Dobkin & Huhdanpaa (1993). This procedure actually calculates the 'convex hull' of a set of points in any number of dimensions. (The convex hull is the smallest convex set of nodes that enclose all nodes. In 2-D the line joining all nodes in the convex hull forms an 'outer perimeter'. See the thicker line in Fig. 1(b).) Brown (1979) discovered a connection between Delaunay tessellations in dimension d and convex hulls in dimension $d + 1$, which means that any multidimensional convex-hull algorithm can be used to calculate a Delaunay tessellation. The quickhull algorithm uses this approach to determine the Delaunay tessellation in any dimension.

The data structure used to specify the Delaunay tessellation is a triplet of nodal indices for each triangle. In this paper we calculate this list with the quickhull algorithm, which we found to be comparable in speed, in 2-D, with Fortune's sweepline method. Since it is mainly the properties of the Delaunay triangulation that are of interest here, we do not describe the 'quickhull' algorithm in any detail. Barber *et al.* (1992) provide full details, and in any case an efficient public domain computer code is available to implement it (Barber & Huhdanpaa 1994).

3 INTERPOLATION METHODS BASED ON DELAUNAY TRIANGULATIONS

3.1 Linear interpolation

If linear interpolation is sufficient for an application then the Delaunay triangles provide an ideal basis. Given the values of some observable $(f_i, i = 1, \ldots, 3)$ at the nodes of a triangle $(x_i, y_i;\ i = 1, \ldots, 3)$, the interpolated value at any point (x, y) interior to the triangle is given by

$$f(x, y) = \sum_{i=1}^{3} \phi_i(x, y) f_i, \tag{3.1}$$

where $\phi_i(x, y)$ is a 2-D basis function which varies linearly from a value of 1 at the node (x_i, y_i) to zero at nodes (x_j, y_j) $(j \neq i)$. In practice, it is convenient to replace eq. (3.1) with the equivalent expression

$$f(x, y) = c_1 x + c_2 y + c_3, \tag{3.2}$$

where the coefficients, $\mathbf{c} = (c_1, c_2, c_3)^{\mathrm{T}}$, are the solution of the linear system

$$\mathbf{Ac} = \mathbf{f}, \tag{3.3}$$

with $\mathbf{f} = (f_1, f_2, f_3)^{\mathrm{T}}$ and where $\mathbf{A}$ is a matrix whose ith row is $(x_i, y_i, 1)$. In 3-D there are four vertices to each tetrahedron and so the number of coefficients in eq. (3.1) increases to four. Similarly, the linear system in eq. (3.3) increases to 4×4.

This linear system is trivial to solve, and so the value of $f(x, y)$ at any point in a triangle is easily found. The main 'book-keeping problem' that occurs is to know which triangle (or tetrahedron) contains a given point. As the

number of nodes increases, it can become very inefficient to search through all triangles systematically. Fortunately, a very efficient method, which we will refer to as the 'walking triangle algorithm' (Lawson 1977; see also Lee & Schachter 1980; Sloan 1987), has been developed for this problem. Details of the algorithm and our extension to 3-D are given in Appendix C. With this method it is possible to find quickly the triangle containing any given point, regardless of the overall complexity of the Delaunay triangles. Therefore linear interpolation may be performed efficiently.

3.2 Natural-neighbour interpolation

In natural-neighbour interpolation the weights, ϕ_i, in eq. (2.1) are taken as the *natural-neighbour* (*n-n*) *coordinates* of the point (x, y) in the plane. Natural-neighbour coordinates were introduced by Sibson (1980, 1981) and may be defined in any number of dimensions. They have a straightforward geometric definition which is most easily explained in 2-D. Consider the five nodes in Fig. 2(a) and the sides of the Voronoi cells. If one adds a point X in cell 3, then a new Voronoi cell can be placed around it (the shaded region in Fig. 2b). The Voronoi cell about X overlaps all of the original cells of its natural neighbours. The n-n coordinate of X with respect to a neighbour is defined as the ratio of the area of their overlapping Voronoi cells to the total area of the Voronoi cell about X. For example, in Fig. 2(b) the n-n coordinate of X with respect to node 3 is the ratio of the areas of the polygon *afghe* to the polygon *abcde*. The five 'overlapping regions' in Fig. 2(b) are known as *second-order* Voronoi cells. The definition of a second-order Voronoi cell between nodes i and j is the region within which all points are closest to node i and second-closest to node j. In Fig. 2(b) the n-n coordinates of X with respect to its natural neighbours are the normalized areas of the five second-order Voronoi cells of X and nodes 1 to 5.

Since n-n coordinates are always normalized areas (or volumes in 3-D), we immediately have $0 \leq \phi_i(x, y) \leq 1$ and $\sum_i \phi_i = 1$. Clearly, if X were placed on top of any node, say node 3, then the boundaries of the (shaded) Voronoi cell would become identical to that of the original Voronoi cell and so there would be total overlap with cell 3 and zero overlap with all other nodes. From this it follows directly that

$$\begin{aligned} \phi_i(\mathbf{x}_j) &= 1, \quad \text{if} \quad i = j, \\ &= 0, \quad \text{if} \quad i \neq j, \end{aligned} \tag{3.4}$$

where the vector $\mathbf{x}_j$ is the position of node j. Eq. (3.4) also holds in 3-D, and means that the interpolated value $f(\mathbf{x})$ at any point, $\mathbf{x}$, has the useful property of being exactly equal to the original function value at the nodes, i.e. we have

$$f(\mathbf{x}_i) = f_i. \tag{3.5}$$

The second important property of natural-neighbour interpolation is that it is a local procedure. One can think of the n-n coordinate, $\phi_i(\mathbf{x})$, as a function describing how node i influences the region around it. It turns out that $\phi_i(\mathbf{x})$ is only non-zero in the union of the n circles that pass through the vertices of the n Delaunay triangles about node i, where n is the number of natural neighbours of node i. Fig. 3(a) shows a mesh of Delaunay triangles with the union of circles about node i shaded. Each circle passes through node i and two of its natural neighbours, and so the boundary is composed of n circular arcs. (In 3-D the circles become spheres and the triangles tetrahedra.) It can be seen that the size and shape of the influence region is a function of the position of node i and its neighbours. Fig. 3(b) shows a plot of $\phi_i(\mathbf{x})$ as an illuminated surface, seen from the direction of the arrow in Fig. 3(a). One can see that $\phi_i(\mathbf{x})$ smoothly decreases from a value of 1 at node i to 0 at the boundary, regardless of the distribution of the neighbours. It can be shown that the gradient of the surface goes to zero at the boundary (see Appendix A).

The third important property of natural-neighbour coordinates is that they are continuously differentiable at all points except at their defining node, i (Sibson 1980). This means that the interpolated function, $f(\mathbf{x})$, given by (2.1) is also *continuously differentiable everywhere* except at the nodes. The smoothness in the interpolated function is a direct result of the smoothness of the natural-neighbour surface seen in Fig. 3(b). In our experience (see the examples in Section 4) the lack of differentiability at the nodes is rarely a problem in practice, and natural-neighbour interpolation does not have the limitation, seen in linear

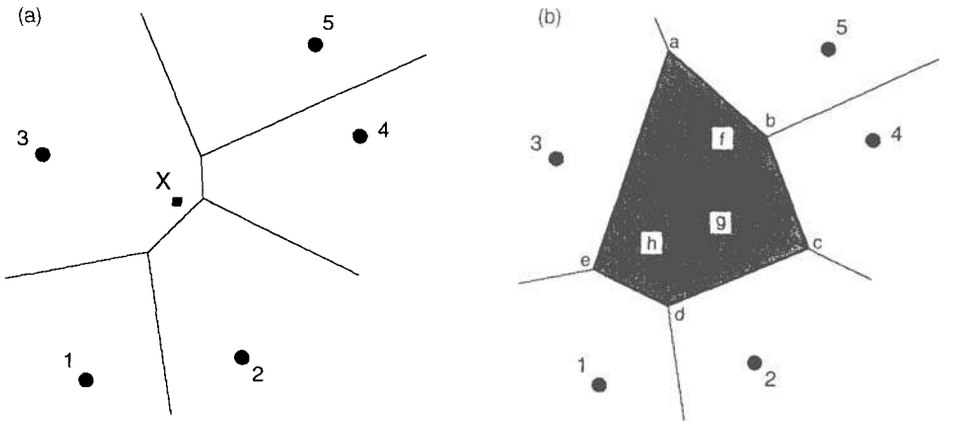


Figure 2. (a) The original Voronoi diagram for five neighbouring nodes and an interpolation point X. (b) The new Voronoi cell about X (shaded). The overlap of the new Voronoi cell with the original cells forms five second-order Voronoi cells between X and its neighoburs.

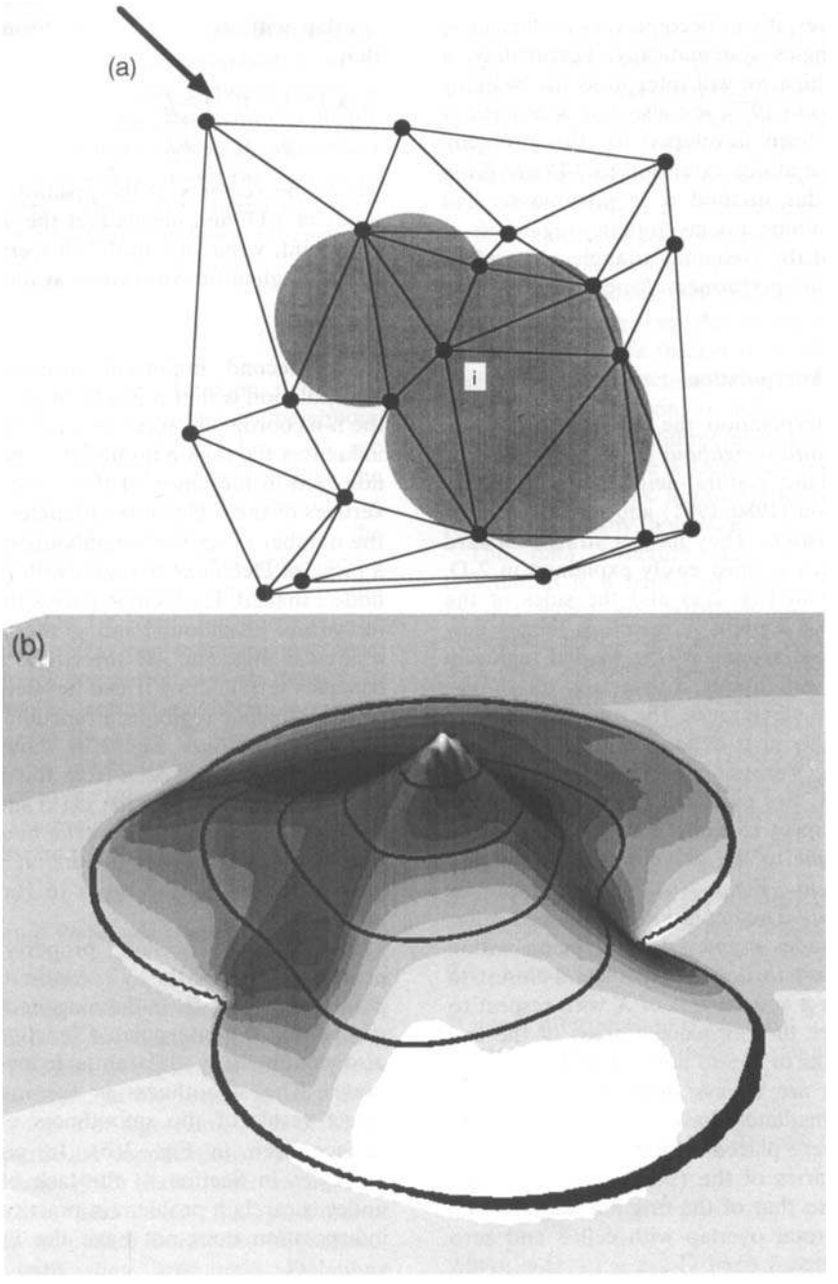


Figure 3. (a) The shaded region about node *i* shows the area that it can influence in natural-neighbour interpolation. (b) A perspective view of the influence surface about node *i* seen from the direction of the arrow in (a). The height of the surface at any point is the value of its natural-neighbour coordinate with respect to node *i*.

interpolation, of discontinuous derivatives across cell boundaries.

