RC24141 (W0612-073) December 19, 2006
Mathematics

IBM Research Report

Higher Dimensional Continuation

Michael E. Henderson
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Higher Dimensional Continuation

Michael E. Henderson

IBM T.J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
mhender@us.ibm.com

1 Statement of the problem

Numerical continuation in one dimension produces a sequence of points $(\mathbf{u}_i, \lambda_i)$ $i \in [0, N)$, along a set of arcs that are connected at their endpoints. Each point is the solution of an equation

$$F(\mathbf{u}_i, \lambda_i) = 0 \qquad\qquad F : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n \tag{1}$$

A *regular point* is a solution $F(\mathbf{u}_0, \lambda_0) = 0$ where the Jacobian $F_{\mathbf{u}}(\mathbf{u}_0, \lambda_0)$ is non-singular. In a small neighborhood of a regular point there is a unique curve $(\mathbf{u}(s), \lambda(s))$ of regular points in some small neighborhood $|s| < \epsilon$. This is the Implicit Function Theorem (IFT). By extending this small piece of the solution curve, detecting singular points and switching branches at singular points, the continuation method produces an approximation of the *connected component* $\Gamma_{(\mathbf{u}_0,\lambda_0)}$ of solutions of $F(\mathbf{u}, \lambda) = 0$ containing $(\mathbf{u}_0, \lambda_0)$. (Fig. 1, (left).)

Most physical problems depend on more than one parameter. While it can be useful to study how the solution depends on each parameter separately, what is really wanted is an understanding of the behavior over some finite piece of the parameter space. This is the higher dimensional continuation problem. Instead of solution curves, we wish to find *solution manifolds* of

$$F(\mathbf{u}, \lambda) = 0 \qquad\qquad F : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^n \tag{2}$$

Using the IFT again, at a regular point in there is a unique manifold of regular solutions $F(\mathbf{u}(\mathbf{s}), \lambda(\mathbf{s})) = 0$, which exists in some small neigborhood $|\mathbf{s}| < \epsilon$, $\mathbf{s} \in \mathbb{R}^k$. The connected component $\Gamma_{(\mathbf{u}_0,\lambda_0)}$ is as defined above, but it is a branched manifold (the solution manifold), rather than a branched curve. (Fig. 1 (right).) Singular points on the manifold where the Jacobian is rank deficient by one form the boundary of each branch. These singular manifolds are generically $k - 1$ dimensional sub–manifolds.

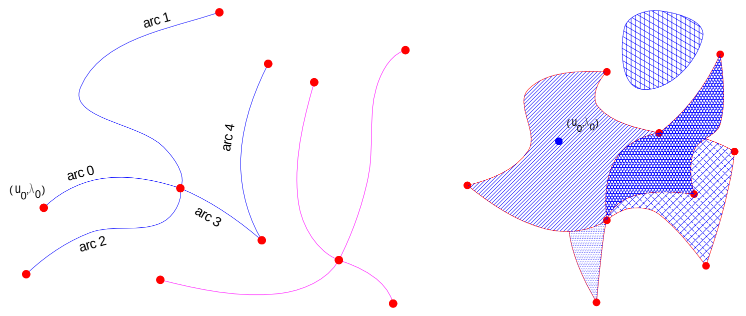


Fig. 1. (left) A connected component $\Gamma_{(\mathbf{u}_0,\lambda_0)}$ consisting of a set of smooth arcs, which meet at the endpoints of the arcs, which are singular points. $F(\mathbf{u},\lambda) = 0$ may consist of more than one connected component. The blue branched curve is part of $\Gamma_{(\mathbf{u}_0,\lambda_0)}$ and the magenta branched curve is a part of a different connected component. (right) A branched manifold consists of a set of smooth manifolds with boundaries, which meet along shared boundary manifolds.

Numerical continuation methods use a local analysis, usually a computational version of the IFT, which approximates the solution manifold in a neighborhood of a regular point $\mathcal{N}_i(\mathbf{u}_i)$, and aggregate n of these neighborhoods into a global approximation of the solution manifold M, M_n. The geometric problem of maintaining the aggregate of neighborhoods, and merging a new neighborhood into the aggregate, is roughly equivalent to advancing front mesh generation on a surface. This is not an easy problem, especially for higher dimensions n and manifolds of dimension k more than three. In Section 2 we give a brief survey of the methods used to represent and manipulate *complexes*, which are the generalization of meshes, and are used to represent the branched manifold which approximates M. In Section 3 we describe five algorithms which are in the literature, and try to classify them by which of the representations discussed in section 2 are used. And finally in section 4 we show the results of the five algorithms applied to a sphere.

2 Background

In computational and pure mathematics it is necessary to chose a representation or notation for the object being computed or manipulated. The parallel between notation and computational representation is quite close, though the computer can deal with expressions far too complicated to deal with by hand. A good notation makes manipulations easier and reduces errors, and a good computational representation makes operations easier to implement, and reduces coding errors. The choice of representation is not often not made explicitly, especially when the objects are simple.

The choice of representation also serves to distinguish between algortihms, since often once the choice of a representation is made there are fewer choices of how to proceed. We use this approach in Section 3 to analyze five algorithms for higher dimensional continuation in the literature.

For one dimensional continuation many issues simply do not arise: curves are easily represented as lists of points. Surfaces and manifolds are more challenging. In section 2.1 we discuss simplicial and cell complexes, which are general meshes that are commonly used in algebraic topology. Then we define manifolds and manifolds with boundary, which are the analogues of curves and arcs in one dimension.

Section 2.2 describes two fundemental geometric abstractions, the Voronoi and Delaunay tesselations, and a particular generalization of the Voronoi which represents the boundary of a union of spherical balls. In section 2.3 we discuss how these topological and geometrical ideas have been used to represent manifolds, and finally two to construct the neighborhood of a regular point.

2.1 Cell and Simplicial Complexes, and Manifolds

There are only a few ways to deal with general surfaces, and most are variations on meshes In this section we discuss representations of cell and simplicial complexes, which are general meshes in higher dimensions. The Voronoi diagram and Delaunay triangulation are two instances of complexes with special properties. The Voronoi tessellation contains information about points that are near each other, and there is a variant, the Laguerre–Voronoi diagram, which provides an efficient means of determining the boundary of a union of spherical balls. The Delaunay triangulation is related to the Voronoi diagram, and is frequently used to generate meshes in two and three dimensions.

This subject can be presented in a very opaque, but abstract way. The author would recommend the book [14] and the paper [16] for reasonable clear exposition of complexes, and hopes that the descriptions which follow are as clear.

A *cell complex* of dimension k is a set of cells of dimension 0 to k. The 0–cells (vertices) may (or may not) be identified with points in an n dimensional embedding space. If they are not embedded the complex represents purely topological information. For our purposes the vertices will be points on or near the solution manifold. Each p–cell (except the 0-cells) has a set of faces, which are $(p-1)$–cells in the complex. Cells must be *compatible* with each other; the intersection of two cells is either empty, or a cell in the complex that both cells have as a common face. This compatiblilty condition turns out to be a major difficulty for some of the algorithms described in Section 3 below. Figure 2 shows some cells that are not compatible.

A complex is a set of lists; of cells of each dimension, the faces of each cell with orientations, and a list of cells of which the cells is a face. Some of this information is redundant, and when storage is an issue it is necessary to avoid storing redundant information. Of course, if information is not stored, it must be computed from available information. Guibas and Stolfi [18] describe

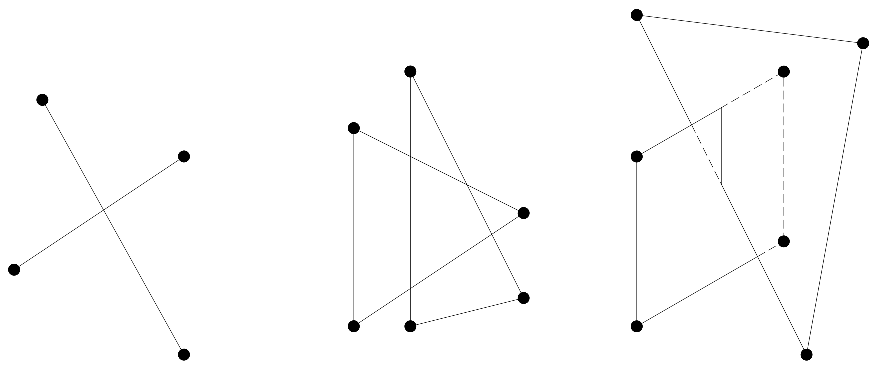


Fig. 2. Cells that are not compatible. (left) 1–cells embedded in $\mathbb{R}^2$, (center) 2–cells embeded in $\mathbb{R}^2$, and (right) 2–cells embedded in $\mathbb{R}^3$.

a minimal data structure for representing two dimensional cell complexes (in general, not just polyhedral or simplical), and show that with operations which are fairly simple in this representation, a Delaunay triangulation algorithm can be built (see section 2.3). For three dimensional complexes there are similar representations, e.g. the "winged edge" (see for example [13], [19]).

The basic geometrical object in higher dimensional continuation is a k–dimensional branched manifold (recall that k is the number of parameters). A branched manifold is a set of *manifolds with boundaries* (the branches), glued together along common boundaries. The boundaries are sets of singular points, which are generically $(k-1)$–dimensional manifolds. Finding the branches which meet at a singular surface is called branch switching.

Convex Polyhedral Cells

Complexes with convex polyhedral cells have a particularly nice structure. The cells are formed by intersecting half spaces whose planar boundaries contain the faces of the polyhedron and can be represented by the list of planes that they lie on. So for example, the set of linear inequalities in three dimensions:

$$P = \begin{cases} (\ \ \hat{\mathbf{e}}_0, -1).(\mathbf{v}, 1) \le 0 \\ (-\hat{\mathbf{e}}_0, \ \ 1).(\mathbf{v}, 1) \le 0 \\ (\ \ \hat{\mathbf{e}}_1, -1).(\mathbf{v}, 1) \le 0 \\ (-\hat{\mathbf{e}}_1, \ \ 1).(\mathbf{v}, 1) \le 0 \\ (\ \ \hat{\mathbf{e}}_2, -1).(\mathbf{v}, 1) \le 0 \\ (-\hat{\mathbf{e}}_2, \ \ 1).(\mathbf{v}, 1) \le 0 \end{cases} \tag{3}$$

defines a polyhedron (a cube). The representation of a vertex, $\mathbf{v}_0 = (0, 2, 4)$ for example, means that the point $\mathbf{v}_0$ satifies the linear system:

$$\begin{array}{l} (\ \ \hat{\mathbf{e}}_0, -1).(\mathbf{v}_0, 1) = 0 \\ (\ \ \hat{\mathbf{e}}_1, -1).(\mathbf{v}_0, 1) = 0 \\ (\ \ \hat{\mathbf{e}}_2, -1).(\mathbf{v}_0, 1) = 0 \end{array} \tag{4}$$

The list of planes that contain each vertex is sufficient to define the rest of the cells. In a k–dimensional polyhedron each vertex must lie on k or more planes. Cells of dimension $p > 0$ must lie on exactly $k-p$ faces, and so have $k-p$ indices. There is an edge between two vertices if the vertices have $k-1$ indices in common. For example:

$$\mathbf{v}_0 \cap \mathbf{v}_1 = (0,2,4) \cap (0,2,5) = (0,2) \mathbf{v}_3 \cap \mathbf{v}_4 = (0,3,4) \cap (1,2,4) = (4) \quad (5)$$

So there is an edge with endpoints $\mathbf{v}_0$ and $\mathbf{v}_1$, but not $\mathbf{v}_3$ and $\mathbf{v}_4$. The same

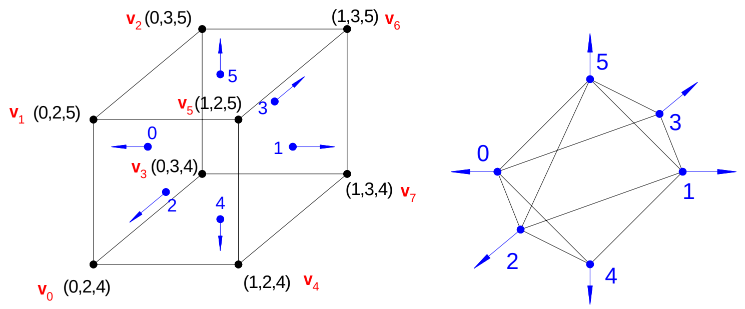


Fig. 3. A cube with the faces and vertices labeled (left). At the right is the dual of the boundary of the cube, which is an octahedron.

holds true for cells of higher dimension. A p–cell exists containing a set of vertices if and only if the intersection of the index sets of all the vertices contains $k-p$ indices. This works because each plane is one linear constraint, and a set of indices is just a linear system. In $\mathbb{R}^k$, $k-1$ linear equations defines a line. For more on polyhedra see for example [17] or [12]. This notation (in a different guise) is introduced in section 2.6 of [17].

