Selçuk J. Appl. Math.
Vol. 3, No. 2, pp. 81–98, 2002

Selçuk Journal of
Applied Mathematics

Locally exact smooth reconstruction of lines, circles, planes, spheres, cylinders and cones by blending successive circular interpolants

Richard Liska[1], **Mikhail Shashkov**[2] and **Blair Swartz**[2]

[1] Faculty of Nuclear Sciences and Physical Engineering,
Czech Technical University, Břehová 7, 115 19 Prague 1, Czech Republic;
e-mail: `liska@siduri.fjfi.cvut.cz`
[2] Group T-7, Los Alamos National Laboratory, Los Alamos, NM 87545, USA;
e-mail: `shashkov@lanl.gov`, e-mail: `bks@lanl.gov`

Received: July 16, 2002

Summary. G^1-smooth curves and surfaces are developed to span a given logically cuboid distribution of nodes. Given appropriate data, they locally reconstruct the curves and surfaces of spherical or cylindrical coordinates. Thus, if a set of nodes consists of a contiguous subset of a tensor product grid of points associated with a (possibly non-uniform) set of coordinate values of some rectangular, cylindrical, or spherical coordinate system; then the appropriate coordinate curves (linear or circular segments) and coordinate surfaces (segments of planes, cylinders, spheres and cones) that interpolate the subset are reconstructed exactly. The underlying construction uses four successive nodes to define a curve spanning the middle pair as follows: One interpolates each of the two successive triples of nodes with the segment of a circle or straight line going through these three points. Then one blends the two segments continuously between the middle pair of nodes. The blend is relatively linear in terms of arc-length along each segment. The union of such successive curve-sections forms a G^1 curve. Wire-frames of such curves define cell edges. Similar intermediate curvilinear interpolation of the wires defines cell faces, and their union defines G^1 coordinate-like surfaces. The surface generated depends on the direction one interpolates the wires. If the nodes are a tensor product grid associated with a sufficiently smooth reference coordinate system, then the cell edges (and probably also the cell faces) are third-order accurate.

Key words: smooth reconstruction, line, circle, plane, sphere, cylinder, cone, interpolation

2000 Mathematics Subject Classification: 65D15

1. Introduction

Most numerical methods, including Finite-Element Methods (FEMs) and Finite-Difference Methods (FDMs), use a computational grid to solve partial differential equations. Typical grid generation methods (Knupp and Steinberg, 1993; George and Borouchaki, 1998) produce both the nodes of the grid (i. e., the coordinates of the grid-points) and the grid's structure (logically-rectangular, triangular, and so on). The structure of the grid tells us which pair of nodes is associated with an edge, which edges are associated with a face, and which faces are associated with a cell of the grid. But grid generators do not always specify the continua that connect the nodes to bound and separate the cells of the computation. Yet, edge lengths, facial areas, or cell volumes are often needed when approximating differential equations.

The simplest approach to these continua is to connect nodes by segments of straight lines. This is the most common approach for FDMs in two dimensions. The situation is more complicated in three dimensions. Consider a logically cuboid grid, and connect the appropriate nodes by segments of straight lines to form the edges. The four edges associated with a face do not usually lie in the same plane, and there are a couple choices for surfaces that span such edges. Depending on this choice we will have different formulas for facial areas and cell volumes. The simplest approach in practice is to use the tri-linear map from the unit reference cube to a cell of the computational grid. This map retains the line segments as edges, while the faces become ruled surfaces, see, e.g., (Dukowicz, 1988).

Introducing curvilinear grids at curvilinear boundaries or interfaces can increase the accuracy of a simulation, as is well known from the FEM literature (Ciarlet and Raviart, 1972; Huyakorn and Andersen, 1986; Zlamal, 1973 a; Zlamal, 1973 b). But a different goal motivates the curvilinear grid that underlies the difference approximations for the equations of Lagrangian gas-dynamics in (Margolin and Shashkov, 1999): namely, the preservation of the physical symmetry of the solution. For example, the ability to represent and preserve spherical symmetry in numerical simulations of implosion is critically important in the inertial confinement fusion program, (Town and Bell, 1991), where small departures from spherical symmetry due to discrete errors may be amplified by Rayleigh-Taylor instability, leading to unacceptably large errors in problems with strong compressions. Also, uncertainty as to whether a non-symmetric result is due to numerical errors or to the physical design inhibits our understanding of the dynamics of the implosion and so severely limits our predictive capabilities.