In summary, natural-neighbour interpolation results in an interpolant which fits the origin data at the nodes exactly, is local, and guarantees continuity in first and second derivatives everywhere except at the nodes. Furthermore, the size and shape of the local region vary according to the local nodal density distribution, by virtue of its definition using natural neighbours. These properties hold in any dimension. An efficient procedure for calculating the n-n coordinates in 2-D was given by Watson (1992). In Appendix A we expand on Watson's description, and show how it may be extended to give first derivatives of the interpolated function. We also discuss the extension of the procedure to 3-D. In the next section we examine the performance of natural-neighbour interpolation in the gridding of a highly irregularly distributed topographic data set.

4 APPLICATIONS OF THE DELAUNAY MESH AND NATURAL-NEIGHBOUR INTERPOLATION

4.1 Building multiple-scale seismic models with a Delaunay mesh

In many geophysical studies there is a need to build 2- and 3-D parametrizations of the Earth. For example, in seismic

tomography the region of interest is often divided into cubic cells (e.g. Aki, Christoffersson & Husebye 1977). In nearly all cases, the cell sizes are constant throughout the entire model, or vary in only one dimension (e.g. depth). With a local interpolation procedure the size of the cell determines the smallest scale-lengths allowed, and a regular parametrization forces these to be the same everywhere in the model. In many problems it may be useful to concentrate detail in a particular part of a model without introducing a dense parameterization everywhere, for example in teleseismic traveltime tomography where the data often only provide useful constraints within a limited region beneath the receivers, or in mantle convection simulations where large- and small-scale flows interact with each other. The complete freedom allowed by the Delaunay tessellation to build 'well shaped' triangles, or tetrahedra, from arbitrary nodal distributions in 2-D and 3-D makes them ideal for the basis of a parametrization. With this approach it is possible to build 2-D and 3-D models with an enormous variation in cell sizes and complex geometry. Fine detail may be imposed anywhere by simply adding more nodes locally and letting the Delaunay tessellation produce the cells. The resulting 'book-keeping' problem (i.e. how to find the triangle containing a given point) is solved by the walking triangle algorithm (Appendix C), and so a highly flexible method of parametrization is produced.

We now present two examples to demonstrate the power of the technique. These are somewhat artificial in that we have combined subsets of 3-D models, resulting from different tomographic studies over different length-scales, in order to highlight the advantages of the technique. This was necessary because all current 3-D 'tomographic' earth models are produced using regular grids, or long-wavelength spherical harmonic parametrizations and therefore provide a relatively narrow range of length-scales compared to what can be achieved with the new approach. We use subsets of these models to increase the variability in data density and illustrate the power of the new approach to handle highly irregular nodal distributions.

4.1.1 Combining global- and regional-scale models on a sphere

In the first example we use the Delaunay tessellation to combine features of two complex models on the surface of a sphere. The two models we use are a global *a priori* Love-wave velocity model of Ricard, Nataf & Montagner (1995) and a 3-D *P*-wave model of the north-west Pacific produced by seismic tomography (van der Hilst, Engdahl & Spakman 1993). The laterally heterogeneous global model is intended to be representative of a period of ≈73 s and a depth of ≈100 km. It consists of velocity perturbations sampled on a regular $2° \times 2°$ grid. The regional 3-D subduction-zone model of the north-west Pacific consists of *P*-velocity variations on a lateral $1° \times 1°$ grid and layers with depth separations of between 35 and 175 km. The two models are useful because they contain structural features of different shapes and scales.

To demonstrate the power of the tessellation we take an irregularly distributed subset of nodes from each model and combine them on the surface of a sphere. In the global model we choose 1932 nodes from the total of 16 200. The selection of these was biased towards nodes with negative velocity perturbations, which forces many of the nodes to lie in slow regions like mid-ocean ridges, and the overall distribution becomes representative of these large-scale global features. In contrast, the subset of nodes selected from the regional model were biased towards fast velocity perturbations in a horizontal layer at 135 km depth. This forces the distribution of nodes to take on the character of a lateral section through the imaged subduction zone. By combining the two sets of nodes we obtain a highly irregular distribution of nodes which is concentrated about different regional- and global-scale structural features.

Figure 4(a) shows the triangulation of the combined set of 2780 nodes on a sphere. Since the nodes actually lie on a curved surface in 3-D (and not on a plane) the Delaunay tessellation consists of tetrahedra rather than triangles. The triangles in Fig. 4(a) are in fact the exterior faces of the Delaunay tetrahedra as seen from the outside (which is equivalent to the convex hull of the surface nodes). The internal connections between nodes are ignored in this case, although one could just as easily add nodes throughout the volume of the sphere and build a complete 3-D parametrization. (Note that if the internal nodes were all within the convex hull then the exterior faces would be unchanged.) In Fig. 4(a) it can easily be seen how the size of the triangles varies with the local nodal density and yet there are very few highly distorted (long and thin) triangles. The global structure of the mid-ocean ridges is clearly visible, and the dense set of triangles in the north-west Pacific region is too small to be distinguished at this scale.

Figure 4(b) shows an enlarged view of the north-west Pacific region. (The grey border is the same in the two diagrams.) In this figure we see more clearly how the regional model with shorter-scale features has been knitted into the global model. The lateral expression on the subduction zone beneath the Kurile and Izu Bonin trenches is represented by a dense set of triangles. In this case the size of the triangles varies by two orders of magnitude across the figure, but we could just as easily add much denser nodal distributions within the current mesh, for example down to separations of the order of kilometres or metres, and calculate the corresponding Delaunay triangles without difficulty. Similarly, it is possible to add extra nodes to the interior of the sphere to form a complex 3-D parametrization. (In this case the diagram would be unchanged because the new tetrahedra would have the same exterior faces.)

4.1.2 From subduction zones to spherical harmonic models of the Mantle

A second example, which illustrates the power of the parametrization for complex structures, is shown in Figs 5(a) and (b). Here we have combined a great-circle cross-section through a global spherical harmonic *P*-wave model of the lower mantle (i.e. model L02.56 of Dziewonski 1984) with the same profile through the north-west Pacific model used above. The global spherical harmonic model contains very large-scale features, and nodes (shown as black in Fig. 5a) have been selected at intervals of ≈300 km along 11 contours of velocity perturbation between ±1.0 per cent. The subset of nodes from the subduction-zone model has been chosen randomly from those nodes with positive

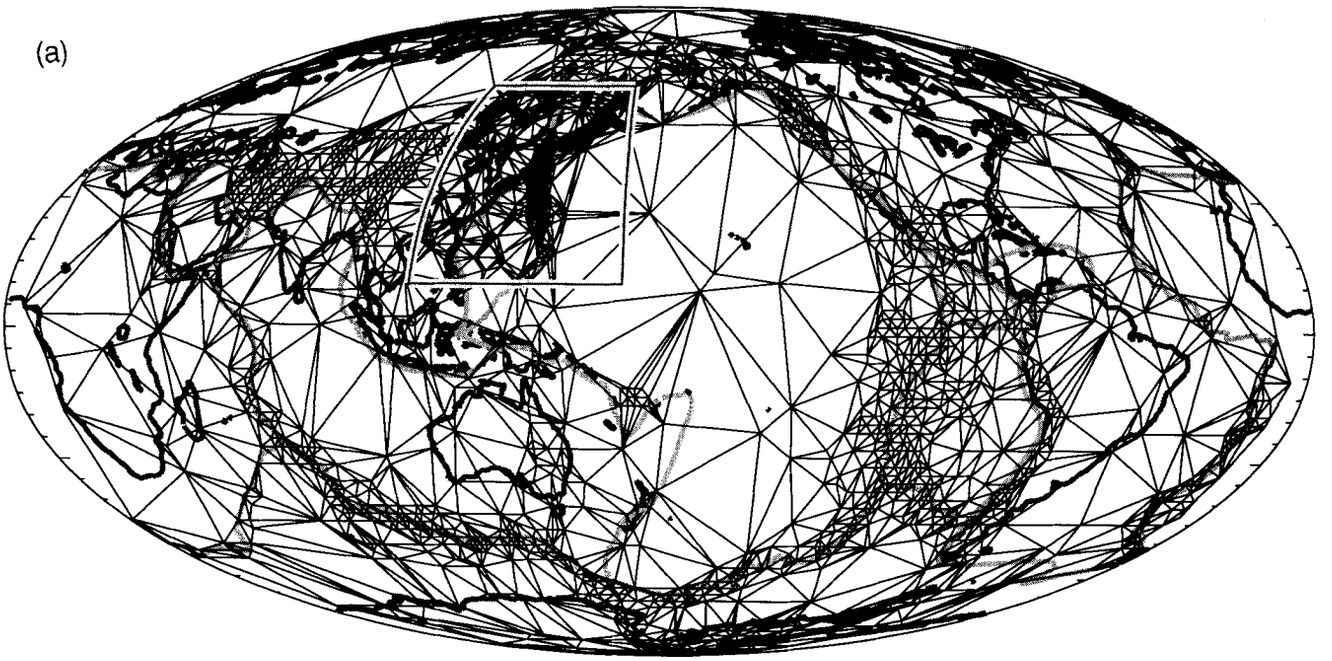


Figure 4. (a) A triangulation of a subset of nodes from the *a priori* Love-wave model of Ricard *et al.* (1994) with a subset of nodes from the model of van der Hilst *et al.* (1993) embedded in the north-west Pacific region. The distance between nodes varies by nearly two orders of magnitude but the tessellation procedure is able to produce 'well shaped' triangles. (b) An enlargement of the north-west Pacific region showing the triangles in the subduction-zone model.

anomalies down to 660 km depth. The two sets of nodes have been combined with a regular grid in the upper mantle (white nodes) with 4° angular spacing. In total, 1471 nodes are used and 2849 Delaunay triangles are produced. In this case we see how the Delaunay triangles adapt from the long-wavelength global-scale features in the spherical harmonic model to the much smaller-scale features in the subduction zone. Fig. 5(b) shows an enlarged view of the subduction zone mesh. As in the previous example, the size of the triangles varies by two orders of magnitude, and one can see how very complex structures with large variations in scale can easily be incorporated in this type of mesh.