Duality

We expressed the halfspaces in Eq. 4 in terms of an inner product of a point in $\mathbb{R}^{k+1}$ and a point in $\mathbb{R}^k \times \mathbb{R}$. The additional coordinate is a homogeneous coordinate, so that $(\mathbf{v}, h)$ corresponds to $\mathbf{v}/h$. This highlights the duality that exists in cell complexes. A plane is represented by an $\mathbb{R}^{k+1}$ vector, which is a linear functional acting on points in the space in which the vertices are embedded (in homogeneous coordinates). (Fig. 3.) So, for example, the dual of the 0–cell (1) is the set of three linear functionals (points) $(\hat{\mathbf{e}}_0, -1)$, $(\hat{\mathbf{e}}_1, -1)$, and $(\hat{\mathbf{e}}_2, 1)$, or $\overline{(0,2,5)}$. This is one of the triangles in the octahedron shown in Figure 3, which is the dual of the cube. The dual of a cell is the intersection of the index sets for the vertices in the cell, so the dual of 1–cell $(4,5)$ is $(1,2,4) \cap (1,2,5) = \overline{(1,2)}$. This duality relates the Voronoi diagram and Delaunay triangulation described below, and is called Poincaré duality.

p	cell	dual	p	cell	dual	p	cell	dual
2	(0,1,2,3)	$\overline{(0)}$	1	(0,1)	$\overline{(0,2)}$	0	(0)	$\overline{(0,2,4)}$
2	(4,5,6,7)	$\overline{(1)}$	1	(0,3)	$\overline{(0,4)}$	0	(1)	$\overline{(0,2,5)}$
2	(0,1,5,4)	$\overline{(2)}$	1	(2,3)	$\overline{(0,3)}$	0	(2)	$\overline{(0,3,5)}$
2	(3,2,6,7)	$\overline{(3)}$	1	(1,2)	$\overline{(0,5)}$	0	(3)	$\overline{(0,3,4)}$
2	(1,2,6,5)	$\overline{(4)}$	1	(4,5)	$\overline{(1,2)}$	0	(4)	$\overline{(1,2,4)}$
2	(0,3,7,4)	$\overline{(5)}$	1	(4,7)	$\overline{(1,4)}$	0	(5)	$\overline{(1,2,5)}$
			1	(5,6)	$\overline{(1,5)}$	0	(6)	$\overline{(1,3,5)}$
			1	(0,4)	$\overline{(2,4)}$	0	(7)	$\overline{(1,3,4)}$
			1	(1,5)	$\overline{(2,5)}$			
			1	(3,7)	$\overline{(3,4)}$			
			1	(2,6)	$\overline{(3,5)}$			

Table 1. The cells of a cube and their duals.

Subtracting a halfspace from a convex polyhedron

One operation on convex polyhedra requires mention, and that is intersecting the polyhedron with a halfspace. This adds a plane to the polyhedron, and the vertices that are the "wrong" side of the plane and any cell containing those vertices must be removed. If the convex polyhedron is represented by a list of inequalities and a list of vertices (the 0–cells and dual 0–cells), then the algorithm described in [12] may be used.

The algorithm is simple. First the halfspace is added to the list of inequalities, then the vertices are each tested for inclusion in the half space. Those that are not in the halfspace are removed and a new vertex is added for each edge containing the deleted vertex. The index of these new points is the index of the edge, plus the index of the new inequality. The performance of the algorithm can be improved by storing the edges.

Simplices

A *simplex* of dimension k is a set of $k+1$ vertices that do not lie in a linear subspace of $\mathbb{R}^k$. A *simplicial complex* is a cell complex whose cells are all simplices. Simplices are represented as dual polyhedra, but for simplices the cells can be enumerated by simple operations on their vertices, so do not have to be tabulated. A 2 dimensional simplex (a triangle) with by vertices 1, 4, and 5 is represented as $(1,4,5)$. The natural ordering of the integers determines the order in which the vertices are listed.

The $(k-1)$–cells lying in a k dimensional simplex are simplices whose representation is the same as that of the "parent" simplex, but with one vertex dropped. The $(k-1)$–cell is the one opposite the vertex that is dropped. (Fig. 4.) This enumeration holds for cells down to 0–simplices, which are points, and have a single index.

The dual of a simplex is the list of missing vertices. (Table 2.)

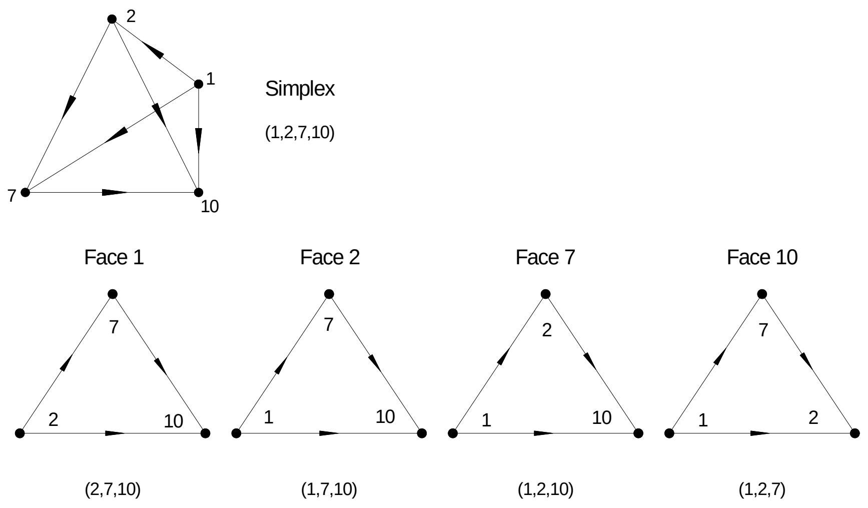


Fig. 4. The simplex $(1, 2, 7, 10)$ and it's faces. The faces can be enumerated by removing from the label of the simplex the index of each vertex in turn. Removing a vertex index yields the label of the face opposite that vertex. The arrows indicate the positive ordering of the vertices on the edges.

Orientation and the boundary and co–boundary operators

Simplices (and cells in general), have an orientation of ± 1. If two adjacent vertices in the representation of a simplex are interchanged, the sign changes, with the simplex with vertices in order being positive. So for an arbitrary list of vertices the sign is the parity of the permutation which orders the vertices.

The *boundary operator* d_p maps a p–cell into a sum of $(p-1)$ cells (the faces of the cell). Each cell in the sum, has an orientation of ± 1, assigned so that the boundary of the boundary is empty (Fig. 5). For simplices there is a simple expression for the boundary operator.

$$d_p(v_0, ..., v_i, ..., v_p) = \sum_{i=0}^{p} (-1)^i (v_0, .., v_{i-1}, v_{i+1}, ...v_p) \tag{6}$$

(Figure 5.) For example, the face $(2, 7, 10)$ in the simplex $(1, 2, 7, 10)$ (a tetrahedron) has boundary

$$\begin{aligned} d_3(2, 7, 10) &= (7, 10) - (2, 10) + (2, 7) \\ d_2 d_3(2, 7, 10) &= \{(10) - (7)\} - \{(10) - (2)\} + \{(7) - (2)\} = 0 \end{aligned} \tag{7}$$

The *co-boundary operator* d^p is the adjoint of the boundary operator d_p. The co–boundary operator is the boundary of the dual. If σ_p is a p–cell, and f^p

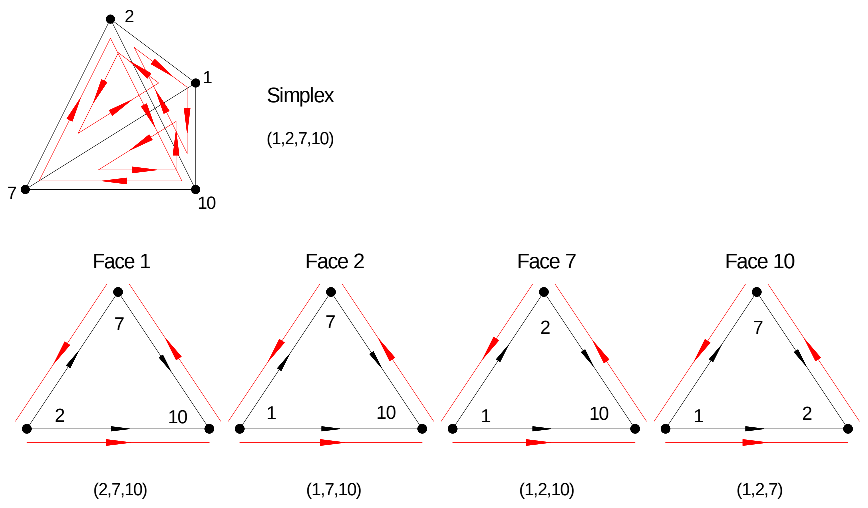


Fig. 5. The Simplex $(1, 2, 7, 10)$ and it's faces oriented so that the boundary of it's boundary is empty.

is a set of $p+1$ dual vertices, the the co–boundary is related to the boundary operator by

$$(d^p f^p)(\sigma_{p+1}) = f^{p+1}(d_{p+1}\sigma_{p+1}), \quad \textit{or simply} \quad d^p f^p = f^{p+1} d_{p+1} \qquad (8)$$

The co–boundary operator increases the dimension of a simplex (decreases the dimension of a dual simplex), while the boundary operator decreases the dimension of a cell. For a simplex there is an expression for the boundary operator d_{p+1}, so the co–boundary operator can be written explicitly as well:

$$d^p f^p = f^{p+1} d_{p+1} = \sum_{i=0}^{p+1} (-1)^i (v_0, .., v_{i-1}, w_i, v_i, ...v_p) \qquad (9)$$

Here the w_i is in turn each of the vertices that are missing from the index of the cell. For a simplical complex w_i ranges over all vertices that make the simplex in the sum a cell in the complex. That is, it indexes the $p+1$ cells which have the p–cell as a face. For cell complexes the boundary and co–boundary need to be tabulated.

The boundary of a simplex is a simplicial complex, so we can use the boundary of our simplex $(1, 2, 7, 10)$ to give a concrete example of a simplicial complex:

One of the advantages of this notation is that two simplicial complexes can be added. The example in Fig. 7 removes the face they share from the boundary. The result is still a simplicial complex, but is the boundary of an octahedron. This makes it possible to perform such operations as adding a "handle" to a cell complex [28]. For simplices there is no need to represent

p	σ_p	σ^{k-p+1}	$d_p\sigma_p$	$d_{k-p+1}\sigma^{k-p+1}$	$d^p\sigma_p$
0	(10)	$\overline{(1,2,7)}$		$\overline{(2,7)} - \overline{(1,7)} + \overline{(1,2)}$	(1,10)-(2,10)+(7,10)
0	(7)	$\overline{(1,2,10)}$		$\overline{(2,10)} - \overline{(1,10)} + \overline{(1,2)}$	(1,7)-(2,7)+(7,10)
0	(2)	$\overline{(1,7,10)}$		$\overline{(7,10)} - \overline{(1,10)} + \overline{(1,7)}$	(1,2)-(1,7)+(1,10)
0	(1)	$\overline{(2,7,10)}$		$\overline{(7,10)} - \overline{(2,10)} + \overline{(2,7)}$	(1,2)-(2,7)+(2,10)
1	(7,10)	$\overline{(1,2)}$	(10)-(7)	$\overline{(2)} - \overline{(1)}$	(1,7,10)-(2,7,10)
1	(2,10)	$\overline{(1,7)}$	(10)-(2)	$\overline{(7)} - \overline{(1)}$	(1,2,10)-(2,7,10)
1	(2,7)	$\overline{(1,10)}$	(7)-(2)	$\overline{(10)} - \overline{(1)}$	(1,2,7)-(2,7,10)
1	(1,10)	$\overline{(2,7)}$	(10)-(1)	$\overline{(7)} - \overline{(2)}$	(1,2,10)-(1,7,10)
1	(1,7)	$\overline{(2,10)}$	(7)-(1)	$\overline{(10)} - \overline{(2)}$	(1,2,7)-(1,7,10)
1	(1,2)	$\overline{(7,10)}$	(2)-(1)	$\overline{(10)} - \overline{(7)}$	(1,2,7)-(1,2,10)
2	(2,7,10)	$\overline{(1)}$	(7,10)-(2,10)+(2,7)		
2	(1,7,10)	$\overline{(2)}$	(7,10)-(1,10)+(1,7)		
2	(1,2,10)	$\overline{(7)}$	(2,10)-(1,10)+(1,2)		
2	(1,2,7)	$\overline{(10)}$	(2,7)-(1,7)+(1,2)		

Table 2. The simplicial complex which is the surface of the simplex $(1, 2, 7, 10)$.

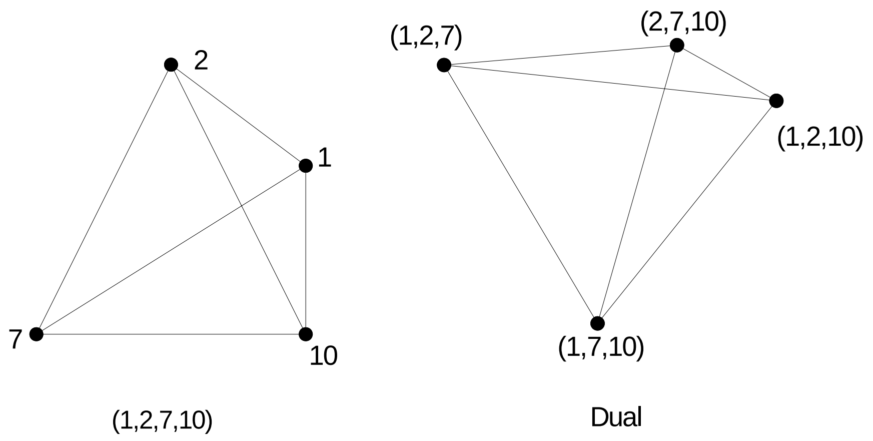


Fig. 6. The dual (right) of the boundary of the simplex $(1, 2, 7, 10)$.

the faces, since they can be generated from the vertex list. For a simplicial complex the dual edges to each face of a simplex must be stored (i.e. the pair of simplices which share the faces). For a cell complex there needs to be a data structure for storing the lists associated with each cell.