Curvilinear grids used for the construction of symmetry-preserving FDMs in two dimensions must satisfy some particular requirements: if points from the set corresponding to a logical direction belong to a straight line or a circle, then the edge-reconstruction algorithm must reproduce those curves. In (Margolin and Shashkov, 1999) a simple two-dimensional method that satisfies these requirements is described: points are connected either by a straight line segment or by the arc of some circle. But the tangent to the resulting curvilinear grid lines can be discontinuous at nodes — which can cause some problems in FDMs. Therefore, it is desirable to use a reconstruction algorithm that produces curves having continuously changing tangent lines. When developing symmetry-preserving methods for three dimensions, there are additional requirements for the reconstruction of surfaces: certain families of planes, spheres, cylinders and cones need to be reproduced; and we would like unit surface normals to vary continuously.

Concerning related work: we were inspired by the use that (Hill et al., 1990; Hall and Hibbs, 1990) made of Overhauser parabolic patches (Overhauser, 1968); see also, e.g., (Brewer and Anderson, 1977). Overhauser blending has been successfully incorporated in boundary element methods in, e.g., (Hall and Hibbs, 1987; Hall and Hibbs, 1988); while (Sladek and Sladek, 1995) presents a deficiency of this approach related to normal vectors.

We mention two references that explicitly blend portions of circles. In order to move smoothly between a succession of orientations of a solid, i. e., between a sequence of points on the unit 3-sphere in 4-space, (Kim and Nam, 1995) blend two successive three-point interpolating circles between the common pair of points they each interpolate, using weights that are linear (or, even smoother) in the relative distances along the two segments. Because they must remain on the unit 3-sphere in 4-space, they use the weights to locate the blended point on the unit circle centered on the origin that contains the two points being blended (instead of locating it on the line segment between the two points as we do). On the other hand, given four successive points in the plane, (Wenz, 1996) interpolates the middle pair (only) with two successive circles and uses relatively linear weights to locate the blended point on the line between the two points being blended. Our own choice of the circle that we associate with three successive points, now in 3-space, is the one that interpolates all three points. This not only allows our blend of two such successive interpolants to reproduce circles but also improves the space-curve approximation for our subsequent blend (now a non-planar curve) from Wenz's second-order accuracy (as we illustrate) to a more natural third-order accuracy in the local mesh size (as we prove). But more significantly for our purposes, our choice also ensures locally exact reconstruction of circular arcs in 3-space from non-uniform grids on those arcs — in contrast (Wenz, 1996) reconstructs them in the plane only when such grid-spacing is uniform.

In summary: we shall describe a new method for generating curves and surfaces spanning a logically cuboid grid in three dimensions that has the following properties:

- When the given nodes are associated with a contiguous subset of a tensor product grid of points associated with a (possibly non-uniform) set of coordinate values of some rectangular, polar, cylindrical, or spherical coordinate system, the interpolants exactly reconstruct
  - the appropriate lines segments and circular arcs for the curve interpolant
  - the appropriate segments of planes, spheres, cylinders and cones for the surface interpolant.
- The continua that interpolate the nodes are G^1 continuous; i. e., tangent lines and planes vary continuously.

Our main contribution here is an interpolation method that has these properties while depending continuously on the data. The third-order curve-approximation accuracy we prove is an added bonus.

The paper is organized as follows. In Section 2 we present the algorithm for our local blend of two successive circular interpolants. Section 3 describes the properties of the resulting curves, and Section 4 presents several examples. Sections 5, 6 and 7 describe an algorithm, properties and examples for our surface interpolant — which is obtained using blended local circular nodal interpolation to form cell edges or "wires", followed by similar interpolation across the wires to define a surface interpolant. An Appendix summarizes proof of the third-order approximation of space curves by successive three-point circular interpolants and their blendings.

2. Blending Successive Circular Interpolants

The curve we construct connects all but the first and last of a given set of nodes because our algorithm requires an additional point at each end. Specifically, given an ordered set of $n+2$ distinct points $p_0, p_1, \ldots, p_n, p_{n+1}$, we shall define a curve C going through $p_1, \ldots, p_n$. Two successive circular interpolants — of the two overlapping triples of four successive points from this sequence — are blended between the middle pair to yield the segment of C between that pair. For completeness we describe our entire algorithm. To facilitate comparison we follow a corresponding description in (Wenz, 1996) as closely as possible.

