A numerical method for solving partial differential equations on highly irregular evolving grids

Jean Braun* **& Malcolm Sambridge*†**

* Research School of Earth Sciences, Institute of Advanced Studies, † Centre for Information Science Research, Australian National University, Canberra, ACT 0200, Australia

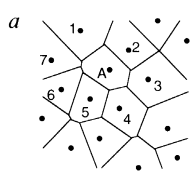
An efficient numerical method is described for solving partial differential equations in problems where traditional eulerian and lagrangian techniques fail. The approach makes use of the geometrical concept of 'natural neighbours', the properties of which make it suitable for solving problems involving large deformation and solid–fluid interactions on a deforming mesh, without the need for regridding. The approach can also be applied to high-order partial differential equations (such as the Navier–Stokes equation), even in cases where the evolving mesh is highly irregular.

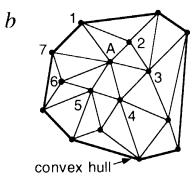
NUMERICAL methods for solving partial differential equations (PDEs) require some form of spatial discretization, or mesh of nodes, at which the solution is specified. The nature of the mesh is an important factor in determining the accuracy and stability of the method as well as the type of PDE that can be solved. Straightforward methods, such as the finite-difference method[1] (FDM), are based on a regular spatial discretization, whereas more flexible techniques such as the finite-element method (FEM) divide the plane into triangles or rectangles, often with an irregular distribution[2]. In many cases, the spatial mesh is made to change during the course of a calculation. An example is a 'self-adapting mesh' where nodes are added during the calculation to improve accuracy in a particular region, the position of which is determined by the solution itself[3]. The most difficult situation to handle is when the mesh is required to evolve with the solution[4], that is, mesh nodes move during the calculation. An example is the case of advection in which the mesh nodes are 'attached' to a deforming medium. Although an evolving mesh allows for material properties to be accurately transported at the nodes, large displacements quickly result in severe mesh distortion which in turn increases numerical instability and restricts accuracy.

Here we show how the fundamental geometrical concept of 'natural neighbours'[5] may be used to overcome these problems. The result is a new method which can be applied to problems where traditional eulerian and lagrangian techniques fail; for example, those involving large deformation, or fluid–solid interactions. The two essential features are the way in which both the mesh nodes and the connections between nodes are updated during the calculation to maintain an appropriate 'well-shaped' triangulation, and the use of 'natural neighbour' coordinates and their derivatives to interpolate smoothly between arbitrarily distributed nodes. We use a two-dimensional (2D) example of a sinking elasto-plastic plate in a linear viscosity fluid to illustrate these features, but all the theory described is equally valid in three dimensions. The crucial factor that makes the new approach possible is that the computational cost of all of the numerical algorithms increases at most linearly with the number of nodes.

'Natural neighbour' coordinates

A unique and fundamental property of an arbitrarily distributed set of nodes in a plane (or in any dimension) is the set of 'natural neighbours' about each node. Natural neighbours are defined in Fig. 1*a* and *b*. Any two nodes are said to be natural neighbours if their Voronoi cells[6] have a common boundary, and the Voronoi cell about each node is the part of the plane which is closest to that node. Natural neighbours have some very interesting and useful properties. The number, and distribution, of neighbours about each node vary with the density of the nodes. Their distribution is a compromise between being 'close to' and 'surrounding' a node, and yet they are uniquely determined by the nodal distribution. By connecting each node to its neighbours we obtain the Delaunay triangulation[7] which results in a 'well-shaped' set of triangles, that is, as near equilateral as possible.





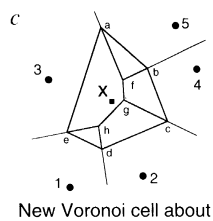


FIG. 1 *a*, A set of randomly distributed nodes and their Voronoi cells in a plane. The natural neighbours of node A are numbered 1–7. *b*, The Delaunay triangulation is obtained by connecting all pairs of natural neighbours together. The Voronoi cells are unique in any number of dimensions and their boundaries consist of the perpendicular bisectors between points. The Delaunay triangulation is unique unless four neighbours lie on a circle (or five on a sphere in three dimensions). *c*, A test point **x** and its five natural neighbours. The shaded area is the Voronoi cell about **x**. This first-order Voronoi cell can be divided into five second-order Voronoi cells. Each second-order Voronoi cell contains that part of the plane which is closest to **x** and second closest to one of its neighbours. Natural-neighbour coordinates, $N_i(\mathbf{x})$, are defined as the ratio of the area of the second- to first-order Voronoi cells. For example, $N_3(\mathbf{x})$ is equal to the area of the polygon (a, f, g, h, e) divided by the area of the polygon (a, b, c, d, e). Highly efficient 'local' algorithms have been found for calculating Delaunay triangulations and natural-neighbour coordinates in two dimensions[8,13,15]. Sambridge *et al.*[15] also present analytical expressions for the derivatives of $N_i(\mathbf{x})$ with respect to the components of **x** in two dimensions. Recently, we have implemented a different algorithm for calculating natural-neighbour coordinates and derivatives in three or more dimensions (see Box 1).