2.2 Manifolds

A manifold is the generalization of a surface, but has a lot of the freatures of a cell complex. A k–dimensional manifold is a set of k–dimensional neighborhoods of the origin (called charts) which are isomorphic to the k–dimensional

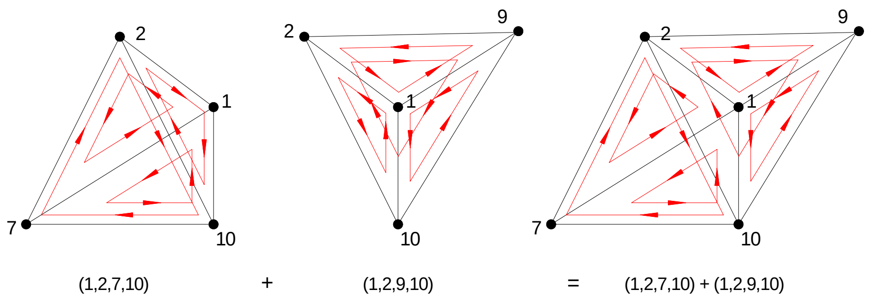


Fig. 7. The sum of the boundaries of two simplices $(1, 2, 7, 10)$ and $(1, 2, 9, 10)$.

unit ball $B_1(0) = \{x \mid |x| < 1\}$, along with adjacency relations indicating which charts "overlap". Two adjacent charts must agree on some common non-empty sub–region, which means that there is a one–to–one and onto mapping from the subregion of one chart to the corresonding subregion of the other (Fig. 8). The neighborhoods are called *chart domains*, the collection of charts is called an *atlas*. The reference is to navigational charts which cover a small piece of the globe, and are bound together into an atlas. In this analogy the overlaps are needed to move from one chart to the next, and the adjacency is usually indicated by the number of the neighboring chart in the margin. (Fig. 8.)

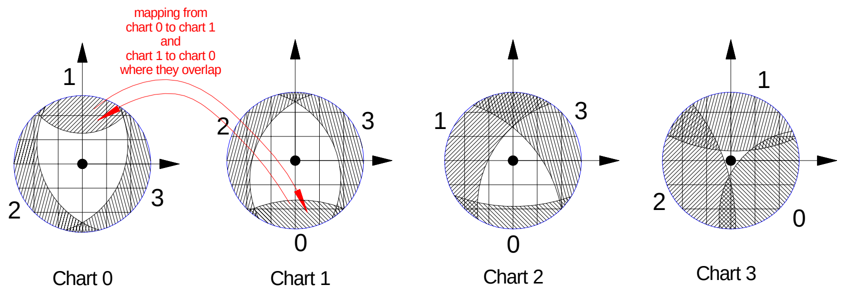


Fig. 8. A manifold is an "atlas" of overlapping "charts", with an identification of overlapping subregions and a one to one and onto mapping between the subregions.

The relation of a manifold to a cell complex is straightforward. A chart is a k–cell, the overlap between two charts is a $k - 1$ cell and a common face of each chart, and so on. With this interpretation the manifold shown in Fig. 8 is a tetrahedron.

Manifolds with Boundary

A manifold with boundary has special charts called *boundary charts.* Instead of a full neighborhood of the origin, the domain of a boundary chart is isomorphic to the half ball $B_1(0) = \{x \mid |x| \leq 1 \;\; x_0 \geq 0\}$. The restriction to the ball $B_1(0) = \{x \mid |x| \leq< 1, \;\; x_0 = 0\}$, a full ball of one dimension less, is a chart on one of the *boundary manifolds.* The boundaries themselves can have boundaries (just as the edges of a square have end points), so instead of a half ball we really mean a ball restricted so that some set of the coordinates are non–negative. (Fig. 9.)

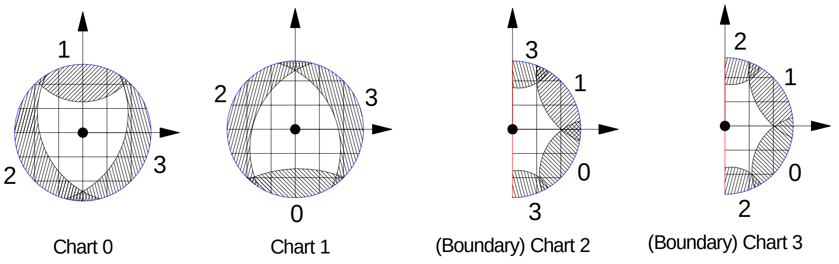


Fig. 9. A manifold is an "atlas" of overlapping "charts", with an identification of overlapping subregions and a one to one and onto mapping between the subregions.

This is a second way that manifolds with boundary are like a cell complex. The manifold itself is a cell, and the faces of cell are the boundary manifolds.

Embedding a Manifold

The solution manifold M of $F(\mathbf{u}, \lambda)$ is a set of points in $\mathbb{R}^n$, but so far the definition of a manifold has been in terms of neighborhoods of the origin in $\mathbb{R}^k$. The relation between a k–dimensional manifold and the solution space $\mathbb{R}^n$ is an embedding of the manifold (if n is large enough so that the manifold does not cross itself). Each chart is assigned a one to one mapping to $\mathbb{R}^n$, called the *chart mapping*, which must map points in the overlap of two charts to the same point in the embedding space. With an embedding the mapping between the overlap of two charts can be expressed using the chart mapping of one and the inverse of the chart mapping of the other. For a manifold defined by a system of equations we considered branches which consist of regular points, and the implicit function theorem gives the chart mapping. The singular boundary manifolds are the where this embedding fails. (Fig. 10.)

Representing a Manifolds

The algorithms described in Section 3 use different representations of the solution manifold. Allgower and Schmidt's algorithm [3] represents the solution

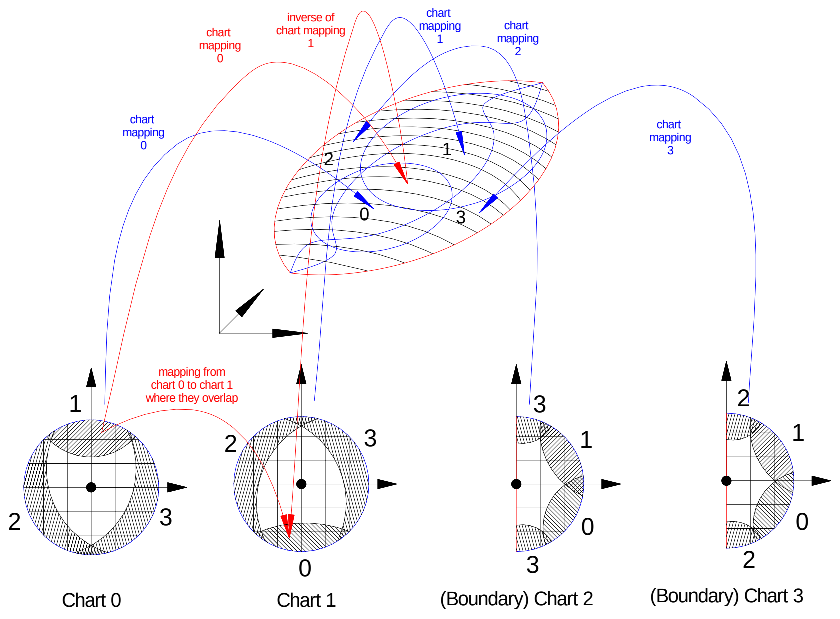


Fig. 10. An embedded manifold with boundary.

manifold as a set of simplices in the embedding space $\mathbb{R}^n$, each of which contains a piece of the manifold. Rheinboldt's moving frame algorithm [31], Brodzik's algorithm [9] and Melville and Mackey's boundary representation [29] instead use a simplicial complex whose vertices are points on the solution manifold. The author's algorithm [20] represents the solution manifold as a complex with convex polyhedral cells. These are the three representations that are used by the five algorithms described in secion 4. (Fig. 11.)

2.3 Basic Computational Geometry

The representations of manifolds which are used for higher dimensional continuation are all based on cell or simplicial complexes. In computational geometry two of the most frequently used complexes are the Delaunay triangulation of a set of points and the dual Voronoi diagram. Of course, the Delaunay triangulation in higher dimensions isn't a triangulation, but it still is refered to as a triangulation. Generically, the Delaunay triangulation is a simplicial complex, and the Voronoi diagram is a complex with convex polyhedral cells. There is a large literature on both but the reader might begin with [7], [15], [34], [11], [8], or any introductory text on computational geometry.

The Delaunay triangulation has good properties for mesh cells (it creates "fat" simplices). In two dimensions it has been proven [30] that over all triangulations of a fixed set of points, the Delaunay triangulation is the one the

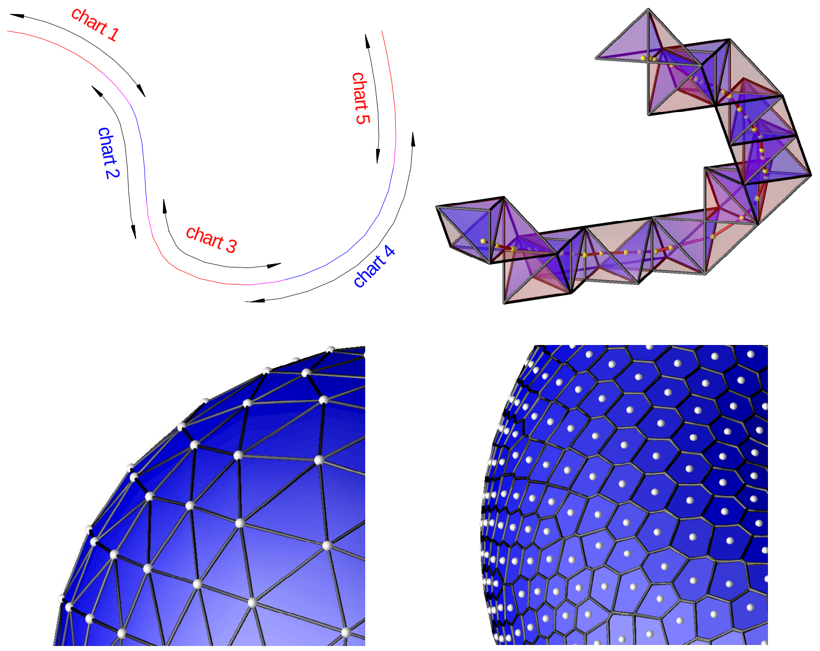


Fig. 11. The four main ways of representing a manifold. (upper left) The definition, a list of overlapping chart mappings from $\mathbb{R}^k$ to $\mathbb{R}^n$, in this case $k = 1$, $n = 2$. (top right) A covering, or containment in solution space, $k = 1$, $n = 3$. (lower left) A triangulation $k = 2$, $n = 3$, and (lower right) A polygonal tiling $k = 2$, $n = 3$.

maximizes the smallest angle in any of the triangles. The Voronoi diagram contains information about "nearest neighbors", so is used in many pattern matching applications.

Voronoi Diagrams

Given a set of points $\mathbf{u}_i$ in $\mathbb{R}^n$, the Voronoi diagram of the points is a decomposition of $\mathbb{R}^n$ into n–cells, each associated with one of the points. The Voronoi cell V_i of $\mathbf{u}_i$ is

$$V_i = \{\mathbf{u} \in \mathbb{R}^n \mid |\mathbf{u} - \mathbf{u}_i| < |\mathbf{u} - \mathbf{u}_j| \text{ for all } j \neq i\} . \qquad (10)$$

Here $|\cdot|$ is the Euclidean 2-norm. Figure 12 shows an example of a Voronoi diagram for a set of points in the plane.

The Voronoi regions are a "domain of influence" of the points. They also give information about the nearest neighbors of each point (the $n-1$ dimensional faces separate the Voronoi regions of two nearby charts).

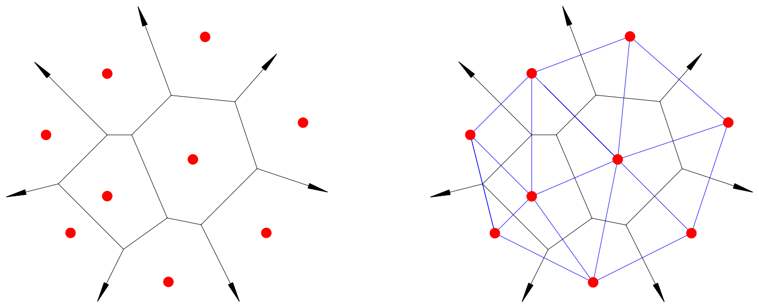


Fig. 12. (Left) the Voronoi diagram of a set of points (red) in the plane. Each region is the set of points closest to a particular point. (Right) the duality of the Voronoi (black) and Delaunay diagrams (blue).

The Laguerre–Voronoi Diagram and the boundary of a union of spherical balls

There are several generalizations of the Voronoi diagram [24]. One of these, the Laguerre–Voronoi diagram, or power diagram, [5], [4], and [22], [6], can be used to find points on the boundary of a union of spherical balls, an operation which will be refered to in Section 3. In the Laguerre-Voronoi diagram each point is given a weight, which is the radius of the spherical ball about the point.