Algorithm 1 (*Construction of the curve interpolant*)

```
1. For $i := 1$ to $n-1$ do (construct the interpolant between $p_i$ and $p_{i+1}$)
   (a) determine the circle or line segment $C_i^L$ going through the points $p_{i-1}, p_i, p_{i+1}$
   (b) determine the circle or line segment $C_i^R$ going through the points $p_i, p_{i+1}, p_{i+2}$
   (c) between $p_i$ and $p_{i+1}$, blend the two segments $C_i^L$ and $C_i^R$ into the resulting curve $C_i$, doing this in a manner ensuring that the tangents of $C_i$ and $C_i^L$ are the same at the point $p_i$, and the tangents of $C_i$ and $C_i^R$ are same at the point $p_{i+1}$
2. The union of all such curve segments $C_i$ defines the resulting interpolating curve $C$
```

Note that we use both pieces of each circle, i. e., $C_{i+1}^L = C_i^R$; so we need to locate only about half as many circles as (Wenz, 1996) when interpolating the same data. We apply our algorithm in both two and three dimensions. If the three points are collinear we construct the line segment. If not, we construct the circle going through the points — but in three dimensions we first construct the plane going through the points.

The blending in step 1(c) of the algorithm requires a parameterization of circular/line segments. The line segment L between the points p_i and p_{i+1} is parameterized by $L(t) := (1-t)p_i + tp_{i+1}, t \in [0,1]$, so that $L(0) = p_i$ and $L(1) = p_{i+1}$. The circular segment c with center $a = (x_c, y_c)$ and radius r is parameterized between p_i and p_{i+1} as $c(t) = (x(t), y(t))$, where

$$x(t) = r\sin(t\varphi - \vartheta) + x_c, \quad y(t) = r\cos(t\varphi - \vartheta) + y_c, \qquad t \in [0,1].$$

Here, $\varphi = \angle(p_i, a, p_{i+1})$, and ϑ is a rotational angle chosen so that $C(0) = p_i$ and $C(1) = p_{i+1}$. In three dimensions the parameterization is the same in principle, but necessarily involves the directional cosines for rotating a plane determined by three points in the coordinate frame. As in the simplest form in (Kim and Nam, 1995) — but here along a line rather than along a circle — and like (Wenz, 1996) but in 3-space, we choose our blend for step 1(c) to be the relatively linear Overhauser-type blend C_i of our parameterized segments C_i^L and C_i^R. It yields the (generally non-planar) curve-segment

$$C_i(t) = (1-t)\,C_i^L(t) + t\,C_i^R(t), \qquad t \in [0,1]. \tag{1}$$

An (extreme) example of such blending is shown in Fig. 1.

Finally, the curve interpolant $C(t)$, $t \in [1,n]$ is obtained by adjoining the segments C_i so that

$$C(t) = C_i(t-i), \quad t \in [i, i+1], \quad i = 1, \dots, n-1.$$

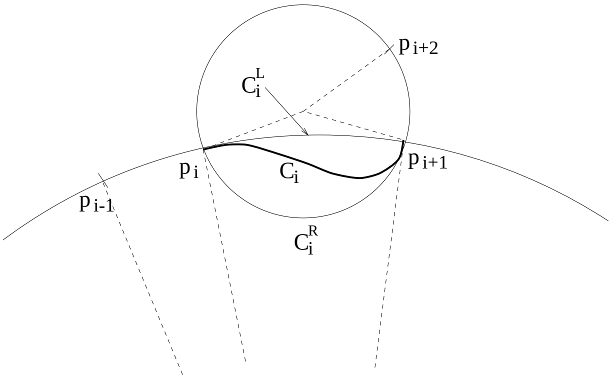


Fig. 1. The successive interpolating arcs C_i^L, C_i^R, with their blend C_i in bold.

3. Properties of the blend of successive circular interpolants

We list the properties of our blend of successive circular interpolants.

Locality: The segment of the interpolant between the points p_i and p_{i+1} depends only on the four points p_{i-1}, p_i, p_{i+1}, and p_{i+2}.

Exactness for segments of straight lines or circles: If four points $p_{i-1}, p_i, p_{i+1}, p_{i+2}$ lie linearly ordered on a straight line (circle), then the interpolant between the points p_i, p_{i+1} is the segment of the line (circle) going through these two points. If all points $p_0, p_1, \dots, p_n, p_{n+1}$

lie linearly ordered on a line (circle), then the whole interpolant is a segment of that line (circle). In contrast to (Wenz, 1996) we do not need equidistant points for this property.