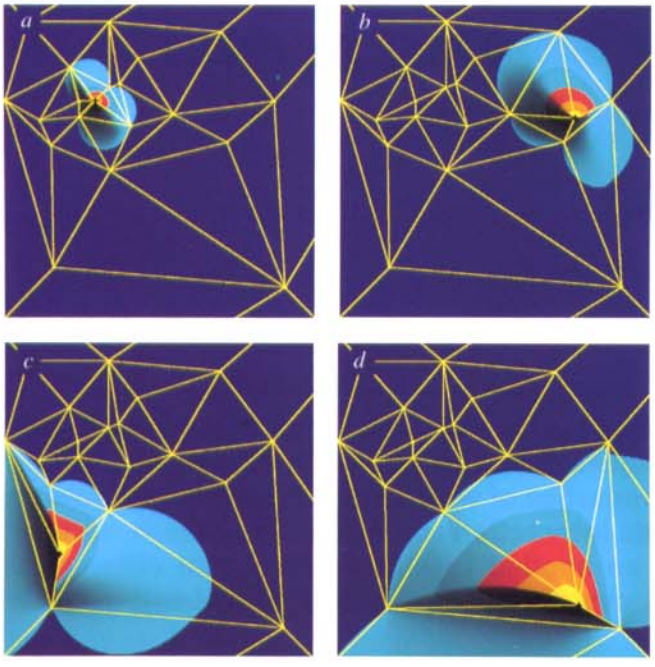


FIG. 2 The natural-neighbour influence function, $N_i(\mathbf{x})$, associated with a node i at various locations within a set of fixed nodes irregularly distributed in the plane. The influence function is displayed as an artificially illuminated surface with the height proportional to the value of the trial function about the node i marked by a cross (one at the node; zero at all other nodes). The surface is divided into five contours from yellow to blue; each band represents an interval of 0.2 in the value of the trial function. The surface shows how the node influences its surrounding region and that its shape is solely dictated by the distribution of nodes: it expands in regions of low nodal density (*a*) and shrinks in regions of high nodal density (*b–d*). An example of a 3D natural-neighbour influence function is shown in Figs. 3 and 4.

Efficient methods have been developed to calculate Delaunay triangulations (and the dual Voronoi diagram) in two and three dimensions[8–10]. To make the Delaunay mesh useful in solving PDEs an interpolation method is required to estimate the solution (and its derivatives) between the nodes. A simple linear interpolation could be used[11] but this would restrict the approach to low-order PDEs. We use natural-neighbour

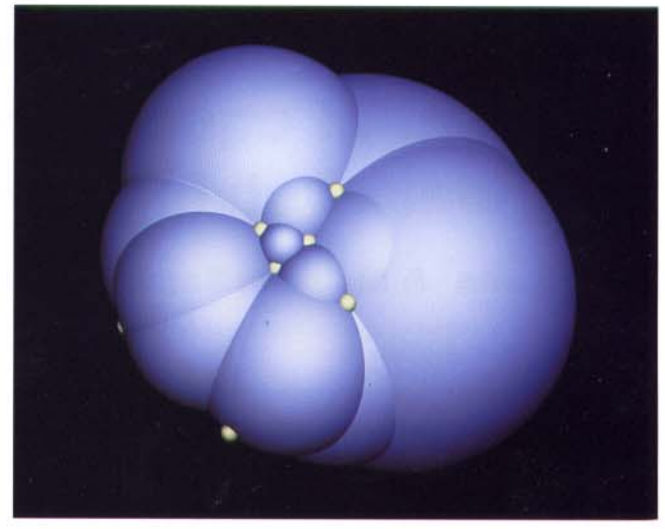


FIG. 4 The isosurface for which the natural-neighbour coordinate is zero for the nodes shown in Fig. 3. The surface is shown from a different perspective than employed in Fig. 3 to highlight the intersection of the (large) spheres. The spheres always intersect at a neighbour node (smallest spheres), and all spheres pass through the central node, which is obscured in this view.