The face between two Laguerre–Voronoi cells is defined by the equation

$$|\mathbf{s}-\mathbf{s}_i|^2 - R_i^2 = |\mathbf{s}-\mathbf{s}_j|^2 - R_j^2 \tag{11}$$

which is easily solved. We find that points $\mathbf{s}$ on this Voronoi face are solutions of

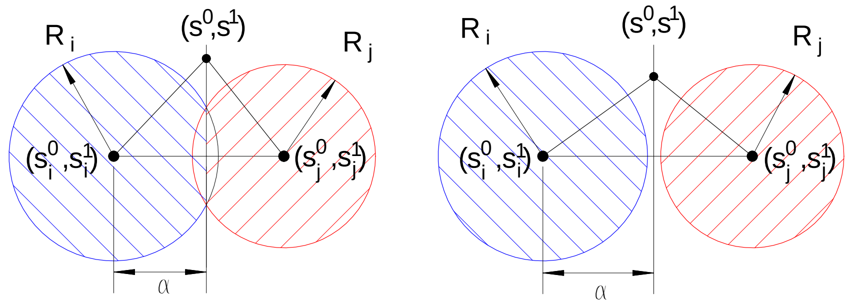
$$2(\mathbf{s}_j-\mathbf{s}_i).\mathbf{s} = R_i^2 - R_j^2 + |\mathbf{s}_i|^2 + |\mathbf{s}_j|^2 \tag{12}$$

which is a plane orthogonal to the line connecting the centers of the two cells. So the Laguerre–Voronoi diagram has cells with planar faces. Substituting $\mathbf{s} = \mathbf{s}_i + \alpha(\mathbf{s}_j - \mathbf{s}_i)$, we find that the planes intersects the line between the two points at

$$\alpha = \frac{R_i^2 - R_j^2 + |\mathbf{s}_i - \mathbf{s}_j|^2}{2\ |\mathbf{s}_i - \mathbf{s_j}|^2} \tag{13}$$

If $|\mathbf{s}_i - \mathbf{s}_j| \leq R_i + R_j$, the plane contains the intersection of the two spheres, and divides the part of the first sphere that lies outside the second from the part that lies inside the second (and vice versa). (Fig. 13). The n–cell is the intersection of the halfspaces defined by each pair of points, so is a convex polyhedron, and the boundary of the union of balls which define the diagram is the union of the parts of each sphere that lie inside the corresponding Laguerre–Voronoi n–cell.

The boundary of a union of spherical balls is the union of the parts of the spherical boundaries that do not lie inside another ball. So the boundary of



$dB_i - B_j$:

$$\begin{array}{l} (s^0 - s^0_i)^2 + (s^1)^2 = R^2_i \\ (s^0 - s^0_j)^2 + (s^1)^2 \le R^2_j \end{array} \quad \equiv \quad \begin{array}{c} (s^0 - s^0_i)^2 + (s^1)^2 = R^2_i \\ 2(s^0_i - s^0_j)s^0 + (s^0_j)^2 - (s^0_i)^2 \le R^2_j - R^2_i \end{array}$$

Fig. 13. (left) The Laguerre–Voronoi face when $(s^0_i - s^0_j)^2 \le R_i + R_j$, and (right) when $(s^0_i - s^0_j)^2 > R_i + R_j$.

the union consists of the parts of the spheres which lie inside their restricted–Laguerre Voronoi diagram. (Fig. 14.)

The Delaunay Triangulation

The Delaunay triangulation is the dual of the Voronoi diagram in the sense discussed above for complexes (Fig. 15). In two dimensions it has been shown [30] that Delaunay triangulations, among all possible triangulations of a set of points, maximizes the smallest angle in any triangle. In three dimensions there appears to be a similar property, but it is not clear exactly what maximum principle there might be. Because of this property of having "fat" cells, the Delaunay triangulation is widely used for mesh generation. For an more complete explanation of the properties of Delaunay triangulations, and how to they are constructed see [15], [18], [7] and [33].

The Delaunay tessellation can be defined independantly of the Voronoi diagram by means of the "in–circle" test. Each n-cell of the triangulation is such that no point is inside the Euclidean sphere which contains the vertices of the n-cell. (Fig. 15.)

The Coxeter–Freudenthal–Kuhn Triangulation

The Coxeter–Freudenthal–Kuhn triangulation, or tessellation [2] is a simplicial decomposition of $\mathbb{R}^n$. (It is also described in [3], but with quite a few typographic errors.) The tessellation is defined in terms of an initial n-dimensional simplex, with $n+1$ vertices labeled v_i for $i = 0$ to n. An $n-1$ dimensional face

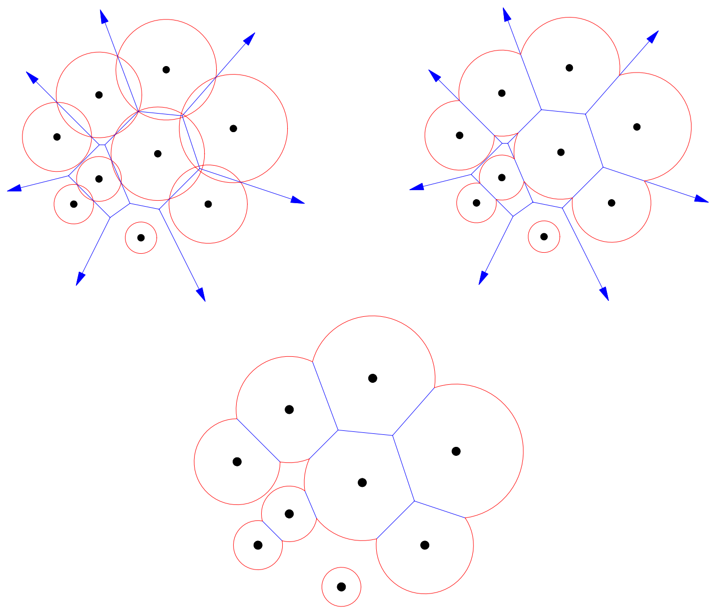


Fig. 14. (Top left) the Laguerre–Voronoi diagram of a set of points (red) in the plane. Each region is the set of points closest to a particular point, using the distance $|\mathbf{s}-\mathbf{s}_i|^2-R_i^2$. (Top right) the Laguerre–Voronoi diagram contains information about the boundary of the union of balls. The boundary of the union of balls is the union of the part of the boundary of each ball that lies within it's Voronoi region. (Bottom) Only the part of the Laguerre–Voronoi diagram that corresponds to overlapping balls is needed. This is the restricted Laguerre–Voronoi diagram.

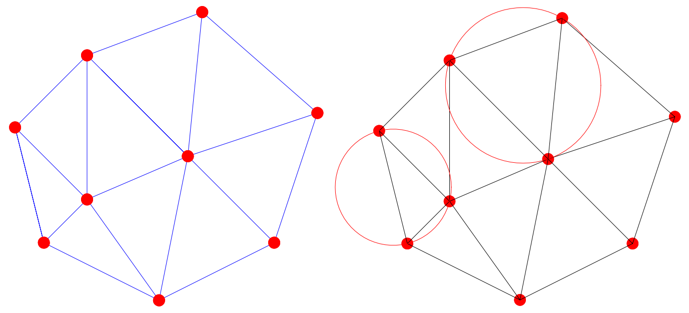


Fig. 15. (left) The Delaunay tessellation of a set of points in the plane. (right) Each simplex in the tessellation satisfies the “in circle” test. No point lies interior to a Euclidean sphere passing through the vertices of the simplex.

separates two n dimensionalsimplices which share the face. If one of the simplices has vertices $(v_0, v_1, ..., v_n)$ then in this triangulation the simplex across the face v_i is defined to be the simplex with vertices

$$P_{v_i}(v_0, v_1, ..., v_n) := (v_0, v_1, ..., v_{i-1}, \tilde{v}_i, v_{i+1}, ..., v_n) \tag{14}$$

where

$$\tilde{v}_i = \begin{cases} v_1 + v_n - v_0 & i = 0 \\ v_{i+1} + v_{i-1} - v_i & 0 < i < n \\ v_{n-1} + v_0 - v_n & i = n \end{cases} \tag{15}$$

This operation – of moving across a face to an adjacent simplex – is called a

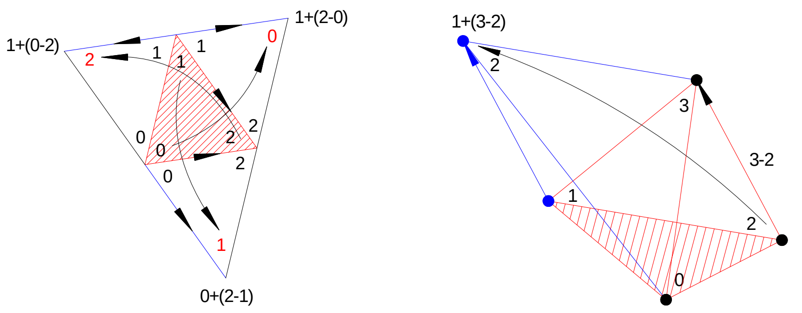


Fig. 16. (left) A pivot in the two dimensional Kuhn triangulation. (right) A pivot in the three dimensional Kuhn triangulation.

pivot.

This pivoting works for any initial simplex, but one particular choice has an explicit representation, and is well suited to computations. The initial simplex is a *path simplex*. A path simplex has vertices which are defined by the endpoints of the segments of a piecewise linear path. In particular, if the coordinate directions are used as segments, and no coordinate direction is allowed to appear twice, the set of path simplices decomposes the interior of the unit cube [25] (which attributes the idea to Tucker and Problem 3 on page 140 of [26]). (Fig. 17). It is trivial to tile $\mathbb{R}^n$ with cubes. So this decomposition of the cube also gives a simplicial decomposition of $\mathbb{R}^n$. This is not true for an arbitray decomposition of the cube, since translation in a coordinate direction will generate incompatibile faces unless opposite faces of the cube have the same decomposition.

If the coordinate of the “lower left ” corner of a cube (the corner which is closest to the origin) is $\mathbf{v}^0$, and the side of the cube is length δ, the simplices in this decomposition can be represented by a translation vector $\mathbf{z} \in \mathbb{N}^n$, and a permutation $\pi \in [0...n-1]^n$ (every number in $[0...n-1]$ appears once and only once in π). The permutation identifies the simplex in the decomposition of the cube. The vertices of the simplex with representation $(\mathbf{z}, \pi)$ are given by the recursion:

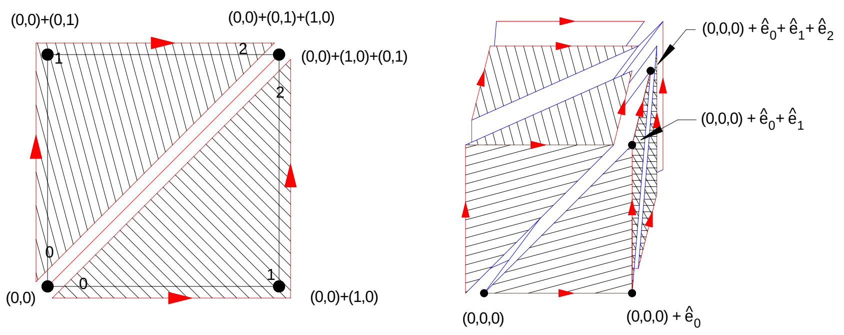


Fig. 17. (left) A path simplex decomposition of a square, and (right) of a cube. The simplices for the n dimensional cube are generated by paths with n segments, each segment parallel to a coordinate direction, with no direction repeated.

$$\begin{aligned}\mathbf{v}_0(\mathbf{z},\pi) &= \mathbf{v}^0 + \delta\mathbf{z} \\ \mathbf{v}_{j+1} &= \mathbf{v}_j + \delta\hat{\mathbf{e}}_{\pi_{j+1}}\end{aligned} \tag{16}$$

The effect of the pivot across face j ($\mathcal{P}_j$) on the permuation π is:

$$\mathcal{P}_j\pi = \begin{cases} \{\pi_0,...,\pi_{j-1},\pi_{j+1},\pi_j,\pi_{j+2},...,\pi_{n-1}\} & 0<j<n \\ \{\pi_1,...,\pi_{n-1},\pi_0\} & j=0 \\ \{\pi_{n-1},\pi_0,...,\pi_{n-2}\} & j=n \end{cases} \tag{17}$$

And the action on $\mathbf{z}$ is

$$\mathcal{P}_j\mathbf{z} = \begin{cases} \mathbf{z} & 0<j<n \\ \mathbf{z}+\hat{\mathbf{e}}_{\pi_0} & j=0 \\ \mathbf{z}-\hat{\mathbf{e}}_{\pi_{n-1}} & j=n \end{cases} \tag{18}$$

We now have the language and tools for describing algorithms for higher dimensional continuation. Cell and simplicial complexes will be used to the solution manifold, Delaunay triangulations for decompositions of the manifold, the restricted Laguerre–Voronoi tellessation to obtain information about the boundary of a collection of balls, and the Coxeter–Kuhn–Freudenthal tessellation for finding a set of simplices which encloses the solution manifold.

3 Five Algorithms

Below we describe five algorithms for higher dimensional continuation: Allgower and Schmidt's "pattern algorithm" [3], Rheinboldt's moving frame algortihm [31], Rheinboldt and Brodzik's $k = 2$ [10] and Brodzik's [9] general dimension tiling algorithms, Melville and Mackey's $k = 2$ boundary representation algorithm [29], and the author's covering algorithm [20].

All of these methods can be viewed as an iterative application of three steps, beginning with M_0 being the initial point:

1. Find a point $\mathbf{u}_i$ on the boundary of M_i
2. Build a neighborhood $\mathcal{N}_i$ of $\mathbf{u}_i$
3. Merge $\mathcal{N}_i$ into M_i to obtain M_{i+1}

The algorithms differ in how M_i is represented, and the three operations are performed.