Exactness for Cartesian, polar, cylindrical, and spherical grid edges: A two-dimensional (2D) grid is given by points p_{ij}; and a 3D grid, by points p_{ijk}. If our set of interpolated points $p_0, p_1, \ldots, p_n, p_{n+1}$ is a subset of a 2D or 3D grid that consists of consecutive points along a line parallel to an axis of some set of spherical, cylindrical, polar, or Cartesian coordinates; then our interpolant coincides with the corresponding coordinate curve in Euclidean space. Thus, we can exactly reconstruct by our interpolation the edges of a 2D or 3D tensor-product set of grid cells whose faces, in such curvilinear coordinates, are all parallel to the appropriate coordinate planes. As with the previous properties this one is a local property: if we have a grid of one of the above types which is exact in some region but deformed in the rest, then our interpolant reconstructs the grid edges in the interior of that region exactly.

Approximation: If the points $(p_i)_0^n$ lie successively on a thrice-differentiable space curve p, then the both blend and the local circular interpolants themselves are third-order $O(h^3)$ accurate as an approximation to p, where h is the maximum of the three successive secant distances between the four points of local data. See the Appendix for details. In particular, the distance $\|p(t) - C_i(t)\|$ is $O(h^3)$ in size — here the point $p(t)$ is the relative distance t from p_i to p_{i+1} along p.

4. Examples of blends of successive sircular interpolants

We have compared graphs of our interpolants with the pictorial examples presented in (Wenz, 1966) and did not find important differences (see Liska, Shashkov and Swartz 1998).

As examples from our motivating field of interest we present, in Fig. 2, two-dimensional deformed non-uniform Cartesian and polar grids, where we used our curve interpolant to connect the nodes. The interior nodes of the deformed grids are randomly moved from their non-deformed locations. The edges of the deformed grids form G^1 continuous curves. The bounding edges (undeformed lines or circles) are reconstructed exactly by our interpolant.

Fig. 3 (a) shows the edges of part of a deformed uniform 3D Cartesian grid as constructed by our curve interpolant. The edges delineate four grid cells. The deformation is done by random shift of the eight nodes corresponding to the closest cell, while the remaining ten points

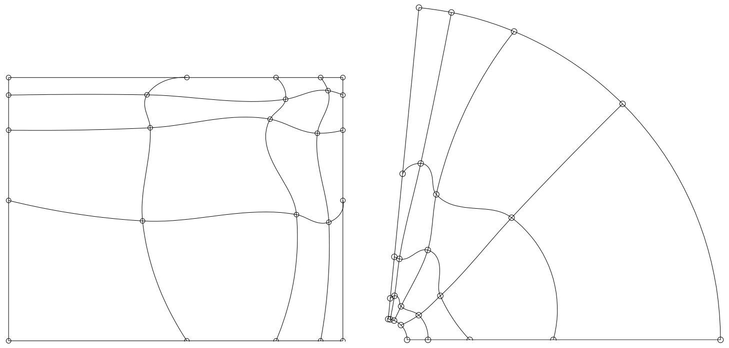

(a) (b)

Fig. 2. Our curve interpolant in two dimensions: cell edges of (a) a deformed non-uniform Cartesian grid, and (b) a deformed non-uniform polar grid.

remain fixed in the original uniform Cartesian grid. Fig. 3 (b) shows the edges of a part of a deformed uniform 3D cylindrical grid as constructed by our curve interpolant. The edges delineate three grid cells. The deformation is done by random shift of the eight nodes corresponding to the middle cell, while the remaining eight points remain fixed in the original uniform cylindrical grid. The edges for non-deformed Cartesian, cylindrical and spherical grids would be reconstructed exactly, see examples in (Liska, Shashkov and Swartz 1998) where an example of deformed spherical grid is presented, too.

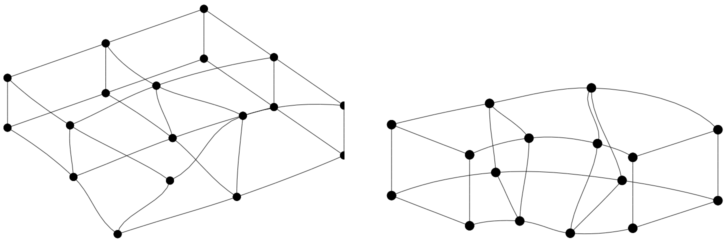

(a) (b)

Fig. 3. Cell edges of (a) a deformed uniform Cartesian grid, and (b) a deformed uniform cylindrical grid via our curve interpolant in three dimensions.

