SIAM REVIEW
Vol. 50, No. 4, pp. 723–752

Logically Rectangular Grids and Finite Volume Methods for PDEs in Circular and Spherical Domains*

Donna A. Calhoun†
Christiane Helzel‡
Randall J. LeVeque§

Abstract. We describe a class of logically rectangular quadrilateral and hexahedral grids for solving PDEs in circular and spherical domains, including grid mappings for the circle, the surface of the sphere, and the three-dimensional ball. The grids are logically rectangular and the computational domain is a single Cartesian grid. Compared to alternative approaches based on a multiblock data structure or unstructured triangulations, this approach simplifies the implementation of numerical methods and the use of adaptive refinement. A more general domain with a smooth boundary can be gridded by composing one of the mappings from this paper with another smooth mapping from the circle or sphere to the desired domain. Although these grids are highly nonorthogonal, we show that the high-resolution wave-propagation algorithm implemented in CLAWPACK can be used effectively to approximate hyperbolic problems on these grids. Since the ratio between the largest and smallest grids is below 2 for most of our grid mappings, explicit finite volume methods such as the wave-propagation algorithm do not suffer from the center or pole singularities that arise with polar or latitude-longitude grids. Numerical test calculations illustrate the potential use of these grids for a variety of applications including Euler equations, shallow water equations, and acoustics in a heterogeneous medium. Pattern formation from a reaction-diffusion equation on the sphere is also considered. All examples are implemented in the CLAWPACK software package and full source code is available on the web, along with MATLAB routines for the various mappings.

Key words. circular domain, spherical domain, grid transformations, finite volume methods, hyperbolic PDEs

AMS subject classifications. 65M06, 65M50, 65Y15, 35L60, 58J45

DOI. 10.1137/060664094

1. Introduction.

Uniform Cartesian grids are well suited to solving partial differential equations (PDEs) in rectangular regions. Logically rectangular quadrilateral grids work well for many problems where the domain is a deformed rectangle. For

*Received by the editors June 30, 2006; accepted for publication (in revised form) October 26, 2007; published electronically November 5, 2008. This work was supported in part by DOE grant DE-FC02-01ER25474 and NSF grant DMS-0106511 as well as the German Science Foundation through SFB 611.
http://www.siam.org/journals/sirev/50-4/66409.html

†Commissariat à l'Energie Atomique, Paris, France (calhoun@amath.washington.edu).

‡Department of Mathematics, Ruhr University Bochum, Bochum, Germany (christiane.helzel@rub.de).

§Department of Applied Mathematics, University of Washington, Seattle, WA (rjl@amath.washington.edu).

example, in an annulus a rectangular grid in polar r-θ coordinates can be used with periodic boundary conditions in the θ direction.

It is not so clear that quadrilateral grids are appropriate for solving problems in smooth domains such as a circle (i.e., a planar disk). Polar coordinates could again be used over a rectangular computational domain, but when polar coordinates are used over the full circle there may be numerical difficulties arising at the center of the circle where all of the radial grid lines meet at a single point in physical space. While it is possible to use finite volume methods in this context, these converging grid lines give rise to cells near the center that are much smaller than cells elsewhere in the domain. For explicit finite volume methods the CFL condition may then require a very small time step everywhere. Similar problems arise when using a latitude-longitude grid on the sphere.

In this paper we consider a family of quadrilateral and hexahedral grids for the numerical solution of PDEs on smooth "circular" domains in two dimensions, on the surface of the "sphere" as a two-dimensional manifold, and within a "ball" or "spherical shell" in three dimensions. We use quotes to indicate that the grids discussed in this paper for circles, spheres, and balls can be easily extended to other smooth domains that can be obtained by applying a mapping to the circle or sphere. We give one such example in section 8.3, but mostly concentrate on the basic mappings for circles and spheres.

The grids we discuss are logically rectangular quadrilateral grids in two dimensions indexed by (i, j) and hexahedral grids in three dimensions indexed by (i, j, k). (By hexahedral we mean simply six-sided; the faces are not generally planar.) These grids are obtained as the mapping of a single rectangle or rectangular block and do not require multiblock data structures. The ratio of largest to smallest cell is modest, i.e., for the sphere this ratio is less than two. On the other hand, the mapping functions are not smooth and the grids are far from orthogonal near the boundary of the computational domain, and so care must be taken when implementing solvers on these grids. In section 8 we consider a variety of test problems and show that, when suitable methods are used, very good results can be obtained. We focus primarily on hyperbolic problems using the CLAWPACK software [33], which implements high-resolution finite volume methods based on Riemann solvers in a framework that works well on arbitrary logically rectangular grids. The grids we discuss may be useful for other problems as well and in section 8.7 we consider an example of pattern formation on the sphere where a set of reaction-diffusion equations are solved.

The grid mappings we will discuss are most easily described as MATLAB functions. These are included in the text below and are also available on the webpage [14]. Complete source code for all of the numerical examples presented in section 8 is also available there, along with additional examples and figures and animations of some results. Python versions for some of the mapping functions are also available on the webpage.

2. Other Approaches.

There are many possible grids that can be used for computations in circular or spherical domains. We focus here on the development of logically rectangular grids because there are situations in which such grids are needed. In particular, the CLAWPACK software for hyperbolic problems requires logically rectangular grids, but has the advantage that a wide variety of hyperbolic problems can then be robustly solved, as illustrated in our examples in section 8. Moreover, adaptive mesh refinement is available in this package, based on refinement on logically rectangular patches of the domain. An example where this is used is given in section 8.5, with others on the webpage [14].

Among logically rectangular grids, polar coordinates are an obvious choice for circular domains as well as the sphere, but have the problem of the singularity at the center where all grid lines coalesce (so that the grid is not truly quadrilateral everywhere). The disparity in the sizes of grid cells can lead to severe time step restrictions when using explicit methods due to the small cells near the center. Conversely, the cells are highly stretched in the θ direction near the edges relative to the center, which may lead to poor accuracy.

Grids on the surface of the sphere can be obtained by inscribing a suitable polyhedron inside the sphere and using the gnomic projection (projection from the inside of the sphere) to map the surface of the polyhedron to the surface of the sphere. For example, if an icosahedron is inscribed in the sphere and projected to the surface, a quasi-uniform cell distribution on the sphere is obtained. The faces of the icosahedron can be further triangulated to obtain a finer mesh. This approach was introduced by Sadourny, Arakawa, and Mintz [45] in 1968. Logically quadrilateral grids can be obtained by using a polyhedron with rectangular faces. The most elementary such polyhedron is the cube, which creates six large rectangular elements which can be further refined. There are eight singular vertices where three grid cells come together instead of four. Computing on such a grid can be done if boundary data from each grid is properly transferred to the boundaries of neighboring grids. Rossmanith has explored the use of such grids for finite volume methods [42], [43]. He found that the wave-propagation algorithms of CLAWPACK are quite robust on these grids and have no difficulties at the singular vertices, although properly transferring boundary data between the blocks is a bit subtle and the multiblock implementation makes it harder to apply adaptive mesh refinement. This type of grid on the sphere is sometimes called a "gnomonic grid" and was also introduced by Sadourny; see [44]. It has more recently been used by other researchers for geophysical flow problems as well, e.g., [1], [40], [41]. This kind of discretization can also be used for a circular domain, by mapping five rectangular grids to a circle; see [42].

Another approach for discretizing the sphere that has recently been proposed is the yin-yang grid; see [28], for example. Two logically rectangular latitude-longitude-type grids overlap to cover the sphere and eliminate the pole problem. A related approach is to use a true latitude-longitude grid over the midlatitudes with overlapping rectangular patches near the two poles. The disadvantage of these grids is that some form of interpolation must be used in the overlap regions. If solutions of conservation laws are approximated on such grids, the interpolation leads to a loss of conservativity. In [31], a combined grid consisting of a reduced longitude-latitude grid and a stereographic grid at the polar caps was used that avoids the need for interpolation but still requires different grids that have to be merged together.

Logically rectangular grids can also be obtained by variational (or PDE) methods. There exists extensive literature on such methods [15, 37]. Geometric considerations are used to derive functionals for controlling the smoothness, the area or volume of grid cells, or the degree of nonorthogonality. It would be interesting to apply such techniques to the grids presented in this paper in order to generate smoother meshes that still maintain reasonable cell-size ratios and can be represented by a single logically Cartesian grid in computational space.

One way to use rectangular grids in general nonrectangular regions is to employ a "cut-cell Cartesian" or "embedded boundary" approach, in which a geometry is embedded in a Cartesian grid and grid cells are arbitrarily cut by the boundary. Cut-cell methods are attractive for complex geometries for several reasons. Grid generation is basically trivial once a general procedure has been developed to specify the geometry. The grid is a uniform Cartesian grid away from the boundary, which typically simplifies the numerical method and may improve its accuracy over most of the domain. On the other hand, some special treatment of the cut cells is required to maintain good accuracy and stability properties.

We have worked on the development of such methods in the past (e.g., [5], [6], [9], [11], [13], [25], [35]) and believe that it is a good approach for complex geometries. This approach has been advanced by many other researchers as well, for hyperbolic equations (e.g., [16], [17], [19], [20]), elliptic equations (e.g., [27], [38], [39]), and incompressible Navier–Stokes equations (e.g., [2], [32]).

In this paper, we focus our attention on simple domains in two space dimensions where the use of a mapped coordinate system is natural and often easier to work with computationally than embedded boundaries or cut cells.

3. Quadrilateral Grids in the Circle.

We begin by discussing several possible mappings to the circle. Similar mappings appear elsewhere in the literature, e.g., [3], [26], though we have not seen this approach explored systematically in the form of the specific mappings we introduce below. We also test these grids in the context of high-resolution finite volume methods for hyperbolic equations.

3.1. The Radial Projection Mapping.

Figure 3.1 shows one possible quadrilateral grid obtained by a simple mapping from a Cartesian grid on a square domain. Two lines of grid cells are shaded to help visualize the mapping. This mapping is obtained by taking each point on the square $[-1, 1] \times [-1, 1]$ and projecting it radially to the unit circle. All points along the radial line through this point are rescaled linearly and so the square is contracted into a circle. This mapping has the property that the grid cells along the 45 degree lines end up being nearly triangular, with two edges of the quadrilateral cell nearly colinear even though in computational space these edges are orthogonal. This grid may not be suitable for use with standard finite difference methods, but still works well when appropriate methods are used, such as the high-resolution wave-propagation algorithms [36] for hyperbolic problems, as demonstrated in section 8. The ratio of largest cell size to smallest cell size is roughly 2 for grids of this form, which results in a much less stringent restriction on the time step than would result from a polar grid, for example, where grid cells near the center are much smaller than the average cell size (see Table 3.1).

The grid shown in Figure 3.1 has the interesting property that one set of cell edges lies along concentric circles and the grid is similar to a polar coordinates grid in this regard. However, note that each concentric circle of cells in the mapped grid now corresponds to a concentric square in the computational domain rather than a single grid line.