Continuation in higher dimensions is conceptually different from one dimension. There are only two ways to represent a 1–manifold, and these correspond to Pseudo–arclength continuation [23], which uses a set of polygonal arcs to represent M, and simplical or piecewise–linear continuation [1], which uses n–dimensinal simplices in $\mathbb{R}^n$ to cover M.

Pseudo–arclength continuation exploits the fact that for 1–manifolds the boundary is a set of points, and the merge operation is just a matter of discarding one of the two intervals which make up the neighborhood of a point on M. If a simplicial approximation of M is used in higher dimensions the boundary of M_i is a $k-1$ dimensional simplicial complex. Constructing a simplicial neighborhood of a boundary point so that the simplices are compatible with M_i seems tractable, but turns out to be a difficult problem.

For 1–manifolds in $\mathbb{R}^{n+1}$ simplicial continuation uses an $n+1$ dimensional Coxeter–Kuhn–Freundenthal simplicial complex, and only requires pivots across n–cells. The is because a boundary "point" is the face of an $n+1$–dimensional simplex, and with $k = 1$, n–dimensional faces intersect M at a point (or not at all). There are exactly two simplices which contain this boundary face. One is the simplex on the boundary and the other can be found with the pivot operation. In higher dimensions the intersection of M with an $n+k$ dimensional simplex is a k–cell, and there are more than two simplices containing an n dimensional simplex which intersects M at a point.

It is easier to understand the algorithms when the manifold is flat, so we will make use of natural parameter continuation to present the five algorithms.

3.1 Natural Parameter Continuation

Let us begin with the case that there is a unique solution at each point in parameter space, and consider the generalization of natural parameter continuation. (Fig. 18). Natural parameter continuation is a fairly obviousmeans of adapting an iterative solver for $F(\mathbf{u}, \lambda)$ at a fixed parameter value to map out the solution manifold. Iterative methods require an initial guess, and rather than start with the same initial guess at each new parameter value, the solution at a nearby parameter value is used. (Fig. 18).

If a set mesh is used, natural parameter continuation selects a point from the mesh at which the solution is known, and which has a neighbor at which

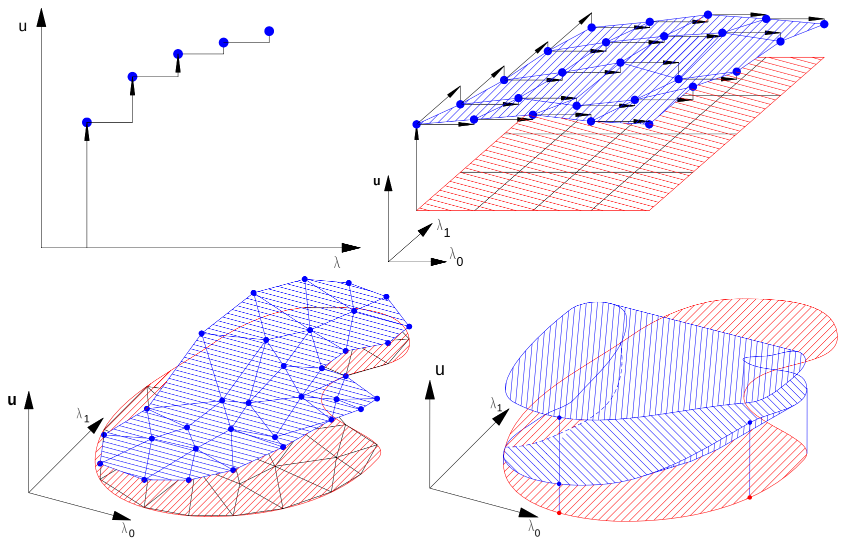


Fig. 18. (top left) Natural parameter continuation with one parameter. The solution at the point to the left is used as an initial guess for an iterative method at the next point. (top right) Natural parameter continuation with two parameters, using a rectangular grid on parameter space. The technique is the same as one parameter, but the grid vertices are travelled in a predetermined order. (bottomleft) Natural parameter continuation with two parameters, but with a triangular grid on the region of interest in parameter space. (bottom right) A case where natural parameter continuation fails.

the solution is not yet known. (Fig. 18) If the points in parameter space are chosen adaptively, the method becomes a generalization of advancing front mesh generation, which is a challenging problem in higher dimensions. The main consideration is that the new point lie on the edge of the previously computed points, but lies near enough so that one of the known points provides a good initial. This makes the boundary of the meshed region an important object.

3.2 Solution Space Continuation

For many interesting problems the solution manifold cannot be expressed as a function of the paramters. Fig. 18 shows such a case for $k = 2$. Natural parameter continuation would find the lower or upper sheet, depending on the initial guess, and the iteration would fail to converge beyond the fold.

When $k = 1$ there are two choices of algorithm. If n (the dimension of the solution space) is small simplicial continuation can be used. On the other hand, pseudo–arclength continuation may be used for any n.

Allgower and Georg's $k = 1$ simplicial continuation begins with a simplex in $\mathbb{R}^{n+1}$ which contains a point on the solution manifold, which is a branched curve. If the simplex is sufficiently small, and the point is a regular point, the curve enters this simplex through one face and leaves through another. With these assumption a linear approximation of F over the simplex results in a good approximation to these two points on the $n-1$ dimensional faces of the simplex. It is important that simplices are used, so that there is a unique, continuous piecewise linear interpolant. This also produces a piecewise linear solution curve. Figure 19 shows the effect of a second parameter on the piecewise linear solution of $F = 0$.

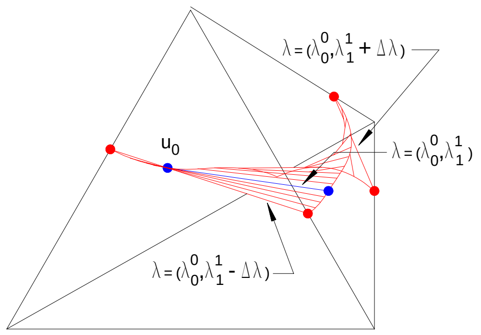

$\lambda=(\lambda_0^0,\lambda_1^1+\Delta\lambda)$
u_0
$\lambda=(\lambda_0^0,\lambda_1^1)$
$\lambda=(\lambda_0^0,\lambda_1^1-\Delta\lambda)$

Fig. 19. The solution of $F = 0$ for a piecewise linear interpolant with the second parameter fixed. As the second paramter is allowed to change the entry and exit points move and the line segment connecting them sweeps out a ruled surface.

Pseudo–arclength continuation (PSALC) [23] uses the tangent of the solution curve to define a new parameter. This is done by appending a constraint that the projection of a solution point $(\mathbf{u}(s), \lambda(s))$ onto the tangent vector is s. The curve of solutions is represented as a piecewise linear arc, and is extended by using one of the end points and a linear extrapolation for an intial guess to Newton's method for the system plus the pseudo–arclength constraint. (Figure 20 (left).) When the solution set is a closed curve, or isola, as shown on the right in Figure 20, the PSALC algorithm will repeatedly trace the isola. Usually an upper limit on the arclength is included in the conditions that terminate the algorithm. In higher dimensions we will call this the *self–intersection* problem, and it is related to the problem of incompatible simplices.

Figure 21 shows the effect of adding an additional parameter to PSALC. The tangent space is no longer one dimensional, and it is possible to make a step in more than one direction. The continuation algorithm must choose what direction to step while ensuring that the resulting points sample the solution manifold uniformly.

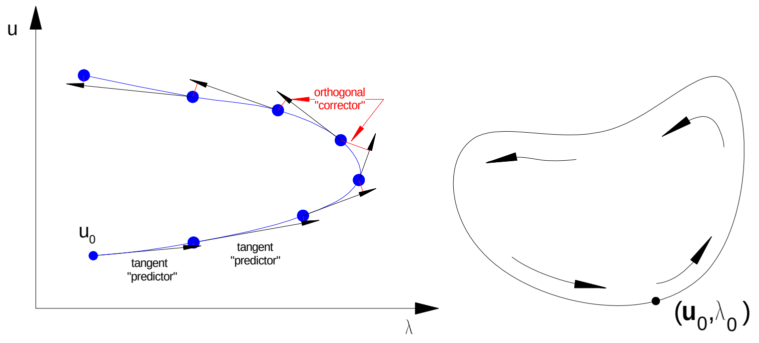


Fig. 20. (left) Pseudo arclength continuation. (right) An isola, which causes pseudo–arclength continuation to repeatedly trace the same curve. on the solution manifold M.

.

3.3 Local Analysis.

To build a neighborhood of M at a point $\mathbf{u} \in \mathbb{R}^n$ a local analysis is required. The operations used by the five algorithms are

- Find a basis for the k dimensional tangent space of M at a point $\mathbf{u}$,
- Project a point from the tangent space onto M,
- Estimate the size of a ball in the tangent space so that points within the ball project uniquely onto M.
- Determine if an n dimensional simplex contains a point on a piecewise linear approximation to M.

Finding a basis for the tangent space.

M is defined by the equation $F(\mathbf{u}) = 0$. If $\mathbf{u}$ is a point on M then the tangent space of M (Fig. 21) at $\mathbf{u}$ satisfies:

$$\begin{aligned} &F_{\mathbf{u}}(\mathbf{u})\Phi = 0 \qquad\qquad \text{or} \qquad\qquad F^i_{,p}\Phi^p_j = 0. \\ &\Phi^T\Phi = I \end{aligned} \tag{19}$$

Finding the tangent space means finding a basis for the nullspace of the Jacobian. At a regular point F^i_j is full rank, which is rank n, so the nullspace is k-dimensional. (Figure 22.) If n is moderately small a singular value decomposition or QR decomposition might be used. If the problem is large, one approach is to use the tangent space at a nearby point. This replaces the first Φ in the normalization, and results in the linear system:

$$\begin{bmatrix} F_{\mathbf{u}}(\mathbf{u}) \\ \\ \Phi_0^T \end{bmatrix} \Phi = \begin{bmatrix} 0 \\ \\ I \end{bmatrix} \tag{20}$$

The columns of Φ_0 are the orthonormal basis for the nearby tangent space. The columns of Φ will not be orthonormal, but Gram–Schmidt can easily be used to find an orthonormal basis from Φ. This linear system aligns the new tangent space as much as possible with the old, which will be used in Rheinboldt's wrapping algorithm.

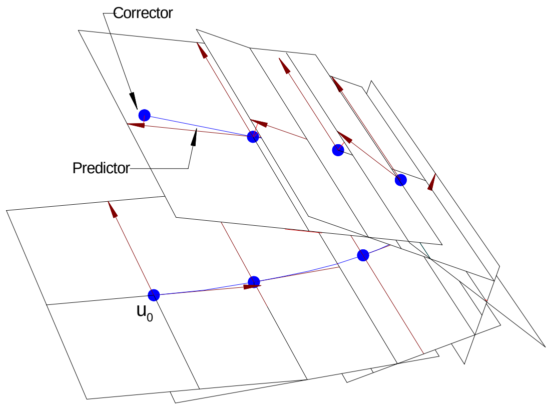


Fig. 21. Pseudo–arclength continuation in higher dimensions by choosing one parameter and following the corresponding curve on the solution manifold M.

Whichever system is used, any structure in F_u should be exploited. For example, if F is a two point boundary value problem and collocation is used, then the collocation points can be eliminated to yield a smaller system.

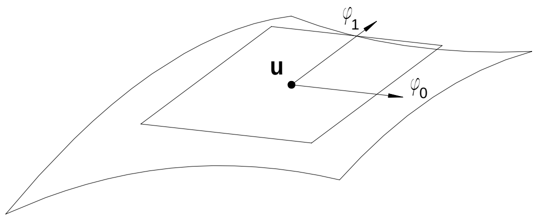


Fig. 22. The tangent space of the solution manifold M at $\mathbf{u}$, spanned by the orthonormal basis $\Phi = [\phi_0, \phi_1]$.

Projecting a point in the tangent space onto M.

The projection of a point $\mathbf{s} \in \mathbb{R}^k$, in the tangent space of M at $\mathbf{u}_0$ (with orthnormal basis Φ_0) orthogonal to tangent space is the solution of the system

$$\begin{aligned} F(\mathbf{u}) &= 0 \\ \Phi^T(\mathbf{u}-\mathbf{u}_0) &= \mathbf{s} \end{aligned} \tag{21}$$

(Figure 23.) If $\mathbf{u}_0$ is a regular point of M and $\mathbf{s}$ is small enough, the Jacobian of this system is non-singular. Modified Newton's method for the nonlinear system results in linear systems with the same matrix as for the tangent space, but with different right hand sides.

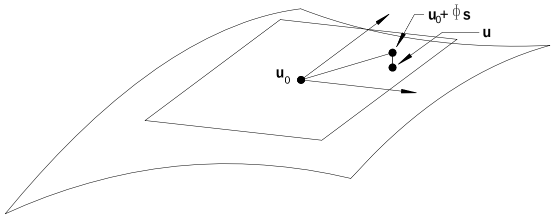


Fig. 23. Projecting a point $\mathbf{s}$ onto the solution manifold M orthogonal to the tangent space at $\mathbf{u}$.

Estimating the size of a ball in the tangent space.

The IFT in finite dimensional spaces is a modified Newton's method, and there are bounds on the size of the ball in terms of norms of the Jacobian, it's inverse, and Lipschitz bounds on the Jacobian. Experience indicates that local estimates of these quantities are expensive to compute, and provide a very conservative radius. Global bounds can sometimes be found, but then the estimated radius is even more conservative.