5. Surface interpolation via blended circular interpolation

As in the case of the circular curve interpolant we shall avoid treatment of boundaries and define the nodal interpolation problem for surfaces as follows. Given a matrix of $(m+2) \times (n+2)$ points $p_{i,j}$; $i = 0, 1, \ldots, n, n+1$; $j = 0, 1, \ldots, m, m+1$, we shall define a surface going through the points $p_{i,j}$, $i = 1, \ldots, n$; $j = 1, \ldots, m$. We do this by constructing our curve interpolant through each ”column” of data points, thereby defining $m+2$ space curves or ”wires” — and then interpolating corresponding points across the wires.

Algorithm 2 (*Construction of the surface interpolant*)

```
1. For $j := 0$ to $m+1$ do
     • construct the curve interpolant $I_j(t), t \in [1,n]$ of the points
       $p_{0,j}, p_{1,j}, \ldots, p_{n,j}, p_{n+1,j}$ by using Algorithm 1
2. fix $t = t_0 \in [1,n]$; and construct the curve interpolant $S_{t_0}(u), u \in [1,m]$ of the points
       $I_0(t_0), I_1(t_0), \ldots, I_m(t_0), I_{m+1}(t_0)$
   by using Algorithm 1
3. the surface $S$ of blended circular interpolants is given by the parameterization
   $S(t,u) = S_t(u),\ t \in [1,n],\ u \in [1,m]$
```

One can interpolate rows of nodes first instead of columns — that is, construct the initial curve interpolants (wires) through the points $p_{i,j}$ for fixed i (instead of fixed j as above). The resulting $i =$ constant wires (explicitly produced by one process, implicitly by the other) coincide mathematically, as do the $j =$ const. wires. Nevertheless, our numerical experiments indicate that the two processes produce surfaces that can differ between the wires.

One can also construct a truly two-dimensional type of surface blending that produces yet a third surface between the wires. Consider a 4×4 pile of data consisting of 16 points in three dimensions and assume that the four middle points are $p_{I,J}, p_{I+1,J}, p_{I,J+1}, p_{I+1,J+1}$. To each of these four points corresponds the 3×3 pile of neighboring points that is centered on it. Assume that we have already constructed a circular interpolation surface $S_{i,j}(t,u), i = I, I+1, j = J, J+1$ of each of these four 3×3 piles, and that each interpolation is parameterized by $(t,u) \in [0,1] \times [0,1]$ between the four middle points. Then a truly two-dimensional blend of these four interpolating surfaces between these four middle points is given by

$$S = (1-t)(1-u)\ S_{I,J} + t(1-u)\ S_{I+1,J} + (1-t)u\ S_{I,J+1} + tu\ S_{I+1,J+1}.$$

The remaining issue here is how to construct the circular interpolation surface $S_{i,j}(t,u)$ of the 3×3 pile of points centered on $p_{i,j}$. One possibility is to construct three circle/line segments ($j =$ const. wires), each going through three points with fixed index j but with i varying; parameterize them as we did in Section 2 (i. e., linear in the relative curve-length parameter t); and then for each value of t do circle/line interpolation in the "j direction" of the three points having that fixed value of t on those three $j =$ const. wires. Then, u becomes the relative distance along this curve. To get full directional symmetry one can average this circular interpolating surface of the three local $j =$ const. wires with the alternative circular interpolating surface of the three local $i =$ const. wires.

The following properties hold for all these blended surface interpolations.

6. Properties of a surface of blended successive circular interpolants

Here we list the properties of the surface of blended successive circular interpolants constructed by Algorithm 2 (or its variations described in the previous section).

Locality: The segment of the interpolant between the four points $p_{i,j}$, $i = I, I+1$, $j = J, J+1$ depends only on the 16 points $p_{i,j}$, $i = I-1, I, I+1, I+2$, $j = J-1, J, J+1, J+2$ for any I, J.