This grid is created by the following mapping, implemented in MATLAB in the form needed for the CLAWPACK software plotting routines. The function `mapc2p.m` is used to map an arbitrary point (xc,yc) in the computational domain to the point (xp,yp) in the physical domain.

```
function [xp,yp] = mapc2p(xc,yc)
r1 = 1;                 % map [-1,1] x [-1,1] to circle of radius r1
d = max(abs(xc),abs(yc));% value on diagonal of computational grid
r = sqrt(xc.^2 + yc.^2);
r = max(r, 1e-10);      % to avoid divide by zero at origin
xp = r1 * d .* xc./r;
yp = r1 * d .* yc./r;
```

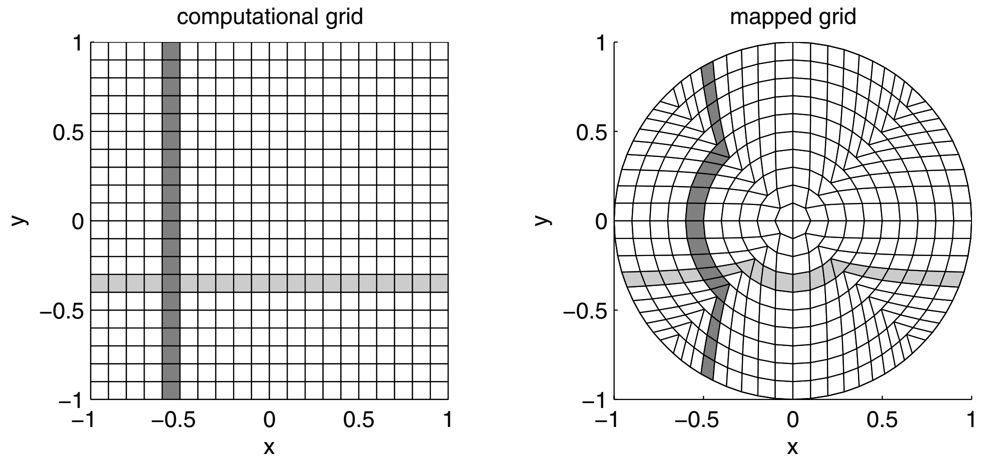

Fig. 3.1 *Rectangular computational grid (left) and the mapped quadrilateral grid in a circular domain (right). The same two rows of grid cells are shaded in each figure.*

To quickly view this or other grids described in this paper in MATLAB, the following commands can be used:

```
n = 40;   % plots an n by n grid of quadrilateral cells
[xc,yc] = meshgrid(linspace(-1,1,n+1), linspace(-1,1,n+1));
[xp,yp] = mapc2p(xc,yc);
pcolor(xp,yp,0*yp);   colormap([1 1 1]);   daspect([1 1 1])
```

These and other MATLAB scripts in this paper may be found on the webpage [14].

The "radial projection" mapping described above has the virtue of simplicity. However, it gives rather skewed mesh cells along the diagonal and does not have much flexibility. In section 3.2 we define a class of smoother maps that are generally better to use in practice.

3.2. A Smoother Circle Mapping. A smoother and more flexible family of mappings can be defined by the following approach. As in the case of the radial projection mapping, the mapping is defined on each of the four sectors determined by splitting the computational square on the two diagonals. Consider a computational point (x_c, y_c) that satisfies $x_c > 0$ and $|y_c| < x_c \equiv d$, for example, which lies in the eastern sector. The vertical line segment between $(d, -d)$ and (d, d) is mapped to a circular arc of radius $R(d)$ that passes through the points $(D(d), -D(d))$ and $(D(d), D(d))$. The mapping is fully defined by the functions $R(d)$ and $D(d)$ as discussed further below. It is easy to compute that the center of this circle of radius $R(d)$ is at $(x_0, y_0) = (D(d) - \sqrt{R(d)^2 - D(d)^2},\ 0)$. The point (x_c, y_c) is then mapped to

$$\begin{aligned} y_p &= y_c D(d)/d, \\ x_p &= D(d) - \sqrt{R(d)^2 - D(d)^2} + \sqrt{R(d)^2 - y_p^2}. \end{aligned} \tag{3.1}$$

Similar mappings are used in the other four sectors; for example, points on the horizontal line segment between $(-d, d)$ and (d, d) are mapped to points on the circular arc of radius $R(d)$ passing through $(-D(d), D(d))$ and $(D(d), D(d))$.

The full mapping is specified in MATLAB as follows:

```
  function [xp,yp] = mapc2p(xc,yc)
  r1 = 1;                    % map [-1,1] x [-1,1] to circle of radius r1

  d = max(abs(xc),abs(yc));  % value on diagonal of computational grid
  d = max(d, 1e-10);         % to avoid divide by zero at origin

  D = r1 * d/sqrt(2);        % mapping d to D(d) from (3.2)
  R = r1 * d;                % mapping d to R(d) from (3.2)
% R = r1 * ones(size(d));    % alternative for (3.3)

  center = D - sqrt(R.^2 - D.^2);
  xp = D./d .* abs(xc);
  yp = D./d .* abs(yc);

  ij = find(abs(yc)>=abs(xc));
  yp(ij) = center(ij) + sqrt(R(ij).^2 - xp(ij).^2);
  ij = find(abs(xc)>=abs(yc));
  xp(ij) = center(ij) + sqrt(R(ij).^2 - yp(ij).^2);

  xp = sign(xc) .* xp;
  yp = sign(yc) .* yp;
```

The mapping is defined once the functions $D(d)$ and $R(d)$ for $0 \leq d \leq 1$ are determined. To map $[-1,1] \times [-1,1]$ to a circle of radius r_1, we want $R(1) = r_1$ and the function $D(d)$ to be be monotonically increasing with $D(0) = 0$ and $D(1) = r_1/\sqrt{2}$. If $R(d)$ varies too rapidly, then grid lines may cross, but various natural choices for $R(d)$ work well. One obvious choice is

$$D(d) = r_1 d/\sqrt{2}, \qquad R(d) = r_1 d. \tag{3.2}$$

This maps concentric squares $\max(|x_c|, |y_c|) = d$ to concentric circles of radius $R(d)$, and is similar to the radial projection grid of section 3.1 in this respect, but is less skewed along the axis. Figure 3.2(a) shows a portion of this grid (only the mapping of the quadrant $[0,1] \times [0,1]$ is shown in this figure since the grid is symmetrically reflected in the other quadrants). The fact that grid lines follow concentric circles may be useful in modeling radially layered circles or cylinders in three dimensions.

It is possible to obtain a grid that is more nearly Cartesian near the center by choosing $R(d)$ to be bounded away from 0 as $d \to 0$. For example,

$$D(d) = r_1 d/\sqrt{2}, \qquad R(d) \equiv r_1 \tag{3.3}$$

gives the grid of Figure 3.2(b), in which each line segment of concentric squares in computational space is mapped to a circular arc with radius of curvature r_1.

A nonlinear mapping $D(d)$ can be used to redistribute points in the radial direction. For example,

$$D(d) = r_1 d(2-d)/\sqrt{2}, \qquad R(d) \equiv r_1 \tag{3.4}$$

gives the mapping shown in Figure 3.2(c), which has grid cells clustered near the boundary. This could be useful for solving problems with a boundary layer and will also prove useful when extending this circle mapping to the surface of the sphere, as is done in section 4.

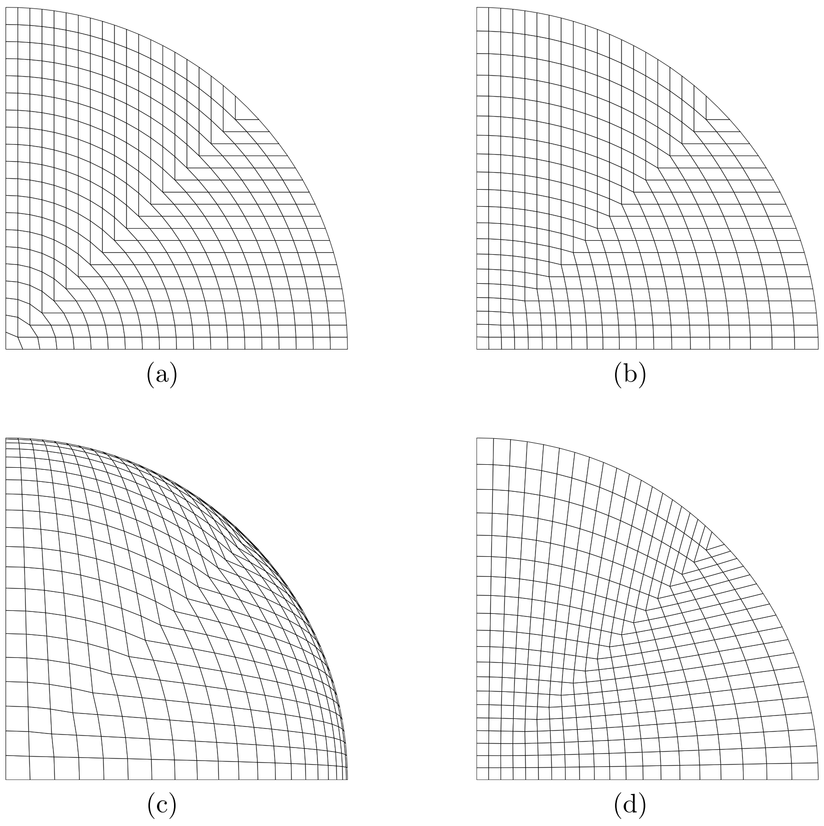

Fig. 3.2 *Four grid mappings discussed in section* 3.2. *Only the positive quadrant is shown in each case.*

3.3. Interpolated Grids.

There is another approach to obtaining a smoother grid on the circle that is somewhat easier to implement than the above procedure, and which also extends easily to mappings in the ball. Start with the radial projection grid of section 3.1 and then use a convex combination of this grid and a pure Cartesian grid, weighted entirely towards the Cartesian grid near the center and towards the radial projection grid near the boundary. This can be accomplished by the following:

```
function [xp,yp] = mapc2p(xc,yc)
r1 = 1;   % map [-1,1] x [-1,1] to circle of radius r1
d = max(abs(xc),abs(yc));
r = sqrt(xc.^2 + yc.^2);
r = max(r,1e-10);
xp = r1 * d .* xc./r;
yp = r1 * d .* yc./r;
w = d.^2;
xp = w.*xp + (1-w).*xc/sqrt(2);
yp = w.*yp + (1-w).*yc/sqrt(2);
```

The weighting function is chosen to be $w = d^2$, which seems to give a nice grid, but other functions $w(d)$ varying monotonically from 0 to 1 could be used. This approach gives the grid shown in Figure 3.2(d).

Table 3.1 *The ratios of largest to smallest grid cells for the $N \times N$ circular disk grid mappings. The ratios for the polar grid were computed analytically. The areas of cells in the ghost cell region are not included for any grids.*

	Cell ratios on the disk			
N	Polar grid	Radial proj.	Constant curv.	Convex combi.
100	199	1.979899	1.966483	3.117242
400	799	1.994994	1.991504	3.158103
1600	3199	1.998750	1.997869	3.168213

Table 3.1 gives the ratios of the largest to the smallest grid cell for several quadrilateral grids of the disk described above. For comparison we also give the ratios of the polar grid.

4. Grids on the Surface of a Sphere. If the mapping of section 3.2 with the choice (3.4) is used, then the grid shown in Figure 3.2(c) is obtained. If we now set

```
zp = sqrt(1 - (xp.^2 + yp.^2))
```

then the points (xp,yp,zp) lie on the upper hemisphere. We can apply a similar mapping to points in the computational domain $[-3,-1]\times[-1,1]$ to map these points to the lower hemisphere. The two mappings together map the rectangle $[-3,1]\times[-1,1]$ to the surface of the sphere. Figure 4.1(a) shows the resulting grid. This is easily accomplished in MATLAB by

```
function [xp,yp] = mapc2p(xc,yc)
ijlower = find(xc<-1);           % indices of points on lower hemisphere
xc(ijlower) = -2 - xc(ijlower);  % flip across the line x = -1
%   compute xp and yp by mapping [-1,1]x[-1,1] to the unit circle
%   using the mapping of section 3.2 with (3.4)
zp = sqrt(1 - (xp.^2 + yp.^2));
zp(ijlower) = -zp(ijlower);      % negate z in lower hemisphere
```

Proper communication of data between the hemispheres requires that periodic boundary conditions be used in the x direction and that appropriate boundary conditions be used along the top and bottom of the rectangular domain where the segment $-3 < x < -1$ is connected to the segment $-1 < x < 1$ at the same boundary. These

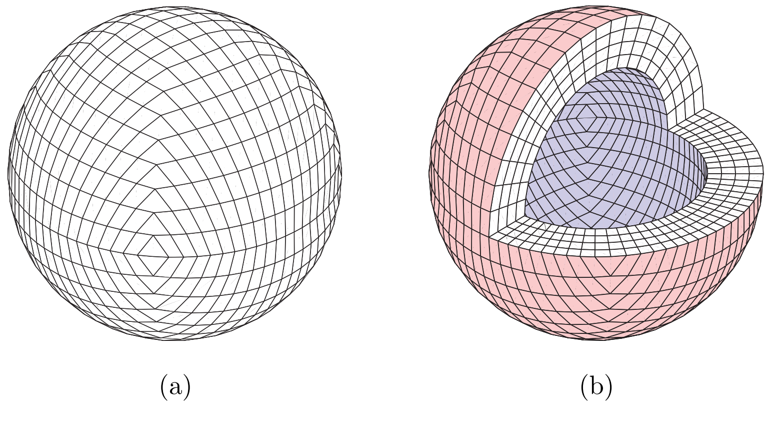

Fig. 4.1 (a) 40×20 *grid on the surface of the sphere.* (b) $40 \times 20 \times 4$ *grid in a spherical shell.*

Table 4.1 *The ratios of largest to smallest grid cells for the $2N \times N$ sphere mappings. The ratios for the latitude-longitude grid were computed analytically.*

	Cell ratios on the sphere	
N	Latitude-longitude grid	New grid
100	63.6567	1.659386
400	254.6466	1.676421
1600	1018.5913	1.680479

are easy to implement using the standard ghost cell technique, as described, for example, in Chapter 7 of [36] (and are implemented in the source code for the example of section 8.6).