interpolation[12] in which the value of $f(\mathbf{x})$ at a point $\mathbf{x}$ is given by

$$f(\mathbf{x}) = \sum_i N_i(\mathbf{x}) f_i \qquad (1)$$

where the weights $N_i(\mathbf{x})$ are the natural-neighbour coordinates of $\mathbf{x}$ with respect to its natural-neighbour nodes, and f_i is the corresponding field value at the node. Natural-neighbour coordinates of the point $\mathbf{x}$ are defined in Fig. 1*c*. Each natural-neighbour coordinate, $N_i(\mathbf{x})$, can be displayed as an influence function about node i. Examples in two and three dimensions are shown in Figs 2, 3 and 4. They are uniquely defined for any distribution of nodes in any number of dimensions[5,13]. They also have the important property of being orthogonal, which means that the interpolation will recover the original function values at the nodes, that is, we have:

$$N_i(\mathbf{x}_j) = \delta_{ij} \qquad (2)$$

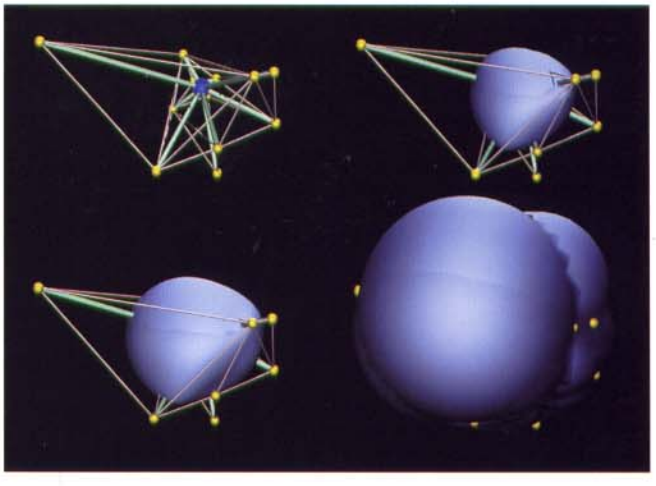


FIG. 3 Examples of the natural-neighbour influence function in three dimensions, calculated with the new method described in Box 1. Top left, a ray-traced image of a central node (purple) connected to its 12 natural-neighbour nodes (yellow). The connections between the neighbours are marked with a thin pipe (pink), while the connections between the central node and its neighbours are marked with a thicker pipe (green). Together they form 20 Delaunay tetrahedra, all of which have the central node at a vertex. The network shown is only a subset of a larger number of surrounding nodes distributed randomly in three dimensions. Top right, a surface on which the natural-neighbour coordinate with respect to the central node is a constant ($N(\mathbf{x}) = 0.1$). The value of the natural-neighbour coordinate represents the weight given to that node in natural-neighbour interpolation. Two other isosurfaces, at values of 0.05 and 0.0000001, are shown in the bottom left and bottom right panels, respectively. Notice how each surface is smooth. The smoothness properties of natural-neighbour interpolation arises directly from the continuity of this influence function. When the natural-neighbour coordinate is equal to zero, the isosurface consists of the union of 20 spheres (see Fig. 4). Each of these spheres passes through the four vertices of one of the Delaunay tetrahedra connected to the central node.

where $\mathbf{x}_j$ is the position of node j (δ_{ij} is the Kronecker delta). Another important property is that they are isoparametric[2], which means that geometrical coordinates are interpolated exactly, that is, we have:

$$\mathbf{x}=\sum_i N_i(\mathbf{x})\mathbf{x}_i \tag{3}$$

The third important property is that they are continuously differentiable (C_∞) everywhere except at the nodes, $\mathbf{x}_i$, where all derivatives are discontinuous[5]. These three properties make them very powerful when used as the basis of numerical methods for solving PDEs.

The 'natural-element method'

In this section we present a new approach for solving PDEs which we called the 'natural-element method' (NEM). It is an example of a general Galerkin form of the weighted-residuals method[14] in which the newly defined natural-neighbour coordinates are used as geometrical trial functions. In this new method, the solution $u(\mathbf{x})$ to a general set of PDEs

$$\Lambda(u)=0 \tag{4}$$

in a domain V, with boundary conditions

$$\Gamma(u)=0 \tag{5}$$

on boundary S, is approximated by the solution of the following integral equation,

$$\int N_j^T\Lambda\left(\sum_i N_i u_i\right)\mathrm{d}V+\int N_j^T\Gamma\left(\sum_i N_i u_i\right)\mathrm{d}S=0 \tag{6}$$

where $u(\mathbf{x})$ has been expanded in terms of trial functions $N_i(\mathbf{x})$ ($i=1,\ldots,n$),

$$u(\mathbf{x})\approx\sum_i N_i(\mathbf{x}-\mathbf{x}_i)u_i \tag{7}$$

The u_i are the estimated values of the solution, $u(\mathbf{x})$, at the nodes $\mathbf{x}_i$.