The two examples above demonstrate the large degree of flexibility possible with the Delaunay parametrization. However, they by no means extend the technique to its full potential. One could just as easily embed smaller-scale models within the variable mesh. Even with single-precision computation, it would be just as easy to build 3-D earth models with nodal separations varying from 1 m to the radius of the Earth. This flexibility can be very useful in geophysical inversion problems like seismic tomography, where one might wish to use a variable density of nodes to reflect the varying density of ray paths, or perhaps make the positions of the nodes variable in the inversion. It seems likely that Delaunay parametrizations together with constant cell, linear, or natural-neighbour interpolation will find applications in many geophysical problems of this kind.

4.2 Application to the gridding of irregularly distributed topographic data

Most geophysical field data (topography, gravity, magnetic anomalies, etc.) are collected at irregularly spaced locations, whereas visualization tools and processing software commonly assume that geophysical observables are known at the corners of a regular grid. A necessary step in much geophysical data analysis is therefore to interpolate a field, known at a series of irregularly positioned points, onto a regular mesh. Here we demonstrate the usefulness of the n-n interpolation method for interpolating highly irregularly distributed data.

The data set we use consists of measurements of topography at 13 932 irregularly spaced 'reference points' in south-eastern Australia (Fig. 6a). All points lie within the boxed region in Fig. 6(a), and their geographical distribution is shown in Fig. 6(d). Notice that the reference points are characterized by large local variations in density. Most areas are covered by approximately 10 points per degree, but the Melbourne region, for instance, (Fig. 6f) has a much denser data coverage at 1 point every 2 km. In some places the data distribution is highly anisotropic, e.g. where the topography has been measured along roads, or ship tracks. The 27 838 Delaunay triangles are shown in Fig. 6(e) (which took only 6.9 CPU seconds to generate on a Sun Sparc 10/51 workstation). A large variation in the size of triangles produced can clearly be seen. The areas of high data density are covered by triangles too small to distinguish at this scale, while the rural areas have a nearly regular pattern of large triangles. The variability of the nodal density is also reflected in the number of neighbours about each node, which varies between 3 and 83 and has an average value of 12. Note how the distribution of the triangles adapts to incorporate the linear 'road' data embedded in the more diffuse regional data.

We have interpolated the entire topographic data set on a 289×193 point rectangular mesh using the n-n interpola-

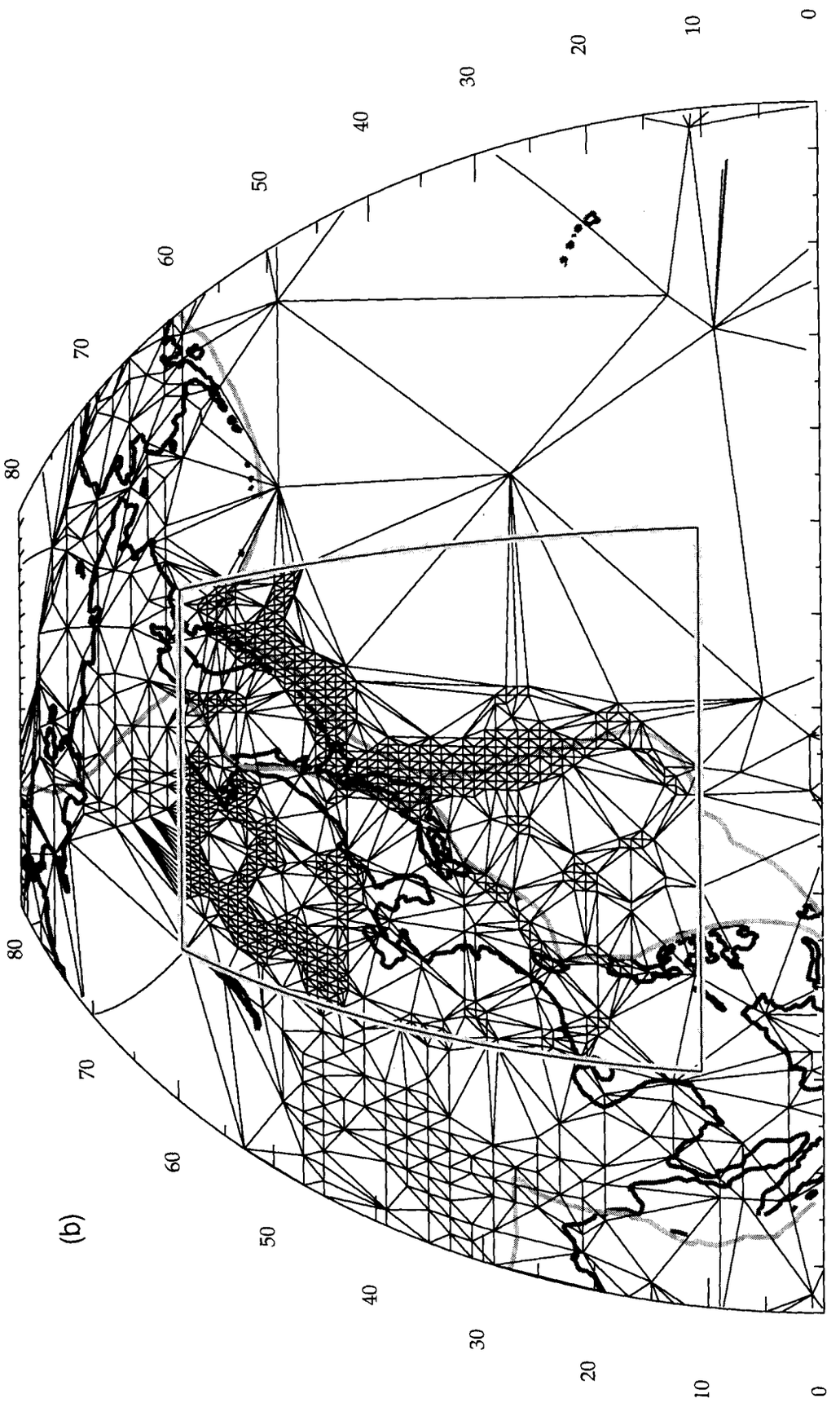
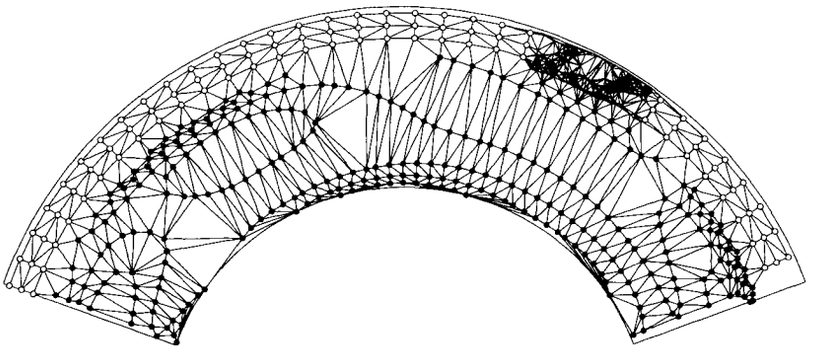


Figure 4. (*Continued.*)

(a) Spherical harmonic (L02.56) + subduction zone models (NWP91)



(b) Spherical harmonic (L02.56) + subduction zone models (NWP91)

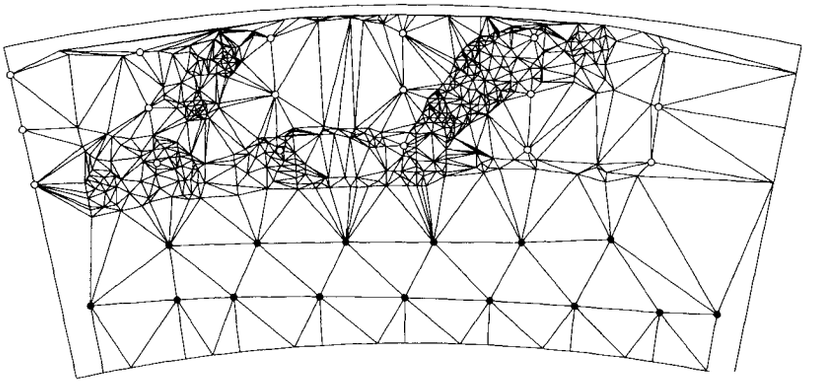


Figure 5. (a) Whole-mantle cross-section of a Delaunay triangulation. The black nodes are produced by contouring the 3-D spherical harmonic lower-mantle model (L02.56) of Dziewonski (1984). A subset of the nodes from the subduction zone model of van der Hilst *et al.* (1993) has been embedded in a regular set of upper-mantle nodes (white). (b) An enlargement of the cross-section through the subduction zone showing the smallest triangles. With the Delaunay tessellation a large range of scale-lengths can be produced in the parametrization.

tion method (45.8 CPU seconds on a Sparc 10/51 workstation). The result is shown in Fig. 6(b) as a contour map with artificial illumination from the north-east. The most interesting result is that, despite the large variations in data density over the area, the n-n interpolation yields a very smooth surface. All contours are characterized by the same curvature, even in areas of very high or very low data density. Comparison of Figs 6(b) and (d) shows that the large gradients in data density have not biased the reconstruction. For example, the area of high data density in the north-west and the co-linear ship track data in the south-east both lie in regions of small or zero gradients in topography, and in the reconstruction there is no noticeable topography produced in those areas. Similarly, in the central highland regions the topographic features in Fig. 6(b) do not appear to be distorted by the trends in the data distribution (Fig. 6d). It is, of course, true that the quality of the reconstructed topography field will be limited by the information content of the data—one cannot reproduce features that are not sampled—but the important point is that the high irregularity of the data sampling has not contaminated the reconstruction in any other way. The n-n interpolation allows us to look at the information contained in the data while minimizing the bias imposed by the irregular distribution of the data.

Another important feature is the ability of the reconstructed surface to respond to data on multiple scales. We have already commented on the large variation in size of Delaunay triangles. A close-up of the data distribution and the Delaunay triangulation for the Melbourne area (boxed in Fig. 6b) is shown in Figs 6(f) and (g) respectively. We have applied the n-n interpolation to the same data, to produce a very fine-scale 241×121 rectangular grid in this region. The result is shown in Fig. 6(c) (which took 11.0 CPU seconds on a Sparc 10/51 workstation). The contours are also equally smooth in all parts of the gridded area. In this case, however, the smaller-scale features present in the data have been reproduced because of the local nature of the interpolation procedure. The interesting point here is that, since the n-n method produces a single well-defined interpolation surface, the two reconstructions in Figs 6(b) and (c) are, essentially, images of the same surface at different scales. That is, if we gridded the entire region in Fig. 6(b) at the fine scale-length of Fig. 6(c), and then zoomed in to the smaller region, we would obtain exactly the same surface as in Fig. 6(c). Therefore the n-n surface contains structural features at all length-scales that are present in the data, and there is no artificial smoothing length imposed by the interpolation method. Furthermore, the minimum scale-lengths produced in the reconstruction are solely determined by the separation of the data points.