Table 4.1 gives the cell-size ratios for the new sphere grid and the latitude-longitude grid.

5. Hexahedral Grids in Three Dimensions.

The mappings described in the previous sections can be extended to a variety of geometries in three space dimensions, including spherical shells, the full sphere, and cylinders or tori.

5.1. Spherical Shells.

The two-dimensional grid on the surface of a sphere that was described in section 4 can be extended radially outwards some distance orthogonal to the surface, yielding a three-dimensional grid that occupies $[-3,1] \times [-1,1] \times [0,1]$ in computational space and maps to the spherical shell between radii r_1 and r_2, as shown in Figure 4.1(b). This can be accomplished by a three-dimensional mapc2p function of the form

```
function [xp,yp,zp] = mapc2p(xc,yc,zc)
% map [-3,1]x[-1,1]x[0,1] to a spherical shell r1 <= r <= r1+dr
r1 = 1;  dr = 0.5;
ijlower = find(xc<-1);           % indices of points on lower hemisphere
xc(ijlower) = -2 - xc(ijlower);  % flip across the line x = -1
% compute xp and yp by mapping [-1,1]x[-1,1] to unit circle using
% the mapping of section 3.2 with (3.4)
zp = sqrt(1 - (xp.^2 + yp.^2));
zp(ijlower) = -zp(ijlower);      % negate z in lower hemisphere
Rz = r1 + zc*dr;                 % radius based on zc
xp = Rz.*xp;
yp = Rz.*yp;
zp = Rz.*zp;
```

This might be useful for problems in atmospheric flow, for example, or for elastodynamics in spherical shells.

5.2. Hexahedral Grids in the Ball.

The spherical shell grid of the previous section could be used for a solid spherical ball by taking the inner radius to be r1=0. In this case all the grid lines in the zc direction meet at the origin and this grid has similar problems at the origin as the standard spherical polar coordinates.

Alternatively, we can construct mappings that are three-dimensional generalizations of the circle mappings considered in section 3. The two-dimensional radial projection circle mapping of section 3.1 is easily extended to map the computational cube $[-1,1] \times [-1,1] \times [-1,1]$ to a ball:

```
function [xp,yp,zp] = mapc2p(xc,yc,zc)
r1 = 1;  % map [-1,1] x [-1,1] x [-1,1] to sphere of radius r1
d = max(max(abs(xc),abs(yc)),abs(zc));
```

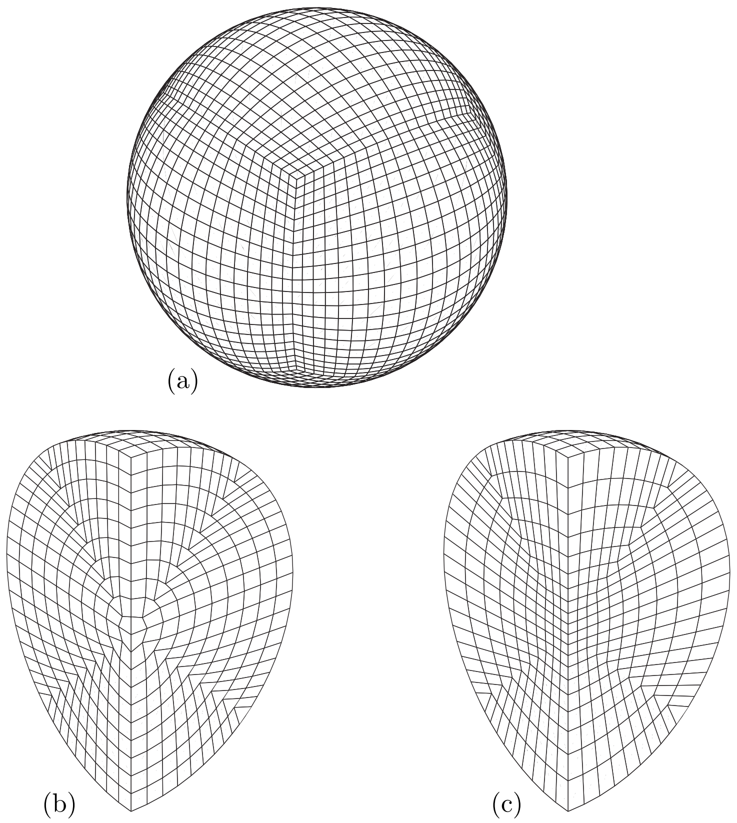

Fig. 5.1 *Hexahedral grid in a ball.* (a) *Mapping of the outer edges of the computational cube.* (b) *Cut-away view of one quarter of ball.* (c) *Cut-away view when convex combination with Cartesian grid is applied.*

```
    r = sqrt(xc.^2 + yc.^2 + zc.^2);
    r = max(r,1e-10);
    xp = r1 * d .* xc./r;
    yp = r1 * d .* yc./r;
    zp = r1 * d .* zc./r;
```

Figure 5.1(a) shows the surface of the resulting hexahedral grid. Note that on the surface this mapping looks like the cubed sphere surface grid described in section 2, but these are now faces of hexahedra in the ball. Figure 5.1(b) shows a cut-away view, obtained by mapping $[0, 1] \times [0, 1] \times [-1, 1]$ to a quarter of the ball. Concentric cubes in computational space are mapped to concentric spheres in physical space, which may be useful in solving problems in a layered sphere, e.g., in global seismology.

As in the case of the circle, this mapping can be smoothed out by taking a convex combination with a Cartesian grid as described at the end of section 3.2. To the above function add the lines

```
    w = d.^2;
    xp = w.*xp + (1-w).*xc/sqrt(3);
    yp = w.*yp + (1-w).*yc/sqrt(3);
    zp = w.*zp + (1-w).*zc/sqrt(3);
```

Figure 5.1(c) shows a cut-away view of the resulting hexahedral grid.

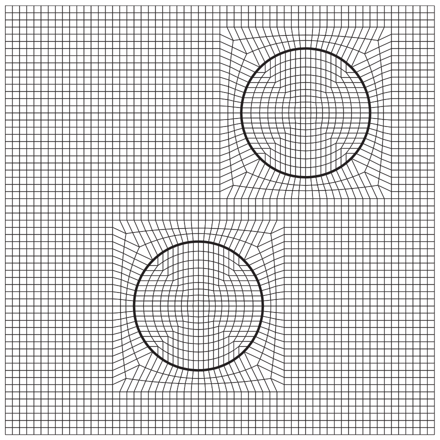

Fig. 6.1 *A quadrilateral grid for a square domain with two circular inclusions.*

6. Circular Inclusions in Rectangular Grids. For problems in more complicated geometry it may be useful to have a quadrilateral grid that meshes a square domain containing a circular inclusion.

Suppose we wish to map the computational unit square to a square domain $[-r_2, r_2] \times [-r_2, r_2]$ with an embedded circle of radius $r_1 < r_2$. The following $D(r)$ and $R(r)$ functions have been found to give a family of smooth mappings that work over a fairly wide range of r_1/r_2:

$$(6.1) \qquad D(d) = r_1 d/\sqrt{2}, \qquad R(d) = \begin{cases} r_1 & \text{if } d \le r_1/r_2, \\ r_1 \left(\frac{1-r_1/r_2}{1-d}\right)^{r_2/r_1+1/2} & \text{if } d > r_1/r_2. \end{cases}$$

Outside the circular inclusion the grid lines are mapped to circles of radius $R(d) \to \infty$ as $d \to 1$. Other choices of $R(d)$ for $d < r_1/r_2$ can be used as discussed in section 3.2 to give different mappings within the embedded circles. For example, $R(d) = r_2 d/\sqrt{2}$ produces concentric circular grid lines in the embedded circles. This may be useful for modeling hollow cylinders or other radially laminated materials in a square domain. Smooth inclusions with shapes other than circular can be incorporated by composing a further mapping as discussed in section 8.3.

Domains with several inclusions can be handled by piecing together several grids of the above form. Figure 6.1 shows an example. See section 8.4 for some numerical results on this grid. This idea can also be extended to the three-dimensional case, where a ball might be embedded in a Cartesian grid.

7. Finite Volume Methods on Mapped Grids. In order to discretize PDEs on mapped grids one can either transform the equations into equations in computational coordinates and discretize the transformed equations or one can discretize directly in physical space and work with reference to a Cartesian frame. We have found that the latter approach works well for the approximation of hyperbolic problems on the kind of grids considered in this paper; see section 7.1 for a description of the algorithm. In order to approximate diffusion processes we instead discretize the transformed equations; see section 7.2.

7.1. The Wave-Propagation Algorithm for Hyperbolic Equations. Here we will briefly review the discretization of hyperbolic problems on mapped grids using the wave-propagation algorithm; see Chapter 23 of [36] for more details.

We consider systems of conservation laws, i.e., first order PDEs in divergence form,

$$\partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = \mathbf{0}, \tag{7.1}$$

with suitable initial conditions for $\mathbf{q}$. Here $\mathbf{q}$ is a vector of conserved quantities that maps from $\mathcal{X} \times [0,\infty)$ to $\mathcal{U}$, where $\mathcal{X}$ and $\mathcal{U}$ are open subsets of $\mathbb{R}^d$ (i.e., the physical space) and $\mathbb{R}^m$ (i.e., the state space), respectively. The matrix $\mathbf{f}$ is a flux matrix that maps from $\mathcal{U}$ to $\mathbb{R}^{m\times d}$ and

$$\nabla \cdot \mathbf{f}(\mathbf{q}) = \sum_{i=1}^{d} \partial_{x_i} \mathbf{f}_\mathbf{i}(\mathbf{q}) = \sum_{i=1}^{d} \mathbf{A}_\mathbf{i}(\mathbf{q}) \partial_{x_i} \mathbf{q},$$

where $\mathbf{f}_\mathbf{i}$ is the ith column of the flux matrix $\mathbf{f}$ and $\mathbf{A}_\mathbf{i}$ is the Jacobian matrix of $\mathbf{f}_\mathbf{i}$ with respect to $\mathbf{q}$. We can rewrite (7.1) in the more general quasi-linear form

$$\partial_t \mathbf{q} + \sum_{i=1}^{d} \mathbf{A}_i(\mathbf{q}) \partial_{x_i} \mathbf{q} = \mathbf{0}. \tag{7.2}$$

The system is called hyperbolic if any linear combination of the flux Jacobian matrices is diagonalizable with real eigenvalues. The eigenvalues describe wave speeds and the eigenvectors lead to a decomposition of the conserved quantities into waves.