A different approach is to impose a maximum number of Newton iterations required for the projection, and reduce the radius by a fixed factor if the number exceeds the maximum allowed. This is an estimate that comes after the fact, and the way we posed the continuation method above requires an estimate before the projection is done.

Here we present a method from [20] that chooses the radius of the ball so that the distance from the tangent space to the manifold is roughly constant. If the rate of convergence of Newton's method is constant over the manifold this will be the same as limiting the number of Newton steps.

Using Taylor's remainder theorem, we have:

$$|\mathbf{u}(\mathbf{s}) - \mathbf{u}_0 - \Phi\mathbf{s}| \leq \frac{1}{2}|\mathbf{u}_{\mathbf{ss}}\xi\xi| \tag{22}$$

where ξ is some point in the ball $\xi \leq |\mathbf{s}|$. If a tolorance ϵ is given on the error, $\mathbf{s}$ must lie in a ball of radius radius of the ball in which the error is less than that tolerance is

$$R(\mathbf{s} = \sqrt{\frac{\epsilon}{2|\mathbf{u_{ss}}|}} \tag{23}$$

(Figure 24.) The second derivative of $\mathbf{u}$ is the solution of the system

$$\begin{aligned} F_{\mathbf{u}}(\mathbf{u}_0)\mathbf{u_{ss}} &= -F_{\mathbf{uu}}\Phi\Phi \\ \Phi^T\mathbf{u_{ss}} &= 0 \end{aligned} \tag{24}$$

It is therefore possible to find the second derivative (which is a $k \times k$ matrix whose entries are vectors in $\mathbb{R}^n$), and estimate it's norm.

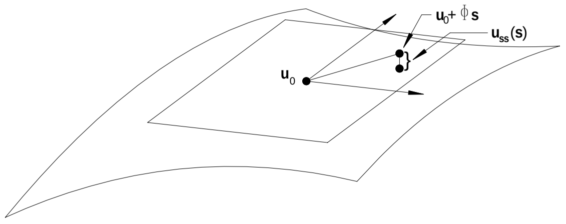


Fig. 24. The curvature of the solution manifold M at $\mathbf{u}$.

Determining whether an n dimensional simplex crosses M.

The mapping $F(\mathbf{u})$ takes a simplicial complex with vertices $\mathbf{v}_i \in \mathbb{R}^{n+k}$ and assigns new coordinates $F(\mathbf{v}_i)$. In particular, F maps an n dimensional simplex with vertices in $\mathbb{R}^{n+k}$ into an n dimensional simplex in $\mathbb{R}^n$ (provided the vertices are regular points and the simplices are small enough). (Figure 25.) If an n dimensional simplex contains a point on the solution manifold, it will generically be a single point. Using barycentric coordinates α, the system for the point is

$$F(\sum_0^{n+k} \alpha_i\mathbf{v}_i) = 0 \qquad\qquad \sum_0^{n-k} \alpha_i = 1. \tag{25}$$

This is a square system for the unknowns α_i, and is a mapping from a reference simplex in $\mathbb{R}^{n+1}$ to the simplex with vertices $F(\mathbf{v}_i)$. In practice a piecewise linear approximation to F is used. Again using barycentric coordinates, the approximation within a p–cell is

$$F(\sum_0^{p} \alpha_i\mathbf{v}_i) \sim \sum_0^{p} F(\mathbf{v}_i)\alpha_i \tag{26}$$

This expression only depends on the vertices of the p–cell, so the approximation is continuous between adjacent simplices. The point where the interpolant is zero is the solution of the $(n+1) \times (n+1)$ linear system.

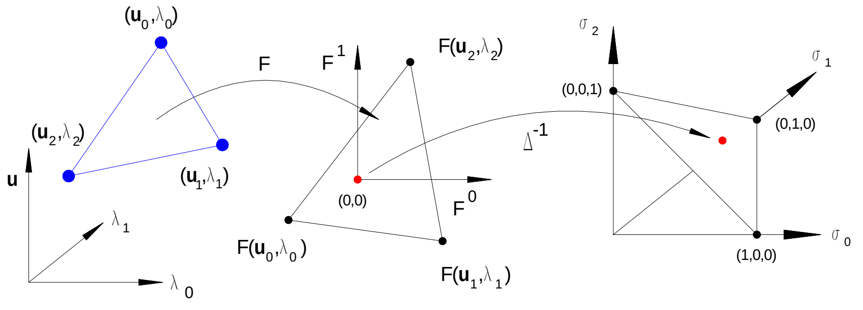

$(\mathbf{u}_0,\lambda_0)$
$(\mathbf{u}_2,\lambda_2)$
$(\mathbf{u}_1,\lambda_1)$
u
λ_1
λ_0
F
F^1
$F(\mathbf{u}_2,\lambda_2)$
(0,0)
F^0
$F(\mathbf{u}_0,\lambda_0)$
$F(\mathbf{u}_1,\lambda_1)$
Δ^{-1}
σ_2
σ_1
σ_0
(0,0,1)
(0,1,0)
(1,0,0)

Fig. 25. The image of an n dimensional face in a simplicial complex under F (with the mapping applied only to the vertices). This is equivalent to a global piecewise linear approximation to F over the complex. The question of whether an n dimensional face in a simplical decomposition of $\mathbb{R}^{n+k}$ contains a zero of the interpolant is whether 0 lies in the image of the face.

$$\Delta\alpha \equiv \begin{bmatrix} F^0(\mathbf{v}_0) & \dots & F^0(\mathbf{v}_n) \\ \vdots & & \vdots \\ F^{n-1}(\mathbf{v}_n) & \dots & F^{n-1}(\mathbf{v}_n) \\ 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{n-1} \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{27}$$

This is one step of Newton's Method using differences for the Jacobian. If α_0 (for example) is eliminated using the last row, we have that $\alpha_0 = 1 - \alpha_1 - \dots - \alpha_n$, and the system for the remaining α's is

$$\begin{bmatrix} F^0(\mathbf{v}_1) - F^0(\mathbf{v}_0) & \dots & F^0(\mathbf{v}_n) - F^0(\mathbf{v}_n) \\ \vdots & & \vdots \\ F^{n-1}(\mathbf{v}_n) - F^0(\mathbf{v}_0) & \dots & F^{n-1}(\mathbf{v}_n) - F^0(\mathbf{v}_n) \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} -F^j(\mathbf{v}_0) \\ \vdots \\ -F^j(\mathbf{v}_n) \end{bmatrix} \tag{28}$$

3.4 Allgower and Schmidt's Pattern Algorithm.

This algorithm, described in [3] extends Allgower and Georg's one dimensional simplicial continuation [1] to arbitrary dimension. It produces a list of $n+k$ dimensional simplices and points at which which a piecewise linear approximation to F is zero.

The solution space $\mathbb{R}^{n+k}$ is decomposed as a simplicial complex, using the Kuhn–Freudenthal triangulation. (Section 2.3.) The initial solution $(\mathbf{u}_0, \lambda_0)$ is used to choose the origin z so that the simplex (z, π) with π the identity permutation contains the initial solution. For example, the expression for the vertices of the simplex can be used to choose z so that the point with barycentric coordinates $(1, \dots, 1)/\sqrt{n+k}$ is the initial point. A list of the indices (z, pi) of the simplices which will be output is kept, starting with the initial simplex.

Iteratively, the n–dimensional of the simplices in the aggregate which lie on the boundary are tested to find those that cross M (Section 3.3). The candidate faces are kept in a list, which is updated as simplices are added to the aggregate. The n–faces of the initial simplex are found by removing all combinations (without order) of k vertices from the index of the simplex. The n dimensional simplices that are found to cross M play the role of points on the boundary. The points on the piecewise linear interpolant can be stored, but are not needed by the algorithm.

With a simplex that crosses M, all of the $n+k$ dimensional simplices which have it as a face are checked. Any that are not on the list are added. This is a process of adding k vertices to the index of the face, with vertices which lie across any $n+k-1$ face that contains the boundary simplex. This set of simplices is the neighborhood of the boundary. (Figure 26.)

Finally, the list of boundary points is updated, removing those which lie on an $n+k-1$ dimensional face for which both cells on either side of the face are in the list, and adding those n dimensional faces of the newly added simplices which lie on $n+k-1$ dimensional simplices with only one cell on the list. Finding the index of the simplex on the opposite side of an $n+k-1$ dimensional face is a pivot, and the action of the pivot on the index is known.

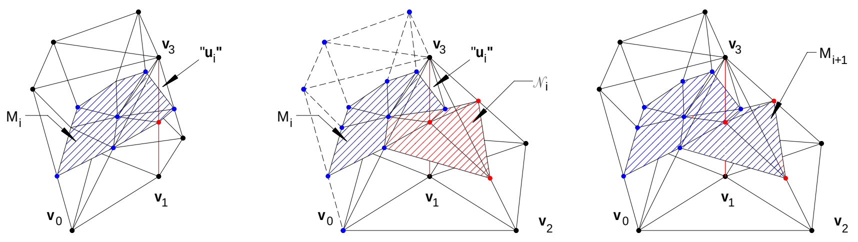


Fig. 26. Allgower and Schmidt's algorithm: A transverse $n-k=1$ dimensional face (edge), shown in red, and the simplices on which it lies. This collection of simplices is the neighborhood of the transverse face. The merge is easy, since the simplices are all drawn from a larger simplicial complex.

3.5 Rheinboldt's Moving Frame Algorithm.

Rheinboldt's moving frame algorithm [32] represents M as a k dimensional mesh with vertices on M. Any fixed mesh can be used, but there must be an ordering on the mesh points $\mathbf{s}_i$, and a way to find a point $\mathbf{s}_{\tilde{i}}$ with $\tilde{i} < i$ and close to $\mathbf{s}_i$. For a rectangular mesh this is straightforward.

The initial point is $\mathbf{s}_0$, which is mapped to the initial point $(\mathbf{u}_0, \lambda_0)$, and Φ_0 is any orthnormal basis for the null space of the Jacobian at the initial point. (Figure 27.)

Each mesh point is computed in sequence. At step i, $(\mathbf{u}_{\tilde{i}}, \lambda_{\tilde{i}})$ is the boundary point.

The neighborhood of the boundary point isn't explicitly represented, but might be defined as the mesh cells with index less than i that include $\mathbf{s}_{\tilde{i}}$ as a vertex. The only point without a mapping to M is the point i. The mapped point $(\mathbf{u}_i, \lambda_i)$ is found by projecting $\mathbf{s}_i - \mathbf{s}_{\tilde{i}}$ orthogonally onto M using the tangent space $\Phi_{\tilde{i}}$. The new tangent space is found by first solving the linear system

$$\begin{aligned} F_(\mathbf{u}, \lambda)(\mathbf{u}_i, \lambda_i)\tilde{\Phi}_i &= 0 \\ \Phi_{\tilde{i}}^T \Phi_i &= I \end{aligned} \tag{29}$$

and then orthonormalizing Φ_i.

The merge operation is trivial, it just involves incrementing i.

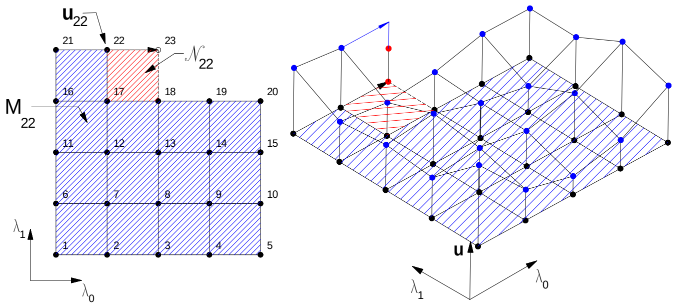


Fig. 27. Rheinboldt's moving frame algorithm using a rectangular mesh. The natural parameter version of Rheinboldt's (left), and (right) the solution spaceversion . The natural order (e.g. left to right, bottom to top) is used on the mesh vertices, and the continuation advances in the same order. Moves from scan line to scan line use the point at the beginning of the scan for an initial guess.

3.6 Brodzik's Tiling Algorithm.

Brodzik and Rheinboldt's continuation method for 2-manifolds [10], and Brodzik's extension to arbitrary dimension [9] represent M as a k-dimensional Delaunay triangulation with vertices lie on M.

The initial simplicial complex is a reference Delaunay triangulation of a k dimensional spherical ball. The vertices on M are found by projecting the set of vertices $\mathbf{s}_i$ onto M using the tangent space at the initial point.

A boundary point is a vertex on any simplex which has a $k-1$ dimensional face without two simplices on opposite sides if the face.

The neighborhood is the projection of the vertices of the reference decomposition of the spherical ball orthogonally onto M, though not all vertices are projected.