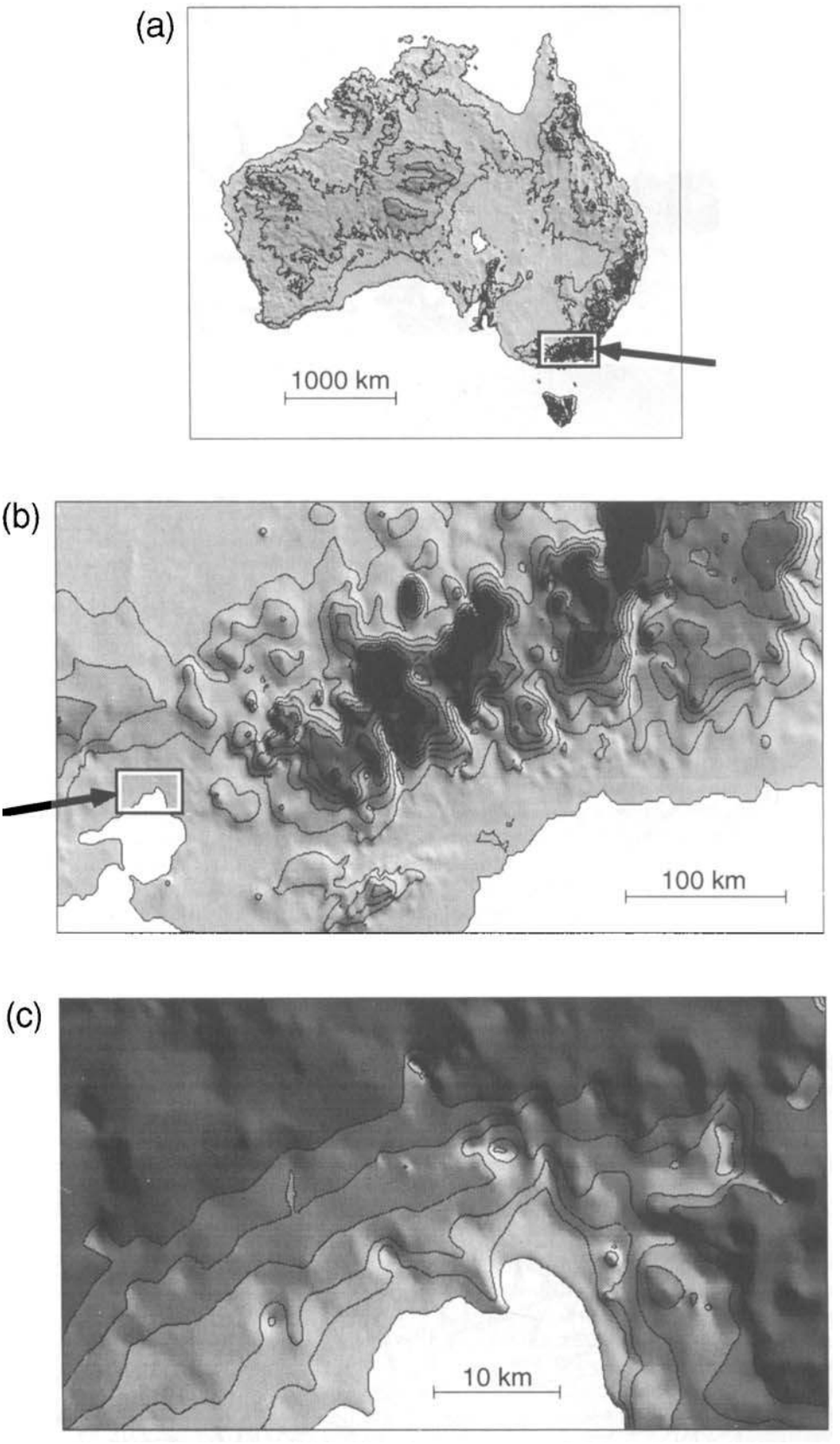


Figure 6. (a) The region covered by the topographic data set (boxed) relative to Australia. (b) The n-n interpolation of topography onto a 289 × 193 regular grid with artificial illumination from the north-east. (c) The fine-scale n-n interpolation of the boxed region in (b) onto a 241 × 121 regular grid. (d) The 13 932 data points used to produce (b). (e) The Delaunay triangulation of the nodes in (d). (f) The data points within the 'fine-scale' region of (c). (g) An enlargement of the Delaunay triangulation in the 'fine-scale' region.

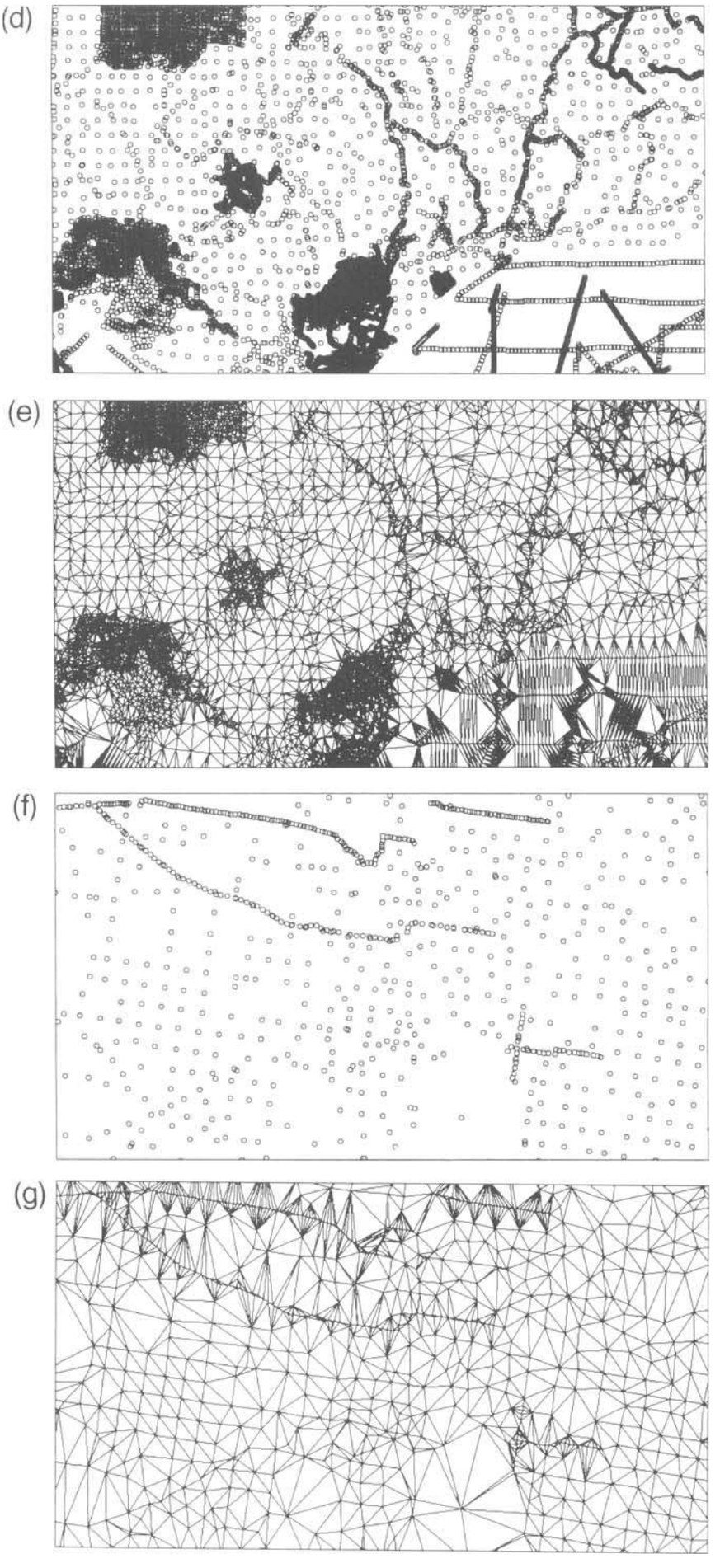


Figure 6. (*Continued.*)

4.3 Other applications of n-n interpolation: solving partial differential equations

Another area where n-n interpolation has applications is in the numerical solution of partial differential equations (PDEs). Finite-element methods (FEM) approximate the solution, $f(\mathbf{x})$, of PDEs at a series of nodes, within a 2-D or 3-D domain. Two important features are the choice of the spatial distribution of the nodes, and the choice of interpolation method used to represent the variation of $f(\mathbf{x})$ between the nodes. Delaunay triangulation and natural-neighbour interpolation provide an excellent basis for these, and offer several advantages over previous methods in cases where the nodal distribution cannot be chosen to suit the interpolation method, for example when a Lagrangian method is used to solve the PDE, or when the mesh adapts during the calculation to improve accuracy.

The essential part of all finite-element methods is the numerical integration of a functional, **A**, defined in terms of $f(\mathbf{x})$ and its spatial derivatives, over the domain of the problem. This integral is performed by dividing the region into elements and evaluating the integrand at a series of 'strategically chosen' integration points. In classical FEM, polynomial functions of the solution values at the nodes are used to evaluate **A** at the integration points within each element. In this case the number of nodes at the boundaries of each element controls the order of the interpolation procedure, and hence the type of PDE that can be solved. If Delaunay triangles (or tetrahedra) are used together with natural-neighbour interpolation then the value of the integrand is influenced by all natural neighbours and not just those at the boundries of the element (see Fig. 2). This produces an interpolant that is smoothly varying and continuously differentiable, and it can therefore be used to solve high-order PDEs (for example the Navier–Stokes equation) even when the nodes are irregularly distributed. In a Lagrangian formulation of a PDE, the mesh 'follows', or evolves with, the solution, and the distribution of nodes can quickly become highly irregular. In this case Braun & Sambridge (1995) have shown that the natural-neighbour interpolant leads to a powerful new type of weighted residual method for solving PDEs, which they call the *Natural Element Method* (NEM).

In the NEM, numerical problems arising from the irregular distribution of nodes are easily dealt with, provided that enough nodes are defined in regions of high gradients in the solution. Similarly, highly distorted meshes may still be used to yield accurate estimates of the solution, $f(\mathbf{x})$, because it is the density of nodes that determines the 'quality' of the n-n interpolation, and not the shape of the Delaunay triangles. Also, the discontinuity that occurs in the gradient of the interpolant at the nodes is of no consequence because we need only evaluate the functional **A** at the integration points within each triangle. It is important to note that the NEM is only possible because we have derived analytical expressions for the spatial derivatives of the interpolant in terms of the function values $f(\mathbf{x})$ at the nodes (Appendix A). [For a detailed description of the NEM and examples of the different types of PDEs, the reader is referred to Braun & Sambridge (1995).]

Table 1. CPU time for tessellation and gridding.

Number of nodes	preprocessing	gridding
10^2	0.97	0.05
10^3	1.33	0.06
10^4	5.63	0.08
10^5	55.79	0.14

CPU time (in seconds) for preprocessing and gridding of n randomly generated data points onto an 11×11 regular grid using a Sun Sparc 10/51 workstation. Preprocessing consisted of calculating the Delaunay tessellation, finding the centres of each circumcircle, and formulating the adjacency matrix (using the algorithm in Appendix C). Note: the interpolation times increase in proportion to $\log n$.