A finite volume method can be applied on any control volume C using the integral form of the conservation law

$$\frac{d}{dt} \int_C \mathbf{q}\, d\mathbf{x} = - \int_{\partial C} \mathbf{f}(\mathbf{q}) \cdot \mathbf{n}\, ds,$$

where $\mathbf{n}$ is the outward-pointing unit normal vector at the boundary ∂C and $\mathbf{f} \cdot \mathbf{n}$ is the flux normal to the boundary. This leads to a finite volume method of the form

$$Q^{n+1} = Q^n - \frac{\Delta t}{|C|} \sum_{j=1}^{N} h_j \breve{F}_j^n,$$

where $\breve{F}_j^n$ represents a numerical approximation to the average normal flux across the jth side of the grid cell, N is the number of sides, and h_j is the length of the jth side (in two dimensions) or the area of the cell interface (in three dimensions) measured in physical space. We could now apply any unstructured grid method (see, for instance, [29]) to approximate hyperbolic problems on the grids discussed in this paper.

In the following we will, however, make use of the fact that our mapped grids are logically rectangular and we will restrict our consideration to the two-dimensional case ($d = 2$), i.e., the physical space could be the circle or the star shaped domain of Figure 8.4. The computational space is a square. In section 8.6, we show how the method can be used to approximate hyperbolic equations on the sphere. Furthermore, the method has been implemented in the three-dimensional case; see section 8.2 for calculations in a ball.

On a curvilinear grid, the finite volume method can be written in the form

$$Q_{ij}^{n+1} = Q_{ij}^n - \frac{\Delta t}{\kappa_{ij} \Delta x_c} \left(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j} \right) - \frac{\Delta t}{\kappa_{ij} \Delta y_c} \left(G_{i,j+\frac{1}{2}} - G_{i,j-\frac{1}{2}} \right), \tag{7.3}$$

where Δx_c, Δy_c denote the equidistant discretization of the computational domain, $\kappa_{ij} = |C_{ij}|/\Delta x_c \Delta y_c$ is the area ratio between the area of the grid cell in physical space and the area of a computational grid cell, and

$$\begin{aligned} F_{i-\frac{1}{2},j} &= \gamma_{i-\frac{1}{2},j} \breve{F}_{i-\frac{1}{2},j}, \\ G_{i,j-\frac{1}{2}} &= \gamma_{i,j-\frac{1}{2}} \breve{F}_{i,j-\frac{1}{2}} \end{aligned} \tag{7.4}$$

are fluxes per unit length in computational space. Here the length ratios are defined as $\gamma_{i-\frac{1}{2},j} = h_{i-\frac{1}{2},j}/\Delta y_c$ and $\gamma_{i,j-\frac{1}{2}} = h_{i,j-\frac{1}{2}}/\Delta x_c$.

Approximations to the intercell fluxes are obtained by solving Riemann problems for a one-dimensional system of equations normal to the interface. For an isotropic system of equations such as the Euler, shallow water, or acoustics equations, the flux $\breve{F}$ at a grid cell interface can be calculated by first rotating the velocity components of the cell average values of the quantity $\mathbf{q}$ on both sides of the interface into the direction normal and tangential to the interface. With these modified data, a one-dimensional Riemann problem is solved and a flux is calculated. Finally the momentum components of the flux are rotated back to obtain the flux in physical space.

To be more specific, we now consider the grid cell interface $(i-\frac{1}{2}, j)$ between the cells $(i-1, j)$ and (i, j). Let $\mathbf{n_{i-\frac{1}{2},j}} = (n^1, n^2)$ and $\mathbf{t_{i-\frac{1}{2},j}} = (t^1, t^2)$ be the normalized normal and tangential vectors to this cell interface. Then the rotation matrix which rotates the velocity components (e.g., for the shallow water or the acoustics equation) has the form

$$R_{i-\frac{1}{2},j} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & n^1 & n^2 \\ 0 & t^1 & t^2 \end{pmatrix}.$$

The wave-propagation algorithm consists of the following steps:

1. Determine $Q^{+\bullet}_{i-1,j}$ and $Q^{-\bullet}_{i,j}$ by rotating the velocity components, i.e.,

$$Q^{+\bullet}_{i-1,j} = R_{i-\frac{1}{2},j} Q_{i-1,j}, \quad Q^{-\bullet}_{i,j} = R_{i-\frac{1}{2},j} Q_{i,j}.$$

   Later we will also use the notation $Q^{\bullet-}_{i,j} = R_{i,j-\frac{1}{2}} Q_{i,j}$ and $Q^{\bullet+}_{i,j} = R_{i,j+\frac{1}{2}} Q_{i,j}$.
2. Solve the Riemann problem for the one-dimensional conservation law

$$\partial_t \mathbf{q} + \nabla \cdot \mathbf{f_1} = \mathbf{0}$$

   with initial values

$$\mathbf{q}(x,0) = \begin{cases} Q^{+\bullet}_{i-1,j} & : \quad x < x_{i-\frac{1}{2}}, \\ Q^{-\bullet}_{i,j} & : \quad x \geq x_{i-\frac{1}{2}}. \end{cases}$$

   This results in waves $\breve{\mathcal{W}}^p_{i-\frac{1}{2},j}$ that are moving with speeds $\breve{s}^p_{i-\frac{1}{2},j}$, $p = 1, \ldots, M_w$, where M_w is the number of waves; see [36] for details.
   The waves lead to a decomposition of the jump in the conserved quantities, i.e.,

$$Q^{-\bullet}_{ij} - Q^{+\bullet}_{i-1,j} = \sum_{p=1}^{M_w} \breve{\mathcal{W}}^p_{i-\frac{1}{2},j}.$$

The solution of the Riemann problem at the interface can be expressed in the form

$$\breve{Q}^*_{i-\frac{1}{2},j} = Q^{+\bullet}_{i-1,j} + \sum_{p:s^p<0} \breve{\mathcal{W}}^p_{i-\frac{1}{2},j}.$$

3. Define (in analogy to (7.4)) scaled speeds

$$s^p_{i-\frac{1}{2},j} = \gamma_{i-\frac{1}{2},j}\breve{s}^p_{i-\frac{1}{2},j}$$

and rotate waves back to Cartesian coordinates by

$$W^p_{i-\frac{1}{2},j} = R^T_{i-\frac{1}{2},j}\breve{W}^p_{i-\frac{1}{2},j}, \quad p = 1, \dots, M_w.$$

4. The waves and associated speeds are used to calculate left- and right-moving fluctuations in the form

$$\mathcal{A}^\pm \Delta Q_{i-\frac{1}{2},j} = \sum_{p=1}^{M_w} (s^p_{i-\frac{1}{2},j})^\pm \mathcal{W}^p_{i-\frac{1}{2},j},$$

with $(s^p_{i-\frac{1}{2},j})^+ = \max(s^p_{i-\frac{1}{2},j}, 0)$ and $(s^p_{i-\frac{1}{2},j})^- = \min(s^p_{i-\frac{1}{2},j}, 0)$.
An alternative description of the fluctuations is given by

$$\mathcal{A}^+\Delta Q_{i-\frac{1}{2},j} = f_{i-\frac{1}{2},j}(Q^{-\bullet}_{i,j}) - f^*_{i-\frac{1}{2},j},$$
$$\mathcal{A}^-\Delta Q_{i-\frac{1}{2},j} = f^*_{i-\frac{1}{2},j} - f_{i-\frac{1}{2},j}(Q^{+\bullet}_{i-1,j}),$$

where $f_{i-\frac{1}{2},j}(\cdot) = \gamma_{i-\frac{1}{2},j}R^T_{i-\frac{1}{2},j}\mathbf{f_1}(\cdot)$ and $f^*_{i-\frac{1}{2},j} = f_{i-\frac{1}{2},j}(\breve{Q}^*_{i-\frac{1}{2},j})$.

5. In an analogous way, up- and down-moving fluctuations $\mathcal{B}\Delta Q^\pm_{i,j-\frac{1}{2}}$ at the interface $(i, j - \frac{1}{2})$ are calculated.

Using these fluctuations, a Godunov-type finite volume method can be defined that calculates the cell average of the conserved quantities at the next time step by adding to the cell average of the conserved quantities at the previous time the contribution of all waves that are moving into the grid cell, i.e.,

(7.5)
$$Q^{n+1}_{ij} = Q^n_{ij} - \frac{\Delta t}{\kappa_{ij}\Delta x_c}\left(\mathcal{A}^+\Delta Q_{i-\frac{1}{2},j} + \mathcal{A}^-\Delta Q_{i+\frac{1}{2},j}\right) - \frac{\Delta t}{\kappa_{ij}\Delta y_c}\left(\mathcal{B}^+\Delta Q_{i,j-\frac{1}{2}} + \mathcal{B}^-\Delta Q_{i,j+\frac{1}{2}}\right).$$

The speeds and limited versions of the waves are used to calculate second order correction terms. These terms are added to the update (7.5) in flux difference form. At the interface $(i - \frac{1}{2}, j)$ the correction flux has the form

$$\tilde{F}_{i-\frac{1}{2},j} = \frac{1}{2}\sum_{p=1}^{M_w} |s^p_{i-\frac{1}{2},j}| \left(1 - \frac{\Delta t}{\kappa_{i-\frac{1}{2},j}\Delta x_c}|s^p_{i-\frac{1}{2},j}|\right) |s^p_{i-\frac{1}{2},j}|\tilde{\mathcal{W}}^p_{i-\frac{1}{2},j}, \tag{7.6}$$

where $\kappa_{i-\frac{1}{2},j} = (\kappa_{i-1,j} + \kappa_{ij})/2$. To avoid oscillations near shock waves and contact discontinuities, a wave limiter is applied leading to limited waves $\tilde{\mathcal{W}}^p$.

Furthermore, a transverse wave propagation is included as part of the second order correction terms. The left-going and right-going fluctuations are each split into two transverse fluctuations (up-going and down-going); see [36].

One advantage of the wave-propagation algorithm over other finite volume methods is that it can also be applied to hyperbolic problems in variable-coefficient quasilinear form. A possible example is acoustics in heterogeneous media, which is considered in section 8.4. However, if applied to equations in divergence form, it is required that the method leads to a conservative approximation. To show that the method is conservative, we multiply (7.5) by κ_{ij} and sum over all grid cells:

$$\begin{aligned}
&\sum_{i=1}^{l}\sum_{j=1}^{m}\kappa_{i,j}Q_{i,j}^{n+1} = \sum_{i,j}\kappa_{i,j}Q_{i,j}^{n} \\
&-\frac{\Delta t}{\Delta x_c}\left(\sum_j \mathcal{A}^+\Delta Q_{\frac{1}{2},j} + \sum_j\sum_{i=2}^{l}\left(\mathcal{A}^-\Delta Q_{i-\frac{1}{2},j} + \mathcal{A}^+\Delta Q_{i-\frac{1}{2},j}\right) + \sum_j \mathcal{A}^-\Delta Q_{l+\frac{1}{2},j}\right) \\
&-\frac{\Delta t}{\Delta y_c}\left(\sum_i \mathcal{B}^+\Delta Q_{i,\frac{1}{2}} + \sum_i\sum_{j=2}^{m}\left(\mathcal{B}^-\Delta Q_{i,j-\frac{1}{2}} + \mathcal{B}^+\Delta Q_{i,j-\frac{1}{2}}\right) + \sum_i \mathcal{B}^-\Delta Q_{i,m+\frac{1}{2}}\right) \\
&= \sum_{i,j}\kappa_{i,j}Q_{ij}^n - \sum_j \frac{\Delta t}{\Delta x_c}\left(f^*_{l+\frac{1}{2},j} - f^*_{\frac{1}{2},j}\right) - \sum_i \frac{\Delta t}{\Delta y_c}\left(f^*_{i,m+\frac{1}{2}} - f^*_{i,\frac{1}{2}}\right) \\
&-\sum_{i,j}\left(\frac{\Delta t}{\Delta x_c}\left(f_{i-\frac{1}{2},j}(Q_{i,j}^{-\bullet}) - f_{i+\frac{1}{2},j}(Q_{i,j}^{+\bullet})\right) - \frac{\Delta t}{\Delta y_c}\left(f_{i,j-\frac{1}{2}}(Q_{i,j}^{\bullet-}) - f_{i,j+\frac{1}{2}}(Q_{i,j}^{\bullet+})\right)\right).
\end{aligned} \tag{7.7}$$