The difficult step in this algorithm is the merge. The points of the current Delaunay triangulation which are near to the boundary point are projected into the tangent space at the new point

$$\tilde{\mathbf{s}}_j = \Phi_i^T \left((\mathbf{u}_j, \lambda_j) - (\mathbf{u}_i, \lambda_i) \right). \tag{30}$$

The previously computed vertices carry with them part of the entire Delaunay triangulation. First points from the spherical ball which lie inside this projected triangulation are discarded. Next points from the sphere that lie too close to a vertex of the triangulation are also discarded. Finally the remaining points are projected orthogonally onto M, and the Delaunay triangulation is updated to include the new vertices. This last step is done in the tangent space, and involves removing some of the cells from the triangulation, and replacing them with new cells. (Fig. 28.) The extension to an arclength

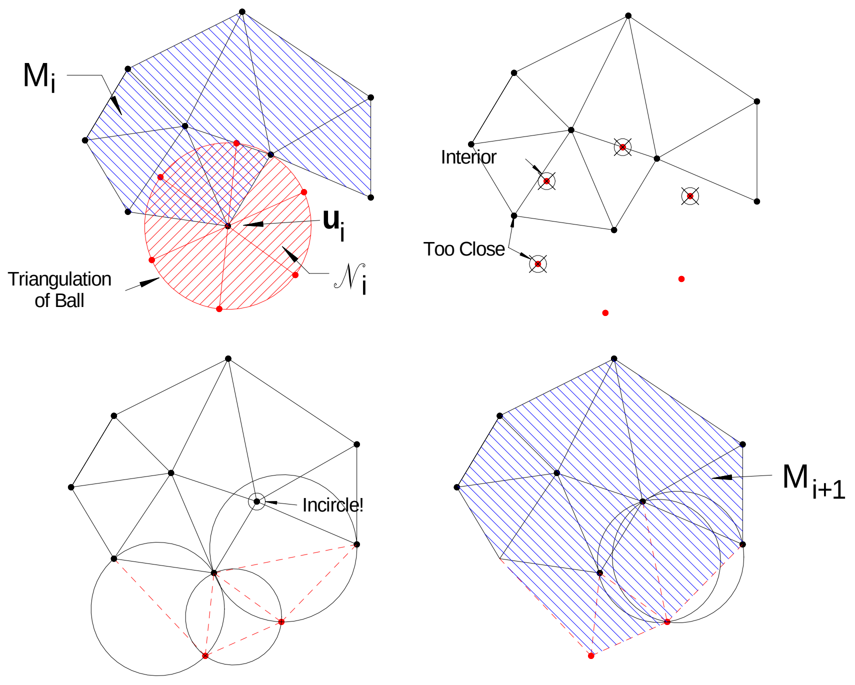


Fig. 28. The natural parameter version of Brodzik and Rheinboldt's algorithm [10] (and Brodizk's extension to arbitrary dimension [9]).

continuation is done by projecting the points which are near the boundary point onto the tangent space at the boundary point, and then proceeding as if the tangent space coordinates were the parameters in the natural parameter continuation. (Fig. 29.)

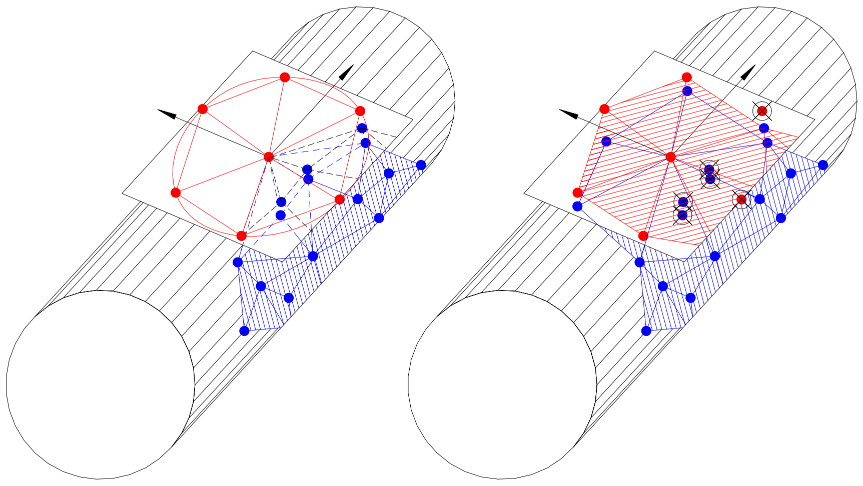


Fig. 29. The arclength version of Brodzik and Rheinboldt's algorithm [10] (and Brodizk's extension to arbitrary dimension [9]).

3.7 Melville and Mackey's Tiling Algorithm

Melville and Mackey's algorithm [29] is strictly two dimensional. M is not explicitely represented, rather the boundary of the triangulation is represented as a polygonal curve with vertices on M.

The initial polygon is found using PSALC, starting at the initial solution, and solving the modified system

$$\begin{aligned} &F(\mathbf{u}, \lambda) = 0 \\ &|\mathbf{u} - \mathbf{u_0}|^2| + |\lambda - \lambda_0|^2 - R^2 = 0 \end{aligned} \tag{31}$$

To obtain an initial solution for this system, a homotopy can be done in the radius R starting at $R = 0$ when $(\mathbf{u}_0, \lambda_0)$ is a solution, to the full value of R, where R is chosen to be roughly the desired resolution of M. With a point on the intersection of M and the sphere, PSALC can be used to trace the intersection of M and the sphere clockwise relative to Φ_0. The polygon formed by the computed points is the initial boundary polygon. The tangent space Φ_i at each point on the boundary is found as in the moving frame algorithm, which preserves the orientation of tangent space. (Figure 30.)

With an explicit representation of the boundary, any vertex will serve as a boundary point.

The neighborhood of the boundary point is a circular disk of radius R, represented by it's boundary. Two PSALC's are done in the tangent space of the boundary point. The first to the circle, and the second around the circle.

The merge operation starts by projecting the part of the existing boundary whose vertices are near the boundary point into the tangent space at the

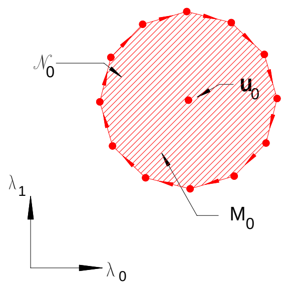


Fig. 30. A homotopy in the radius R of the circle about λ_i for obtaining a starting point on the boundary of the neighborhood of λ_i for Melville and Mackey's boundary algorithm.

boundary point. This leaves a relatively simple two dimensional problem of clipping the boundary of the neighborhood against the existing boundary. (Figures 31 and 32.)

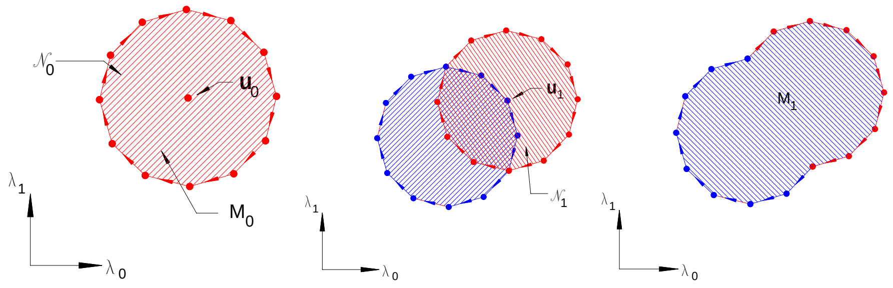


Fig. 31. The natural parameter version of Melville and Mackey's boundary representation algorithm for $k = 2$. Left to right are shown the initial configuration, where a continuation has been used around the boundary of the region of interest, then the construction of a neighborhood by continuing around a circle, and finnaly the merge.

3.8 The author's covering algorithm

This algorithm [20] represents the solution manifold M as a union of the projection of spherical balls. Each ball lives in the tangent space of a point on M, and is represented by it's radius R_i, an orthonormal basis for the tangent space Φ_i, and a restricted Laguerre–Voronoi polyhedral k–cell P_i.

The initial condition is found by finding Φ_0, estimating the size of the ball (Section 3.3), and setting P_0 to a cube slightly larger than the spherical ball.

The boundary point is found by selecting a ball that has a polyhedral vertex which lies outside its spherical ball. If the ratio of the radii of neighboring

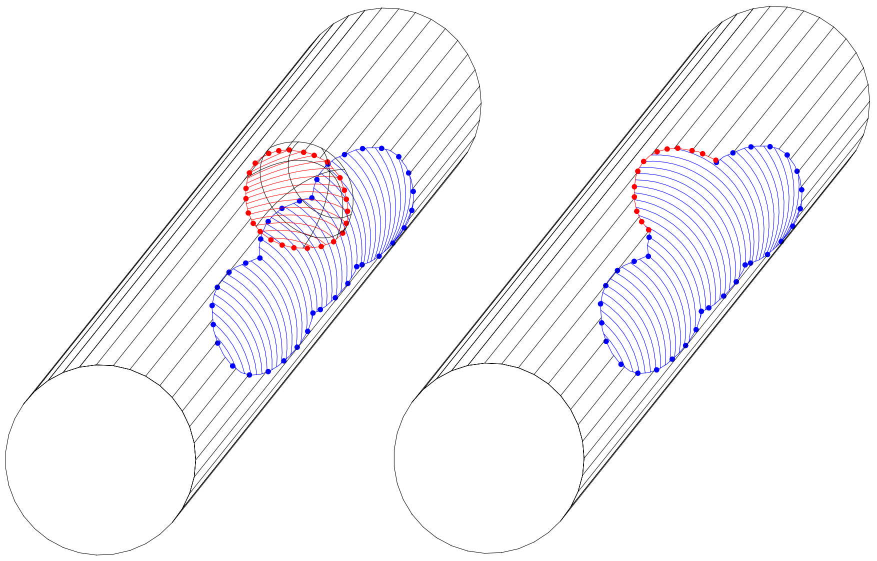


Fig. 32. Melville and Mackey's boundary representation algorithm for $k = 2$ and $n = 3$.

balls is within $\sqrt{2}$ the origin (the center of the ball) is inside the polyhedron, and a boundary point can be found by finding the intersection of a line between the origin and the exterior vertex and the sphere.

The neighborhood of the boundary point is the projection of a spherical ball in the tangent space at the boundary point onto M. The polyhedron for the boundary point is initialized to a cube centered about the origin.

The merge requires finding neighboring balls, and subtracting complementary half spaces from their polyhedra. The half spaces are found by projecting each center into the tangent space of the neighbor, and using a ball of the same radius, but now in the other tangent space. (Figures 33 and 34.)

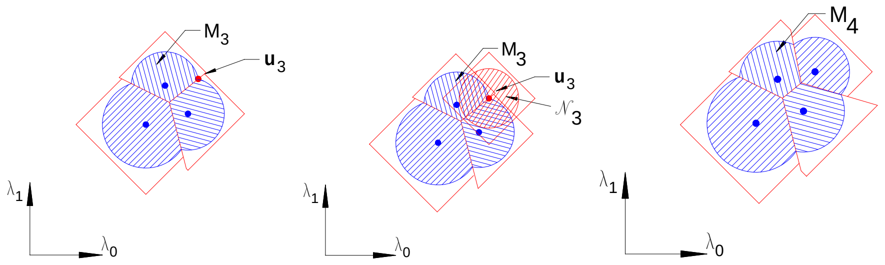


Fig. 33. The natural parameter version of the author's covering algorithm.

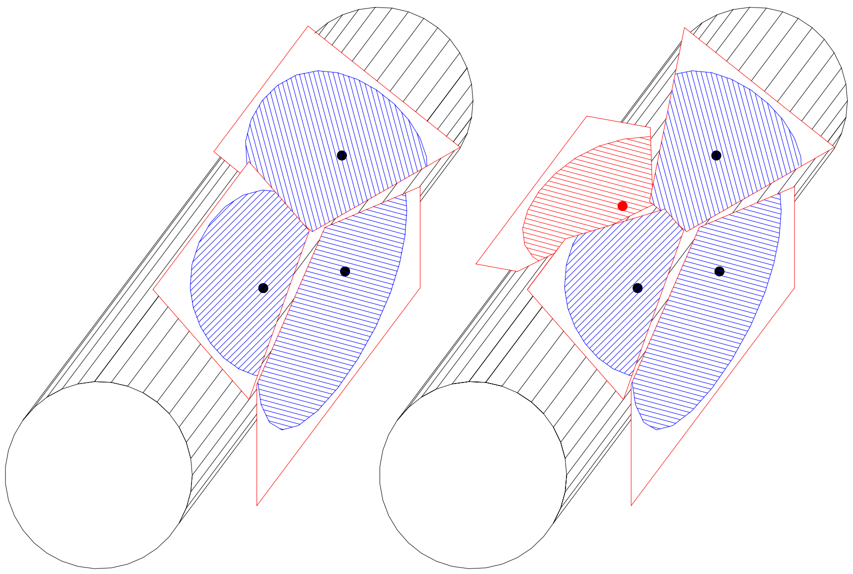


Fig. 34. The author's covering algorithm in solution space. To update the polyhedra P_i the centers of neighboring neighborhoods are projected into the tangent space at $\mathbf{u}_i$

4 Example and Comparisons

We have tried to cast each of these algorithms as an iteration of two steps. If M_i is the part of the manifold that has been computed in i steps, then M_{i+1} is found by first selecting either a point or transverse face $\mathbf{u}_i$ that lies on the boundary of M_i, constructing a neighborhood $\mathcal{N}(\mathbf{u}_i)$ of that point, and merging the neighborhood into M_i to get M_{i+1}.

The algorithms differ in how they perform these steps, but there are two challanges which they all must address. All use a simplicial or cellular approximation to the manifold, and to perform the merge the simplices in the neighborhood must be compatible with those in M_i (that is, they must either be disjoint or have an interseection which is a face). Allgower and Schmidt's pattern algorithm maintains compatibility by selecting simplices from a decomposition of the entire region of interest, as does Rheinboldt's moving frame algorithm, so that compatibility is guarenteed for these two algorithms unless the mapping from simplices to M becomes singular. Melville and Mackey do not attempt to store a representation of M_i, but instead keep it's boundary as a set of polygons, and update the boundary. This is limited $k = 2$, though in principle a simplicial approximation could be maintained for the boundary in higher dimensions. But in that case the compatibility issue arises again. Brodzik's algorithm maintains a Delaunay triangulation on M, and deals with compatibility by removing points, and recomputing the triangulation locally.