5 DISCUSSION

We have shown that the use of new algorithms from computational geometry has applications in several areas of geophysics. These algorithms provide a method for rapidly discretizing a 2-D or 3-D medium into irregularly sized triangles or tetrahedra, which provides a highly flexible parametrization and an ideal basis for linear, or smooth, local interpolation. The parametrization allows one to build models with an arbitrarily large range of length-scales by simply adding nodes at any point in the model and using the Delaunay triangulation to define the 'cells'. The triangle, or tetrahedron, containing any point can be found efficiently with the walking triangle algorithm, which is explained in full together with our extension to 3-D. An important feature of this and all of the other algorithms described here is that they completely avoid global searches across the entire data set. As a consequence, the computation time increases at most linearly with the number of data nodes, as shown in Table 1. They are, therefore, ideally suited to very large data sets.

The flexibility of the parametrization makes it a powerful tool in any numerical modelling, or inverse problem, where an irregular discretization is advantageous. Several examples have been given to illustrate how the parametrization may be used to build 2-D and 3-D seismic models with structures ranging from the scale of subduction zones to that produced by whole-mantle tomography.

We have discussed the concept and use of natural neighbours as a method for obtaining a smooth interpolant over irregularly shaped and sized Delaunay triangles. Full details of an existing algorithm for natural-neighbour interpolation have been presented together with some improvements of our own. An example is given of its use in the gridding of a topographic data set with a highly irregular distribution. We have drawn together several algorithms from the computational geometry literature and paid particular attention in describing how they work. It is our hope that the level of detail given is complete enough to enable the reader to make use of any of the methods presented. We believe that they represent a powerful new tool, and that they will find applications in many geophysical problems.

ACKNOWLEDGMENTS

We thank Rob van der Hilst and Jean-Paul Montagner for providing us with their seismic models, and David Watson

for many helpful discussions and details of his algorithm for calculating natural-neighbour coordinates. Greg Houseman and Cathy Constable provided reviews which were helpful in improving the paper.

REFERENCES

Aki, K., Christoffersson, A. & Husebye, E.S., 1977. Determination of the three-dimensional structure of the lithosphere, *J. geophys. Res.*, **82,** 277–296.

Aurenhammer, F., 1991. Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure, *ACM Computing Surveys*, **23,** 345–405.

Avis, D. & Bhattacharya, B.K., 1983. Algorithms for computing *d*-dimensional diagrams and their duals, *Adv. Comput. Res.*, **1,** 159–180.

Barber, B. & Huhdanpaa, H., 1994. Qhull (computer program), *Available via Internet ftp from geom.umn.edu.*, The Geometry Center, 1300 South Second Street, Minneapolis, MN 55454.

Barber, B., Dobkin, D.P. & Huhdanpaa, H., 1993. The Quickhull Algorithm for Convex Hull, *The Geometry centre technical report* **GCG53,** The Geometry Centre, Univ. of Minesota, Minneapolis, MN 55454.

Blum, H., 1967. A transformation for extracting new descriptors of shape, *Proc. Symp. on Models for the Perception of Speech and Visual Form*, pp. 362–380, ed. Whaten-Dunn, W., MIT Press, Cambridge, MA.

Bowyer, A., 1981. Computing Dirichlet tessellations, *Computer J.*, **24,** 162–166.

Braun, J. & Beaumont, C., 1987. Styles of continental rifting: results from dynamic models of lithospheric extension, in *Sedimentary Basins and Basin-forming mechanisms*, eds Beaumont, C. & Tankard, A.J., *Can. Soc. Pet. Geol. Mem.* **12,** 241–258.

Braun, J. & Sambridge, M., 1994. Dynamical Lagrangian Remeshing (DLR): A new algorithm for solving large strain deformation problems and its application to fault-propagation folding, *Earth planet. Sci. Lett.*, **124,** 211–220.

Braun, J. & Sambridge, M., 1995. Solving partial differential equations on highly irregular evolving grids using the Natural Element Method, *Nature*, in press.

Brown, K.Q., 1979. Voronoi diagrams from convex hulls, *Inf. Process. Lett.*, **9,** 223–228.

Constable, C.G., Parker, R.L. & Stark, P.B., 1993. Geomagnetic field models incorporating frozen flux constraints, *Geophys. J. Int.*, **113,** 419–433.

De Boor, C., 1962, Bicubic spline interpolation, *J. Math. Phys.*, **41,** 212–218.

Delaunay, B.N., 1934. Sur la sphere vide. *Bull. Acad. Science USSR: Class Sci. Math.*, **VII,** 793–800.

Delfiner, P. & Delhomme, J.P., 1975. Optimum interpolation by kriging, in *Display and analysis of spatial data*, pp. 96–114, eds Davis, J.C. & McCullagh, M.J., Wiley, London.

Devijver, A. & Dekesel, M., 1983. Computing multidimensional Delaunay tessellations, *Pattern Recogn. Lett.*, **1,** 311–316.

Dirichlet, G.L., 1850. Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen, *J. Rein u. Angew. Math.*, **40,** 209–227.

Dobkin, D.P., 1988. Computational geometry: then and now, in *Theoretical Foundations of Computer Graphics and CAD*, NATO ASI Series, **F40,** pp. 71–109, ed. Earnshaw R.A., Springer-Verlag, Berlin.

Dziewonski, A.M., 1984. Mapping the lower mantle: Determination of lateral heterogeneity in P velocity up to degree and order 6, *J. geophys. Res.*, **89,** 5929–5952.

Fortune, S., 1987. A sweepline algorithm for Voronoi diagrams, *Algorithmica*, **2,** 153–174.

Fortune, S., 1992. Voronoi diagrams and Delaunay triangulations, in *Computing in Euclidean Geometry*, eds Du, D.Z. & Hwang, F., World Scientific, Singapore.

Frank, F.C. & Kasper, J.S., 1958. Complex alloy structures regarded as sphere packings, *Acta Crystallogr.*, **11,** 184–190.

Fullsack, P., 1995. An arbitrary Lagrangian-Eulerian formulation for creeping flows and its applications in tectonic models, *Geophys. J. Int.*, **120,** 1–23.

Green, P.J. & Sibson, R., 1978. Computing Dirichlet tessellations in the plane, *Comput. J.*, **21,** 168–173.

Hildebrand, J.A. & Parker, R.L., 1987. Paleomagnetism of Cretaceous Pacific Seamounts Revisited, *J. geophys. Res.*, **92,** 12 695–12 712.

Lawson, C.L., 1977. Software for C^1 surface interpolation, in *Mathematical Software*, Vol. 3, ed. Rice, J., Academic Press, New York.

Lee, D.T. & Schachter, B.J., 1980. Two algorithms for constructing a Dalaunary triangulation, *Int. J. Comput. Inf. Sci.*, **9,** 219–242.

Matheron, G., 1973. The intrinsic random functions and their applications, *Adv. Appl. Prob.*, **5,** 439–468.

Niggli, R., 1927. Die topologische Strukturanalyse, *Z. Kristallograph*, **65,** 391–415.

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

O'Rouke, J., 1988. Computational geometry, *Annu. Rev. Comput. Sci.*, **3,** 389–411.

Parker, R.L., Shure, L. & Hildebrand, J.A., 1987. The Application of Inverse Theory to Seamount Magnetism, *Rev. Geophys.*, **25,** 17–40.

Powell, M.J.D. & Sabin, M.A., 1977. Piecewise quadratic approximations on triangles, *ACM Trans. Math. Software*, **3,** 316–325.

Ricard, Y., Nataf, H.C. & Montagner, J.P., 1995. The 3-SMAC Model - Confrontation with seismic data, *J. geophys. Res.*, submitted.

Schumaker, L.L., 1976. Fitting surfaces to scattered data, in *Approximation theory vol. 2*, pp. 203–268, ed. Lorentz, G.G., Academic Press, New York.

Sedgewick, R., 1990. *Algorithms in C*, Addison-Wesley, Reading, MA.

Sibson, R., 1980. A vector identity for the Dirichlet tessellation, *Math. Proc. Camb. Phil. Soc.*, **87,** 151–155.

Sibson, R., 1981. A Brief Description of Natural Neighbour Interpolation, in *Interpreting Multivariate Data*, pp. 21–36, ed. Barnet V., Wiley, Chichester.

Sloan, S.W., 1987. A fast algorithm for constructing Delaunay triangulations in the plane, *Adv. Eng. Software*, **9,** 34–55.

Thiessen, A.H., 1911. Precipitation average for large area, *Monthly Weather Rev.*, **39,** 1082–1084.

van der Hilst, R.D., Engdahl, E.R. & Spakman, W., 1993. Tomographic inversion of *P* and *pP* data for aspherical mantle structure below the northwest Pacific region, *Geophys. J. Int.*, **113,** 264–302.

Voronoi, M.G., 1908. Nouvelles applications des parametres continus a la theorie des formes quadratiques, *J. reine Angew. Math.*, **134,** 198–287.

Watson, D.F., 1981. Computing the *n*-Dimensional Delaunay Tesselation with Applications to Voronoi Polytopes, *Comput. J.*, **24,** 167–172.

Watson, D.F., 1985. Natural Neighbour Sorting, *Australian Comput. J.*, **17,** 189–193.

Watson, D.F., 1992. *Contouring: A Guide to the Analysis and Display of Spatial Data*, Pergamon, Oxford.

Wigner, E. & Seitz, F., 1933. On the constitution of metallic sodium, *Phys. Rev.*, **43,** 804–810.

Zienkiewicz, O.C., 1977. *The finite element method*, 3rd edn, McGraw-Hill, Maidenhead.

APPENDIX A: CALCULATING NATURAL-NEIGHBOUR COORDINATES

The n-n coordinate of X with respect to each node, $\phi_i(\mathbf{x})$, is just the area of the second-order Voronoi cell between X and that node, normalized by the total area of the Voronoi cell around X. We recall that the second-order Voronoi cell between X and node i is the region of overlap between the old Voronoi cell around node i and the new Voronoi cell around X, for example the polygon *afghe* in Fig. 2(b). Watson's method (Watson 1992) is essentially a way of calculating the areas of the second-order Voronoi cells by breaking them down into a sum of signed areas of subtriangles. The procedure is best explained with the aid of an example. Fig. A1 shows the five natural-neighbour nodes about X. We can imagine these nodes to be surrounded by a much larger set of nodes that are not neighbours of X, but only the natural-neighbour nodes need be considered.