We will now show that the final sum is zero. For this we multiply by $\Delta x_c \Delta y_c/\Delta t$ and rewrite each term as

$$\begin{aligned}
&\Delta y_c\left(f_{i-\frac{1}{2},j}(Q_{i,j}^{-\bullet}) - f_{i+\frac{1}{2},j}(Q_{i,j}^{+\bullet})\right) - \Delta y_c\left(f_{i,j-\frac{1}{2}}(Q_{i,j}^{\bullet-}) - f_{i,j+\frac{1}{2}}(Q_{i,j}^{\bullet+})\right) \\
&\quad = h_{i-\frac{1}{2},j}R^T_{i-\frac{1}{2},j}\mathbf{f_1}(R_{i-\frac{1}{2},j}Q^n_{ij}) - h_{i+\frac{1}{2},j}R^T_{i+\frac{1}{2},j}\mathbf{f_1}(R_{i+\frac{1}{2},j}Q_{i,j}) \\
&\quad + h_{i,j-\frac{1}{2}}R^T_{i,j-\frac{1}{2}}\mathbf{f_1}(R_{i,j-\frac{1}{2}}Q_{i,j}) - h_{i,j+\frac{1}{2}}R^T_{i,j+\frac{1}{2}}\mathbf{f_1}(R_{i,j+\frac{1}{2}}Q_{i,j}).
\end{aligned} \tag{7.8}$$

The rotational invariance of the equations implies

$$R^T\mathbf{f_1}(R\mathbf{q}) = n^1\mathbf{f_1}(\mathbf{q}) + n^2\mathbf{f_2}(\mathbf{q}).$$

Using this, the right hand side of (7.8) can be written as

$$\begin{aligned}
&\left(h_{i-\frac{1}{2},j}n^1_{i-\frac{1}{2},j} - h_{i+\frac{1}{2},j}n^1_{i+\frac{1}{2},j} + h_{i,j-\frac{1}{2}}n^1_{i,j-\frac{1}{2}} - h_{i,j+\frac{1}{2}}n^1_{i,j+\frac{1}{2}}\right)\mathbf{f_1}(Q_{i,j}) \\
&+\left(h_{i-\frac{1}{2},j}n^2_{i-\frac{1}{2},j} - h_{i+\frac{1}{2},j}n^2_{i+\frac{1}{2},j} + h_{i,j-\frac{1}{2}}n^2_{i,j-\frac{1}{2}} - h_{i,j+\frac{1}{2}}n^2_{i,j+\frac{1}{2}}\right)\mathbf{f_2}(Q_{i,j}).
\end{aligned} \tag{7.9}$$

The divergence theorem over quadrilateral grid cells ensures that each of these terms is zero, and hence the scheme is conservative, up to fluxes at the boundary described by (7.7).

The three-dimensional wave-propagation algorithm is the natural extension of that for two dimensions, and is described in detail in [30]. The curvilinear grid algorithm is also the natural extension of the two-dimensional method presented above. Just as in two dimensions, where we approximate curvilinear grid cells as quadrilaterals, in three dimensions we approximate the general hexahedral grid cell as a cell with ruled surfaces. This approximation greatly simplifies the task of computing metric terms such as the area of faces, volumes of mesh cells, and the rotation matrix.

7.2. Discretization of Diffusion Terms. For balance laws that also include diffusion and/or reaction terms in addition to transport terms, we typically use a fractional step method; see [36]. Reaction terms can then easily be included by using an ODE solver. If diffusion terms are present, a heat equation is solved during each time step of the fractional step method. The time step will typically be determined by the CFL condition for the explicit update of the transport part of the equations. On our mapped grids we have approximated the solution of the heat equation using a discretization of the equation in computational space, i.e., we discretize

(7.10)

$$\sqrt{|h|}\partial_t q = \partial_{x_c}\left(\sqrt{|h|}\left(h^{11}\partial_{x_c}q + h^{12}\partial_{y_c}q\right)\right) + \partial_{y_c}\left(\sqrt{|h|}\left(h^{21}\partial_{x_c}q + h^{22}\partial_{y_c}q\right)\right).$$

Here $|h|$ is the determinant of the metric tensor $h = J^TJ$, where J is the Jacobian matrix of the grid transformation. Note that $\sqrt{|h|}$ is now the ratio of volume elements in the two coordinate systems and $h^{\alpha\beta}$ denotes the components of h^{-1}, the inverse of the matrix h. (Subscripts on h below refer to the grid point location.)

Equation (7.10) can be discretized in the form of a finite volume method (7.3), with $\kappa_{ij} = \sqrt{|h_{ij}|}$ and numerical fluxes

(7.11)

$$F_{i+\frac{1}{2},j} = -\sqrt{|h_{i+\frac{1}{2},j}|}\left(h^{11}_{i+\frac{1}{2},j}\frac{Q_{i+1,j}-Q_{ij}}{\Delta x_c} + h^{12}_{i+\frac{1}{2},j}\frac{Q_{i,j+1}+Q_{i+1,j+1}-Q_{i,j-1}-Q_{i+1,j-1}}{4\Delta y_c}\right)$$

and an analogous definition of the G terms. Analytical formulas for the metric terms of the sphere grid shown in Figure 4.1(a) have been implemented and were used for the calculations shown in section 8.7. For the time discretization we used the RKC-method, an explicit solver for parabolic PDEs; see [46]. It is a variable time step and variable formula method (based on a family of Runge–Kutta–Chebyshev formulas) that allows the use of reasonable time steps.

8. Numerical Results.

In the remainder of the paper we show some computational results, mostly for hyperbolic problems using the CLAWPACK software.

8.1. Blast Wave in Radial Geometry. We first test the two-dimensional grids described in section 3 to determine how well our proposed grids work for solving the Euler equations. In particular, we wish to determine whether the skewed cells along the diagonals of these grids have a noticeable effect on the quality of the solution. For this test case, we consider two grids, the radial projection grid and the grid consisting of the convex combination of the radial grid and the Cartesian grid. We refer to these grids as the "radial" grid and the "convex" grid, respectively.

The Euler equations for a compressible polytropic gas are given in standard conservation form (7.1) with

$$(8.1)\qquad \mathbf{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ E \end{pmatrix}, \quad \mathbf{f}(\mathbf{q}) = \begin{pmatrix} \rho u & \rho v & \rho w \\ \rho u^2 + p & \rho vu & \rho wu \\ \rho uv & \rho v^2 + p & \rho wv \\ \rho uw & \rho vw & \rho w^2 + p \\ u(E+p) & v(E+p) & w(E+p) \end{pmatrix}.$$

The conserved quantities are the density ρ, momentum $\rho \cdot \mathbf{u} = (\rho u, \rho v, \rho w)$, and total energy E. In addition to these equations, we must supply an equation of state that relates the pressure p to the conserved quantities. For the polytropic gas, the equation

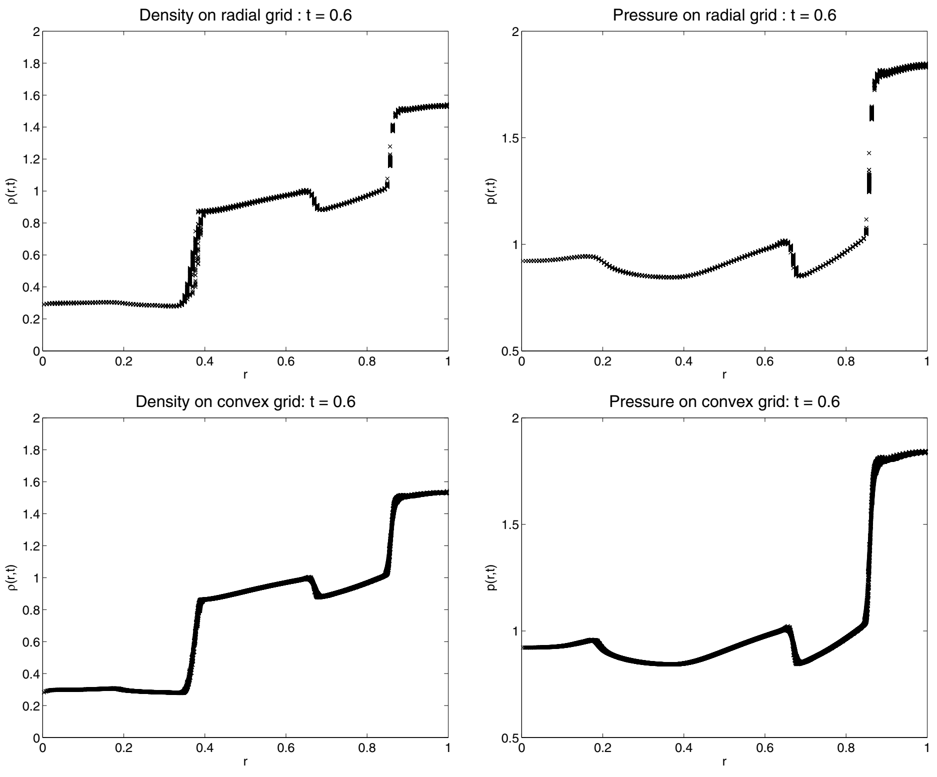

Fig. 8.1 *Scatter plots showing density and pressure as functions of r, the distance from the center of the circle. The top row of plots shows results computed on the radial projection grid, and the bottom row shows results for the smoother convex grid. Each grid is* 150×300*. The data for all plots were taken at time* $t = 0.6$*.*

of state is given by

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho(u^2 + v^2 + w^2), \tag{8.2}$$

where γ is the adiabatic gas constant. We use the value for air and set $\gamma = 1.4$.

For the present two-dimensional problem, we solve the two-dimensional analogue of the equations given above. These equations can be obtained by setting $w = 0$. The problem is initialized with radially symmetric data. Initially $\rho = 1$ and $u = v = 0$ everywhere in the domain. Inside a disk of radius 0.2 centered at $x = y = 0$ we set $p = 5$. Outside this disk, we set $p = 1$. We only compute the solution in the half circle in the right half plane of the Cartesian plane, and use symmetry conditions at $x = 0$ to simulate the effect of the solution in the full domain. The grid size is 150×300. At the physical boundaries on the outer edge of the circle, we impose reflecting, or solid wall, boundary conditions.

We ran the computation long enough to get reflections off the circular outer boundary. In Figure 8.1 we show scatter plots of the computed pressure and density at $t = 0.6$. The density plots clearly show the contact discontinuity near $r = 0.3$ and the shock near $r = 0.9$, which has already reflected off the outer boundary and is in-going at this time. As expected, the pressure is smooth near $r = 0.3$, but also

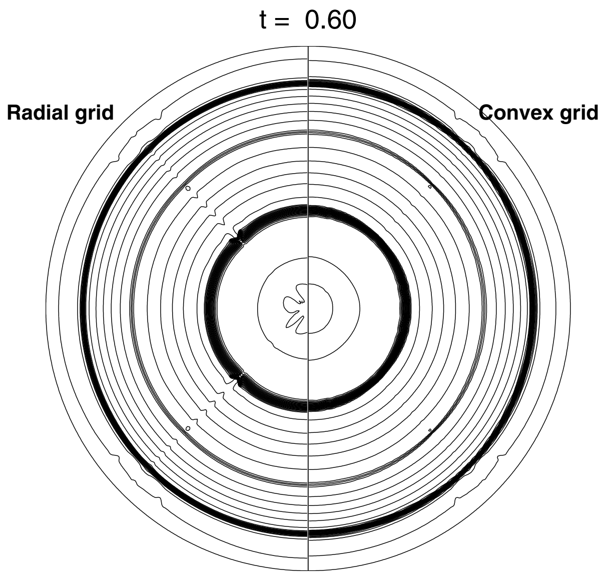

Fig. 8.2 *Contour plot of the density on each grid. The contour lines are drawn at* 31 *equally spaced values between* 0.3 *and* 1.3*.*

exhibits a discontinuity near $r = 0.9$. The solutions on both grids look quite good and suggest that even on this relatively coarse grid, we can get good results with these grids.