In FEM[2], nodes are usually defined at the vertices of triangular or rectangular elements in two dimensions, or polyhedra in three dimensions, and polynomials are used as trial functions within each element. In this case the trial functions are only continuous to the order of the polynomial across element boundaries. In addition, if the PDE is of high order then the polynomials must also be of high order, which often places constraints on the positions of nodes, for example, nodes at the vertices and faces of triangular or rectangular elements. In NEM, natural-neighbour coordinates are used as trial functions and,

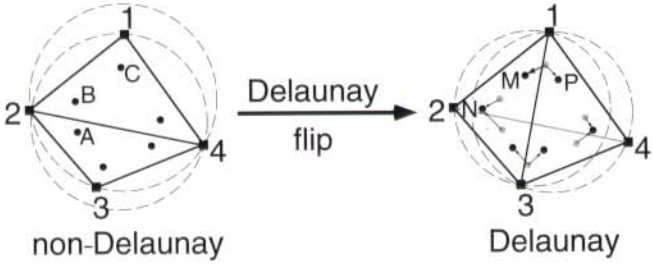


FIG. 5 The basic 'flip operation' used to update the triangulation after each deformation step. Every triangle is checked against each of its neighbours to ensure that the triangulation satisfies the in-circle test: no circle constructed around the corners of the Delaunay triangles contains any node. If two neighbouring triangles do not satisfy the test (note that node 1 is inside the circle built around nodes (2 3 4)) then the side common to the two triangles (2 4) is removed and a new side is added (1 3). The tensors known at the three integration points inside triangles (1 2 4) and (2 3 4) are interpolated onto the integration points inside triangles (1 2 3) and (1 3 4) by simple averaging and splitting, for example $\sigma(\mathbf{N})=[\sigma(\mathbf{A})+\sigma(\mathbf{B})]/2$ and $\sigma(\mathbf{M})=\sigma(\mathbf{P})=\sigma(\mathbf{C})$.