The procedure begins by finding each Delaunay triangle whose circumcircle contains the point X. (The circumcircle is the circle that passes through the three nodes of a triangle.) We will call these 'circum-triangles of X'. In Fig. A1, the circum-triangles of X are Δ_{123}, Δ_{324} and Δ_{345}. A consequence of the empty circle property (described in Section 2) is that the vertices of the circum-triangles of X are in fact just the natural neighbours of X. As the number

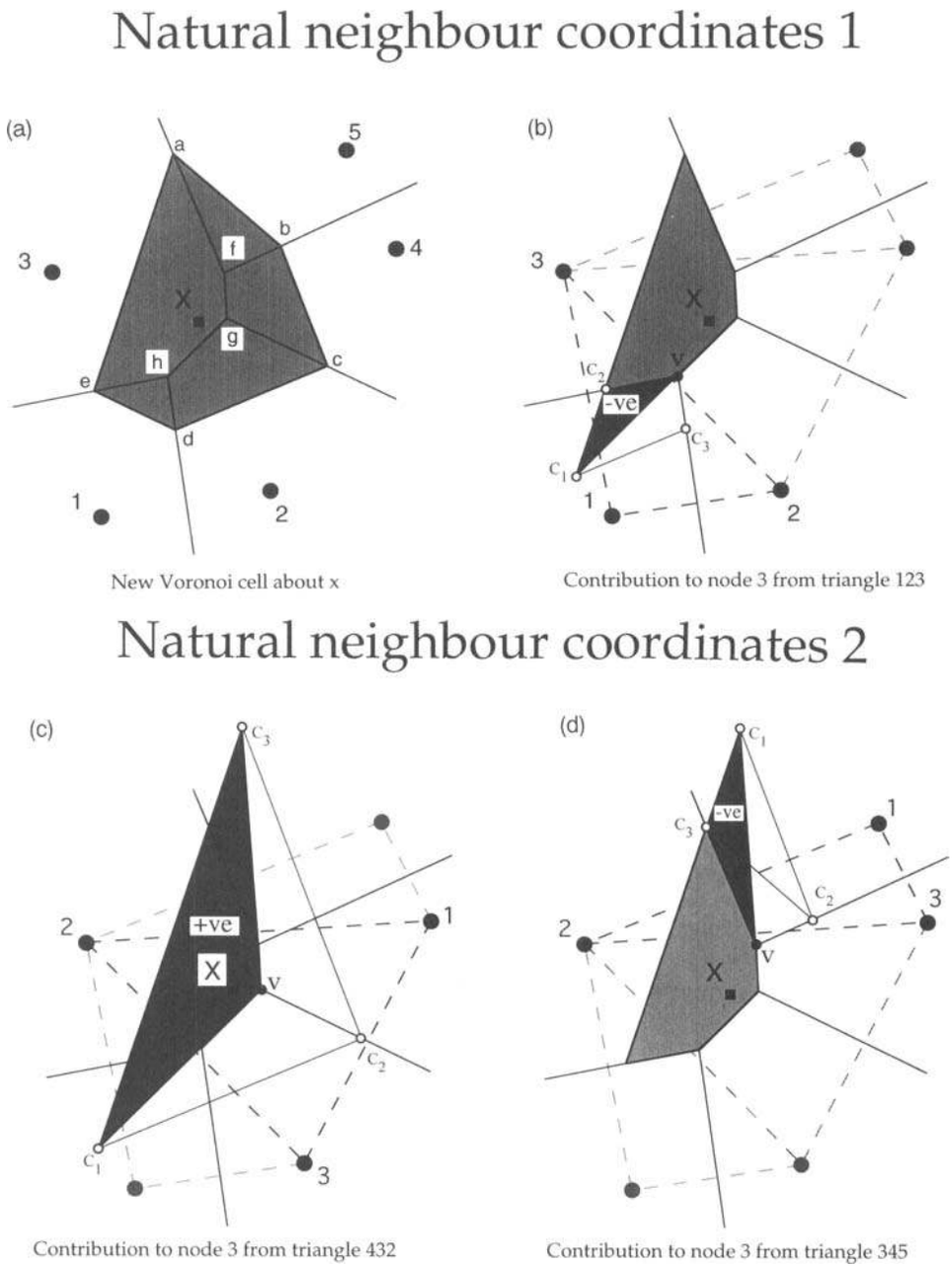


Figure A1. (a) The Voronoi cell about node X (shaded) and its five neighbours. Figs (b), (c) and (d) show how the signed areas of the three sub-triangles (shaded dark) can be added to produce the area of the second-order Voronoi cell between X and node 3. The sub-triangles in (b) and (d) have negative areas while that in (c) is positive. The areas of the other four second-order Voronoi cells are found in a similar way.

of nodes becomes large, it would be expensive to search through all triangles to find the few whose circumcircles contain X, especially since the search would have to be repeated for every new X. Fortunately, the search algorithm of Lawson (1977) can be used to find all circum-triangles. This is a local search method, which remains efficient even for a large number of nodes. Details of the algorithm together with our modifications are given in Appendix B. For each circum-triangle of X we can calculate the vectors $\mathbf{c}_1, \mathbf{c}_2$ and $\mathbf{c}_3$, where

$$\begin{aligned} \mathbf{c}_1 &= \boldsymbol{\Theta}(\mathbf{p}_2, \mathbf{p}_3, \mathbf{x}), \\ \mathbf{c}_2 &= \boldsymbol{\Theta}(\mathbf{p}_3, \mathbf{p}_1, \mathbf{x}), \\ \mathbf{c}_3 &= \boldsymbol{\Theta}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{x}), \end{aligned} \tag{A1}$$

where the function $\boldsymbol{\Theta}(\mathbf{a}, \mathbf{b}, \mathbf{c})$ represents the centre of the circle passing through the three points $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. Each of the points $(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ is the centre of the circle passing through X and a pair of its natural neighbours. Their positions for each of the circum-triangles of X are shown in Figs A1(b), (c) and (d) respectively. It is convenient to write the three expressions in (A1) as one equation using a 'cyclic' rotation of the indices i, j, and k:

$$\boldsymbol{c}_i = \boldsymbol{\Theta}(\mathbf{p}_j, \mathbf{p}_k, \mathbf{x}). \tag{A2}$$

The cyclic rotation simply means that the values of j and k are fixed by i in cyclic rotation, i.e. 123, 231, 312. Each circum-centre $\mathbf{c}_i$ can be found by solving a simple 2×2 linear system,

$$A^{(jk)}\mathbf{c}_i = \mathbf{b}_i, \tag{A3}$$

where

$$A^{(jk)} = [(\mathbf{p}_j - \mathbf{x}), (\mathbf{p}_k - \mathbf{x})]^{\mathrm{T}}$$

and

$$\mathbf{b}_i = \tfrac{1}{2}[(\mathbf{p}_j \cdot \mathbf{p}_j - \mathbf{x} \cdot \mathbf{x}), (\mathbf{p}_k \cdot \mathbf{p}_k - \mathbf{x} \cdot \mathbf{x})]^{\mathrm{T}}. \tag{A4}$$

If we write the circum-centre of the Delaunay triangle Δ_{123} as $\mathbf{v}$, then we have

$$\mathbf{v} = \boldsymbol{\Theta}(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3). \tag{A5}$$

Again, a simple linear system must be solved to find $\mathbf{v}$. [Note that if $\mathbf{x}$ were placed on the circle passing through the nodes at $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ then we would have $\mathbf{c}_i = \mathbf{v}$, $(i = 1, \ldots, 3)$.]

For each circum-triangle we may form three new triangles using the four points $(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{v})$. These have vertices $(\mathbf{c}_2, \mathbf{c}_3, \mathbf{v})$, $(\mathbf{c}_3, \mathbf{c}_1, \mathbf{v})$ and $(\mathbf{c}_1, \mathbf{c}_2, \mathbf{v})$ (see Fig. A1). We will refer to these as the three 'subtriangles of the circum-triangle'. The area of each of these sub-triangles may be written

$$\alpha_{t,i}(\mathbf{x}) = \tfrac{1}{2} \|(\mathbf{c}_j - \mathbf{v}) \times (\mathbf{c}_k - \mathbf{v})\|, \qquad (i = 1, 2, 3) \tag{A6}$$

where $\times$ is the vector product, $\|\cdot\|$ indicates the length of a vector, and t refers to the particular circum-triangle. Again we have used cyclic rotation of coordinates to define j and k. Each sub-triangle can be associated with a vertex of the circum-triangle, i.e. node i is associated with the sub-triangle with vertices $(\mathbf{c}_j, \mathbf{c}_k, \mathbf{v})$. In Figs A1(b), (c) and (d) we have shaded (dark) the three sub-triangles corresponding to node 3. The success of the algorithm rests on the fact that by adding the areas of the sub-triangles, with an appropriate sign factor, we can obtain the areas of the second-order Voronoi cells. For example, consider the three sub-triangles corresponding to node 3 in Figs A1(b) to (d). If we subtract the area of the sub-triangles in Figs A1(b) and (d) from the sub-triangle in A1(c) then we get the required polygon (shaded light grey). In a similar way, the areas of the other four second-order Voronoi cells (i.e. for nodes 1, 2, 4 and 5) are found by adding the signed areas of their sub-triangles. Of course the sign of the areas is important in obtaining the correct sum. Fortunately, however, the correct signs are guaranteed by using cyclic order of indices in eqs (A2) and (A6), and ensuring that the vertices of each Delaunay triangle $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ are in positive (counter-clockwise) order. Once these conditions are met, the sub-triangles will always be determined with the correct sign and we need only sum their signed areas with those from the other circum-triangles that correspond to the same node. The total for each node, i, will then give the area of the second-order Voronoi cell between X and node i.

Since the outer loop is over the circum-triangles and the inner loop is over the three vertices of each circum-triangle, it is convenient to introduce a local node numbering for each triangle, i.e. we refer to $\phi_{t,i}$, for the node with local index i in triangle t. The local node numbering is shown in Figs A1(b) to (d). With this we can write the natural-neighbour interpolated function in eq. (2.1) as

$$f(\mathbf{x}) = \frac{1}{A} \sum_{t=1}^{N} \sum_{i=1}^{3} \alpha_{t,i}(\mathbf{x}) f_{t,i}, \tag{A7}$$

where A is the total area of the Voronoi cell about X, given by the sum of the areas of the second-order Voronoi cells,

$$A = \sum_{t=1}^{N} \sum_{i=1}^{3} \alpha_{t,i}(\mathbf{x}), \tag{A8}$$

where N is the number of circum-triangles of $\mathbf{x}$. Watson's method is therefore to calculate the four points given in eqs (A2) and (A5) for each circum-triangle, evaluate the signed areas for the three vertex nodes using eq. (A6), and calculate the double sums in eqs (A7) and (A8). An algorithm in pseudo-code, which implements this approach, is given in Fig. A2.