Looking at the scatter plots more closely, it appears that the radial grid solutions seem to have fewer data points than the convex grid calculations, especially at the outer shock. This is due to the fact that on the radial grid, one set of coordinate directions lies along concentric circles, and therefore there are fewer distinct values of r at cell centers. On the smoother convex grid, the data points appear more filled in. Also, the convex grid seems to resolve the contact discontinuity slightly better than the radial grid approximation. Finally, the pressure pulse at about $r = 0.2$ is also slightly sharper on the convex grid. This may be due to the fact that, again, the convex grid has data at more distinct r values, and so a wider range of solution values in r is obtained.

In Figure 8.2, we show a contour plot of the density on each grid. The left half disk shows the solution computed on the radial grid, and the right shows the solution on the convex grid. Here, we see more clearly that the highly skewed cells along the diagonal of the radial grid have a noticeable effect on the solution there. The convex grid produces much smoother results along the diagonals, although, on this grid, the contour lines are slightly compressed along the 45 degree lines.

8.2. Spherical Blast Wave. We now use the three-dimensional grid of Figure 5.1(b) to solve the Euler equations for a blast wave problem inside a ball.

We initialize the problem with axisymmetric data, similar to that used for the two-dimensional problem. Inside a sphere of radius 0.2 centered at $x = y = 0$, $z = -0.4$, we set the pressure p to 5. Note that this sphere is offset from the center of the ball domain, leading to a more interesting solution than the radially symmetric solution of the previous example. Outside this sphere, we set the pressure to 1. Density is set to 1 everywhere initially and velocity is set to 0. This is the spherical analogue of the test problem used in section 3.2 of [30], where a spherical blast wave between two parallel walls is used as a test problem of the three-dimensional implementation of CLAWPACK. We compute the solution in a quarter sphere on a $75 \times 75 \times 150$ grid. At each boundary of the computational domain, we impose solid wall boundary conditions. These conditions serve as both symmetry conditions at interior boundaries (i.e., boundaries on the x and y planes bounding the quarter sphere) and as solid wall boundary conditions on the exterior of the sphere.

We compare the three-dimensional calculations with solutions in a two-dimensional disk using the two-dimensional Euler equations with a source that models the axisymmetry. The two-dimensional reference solution is again calculated in only half the domain using solid wall boundary conditions at the axis of symmetry as well as at the physical boundary, using the convex grid with 300×600 mesh cells.

Figure 8.3 shows a single slice of the three-dimensional computed solution alongside the fine grid two-dimensional solution. The main features of the expanding shock wave are visible in both sets of plots. These features include an expanding shockwave (the outermost expanding circle) and the essentially stationary contact discontinuity between different values of density (the innermost ring). Good agreement is seen between the three-dimensional computations and the fine grid axisymmetric results. The only obvious difference is due to the fact that the three-dimensional computation is computed at a lower resolution, and so the shock fronts and contact discontinuities are more diffuse.

8.3. Quadrilateral Grids on Other Smooth Domains. Figure 8.4(a) shows a quadrilateral grid on a noncircular domain with a smooth boundary, obtained by composing the radial projection circle map from section 3.1 with a smooth map from the circle to this starlike domain. The mapping is given by using the `mapc2p.m` function of section 3.1 appended with the transformation

```
    theta = atan(abs(yp) ./ max(abs(xp),1e-10));
    r = 1 + .2*cos(6*theta);
    xp = xp.*r;
    yp = yp.*r;
```

Note that this gives a radial projection to the modified domain. Alternatively, any of the mappings defined in section 3.2 could be appended with these lines to obtain a smoother mapping, or a convex combination of the radial projection grid with a Cartesian grid can be used to smooth the mapping towards Cartesian in the center, as described at the end of section 3.2 and shown in Figure 8.4(b).

Figure 8.5 shows solutions of the shallow water equations solved on a 200×200 version of the grid shown in Figure 8.4 (b). The shallow water equations can be written in the form (7.1) with a flux matrix that consists of the upper left 2×3 submatrix of (8.4). The initial conditions consist of deeper water in a circular region near the center, so it is a radial dam-break problem. Solid wall boundary conditions are used so that we are simulating the resulting sloshing of water in a tank with irregular shape. Notice that the computed solution shows the expected 6-fold symmetry to a remarkable degree in spite of the fact that the grid does not have this symmetry and is more highly skewed in some arms than in others. Calculations on the grid of Figure 8.4(a) have also been performed and gave roughly comparable results which are not shown here.

8.4. Acoustics in a Heterogeneous Medium. The acoustics equations are an example of a linear hyperbolic system in nonconservative form where the wave-

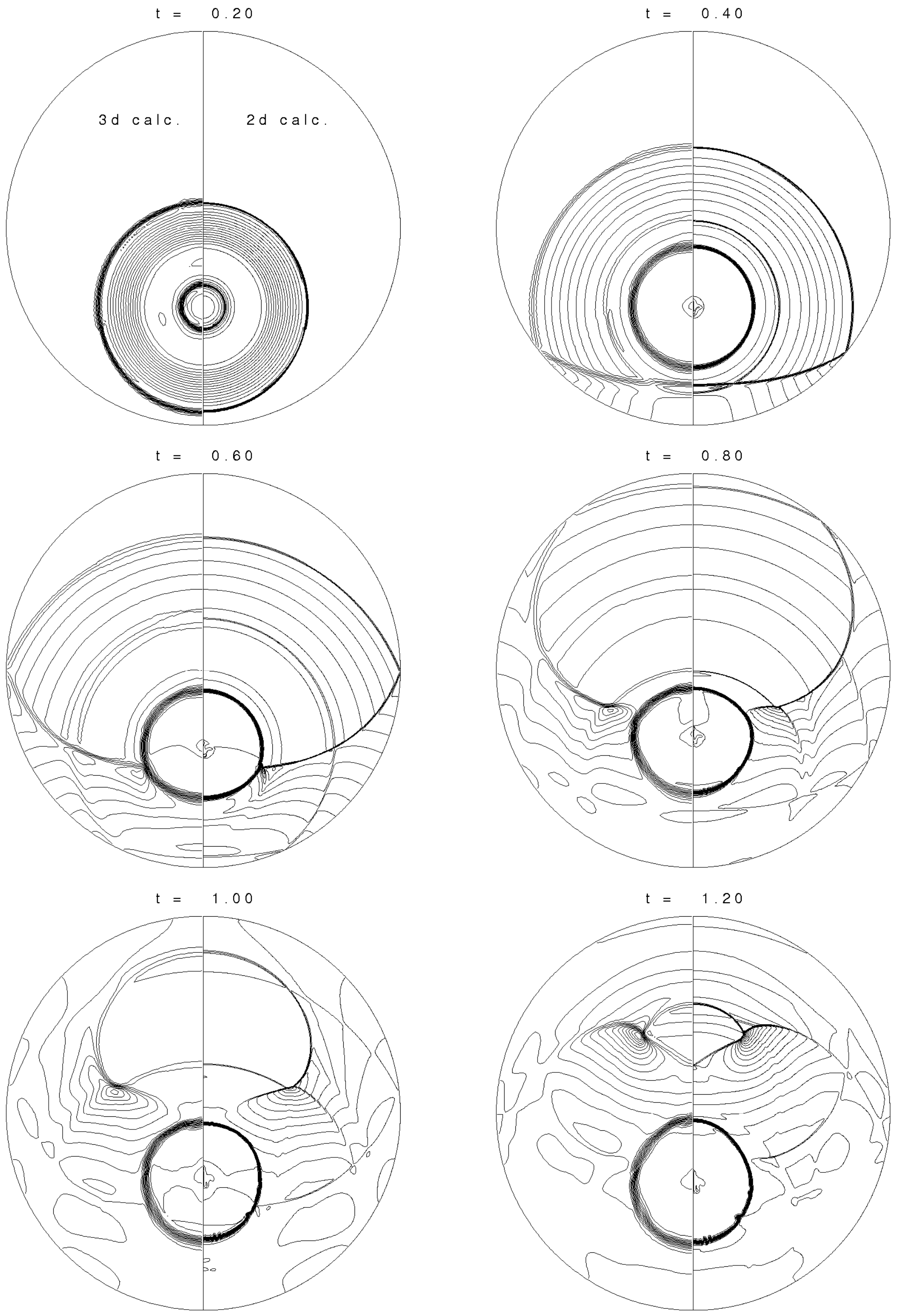

Fig. 8.3 *Contour plots of density computed for the blast wave problem on the three-dimensional radial projection grid (left half of each plot) and a two-dimensional axisymmetric calculation (right half of each plot). The three-dimensional grid is* $75\times75\times150$ *and the two-dimensional grid is* 300×600.

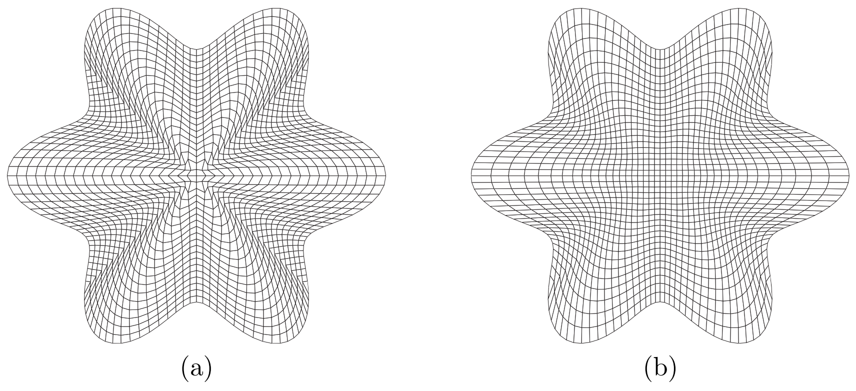

Fig. 8.4 *Quadrilateral grid on a starlike domain using* 40×40 *grid cells.* (a) *The radial projection grid;* (b) *followed by interpolation smoothing as described in section* 3.3.

propagation algorithm can be applied naturally. In two space dimensions the equations can be written in the form

$$\partial_t \mathbf{q} + \mathbf{A}(x,y)\partial_x \mathbf{q} + \mathbf{B}(x,y)\partial_y \mathbf{q} = \mathbf{0},$$

with

$$\mathbf{q} = \begin{pmatrix} p \\ u \\ v \end{pmatrix}, \quad \mathbf{A}(x,y) = \begin{pmatrix} 0 & K(x,y) & 0 \\ 1/\rho(x,y) & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\mathbf{B}(x,y) = \begin{pmatrix} 0 & 0 & K(x,y) \\ 0 & 0 & 0 \\ 1/\rho(x,y) & 0 & 0 \end{pmatrix},$$

where $p,\ u,\ v$ are pressure and velocity in the x and the y direction, respectively, ρ is the density, and K is the bulk modulus. Furthermore, the sound speed and the impedance of the material are defined as $c = \sqrt{K/\rho}$ and $Z = \rho c$ (see [36]).

We solve the acoustics equations on a grid of the form shown in Figure 6.1. The two circles represent cylinders made out of a different material than the background and we consider a plane wave approaching from the left, a square pulse of high pressure generated by setting the velocity of the wall at the left of the domain to be $s = 1$ for $t \leq 0.05$ and $s = 0$ beyond this time. The other three sides of the domain have zero-order extrapolation nonreflecting boundary conditions.

The medium is defined by

background:	$Z = 1$,	$c = 1$,			
circle 1:	$Z = 12$,	$c = 0.3$,	$(x_0, y_0) = (0.45, 0.3)$,	$r_1 = 0.15$,	$r_2 = 0.204$,
circle 2:	$Z = 10$,	$c = 1.5$,	$(x_0, y_0) = (0.7, 0.75)$,	$r_1 = 0.15$,	$r_2 = 0.204$.