Finally, the author's algorithm avoids the compatibility issue by representing M_i as a set of overlapping neighborhoods. Underneath the covers however, is a type of Voronoi triangulation of M_i which aids in finding boundary points, and whose dual is a Delaunay triangulation. So it is possible to view the algorithm as having a merge, but by way of a method that isn't easily described without the Voronoi cells.

The second challange is to avoid computing part of the manifold more than once. This means that the boundary must be maintained in some form or other. In Allgower and Schmidt's pattern algorithm care is taken not to pivot to a simplex which is already in M_i. This is done using a clever integer coding of the simplices, and keeping a coded list of simplices for M_i. Rheinboldt's moving frame algorithm does not address the issue, but since there are a finite number of "boundary" vertices in the reference k dimensional complex, the algorithm at least terminates and does not cycle. Melville and Mackey explicitly represent the boundary, and update the boundary of M_i to find the boundary of M_{i+1}. Brodzik's algorithm detects the overlap of simplices, but simply removes one of the overlapping simplices to leave a gap. THe author's algortihm indirectly represents the boundary in terms of polyhedra in the tangent space, which insure that new points are within a given tolerance of the boundary.

To illustrate how the various algorithms actually perform, we implemented all except Brodzik's, and applied them to the simple example of a sphere:

$$F(\mathbf{u}) = |\mathbf{u}|^2 - 1 \tag{32}$$

with $n = 3$, $k = 2$.

In this case ($k = 2$, $n = 3$) Allgower and Schmidt's algorithm produces a decomposition of the sphere, but even though the simplices in $\mathbb{R}^3$ are fairly uniform, their intersection with the sphere is not, and the method can be expected to produce small triangles. Fig. 35(upper left).

As described above, for a rectangular mesh, Rheinboldt's moving frame algorithm covers parts of the sphere more than once, and although square cells were used in the k dimensional reference space, the projection to the sphere results in clustering of vertices near the pole (the initial point and tangent space define an equator). Fig. 35(upper right).

Melville and Mackey's algorithm is able to cover the sphere, and to avoid covering it more than once. The vertices obtained lie on curves on the sphere which are the intersection of the spherical neighborhood in $\mathbb{R}^3$ that is used to define a curve about the boundary point. In this case these are circular arcs. Fig. 35(lower left).

Brodzik's algorithm not only leaves gaps, but the triangles on the sphere are of a range of sizes. Fig. 36, from [9].

And the author's algorithm sucessfully covers the entire sphere, and produces polygonal Voronoi regions on the sphere. Fig. 35(lower right).

The result of these computations is an approximation to a branched manifold, but no mention has been made yet of how to compute the singular

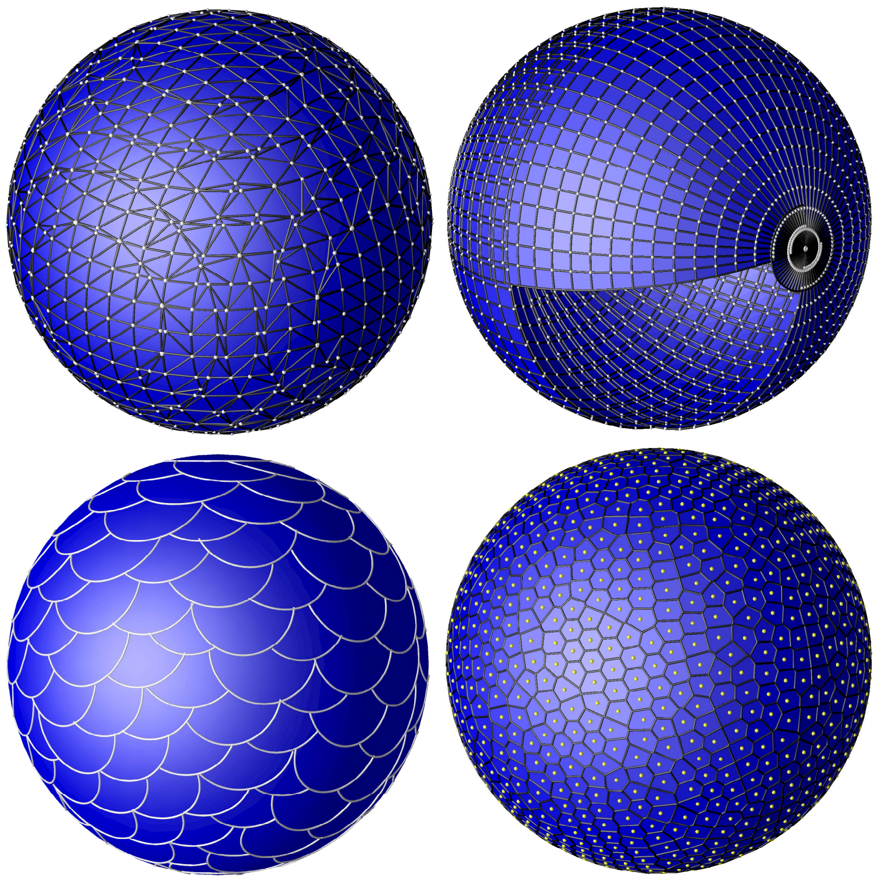


Fig. 35. (top right)Allgower and Schmidt's pattern algorithm with tetrahedra selected from a decomposition of $\mathbb{R}^3$ with $5 * 20 * 20 * 20 = 40000$ tetrahedra (the five tetrahedral decomposition of the cube was used, with eight copies reflected so that there are no incompatibilities). (top left) Rheinboldt's moving frame algorithm with a 50x50 mesh of size 0.1. (lower left) Melville and Mackey's algorithm, which produces a set of boundaries (curves) on the sphere, which in this (lower right) The result of the author's algorithm with the maximum radius of .1 and tolerance $\epsilon = 0.01$. case are circular arcs.

boundary manifolds, or how to move from one branch to another. The author [21] has a discussion of branch switching for manifolds, and though it is cast in terms of the author's algorithm, it could be applied to all of the algorithms except the pattern algorithm. The idea is that moving from a point on M to a point projected orthogonal to the tangent space is one step of PSALC. Therefore singular points may be detected as usual. Branch switching is a little more complicated. For simplicial continuation each of the n dimensional

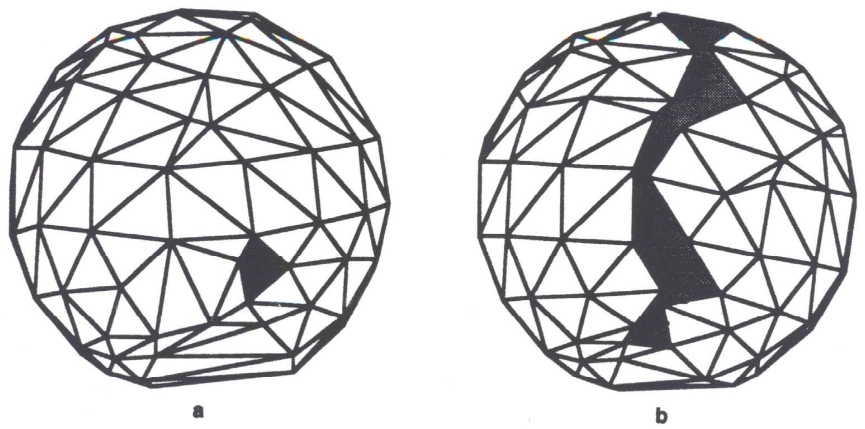


Figure 7. Two triangulations by MAMESH of the 2-sphere. Mesh (a) was generated using $C_\kappa = 2$, and (b) using $C_\kappa = 3$.

Fig. 36. A figure from [9] showing Brodzik's algorithm applied to a sphere with a larger simplex edge than those above. Note that a "gap" is left. The gap is left after removing incompatible simplices generated by non-local overlaps.

faces of the simplices are checked for point on M, so unless a singular point lies at a vertex of the complex, there is no difficulty.

How a mathematical object is represented can effect the complexity of an algorithm. These five algorithms all use different representations of either the solution manifold or the neighborhood of a boundary point. They draw on techniques from algebraic topology, computational geometry, linear programming (the algorithm for subtracting a half space from a polyhedron), and differential geometry. When the results are placed side by side, it can be seen that all are a variation of constructing an approximately spherical neighborhood of a point (or face) on the boundary, and merging the neighborhood into the whole.

Lastly, a practical note. Some work has been done on visualizing solution manifolds [27], but it is still not clear what can be done with a 3–manifold computed with these algorithms.

References

1. E. L. Allgower and K Georg. Simplicial and continuation methods for approximations, fixed points and solutions to systems of equations. *SIAM Review*, 22:28–85, 1980.
2. Eugene L. Allgower and Stefan Gnutzmann. An algorithm for piecewise-linear approximation of implicitly defined two-dimensional surfaces. *SIAM Journal on Numerical Analysis*, 24(2):452–469, April 1987.

3. Eugene L. Allgower and Phillip H. Schmidt. An algorithm for piecewise-linear approximation of an implicitly defined manifold. *SIAM Journal on Numerical Analysis*, 22(2):322–346, April 1985.
4. F. Aurenhammer. Power diagrams: Properties, algorithms and applications. *SIAM Journal of Computing*, 16(1):78–96, 1987.
5. F. Aurenhammer. Improved algorithms for discs and balls using power diagrams. *Journal of Algorithms*, 9:151–161, 1988.
6. F. Aurenhammer and H. Edelsbrunner. An optimal algorithm for constructing the weighted Voronoi diagram in the plane. *Pattern Recognition*, 17(2):251–257, 1984.
7. Franz Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, 1991.
8. A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.
9. M. L. Brodzik. The computation of simplicial approximations of implicity defined p-dimensional manifolds. *Computers Math. Applic.*, 36(6):93–113, 1998.
10. Monica L. Brodzik and Werner C. Rheinboldt. On the computation of simplicial approximations of implicitly defined two-dimensional manifolds. *Computers and Mathematics with Applications*, 28(9):9–21, 1994.
11. Witold Brostow, Jean-Pierre Dussault, and Bennett L. Fox. Construction of Voronoi polyhedra. *Journal of Computational Physics*, 29:81–92, 1978.
12. Pey-Chun Chen, Pierre Hansen, and Brigitte Jaumard. On-line and off-line vertex enumeration by adjacency lists. *Operations Research Letters*, 10:403–409, October 1991.
13. D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.
14. Johan L. Dupont. *Curvature and Charateristic Classes*. Number 640 in Lecture Notes in Mathematics. Springer–Verlag, Berlin, 1978.
15. Steven Fortune. Voronoi diagrams and Delaunay triangulations. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, pages 193–233. World Scientific Publishing Co., 1992.
16. P. W. Gross and P. R. Kotiuga. Data structures for geometric and topological aspects of finite element algorithms. *Progress in Elevtromagnetics Research, PIER 32*, 32:151–169, 2001.
17. Branko Grunbaum. *Convex Polytopes*. Interscience Publishers, London, 1967.
18. Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, April 1985.
19. Patrick M. Hanrahan. Creating volume models from edge–vertex graphs. *ACM SIGGRAPH Computer Graphics*, 16(3):77–84, July 1982.
20. Michael E. Henderson. Multiple parameter continuation: Computing implicitly defined k–manifolds. *International Journal of Bifurcation and Chaos*, 12(3):451–476, 2002.
21. Michael E. Henderson. Multiparameter parallel search branch switching. *Int. J. Bifurcation and Chaos*, 15(3):967–974, 2005.
22. Hiroshi Imai, Masao Iri, and Kazuo Murota. Voronoi diagram in the Laguerre geometry and its applications. *SIAM Journal on Computing*, 14(1):93–105, 1985.
23. H. B. Keller. Numerical solutions of bifurcation and nonlinear eigenvalue problems. In Paul Rabinowitz, editor, *Applications of Bifurcation Theory*, pages 359–384, New York, 1977. Academic Press.

24. Rolf Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1987.
25. H. W. Kuhn. Some combinatorial lemmas in topology. *IBM Journal of Research and Development*, 45(5):518–524, 1960.
26. S. Lefschetz. *Introduction to Topology*. Princeton University Press, 1949.
27. John H. Maddocks, Robert S. Manning, Randy C. Paffenroth, Kathleen A. Rogers, and Jeremy A. Warner. Interactive computation, parameter continuation, and visualization. *Int. J. Bif. Chaos*, 7:1699–1715, 1997.
28. William S. Massey. *Algebraic Topology, An Introduction*. Harcourt, Brace & World, Inc., New York, 1967.
29. Robert Melville and D. Steven Mackey. New algorithm for two-dimensional numerical continuation. *Computers and Mathematics with Applications*, 30(1):31–46, July 1995.
30. V. T. Rajan. Optimality of the delaunay triangulation in rd. In *Proceedings of the seventh annual symposium on Computational geometry*, pages 357–363, New York, NY, USA, 1991. ACM Press.
31. Werner C. Rheinboldt. On a moving-frame algorithm and the triangulation of equilibrium manifolds. In T. Kupper, R. Seydel, and H. Troger, editors, *ISNM79: Bifurcation: Analysis, Algorithms, Applications*, pages 256–267, Basel, 1987. The University of Dortmund, Birkhäuser Verlag.
32. Werner C. Rheinboldt. On the computation of multi-dimensional solution manifolds of parameterized equations. *Numerische Mathematik*, 53(1/2):165–181, July 1988.
33. Jonathan Richard Shewchuk. A condition guaranteeing the existence of higher-dimensional constrained delaunay triangulations. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 76–85, New York, NY, USA, June 1998. ACM Press.
34. D. F. Watson. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.