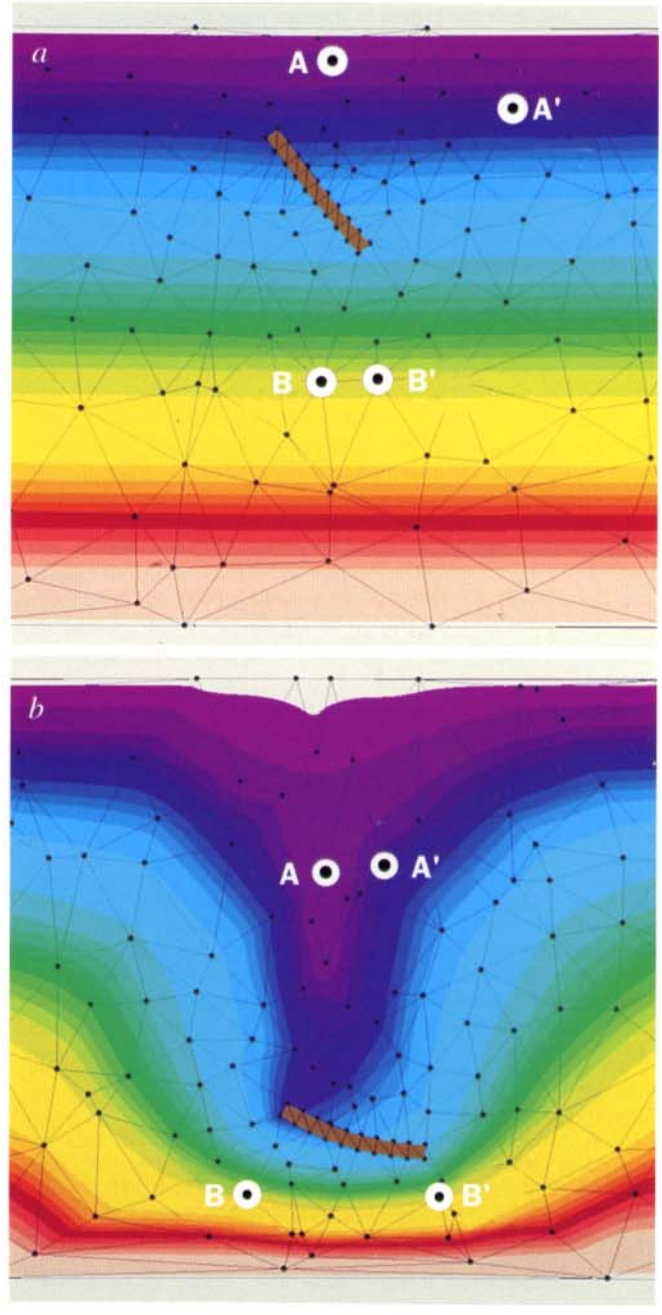


FIG. 6 Natural-element method (NEM) solution of a Stokes flow problem in which motion is driven by the fall of an elasto-plastic plate denser than the viscous fluid. We solve the Stokes equation[20] at the integration points in the linear fluid and the equations of force balance[21] at the integration points located within the elasto-plastic plate, which obeys the Von Mises failure criterion. The solution is shown at times 1 (*a*) and 1,000 (*b*). The nodes on the boundary between the plate and the fluid are shared, which ensures no slip between the fluid and the plate. We use an iterative predictor–corrector scheme to maintain stability in the lagrangian grid update performed at each time step from the computed velocity. The fluid viscosity structure, $\eta(z)$, is carried by the nodes. ($\eta(z)=\eta_0$ for $z>0.75L$ and $\eta(z)=10\eta_0$ for $z<0.75L$, where the box has dimensions $L\times W(=10L)$). The colours from purple to white represent the initial depth of fluid particles. The time-dependent solution shows the plate 'falling' down and dragging along the incompressible fluid which causes a return flow pattern to develop. Notice how the mesh accommodates the movement of the lagrangian nodes: points that were originally natural neighbours (for example, B and B') are moved apart by the divergent flow, whereas points that were originally far apart (for example, A and A') eventually become neighbours.

BOX 1 A new method to calculate natural-neighbour coordinates and their derivatives in three or more dimensions

In any number of dimensions, natural-neighbour coordinates are defined as the ratio of the volume of a second-order Voronoi cell to the volume of the first-order Voronoi cell about a point $\mathbf{x}_p$. First- and second-order Voronoi cells are defined in Fig. 1c, for the 2D case. Because the sum of the second-order cells is equal to the first-order cell, to calculate natural-neighbour coordinates in three dimensions we need only determine the volume of each second-order Voronoi cell. Watson[13] introduced a method to do this for the 2D case. Here we briefly describe a method that can be generalized to any number of dimensions (a fuller treatment is available: see Supplementary Information). Our approach is based on the fact that all second-order Voronoi cells are convex polyhedra (or convex polytopes in higher dimensions). This means that all points inside a second-order Voronoi cell (in dimension n) satisfy a system of linear inequality constraints,

$$A\mathbf{x} \leqslant \mathbf{b} \tag{9}$$

where the matrix A has $m+1$ rows and n columns, and $m+1$ is the number of boundaries (sides) of the cell. By definition the first bounding plane is the perpendicular bisector of $\mathbf{x}_p$ and one of its neighbours, which we shall call $\mathbf{x}_0$, and the m remaining are the bisectors of $\mathbf{x}_0$ and the nodes which are neighbours of both $\mathbf{x}_p$ and $\mathbf{x}_0$. In two dimensions, these facts can easily be verified from Fig. 1c, and they may be generalized readily to any number of dimensions.

A recursive formula for calculating the volume of any convex polytope in any number of dimensions was given by Lasserre[16],

$$V(n, A, \mathbf{b}) = \frac{1}{n} \sum_{i=0}^{m} \frac{b_i}{|a_{it}|} V'_{it}(n-1, \tilde{A}_{i,t}, \tilde{\mathbf{b}}_t) \tag{10}$$

where $\tilde{A}_{i,t}$ is the reduced matrix obtained from A by eliminating the tth variable using the equation $\mathbf{a}_i \cdot \mathbf{x} = b_i$, $\tilde{\mathbf{b}}_t$ is the corresponding reduced vector and a_{it} is the tth element of $\mathbf{a}_i$. The value of t is usually chosen so that $|a_{i,t}|$ is a maximum. The recursive evaluation of equation (10) is best viewed as a tree structure, with V at the root and the first $m+1$ terms as branches attached to V. In the second level of the recursion, each reduced matrix and vector correspond to a system of inequalities with one less constraint and one less variable. Therefore each branch is attached to m other branches, and so on for each level. After $n-1$ recursions we reach the leaf nodes, and for each of them the system of inequalities is reduced to a set of 1D constraints. In each case the length of the region satisfying all constraints becomes the final contribution to the recursive formula in equation (10).

For any given point $\mathbf{x}_p$ the matrix A and vector $\mathbf{b}$ can be easily determined for each second-order Voronoi cell, and the recursive algorithm leads to an efficient evaluation of their volumes. The accuracy of the second-order Voronoi volumes can be independently verified by comparing their sum to the volume of the first-order cell. We have done this for a test set of 220 nodes in three dimensions; in each case we did not find any discrepancies above the level of machine precision. An example of natural-neighbour interpolation using this formula is given in Figs 3 and 4.