Computed results on a 200×200 grid at three different times are shown in Figure 8.6, along with adaptive mesh refinement results with higher resolution.

8.5. Adaptive Mesh Refinement. One advantage of using a single logically rectangular grid for the computational domain is that it is relatively easy to apply adaptive mesh refinement algorithms of the style pioneered by Berger, Oliger, and Colella

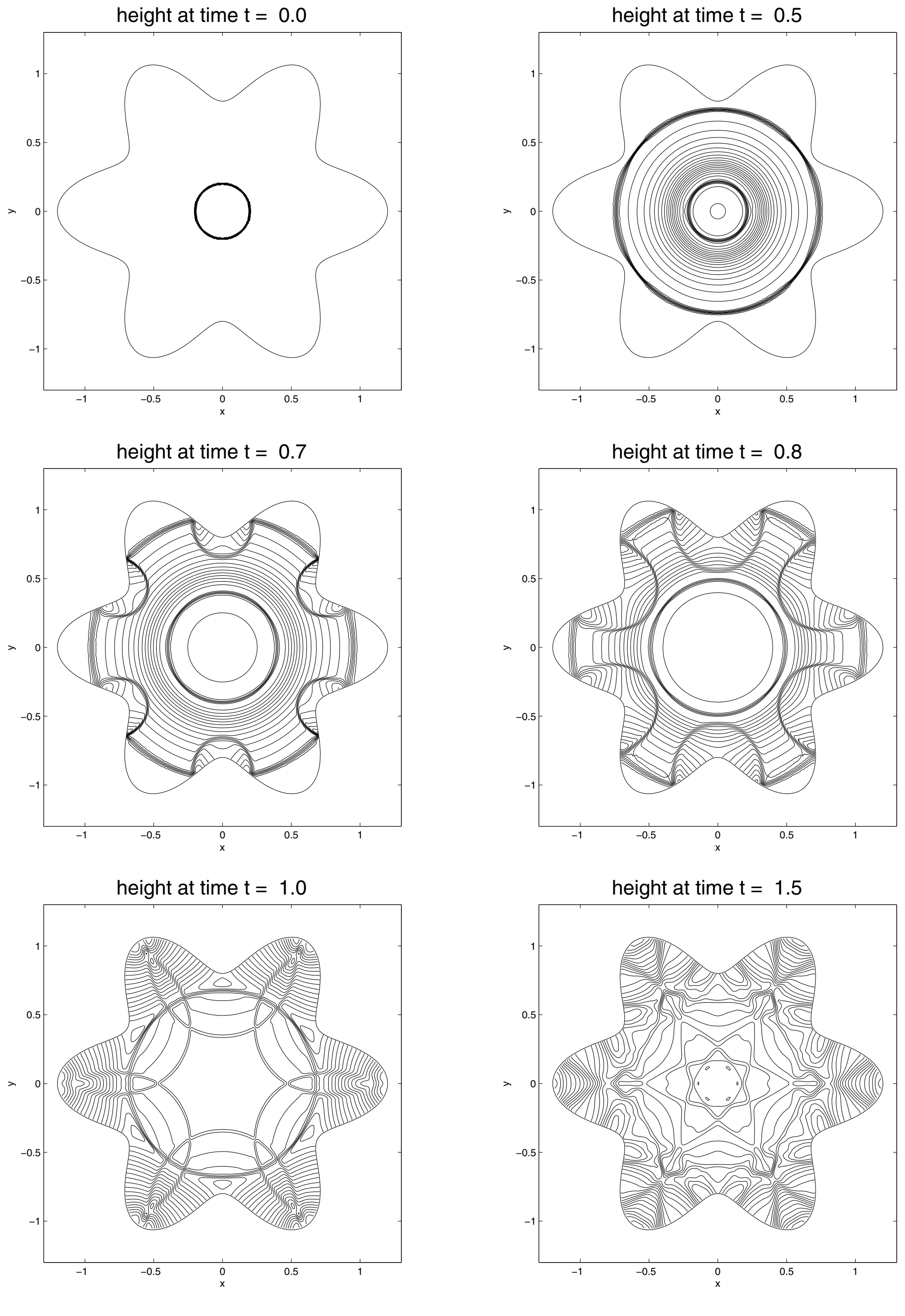

Fig. 8.5 *Computed solution of the two-dimensional shallow water equations on a* 200×200 *version of the grid shown in Figure* 8.4(b) *and with solid wall boundary conditions. The initial data is a circular region, where* $h = 1.5$, *while* $h = 1$ *elsewhere. The computed solutions at six selected times are shown. Contour values are at* 0.505:0.01:1.5.

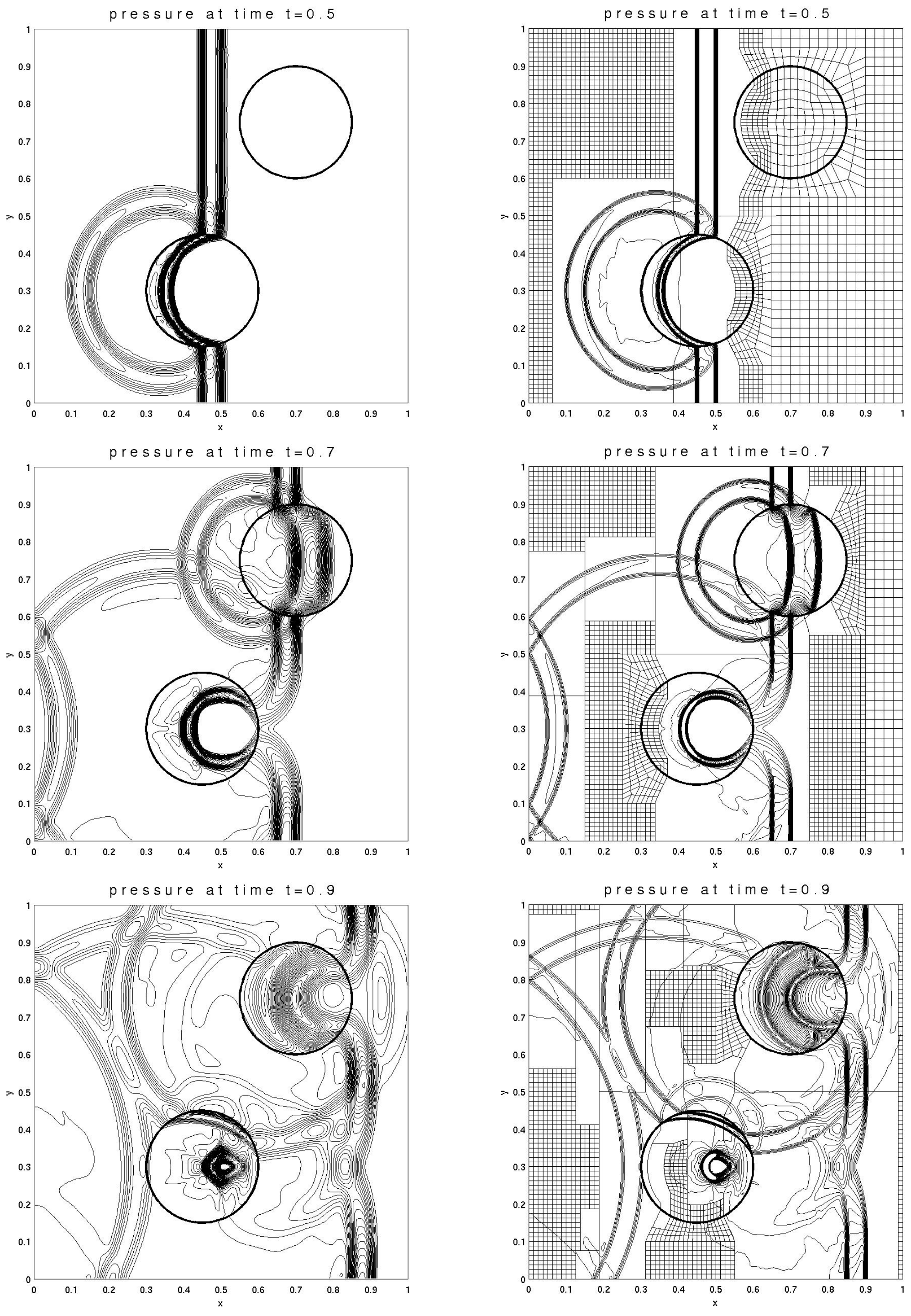

Fig. 8.6 *Acoustics in a heterogeneous medium. The material parameters and sound speed are different in each of the circular regions and a square pulse is propagating from left to right. Left column: Solution at three different times on a* 200×200 *grid. Right column: Using adaptive mesh refinement on rectangular patches (see section* 8.5*). Grid lines are shown on the two coarsest levels, not on the third level where the effective resolution is* 800×800.

[8, 10, 7], in which a rectangular grid is refined on rectangular patches, recursively to several levels. In particular, when solving hyperbolic problems using wave-propagation algorithms, the adaptive mesh refinement routines of AMRCLAW, which are included with CLAWPACK, can be applied directly with little additional work on the part of the user. The approach used for wave-propagation algorithms is described in [10] and the software is documented in [34]. Figure 8.6 shows three frames from a sample calculation where the acoustics problem with inclusions from section 8.4 are solved with three levels. The coarsest grid is 40×40, the level 2 grids are refined by a factor of 2 in both space and time, and the finest level 3 grids are refined by an additional factor of 10 in space and time, giving resolution comparable to a uniform 800×800 grid.

8.6. Shallow Water on the Sphere. The shallow water equations are often solved on a sphere in global atmosphere or ocean models, for example. The equations can be formulated in three-dimensional Cartesian coordinates as

$$\partial_t \mathbf{q} + \nabla \cdot \mathbf{f}(\mathbf{q}) = \mathbf{s}(\mathbf{x}, \mathbf{q}), \tag{8.3}$$

where $\mathbf{q} = (h, hu, hv, hw)^T$ represents the state vector composed of the height h and three Cartesian velocity components $\mathbf{u} = (u, v, w)^T$, all of which are functions of space and time. The flux matrix has the form

$$\mathbf{f}(\mathbf{q}) = \begin{pmatrix} hu & hv & hw \\ hu^2 + \frac{1}{2}gh^2 & huv & huw \\ huv & hv^2 + \frac{1}{2}gh^2 & hvw \\ huw & hvw & hw^2 + \frac{1}{2}gh^2 \end{pmatrix}. \tag{8.4}$$

The source term, $\mathbf{s}(\mathbf{x}, \mathbf{q})$, which acts only on the momentum equations, has the form

$$\mathbf{s}(\mathbf{x}, \mathbf{q}) = -\frac{2\Omega}{a} z \,(\mathbf{x} \times h\mathbf{u}) + \left(\mathbf{x} \cdot (\nabla \cdot \tilde{\mathbf{f}})\right) \mathbf{x}. \tag{8.5}$$

The first term on the right-hand side of (8.5) is the Coriolis force due to the rotation of the earth. Here Ω and a are the rotation rate and the radius of the earth, respectively. The second term in (8.5) is a forcing term which guarantees that the fluid velocity remains on the surface of the sphere, i.e., perpendicular to the position vector $\mathbf{x} = (x, y, z)$ on the sphere. Here $\tilde{f}$ consists of the part of the flux matrix that is associated with the momentum equations. Such a Cartesian form of the equations was also used in [18], [21], [22], [23].

We approximate (8.3) using an operator splitting approach, i.e., Strang splitting. The Cartesian form has the advantage that it is independent of the coordinate transformation being used. The approximation of the three-dimensional equations is restricted to the surface of the sphere, i.e., we only update $\mathbf{q}$ in grid cells on the sphere. Our discretization follows the general outline from section 7.1. In a first step the velocity components of the grid cell values on each side of an interface are rotated into the direction normal ($\mathbf{n}$) and tangential ($\mathbf{t}$) to the interface in the tangent plane; now $\mathbf{n}, \mathbf{t} \in \mathbb{R}^3$. Then a one-dimensional Riemann problem is solved and the momentum components of the flux are rotated back to Cartesian coordinates. During each time step the momentum components are projected to the tangent plane at each cell center. Furthermore, the Coriolis force is calculated using a higher order Runge–Kutta method.