A1 Derivatives of the interpolated function

For some applications it is useful to have derivatives of the interpolated function. These are obtained by differentiating eq. (A7),

$$\frac{\partial f(\mathbf{x})}{\partial x_s} = \frac{1}{A} \sum_{t=1}^{N} \sum_{i=1}^{3} \frac{\partial \alpha_{t,i}(\mathbf{x})}{\partial x_s} f_{t,i} - \frac{1}{A^2} \frac{\partial A}{\partial x_s} \sum_{t=1}^{N} \sum_{i=1}^{3} \alpha_{t,i}(\mathbf{x}) f_{t,i} \qquad (s = 1, 2), \tag{A9}$$

where $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$. The term $\partial A / \partial x_s$ can be found by differentiating eq. (A8). After substituting this into (A9) and simplifying, we obtain

$$\frac{\partial f(\mathbf{x})}{\partial x_s} = \frac{1}{A} \left[\sum_{t=1}^{N} \sum_{i=1}^{3} \frac{\partial \alpha_{t,i}(\mathbf{x})}{\partial x_s} f_{t,i} - f(\mathbf{x}) \sum_{t=1}^{N} \sum_{i=1}^{3} \frac{\partial \alpha_{t,i}(\mathbf{x})}{\partial x_s} \right] \qquad (s = 1, 2). \tag{A10}$$

The only new terms in (A10) are the derivatives of the signed sub-triangle areas, $\partial \alpha_{t,i} / \partial x_s$. These can be found by

An algorithm for natural neighbour interpolation

Set v, dv, f, f_1, f_2, dv_1, and $dv_2 = 0$;
For each Delaunay circum-triangle of X, $t, (t = 1, \ldots, n)$:
  set $\mathbf{v}$ = circumcentre of triangle $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$ (solve eqn. A.3) ;
  for $i = 1, 2, 3$;
    set j and k using cyclic order;
    calculate matrix $A^{(jk)}$ using eqn. (A.4);
    calculate vector $\mathbf{b}_i$ using eqn. (A.4);
    set $\mathbf{c}_i$ = circumcentre of triangle $(\mathbf{p}_j, \mathbf{p}_k, \mathbf{x})$ (solve eqn. (A.3)) and store;
    calculate vector $\mathbf{d}_{i,s}$ for $(s = 1, 2)$ using eqn. (A.13);
    set $\mathbf{c}_{i,s}$ = solution of linear system (A.12) for $(s = 1, 2)$ and store;
  end;
  for $i = 1, 2, 3$;
    set q = global node index corresponding to local node i;
    set $f_q = f_{t,i}$, the function value at current node;
    set $\alpha_{t,i}$ = signed area of triangle $((\mathbf{c}_j, \mathbf{c}_k, \mathbf{v})$ (using eqn. (A.6));
    set $\phi_q = \phi_q + \alpha_{t,i}$;
    set $A = A + \alpha_{t,i}$;
    set $f = f + \alpha_{t,i} f_q$;
    set $d_1 = \partial \alpha_{t,i} / \partial x_1$ using $(\mathbf{c}_{j,1}, \mathbf{c}_{k,1}, \mathbf{c}_j, \mathbf{c}_k, \mathbf{v})$ in eqn. (A.11);
    set $d_2 = \partial \alpha_{t,i} / \partial x_2$ using $(\mathbf{c}_{j,2}, \mathbf{c}_{k,2}, \mathbf{c}_j, \mathbf{c}_k, \mathbf{v})$ in eqn. (A.11);
    set $f_1 = f_1 + d_1 f_q$;
    set $f_2 = f_2 + d_2 f_q$;
    set $dv_1 = dv_1 + d_1$;
    set $dv_2 = dv_2 + d_2$;
  end;
end;
set $f = \frac{f}{A}$ (eqn. (A.7));
set $f_1 = \frac{1}{A}(f_1 - f dv_1)$ (eqn. (A.10);
set $f_2 = \frac{1}{A}(f_2 - f dv_2)$ (eqn. (A.10);
finish;

Upon exit $f = f(\mathbf{x}), f_1 = \partial f / \partial x_1, f_2 = \partial f / \partial x_2$ and $\phi_q (q = 1, \ldots, n)$ are the natural neighour co-ordinates of the point at $\mathbf{x}$.

Figure A2. A pseudo-code algorithm for natural-neighbour interpolation at the test point X.

differentiating eq. (A6):

$$\frac{\partial \alpha_{t,i}(\mathbf{x})}{\partial x_s} = \frac{1}{2}\left\| \frac{\partial \mathbf{c}_j}{\partial x_s} \times (\mathbf{c}_k - \mathbf{v}) + (\mathbf{c}_j - \mathbf{v}) \times \frac{\partial \mathbf{c}_k}{\partial x_s} \right\|,$$
$$(i = 1, 2, 3), (s = 1, 2). \qquad \text{(A11)}$$

[Note here that if $\mathbf{x}$ lies on the circle passing through the vertices of Delaunay triangle, t, then because each $\mathbf{c}_i$ $(i = 1, \ldots, 3)$ is equal to $\mathbf{v}$, the contributions of triangle t to $\alpha_{t,i}(\mathbf{x})$ and its derivatives are all zero.]

The six equations in (A11) contain the derivatives of the centres of the circum-circles with respect to the components of the position vector, $\partial \mathbf{c}_i / \partial x_s$. For ease of notation we will write these vectors as $\mathbf{c}_{i,s}$, and the components of $\mathbf{c}_i$ as (c_1^i, c_2^i). The derivative vectors may be found by differentiating the linear system of equations in (A3), and we obtain

$$A^{(jk)} \mathbf{c}_{i,s} = \mathbf{d}_{i,s}, \qquad (i = 1, 2, 3), (s = 1, 2) \qquad \text{(A12)}$$

where $A^{(jk)}$ is the matrix given by (A4), and

$$\mathbf{d}_{i,s} = [(c_s^i - x_s), (c_s^i - x_s)]^{\mathrm{T}}. \qquad \text{(A13)}$$

All terms in the 2×2 linear system (A12) are known, and so with cyclic rotation of indices the six vectors $\mathbf{c}_{i,s}$ $(i = 1, 2, 3; s = 1, 2)$ can be calculated. The derivatives of the natural-neighbour interpolated function may now be determined, using eqs (A9) to (A13). An algorithm for calculating the function and first derivative natural-neighbour interpolation is given in Fig. A2, in a pseudo-code language.

In practice, the main limitation of the algorithm is that it breaks down when the point $\mathbf{x}$ lies exactly on the line between two nodes, i.e. on the side of a Delaunay triangle. In this case a circle cannot be constructed to pass through all three points, and so the circum-centre, $\mathbf{c}_i$ in eq. (A2), cannot be defined. It is important to note that the theory of natural-neighbour coordinates does not break down here, and that the second-order Voronoi cells are still well defined. It is only the method of calculating their areas

which breaks down. In this special case the second-order Voronoi cell has a pair of parallel sides and it cannot be broken down into triangles of the type used in the algorithm. At present there seems to be no solution to this problem, although in the application to finite-element modelling (Braun & Sambridge 1995) the interpolated function need only be evaluated at points within the Delaunay triangles and never on the sides. Therefore the break-down condition is never encountered, although an extension of the algorithm to handle this special case may be useful for other problems and is the subject of our current work. In the gridding of irregularly distributed data fields, a satisfactory result is almost always achieved by perturbing any co-linear $\mathbf{x}$ points away from the edge of a Delaunay triangle.

The smoothness of the natural-neighbour influence surface is one of its special properties. Sibson (1980) proved that the natural-neighbour coordinates $\phi_j(\mathbf{x})$ $(j = 1, \ldots, n)$ in eq. (2.1) are continuously differentiable at all points except at the nodes, in any dimension. This means that the influence surface in Fig. 3(b) must smoothly tend to zero at its boundary, except at the positions of its neighbouring nodes (which also lie on the boundary).

This property may also be demonstrated in the 2-D case considered here. First we write $\phi_j(\mathbf{x})$ in terms of the triangle areas $\alpha_{t,i}(\mathbf{x})$, and by definition we have

$$\phi_j(\mathbf{x}) = \frac{1}{A} \sum_t \alpha_{t,i}(\mathbf{x}), \tag{A14}$$

where the summation is over all circum-triangles of $\mathbf{x}$ that have node j as the ith vertex. Differentiating, we obtain

$$\frac{\partial \phi_j}{\partial x_s} = \frac{1}{A} \sum_t \frac{\partial \alpha_{t,i}(\mathbf{x})}{\partial x_s} - \frac{1}{A^2} \frac{\partial A}{\partial x_s} \sum_t \alpha_{t,i}(\mathbf{x}). \tag{A15}$$

From the definition of the influence surface of node j we know that if the test point $\mathbf{x}$ is placed on its boundary (and not at the position of any other node) then $\mathbf{x}$ lies on the circle passing through the three vertices of a Delaunay triangle, and node j is at one of those vertices. Furthermore, this is the only triangle in the summation in (A14) and (A15). We recall, from above, that if $\mathbf{x}$ lies on a circum-circle of triangle, t, then we have

$$\mathbf{c}_i = \mathbf{v} \qquad (i = 1, \ldots, 3), \tag{A16}$$

and therefore by substituting in eqs (A6) nd (A11) we find that

$$\alpha_{t,i}(\mathbf{x}) = 0, \qquad (i = 1, \ldots, 3) \tag{A17}$$

and

$$\frac{\partial \alpha_{t,i}}{\partial x_s} = 0, \qquad (\mathrm{i} = 1, \ldots, 3)\ (\mathrm{s} = 1, 2). \tag{A18}$$

Putting these in eqs (A14) and (A15) we have

$$\phi_j(\mathbf{x}) = 0, \tag{A19}$$

and

$$\frac{\partial \phi_j}{\partial x_s} = 0, \qquad (s = 1, 2) \tag{A20}$$

and so we can verify that for all points on the boundary of the influence surface (except at the positions of other nodes) the natural-neighbour interpolant has no discontinuities in its first derivatives. Note, however, that discontinuities in the gradient of the interpolant do exist at the positions of the neighbours to node j, and this can be seen in Fig. 3(b).

A2 Extensions to 3-D

The algorithm of Watson (Watson 1992) is restricted to 2-D. An extension to 3-D follows along very similar lines, although a detailed understanding is the subject of ongoing work. The main geometric components of the 2-D algorithm readily extend to 3-D, i.e. circles become spheres, triangles become tetrahedra. The expressions for areas and circum-centres can easily be replaced with their 3-D counterparts, and the resulting expressions can be differentiated. The only difficulty lies in consistently obtaining the correct signs of the volumes of 'sub-tetrahedra', i.e. the 3-D equivalent of eq. (A.6). Our ongoing work on this problem shows promise and results will be published elsewhere.

APPENDIX B: CALCULATING ALL CIRCUM-TRIANGLES OF X

The outer loop in the natural-neighbour interpolation algorithm is over all circum-triangles of the point X. A method to obtain these triangles without searching through all triangles was described by Sloan (1987) and attributed to Lawson (1977). Before the procedure can be described, some preliminary definitions are needed.

For any triangle, t, and local node i $(i = 1, 2, 3)$, we define side i to be the side opposite node i. In a similar way we can label each face of a tetrahedron with the index of the one vertex not in the plane of the face. Using this notation we define for each side i of triangle t the function $\omega(t, i)$ to be the index of the triangle that shares side i with triangle t. Note that every side is shared by two triangles except those that form the convex hull, and in 3-D every triangular face is shared by two tetrahedra except those on the hull. If the side i of triangle t is on the convex hull then there is no other triangle sharing that side and so we set $\omega(t, i) = 0$.