The natural-element method requires derivatives of the natural-neighbour coordinates with respect to the coordinates of the test point $\mathbf{x}_p$. A differentiation of V in equation (10) leads, after some algebra, to a new recursive expression for each derivative, whose computation takes about the same time as the volume itself. The situation is simplified by the fact that only the first row of A and $\mathbf{b}$ depend upon $\mathbf{x}_p$. However, this property is lost for the $i=0$ term in equation (10). It turns out that for these, and all lower branches of the recursive tree, all elements of the reduced matrix $\tilde{A}_{it}$ and vector $\tilde{\mathbf{b}}_t$ become functions of $\mathbf{x}_p$. With the aid of some extensive algebra it is possible to keep track of the dependent terms and calculate all derivatives. The validity of the resulting derivative formulae have been verified by comparison to finite difference estimates for a set of irregularly distributed nodes in up to four dimensions.

therefore, no discontinuities are present across element boundaries as natural-neighbour coordinates are continuously differentiable at all points except at the nodes themselves[5]. Furthermore, like 'classical' trial functions they are orthogonal and isoparametric (equations (2) and (3)), but as they place no constraint on the positions of nodes they are well suited to solving problems on deforming or self-adapting meshes, even for high-order PDEs like the Navier–Stokes equation.

In the implementation of the natural-element method several computational issues arise. The first, and most important, is the need for a parametric expression for the spatial derivatives of the natural-neighbour coordinates which can be inserted in the integral form of the differential equation (3). In the 2D case, these have been derived by Sambridge *et al.*[15]. A new method is outlined in Box 1 which can handle any number of dimensions. An additional advantage with this method is that, unlike with the approach used by Sambridge *et al.*[15], it does not break down when evaluated along the lines (or planes in three dimensions) connecting the mesh nodes. The new approach is based on the formula of Lasserre[16] for calculating the volume of n-dimensional convex polyhedra. An example of a 3D trial function that was computed with this approach is shown in Figs 3 and 4.

The second computational issue concerns evaluation of the integrals in equation (3). In classical FEMs this is usually done using Gauss–Legendre numerical integration over individual elements[2]. This involves estimating the integral as a weighted sum of the interpolated values of the integrand at a set of Gauss–Legendre points inside each element. Often with polynomial trial functions the integrals can be evaluated exactly. However, natural-neighbour coordinates are not simple polynomial expressions and so Gauss–Lengendre integration in NEM gives only an approximate integral. For a simple 2D elastic problem, we have evaluated the level of accuracy obtained from a series of 2D integration schemes, namely 1, 3, 4, 7 and 13 integration points per element[2,17]. In this test problem, three integration points within each Delaunay triangle appeared sufficient. Note, however, that as the integrand need only be evaluated inside, or along the edge of each triangle, the discontinuity in the derivatives of the natural-neighbour coordinates at the nodes is never encountered. Nevertheless the question of finding an optimal integration scheme for natural-neighbour coordinates remains open.

The third issue arises in problems where tensors (such as an initial stress) rather than scalars (such as viscosity or elastic parameters) have to be stored from one time step to the next. 'Tensorial memory' is required in structural mechanics problems for example, whereas 'scalar-only memory' is required in fluid flow or diffusion problems. Unlike scalar quantities, tensors are characterized by orientations (Euler angles or principal directions). Scalar quantities can be evaluated and stored at the nodes of the numerical mesh (which represent material points), whereas tensors must be evaluated at the integration points inside the elements. As the triangulation is updated at each time step, the geometry of the elements changes and a new set of integration points has to be defined, which means that tensors known at the old integration points have to be interpolated onto the new ones. This interpolation will introduce some numerical diffusion which should be minimized by ensuring fine mesh discretization along material interfaces or in regions where tensors (stresses) are discontinuous. It is important to note, however, that in 2D problems, this interpolation is only local. For example, in our 2D implementation of NEM, the mesh update is performed via a series of local Delaunay 'flips' (see Fig. 5) involving two neighbouring elements. During each Delaunay flip, we take advantage of the distribution of the three integration points within the element to devise a simple local interpolation scheme based on two-point averaging and splitting (see Fig. 5). This simple scheme ensures conservation of linear relationships between tensor invariants. A similar procedure is yet to be developed for 3D problems. It is important to stress, however, that this interpolation is needed only in problems involving 'tensorial memory'. No interpolation is required in cases where only scalar quantities need be carried with the deforming mesh, for example in the Navier–Stokes equation, because, unlike tensors, scalars can be stored at the material nodes. Therefore the present

implementation of NEM may be used for these cases in two or three dimensions.

The fourth computational issue concerns the type of method used to solve the set of algebraic equations which results from combining equations (6) and (7),

$$A_{i,j}u_j = b_i \tag{8}$$

In classical FEMS the nodes can be arranged to minimize the bandwidth of the matrix A which ultimately reduces the number of operations needed to solve equation (8). In NEM, the node connectivity is more complex and may change during the calculation, therefore it is not practical to attempt node numbering that will minimize the bandwidth of A. This problem is avoided by using an iterative linear system solver (for example, Gauss–Seidel with over-relaxation[18], or the 'element by element' method with Cholesky factorization[19]).