Exactness for the surfaces associated with a tensor product of data from Cartesian, cylindrical, or spherical coordinate lines: Suppose the 4×4 block of data being interpolated is the tensor product of four successive points each from two of the coordinate axes — and one fixed point from the third axis — of some set of spherical, cylindrical, or Cartesian coordinates. Then the "middle square" of the surface interpolant coincides with the corresponding piece of the coordinate surface. In particular, the "middle square" reproduces appropriate sections of (a) spheres from a spherical coordinate tensor products of discrete longitude with discrete latitude data (and fixed radius), (b) circular cones from a spherical coordinate tensor product of longitude with radius (and fixed latitude), (c) circular cylinders from a cylindrical coordinate tensor product of longitude with z-axis data (and fixed radius), as well as (d) planes from the remaining tensor products of spherical, cylindrical and Cartesian coordinate lines. Consequently, given as nodes the three-dimensional tensor product of discrete, not-necessarily-uniform set of spherical, cylindrical, or Cartesian coordinates; our surface interpolant reproduces the appropriate circular (or linear) edges that connect these nodes; and then spans them with the appropriate portions of spherical, conical, cylindrical, or planar surfaces. (Recall that a smooth surface interpolant that depended continuously on the data and that that handled these special cases exactly was the primary goal of this research.)

Approximation: The "wires" retain the third-order accuracy previously mentioned for our curve interpolant in three dimensions, but we have no proof of corresponding accuracy between wires.

Smoothness: The surface interpolant is G^1 continuous.

7. Examples of surfaces of blended successive circular interpolants

As examples of our surface interpolant we construct the cell faces of deformed uniform Cartesian, cylindrical and spherical grids. (The faces of non-deformed grids would be reconstructed exactly — for examples see (Liska, Shashkov and Swartz 1998)). The bold curves on the surfaces correspond to the grid edges that were shown earlier in Fig. 3.

Fig. 4 shows some grid faces of deformed uniform 3D Cartesian grids. Three deformed planes, corresponding to fixed indices of the grid, can be seen in Fig. 4(a). Fig. 4(b) shows the faces of four cells that we delineated earlier by their edges in Fig. 3(a).

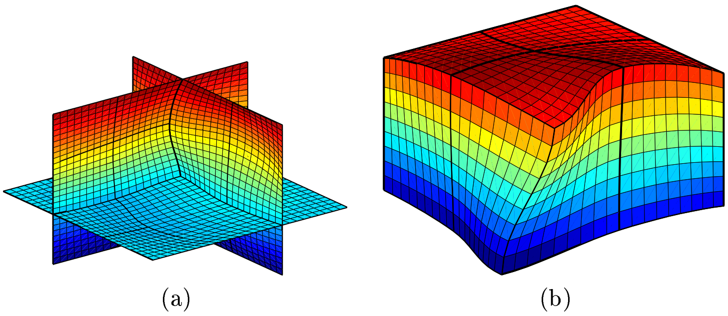
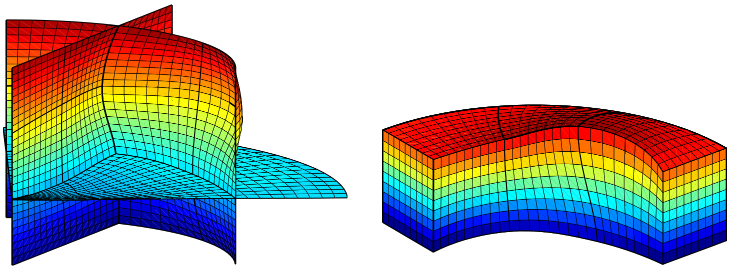

(a) (b)

Fig. 4. Cell faces of deformed uniform Cartesian grids using our surface interpolant: (a) three deformed planes, (b) four deformed cubes, cf. Fig. 3 (a).

Fig. 5 shows some grid faces of a deformed uniform, cylindrical grid. Two deformed planes and a deformed cylinder are in Fig. 5(a). Fig. 5(b) shows the faces of the three cells of the deformed grid that we delineated earlier by their edges in Fig. 3(b).



(a) (b)

Fig. 5. Cell faces of deformed uniform cylindrical grid using our surface interpolant: (a) a deformed cylinder and two deformed planes, and (b) three cells of a deformed uniform cylindrical grid; cf. Fig. 3 (b).

Fig. 6 shows some grid faces of a deformed uniform, spherical grid. A deformed sphere and a deformed cone are in Fig. 6(a). Fig. 6(b) shows the faces of the four cells of the deformed spherical grid.