With these definitions, the circum-triangles of X can be found using the algorithm in Fig. B1. Briefly, the procedure tests a series of triangles to see if they are circum-triangles of X, i.e. if

$$\|(\mathbf{x} - \mathbf{v})\| < \|(\mathbf{p} - \mathbf{v})\|, \tag{B1}$$

where $\mathbf{v}$ is the circum-centre of the triangle, and $\mathbf{p}$ is one of its vertices. The important part of the procedure is the order in which the triangles are tested, which is controlled by two 'last in first out' (LIFO) stacks. These are data structures, commonly used in computer science (see Sedgewick 1990). The procedure starts with the triangle containing $\mathbf{x}$, which is by definition a circum-triangle of X. We can think of the order of triangles to be tested as a walk through neighbouring triangles. In this walk the first stack contains the next triangle to be tested, while the second stores the previous one. Keeping the second stack prevents the procedure walking back on itself. However, it relies on a special property of the Delaunay triangulation to ensure that by walking 'forward' it never encounters a previously found circum-circle of X (Lawson 1977). When the stacks are empty, all circum-circles have been found.

An algorithm for finding all circumcircles containing the point x

```
Find the triangle, t which contains the point x (use the 'walking triangle algorithm');
This must be a circum-triangle of x (FOUND) :
For i = 1,2,3:
    Set t_new = ω(t,i);
    If t_new ≠ 0 then;
        Place t_new on a 'last in first out stack' (stack A);
        Place t on a second 'last in first out stack' (stack B);
    end;
end;
• If stack A is empty then finish;
Take t_new from top of stack A;
Take t from top of stack B;
If t_new is a circum-triangle of x then;
    We have found another circum-triangle of x (FOUND);
    for i = 1,2,3:
        set t_i = ω(t_new,i);
        If (t_i ≠ 0 and t_i ≠ t) then;
        place t_i on top of stack A;
        place t_new on top of stack B;
    end;
end;
go to •;
```

Figure B1. A pseudo-code algorithm for finding all triangles whose circum-circles contain the test point X. A new circum-triangle is found each time the lines labelled 'found' are encountered.

This searching procedure also works in higher dimensions and has been used by Sloan (1987) as the basis of a method for calculating Delaunay tessellations in the plane. The function $\omega(t, i)$ is also used by the walking triangle algorithm, and a method for its efficient calculation is described in Appendix C.

APPENDIX C: THE WALKING TRIANGLE ALGORITHM

The main book-keeping problem encountered with an irregular mesh of triangles (or tetrahedra) is to find the one containing a given point **x**. When the number of triangles is large, a brute force search through all triangles becomes extremely inefficient, especially when many points may need to be located. The walking triangle algorithm (originally due to Lawson 1977; see also Sloan 1987) finds the enclosing triangle, without the need for a search through all triangles. A pseudo-code representation of the algorithm appears in Fig. C1(a). The steps in the algorithm are quite simple: the 'walk' starts with an initial-guess triangle, t, and tests each side, i ($i = 1, 2, 3$), in turn. (Again, triangle sides are defined by the indices of the nodes opposite, which are in counter-clockwise order.) If the point **x** lies to the 'right' of the current side then the walk moves to the neighbouring triangle that shares this edge. The three sides of this new triangle are now considered in the same way as before. The walk stops when **x** lies to the 'left' of all three sides, in which case it must be inside the current triangle.

To test if the point (x, y) lies to the right of side i, we need only check whether the condition

$$(p_y^j - y)(p_x^k - x) > (p_x^j - x)(p_y^k - y) \qquad \text{(C1)}$$

holds, where $\mathbf{p}_j = (p_x^j, p_y^j)^{\mathrm{T}}$ and $\mathbf{p}_k = (p_x^k, p_y^k)^{\mathrm{T}}$ are the nodes on the edge, and j and k are defined by cyclic rotation of indices. The efficiency of the algorithm is due to the nearly direct path it takes to the triangle containing the point. Fig. C2 shows an example of its application to a set of 980 triangles. The initial triangle is in the bottom right corner (black), and the correct triangle is in the top left (black). In this case the correct triangle is located by sampling only 60 triangles. Obviously the number of triangles tested depends on the distance between the starting triangle and the correct one, but the algorithm is guaranteed to work starting from any triangle. If a series of points to be located are close to each other, then the triangle containing the previous point serves as an excellent starting guess for the next. The algorithm is also independent of the size and complexity of the triangulation, and may be used to locate points within any set of convex polygons (e.g. rectangles, hexagons, etc.) whose outer perimeter forms a convex polygon.

C1 An extension to 3-D

The algorithm requires some modifications to work with 3-D tetrahedra. This is because there is no analogue of the counter-clockwise step around the sides of the triangle. Our extension is quite simple—we merely replace the test at each side (of the triangle) with a more sophisticated test at each face of the tetrahedron which distinguishes between the inside and outside of the tetrahedron: we know the point **x** must be outside the current tetrahedron if **x** and the vertex $\mathbf{p}_i$ are on opposite sides of the face i, i.e. if

$$\{(\mathbf{p}_i - \mathbf{p}_j) \cdot [(\mathbf{p}_k - \mathbf{p}_j) \times (\mathbf{p}_l - \mathbf{p}_j)]\} \times \{(\mathbf{x} - \mathbf{p}_j) \cdot [(\mathbf{p}_k - \mathbf{p}_j) \times (\mathbf{p}_l - \mathbf{p}_j)]\} < 0, \qquad \text{(C2)}$$

(a) The walking triangle algorithm

```
Set t to the index of any triangle;
• Triangle t is the current best guess index of the triangle containing x;
For i = 1,2,3:
    Set j and k by cyclic rotation;
    Set t_new = ω(t,i); (t_new is the other triangle which shares side i)
    If x is to the right of the line (p_j, p_k), (use eqn. C.1) then:
        If t_new = 0 then x is outside of convex hull so finish;
        If t_new ≠ 0 then set t = t_new;
        go to • ;
    end;
end;
The point x must be in triangle t;
finish;
```

(b) An algorithm to calculate the adjacency matrix

```
Let V_{t,i} be the global node index of the i th vertex of triangle t.
For q = 1,...,N: set n_q = the number of natural neighbours of node q;
For t = 1,...,N_T:      (loop over triangles)
    For i = 1,2,3:      (loop over sides of current triangle)
        Set j and k by cyclic order;
        Set q = V_{t,i}, q_1 = V_{t,j} and q_2 = V_{t,k}      (global node indices);
        For m_1 = 1,...,n_{q_1}:      (loop over neighbours of node q_1)
            If C(q_1,m_1) = 0 then; (record connections between q_1 and q_2)
                Set C(q_1,m_1) = q_2;
                Set T(q_1,m_1) = t;
                For m_2 = 1,...,n_{q_2}: (loop over neighbours of node q_2)
                    If C(q_2,m_2) = 0 then;
                        Set C(q_2,m_2) = q_1;
                        Set T(q_2,m_2) = t;
                        go to •;
                    end;
                end;
            Else if C(q_1,m_1) = q_2 and T(q_1,m_1) ≠ t, then:
                Set t_new = T(q_1,m_1);
                Triangles t and t_new are neighbours !
                Set i_new = index of the side between nodes q_1 and q_2 in t_new ;
                set ω(t,i) = t_new; (Insert entries into adjacency matrix)
                set ω(t_new, i_new) = t;
                go to •;
            end;
        end;
    • done side i in triangle t;
    end;
end;
finish;
```

Figure C1. (a) A pseudo-code algorithm for calculating the triangle containing the test point X. (b) A pseudo-code algorithm for calculating the adjacency matrix. Two other lists which describe aspects of the Delaunay triangulation are obtained as by-products.

where the three vertices of face i contain the nodes $(\mathbf{p}_j, \mathbf{p}_k, \mathbf{p}_l)$. If this condition is met for any face i, then we move into the tetrahedron on the other side of the face and start again. In this way the algorithm moves through arbitrarily sized and oriented tetrahedra towards the point $\mathbf{x}$ in just the same way as the walking triangle algorithm, although now the faces may be sampled in any order. If many points are to be located then it may be more efficient to calculate, and store in advance, the terms in (C2) that do not depend on $\mathbf{x}$. This can be done with the scalar in the

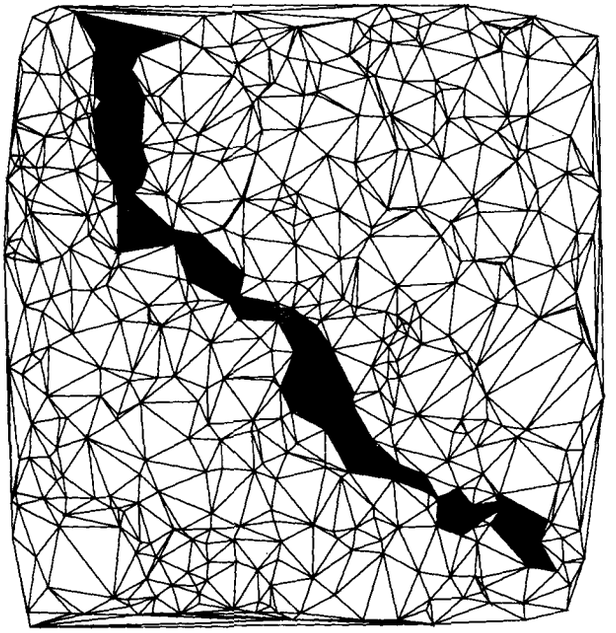


Figure C2. An example of the path (shaded triangles) taken by the walking triangle algorithm. The initial triangle is in the lower right corner (black) and the final triangle is in the top left corner (black). The nearly direct path taken by the algorithm enables it to locate efficiently a point in any triangle.

first curly brackets and the cross product in the second curly brackets. With these values stored, the evaluation of (C2) requires only the dot product of two vectors for every new **x**.

C2 Calculating the adjacency matrix

The main prerequisite for the walking triangle algorithm (and the circum-triangle search in Appendix B) is the adjacency function $\omega(t, i)$, or adjacency matrix Ω, (where $\Omega_{t,i} = \omega(t, i)$). Lawson (1977) and Lee & Schachter (1980) do not give a method for determining this matrix. Sloan (1987) calculates it as a by-product of a 2-D Delaunay triangulation algorithm. A simple 'brute force' method would be to search for all pairs of triangles with two vertices in common. The time taken for this type of approach would increase in proportion to the square of the number of triangles, which can be very inefficient when the number of nodes is large. In Fig. C1(b) we present an approach which requires only a single loop over the triangles. The time taken by this algorithm has only a linear increase with n_t and is therefore much more efficient when the number of nodes becomes large.

The input to the algorithm is the triplet of global node indices for each triangle, which we denote by $V_{t,i}$ ($i = 1, 2, 3; t = 1, \ldots, N_T$), where N_T is the total number of triangles. This is the data structure which defines the Delaunay triangulation, and is the output of the quickhull algorithm discussed in Section 2. Two by-products of the new method are the matrices $C_{q,p}$ and $T_{q,p}$ ($p = 1, \ldots, n(q); q = 1, \ldots, N$), which give the indices of the pth natural neighbour and the pth triangle connected to node q, respectively, where $n(q)$ is the number of natural neighbours to node q, and N is the total number of nodes. (In practice the new arrays may be calculated and stored in a compact format which eliminates all zero entries.) Although cyclic order is used in the algorithm to define the indices j and k, the order in which the sides of each triangle are visited is not important. Once i is fixed we need only set j and k to the other two local indices, in any order. Therefore the algorithm can be extended to work in 3-D by simply increasing the number of local nodes (in each tetrahedron) from 3 to 4.