Efficiency of NEM. Sambridge *et al.*[15] have shown that all operations required to compute natural-neighbour coordinates (including the construction of the Delaunay triangulation) are local and, therefore, require a number of arithmetic operations that increases approximately linearly with the number of nodes. This implies that the computational effort required to form the NEM matrix (equation (8)) is similar to the one required in classical FEM. Table 1 shows the results of computations demonstrating that NEM is only about half as efficient as FEM. Furthermore, it must be stressed that the time required to compute the NEM, or FEM, matrix is usually only a small fraction of that needed to solve the resulting algebraic system of equations, and therefore, in practice, there will be little difference in overall computational time between the two approaches.

An example of NEM: fluid–solid interactions

To illustrate the great geometrical flexibility of NEM, we have selected a problem of creeping flow inside a rectangular box driven by a free-falling elasto-plastic plate of greater density than the fluid (Fig. 6*a*). The Navier–Stokes equations[20] with zero Reynolds number are solved in the incompressible fluid whereas the equations of static equilibrium[21] are solved in the elasto-plastic plate obeying the Von Mises failure criterion[21]. Free slip is assumed on all boundaries. The flow created in the fluid by the falling plate exerts stresses on the plate, which may deform elastically. Increasing internal stresses in the deforming plate leads to irrecoverable plastic deformation (Fig. 6*b*). This type of coupled fluid–solid system could not be solved using 'classical' eulerian or lagrangian methods. In the eulerian formulation of flow, the boundaries of the solid plate would have to be recomputed at every time step by interpolation onto the 'frozen' mesh, which would eventually lead to inaccuracy due to numerical diffusion during interpolation. In a small-deformation-only lagrangian representation, flow in the fluid would require a very large number of remeshing and interpolating steps to avoid excessive mesh deformation.

In our example the initial Delaunay mesh is constructed from a random distribution of nodes with a density inversely proportional to the distance from the plate (Fig. 6*a*). As expected, the Delaunay triangles are as near equilateral as possible. Nevertheless because the nodal distribution is deliberately non-ideal, some triangles are very elongated. These triangles would be unacceptable in a 'classical' finite element mesh as they would yield inaccurate estimates of derivatives. There is no such restriction in NEM as the derivative estimates at every Gauss-integration point are derived from its surrounding natural neighbours and not just the nodes at the vertices of the triangle containing the point. This means that long thin triangles are less problematical with NEM than in classical lagrangian FEM for two reasons. First, the retention of a Delaunay mesh means that these triangles are less likely to occur, and second, if thin triangles are present the natural-neighbour interpolation ensures that the 'quality' of the derivative estimates everywhere depends on the local node density and not just on the shape of a single triangular element.

In a lagrangian formulation, the position of the nodes is updated at each time step, using the velocity field calculated from the previous time step, which can quickly stretch the mesh and lead to highly distorted elements. This is the reason why lagrangian FEM is usually restricted to small-deformation-only problems. At each time step of NEM we update the nodal positions and connections so that a Delaunay triangulation is maintained. This ensures stability and accuracy of the derivative estimates regardless of how far the nodes have moved. Also we have the added advantage that all fluid properties are carried with the flow without the need for re-interpolation or regridding of the mesh. NEM is therefore very well suited to track material interfaces, deforming boundaries or material properties, even in large-scale-flow problems. In our example, the density of the nodes varies because of the divergent nature of the flow around the plate and along the sides of the box. Because the local nodal density can easily be monitored (using the size of the associated Voronoi cells), it is also straightforward in NEM to dynamically 'inject' new nodes in regions where the density falls beyond some critical value. If this is done, it turns out that only local changes to the Delaunay mesh are required and consequently only those elements of the matrix A that correspond to the new nodes and their neighbours need be recomputed.

TABLE 1 Relative efficiency of NEM

Number of elements	FEM	NEM		Delaunay triangulation
		Method 1	Method 2	
32	1	1.1	2.1	0.08
200	1	1.2	2.1	0.03
1,800	1	1.36	2.22	0.03

Computing time required to form the NEM matrix normalized by the time taken to form the equivalent FEM matrix for a simple elastic test problem. Quadratic triangular finite elements with three Gauss integration points were used. NEM method 1 is based on the natural-neighbour algorithm used in Sambridge *et al.*[15]; NEM method 2 is based on the algorithm described in Box 1. The last column gives the time taken to form the Delaunay triangulation and update it at the end of the time step (normalized by the FEM matrix-formation time). These operations have to be performed in NEM, not FEM, but they do not require substantial computational overhead.

Future directions

Our use of the natural-neighbour theory to solve PDEs demonstrates the great geometrical flexibility of the method for solving complex problems on highly irregular evolving grids without compromising accuracy.

The power of NEM is derived from the fundamental geometrical concepts on which it is based, all of which generalize to higher dimensions; for example, triangles in two dimensions become tetrahedra in three dimensions and Voronoi cells become convex polyhedra. More importantly, all natural-neighbour coordinates remain orthogonal, isoparametric and continuously differentiable[5]. The new method we have developed to compute the natural-neighbour coordinates (see Box 1) is valid in any number of dimensions. NEM can therefore be used to solve problems in two or three dimensions. However, because there is, as yet, no direct extension of the 2D 'flip operation' in three dimensions, the Delaunay triangulation has to be computed at every time step by another method (for instance, by using the method of Barber *et al.*[9]). This also means that there is no 3D equivalent of the 2D averaging and splitting strategy that we have used in our implementation of NEM (see Fig. 5), and therefore, in three dimensions, NEM is at present restricted to problems requiring scalar-only memory (like the Navier–Stokes or diffusion equation).

This article has dealt solely with the application of natural-neighbour theory to the weighted residual method for solving PDEs. We expect that many other applications of natural neighbours will appear in the future. For example, it may well be possible to generalize classical finite-difference methods (which rely on estimating derivatives at nodes of a regular grid[1]) to irregular meshes. Although the natural-neighbour coordinates themselves cannot be used to estimate derivatives at nodes, an extension described by Sibson[12] does give first-order derivatives at the nodes. Because most numerical methods are based on the discretization of a field onto a mesh, the theory of natural neighbours raises the exciting possibility of generalizing methods commonly restricted to regular grids to a Delaunay mesh based on any irregular distribution of nodes. □

Received 27 February; accepted 27 July 1995.

1. Roache, P. J. *Computational Fluid Dynamics* (Hermosa, Albuquerque, NM, 1982).
2. Zienkiewicz, O. C. *The Finite Element Method* 3rd edn (McGraw-Hill, Maidenhead, 1977).
3. Zienkiewicz, O. C. & Phillips, D. V. *Int. J. num. Meth. Engng* **8,** 519–528 (1971).
4. Bathe, K.-J., Ramm, E. & Wilson, E. L. *Int. J. num. Meth. Engng* **9,** 353–386 (1975).
5. Sibson, R. *Math. Proc. Camb. phil. Soc.* **87,** 151–155 (1980).
6. Voronoi, M. G. *J. reine angew. Math.* **134**, 198–287 (1908).
7. Delaunay, B. N. *Bull. Acad. Sci. USSR: Class. Sci. Math.* **7,** 793–800 (1934).
8. Fortune, S. in *Computing in Euclidean Geometry* (eds Du, D. Z. & Hwang, F.) (World Scientific, Singapore, 1992).
9. Barber, B., Dobkin, D. P. & Huhdanpaa, H. *Technical Rep. GCG53* (The Geometry Centre, Univ. Minesota, Minneapolis, 1993).
10. Fortune, S. *Algorithmica* **2,** 153–174 (1987).
11. Braun, J. & Sambridge, M. *Earth planet Sci. Lett.* **124,** 211–220 (1994).
12. Sibson, R. in *Interpreting Multivariate Data* (ed. Barnet, V.) 21–36 (Wiley, Chichester, 1981).
13. Watson, D. F. *Contouring: a Guide to the Analysis and Display of Spatial Data* (Pergamon, Oxford, 1992).
14. Finlayson, B. A: *The Method of Weighted Residuals and Variational Principles* (Academic, New York, 1972).
15. Sambridge, M., Braun, J. & McQueen, H. *Geophys. J. int.* (in the press).
16. Lasserre, J. B. *J. Opt. Theory Applic.* **39,** 363–377 (1983).
17. Bathe, K.-J. *Finite Element Procedures in Engineering Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1982).
18. Van Norton, R. in *Mathematical Methods for Digital Computers* Vol. 1 (eds Ralston, A. & Wilf, H. S.) (Wiley, New York, 1960).
19. Hughes, T. J. R., Winget, J., Levit, T. & Tedzuyar, T. E. in *Computer Methods for Non-linear Solids and Structure* (eds Atluri, S. N. & Perrone, N.) 75–109 (Am. Soc. Mech. Eng., New York, 1983).
20. Batchelor, G. K. *An Introduction to Fluid Dynamics* (Cambridge Univ. Press, 1967).
21. Fung, Y. C. *Foundations of Solid Mechanics* (Prentice-Hall, Englewood Cliffs, NJ, 1965).

SUPPLEMENTARY INFORMATION. Requests should be addressed to Mary Sheehan at the London editorial office of *Nature*.

ACKNOWLEDGEMENTS. We thank H. McQueen, K. Lambeck and D. Watson for stimulating discussions, U. Christensen for his review and D. Whitehouse for producing the 3D ray-traced pictures.