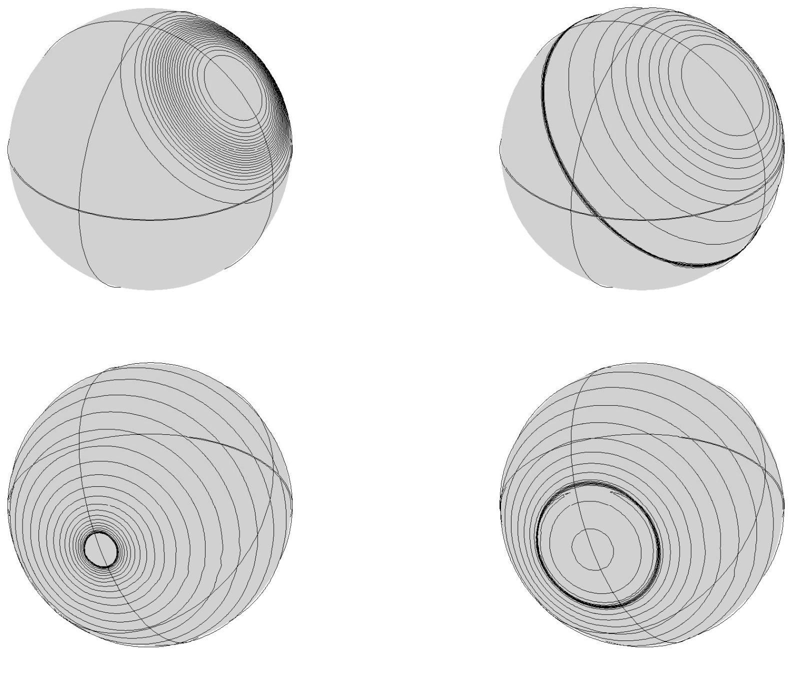

Fig. 8.7 *Shallow water equations on the surface of the sphere with initial data consisting of a hump of fluid. Contours of the fluid depth h are shown computed on a 400×200 grid of the type shown in Figure 4.1(a). The top two figures (times $t = 0$ and 0.6) are viewed from above. The bottom two figures (times $t = 1.8$ and 2.0) are viewed from below.*

Note that on the sphere the term stated in (7.9) is in general not equal to zero. In order to obtain a conservative update, we thus subtract in each grid cell during each time step the term

(8.6)
$$Q^{cor}_{i,j} = \frac{\Delta t}{\kappa_{ij}\Delta x_c \Delta y_c}\left(h_{i-\frac{1}{2},j}\mathbf{n_{i-\frac{1}{2},j}} - h_{i+\frac{1}{2},j}\mathbf{n_{i+\frac{1}{2},j}} + h_{i,j-\frac{1}{2}}\mathbf{n_{i,j-\frac{1}{2}}} - h_{i,j+\frac{1}{2}}\mathbf{n_{i,j+\frac{1}{2}}}\right)\cdot \mathbf{f}(Q_{ij})$$

from the cell average value of $\mathbf{q}$.

We first consider an example on a nonrotating sphere similar to one from [43]. The initial data consist of stationary fluid (zero velocities) with initial depth

$$h(x,y,z) = 1 + 2\exp\left(-40(1-(a_1x+a_2y+a_3z))^2\right), \tag{8.7}$$

where (x,y,z) is a point on the unit sphere and (a_1,a_2,a_3) is a unit vector determining an axis of symmetry. In this example we set the gravitational constant g equal to one. This gives a smooth hump of fluid centered about the point (a_1,a_2,a_3) and the solution remains axisymmetric about the ray through this point. A reference solution for comparison can then be obtained by solving a one-dimensional shallow water equation with a source term for the axisymmetric solution as a function of $s = \sin(a_1x+a_2y+a_3z)$, the "latitude" relative to the axis of symmetry.

Figure 8.7 shows the initial data and solution at three later times as a contour plot of h on the sphere. By time $t = 0.6$ a shock has formed and at time $t = 1.8$

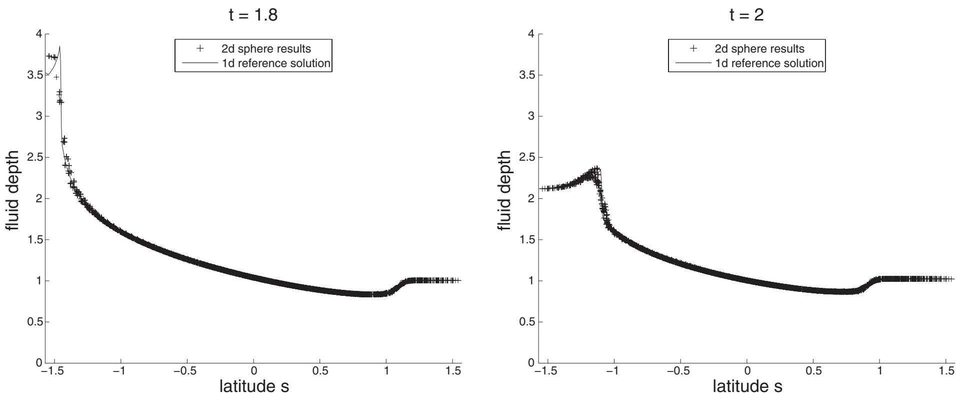

Fig. 8.8 *Shallow water equations on the surface of the sphere. Scatter plots of solution vs. s, the latitude relative to the axis of symmetry, shown at two times corresponding to the bottom two figures of Figure* 8.7 *but on a coarser grid with* 100×50 *grid cells. The solid line is the one-dimensional axisymmetric reference solution computed on a fine grid (*500 *points).*

this shock has just passed through the antipode of the initial hump location (which is taken off center from the north pole to better test the behavior near the equator, where the highly nonorthogonal grid points lie). For the latter two times, a scatter plot of the numerical solution at all points on the sphere vs. the corresponding s value is compared with the one-dimensional reference solution in Figure 8.8.

There is very little scatter in the scatter plots, indicating that there are few grid effects, and the solution is quite good even in the nearly triangular cells near the equator. In [12], we present results for this test case using adaptive mesh refinement. These can also be found on the webpage for this paper [14], along with animations of the evolution of the solution and the grids.

Rossby–Haurwitz Test Problem. In our next example we consider approximations of a wave number 4 Rossby–Haurwitz pattern. This is a standard test problem for the shallow water equations on a rotating sphere. We refer to Williamson et al. [49, Problem 6] for a detailed description of the initial conditions. For our simulations we have set the characteristic time scale to one day and the characteristic length scale to the radius of the earth.

Haurwitz [24] showed that in a nondivergent barotropic atmosphere (modeled by the nonlinear barotropic vorticity equation on the sphere), the flow pattern moves from west to east with constant angular velocity and without change of shape. Although Rossby–Haurwitz waves are not analytical solutions of the shallow water equations, the wave structure is rotated about the z-axis for a long time. For even longer time scales (about 100–150 days), highly resolved numerical simulations (see [47], [43]) show that the symmetric wave structure breaks.

In Figure 8.9 we show contour lines of the height obtained on a grid with 400×200 grid cells at different times. Our results compare well with results obtained on other (smoother) grids; see, for instance, [21] or [43]. In particular we do not observe any grid effects.

For early times, i.e., $t = 1$–4 days, we calculated the experimental order of convergence (EOC) by comparing solutions on three grids with 100×50, 200×100, and 400×200 grid cells. The results of this convergence study are reported in Table 8.1

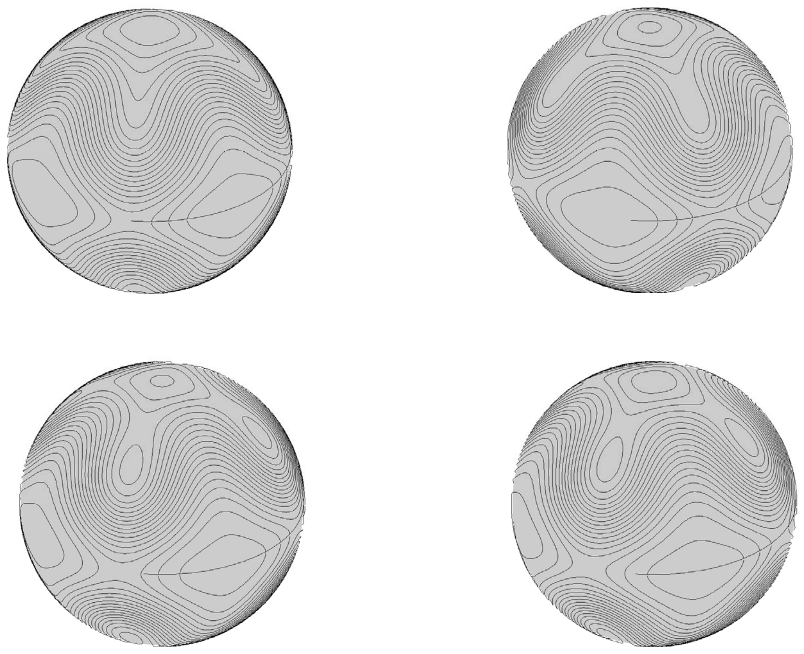

Fig. 8.9 *Contour levels of the fluid height* h *between* $8100m$ *and* $10500m$ *in increments of* $120m$*. The Rossby–Haurwitz wave was computed on a* 400×200 *grid of the type shown in Figure* 4.1(a)*. The top two figures show the solution at times* $t = 0$ *and* $t = 3$ *days and the bottom two figures at times* $t = 7$ *days and* $t = 14$ *days.*

Table 8.1 *Convergence study for the Rossby–Haurwitz wave using three grids with* 100×50*,* 200×100*, and* 400×200 *grid cells and* $CFL \le 0.9$*. The table shows the EOC in all components of* q *at different times.*

	1 day	2 days	3 days	4 days
h	1.64	1.73	1.89	2.04
hu	1.77	1.93	1.93	1.92
hv	1.77	1.93	1.95	1.90
hw	1.76	1.80	2.38	1.85

and confirm second order convergence for this smooth test case at early times. Here we used the second order wave-propagation algorithm without limiters. Note that this convergence test only worked for relatively small times. This might be related to the above-mentioned unstable behavior of the 4-Rossby–Haurwitz wave. In [12], we presented convergence studies for advective transport on the sphere which confirmed the second order accuracy of the method.

8.7. Reaction-Diffusion Equations on the Sphere. Many applications, in particular biological applications, require the approximation of reaction-diffusion systems on the sphere. As an example we present calculations for pattern formation of a Turing model with two interacting chemicals u and v. For this we use the model equations

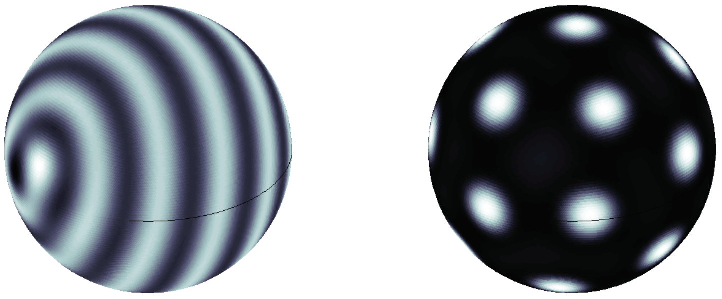

Fig. 8.10 *Turing patterns calculated on the sphere using* 300×150 *grid points (species u is shown). Striped pattern:* $\delta = 0.0021$, $D = 0.516$, $\tau_1 = 3.5$, $\tau_2 = 0$, $\alpha = 0.899$, $\beta = -0.91$, $\gamma = -\alpha$. *Spotted pattern:* $\delta = 0.0045$, $D = 0.516$, $\tau_1 = 0.02$, $\tau_2 = 0.2$, $\alpha = 0.899$, $\beta = -0.91$, $\gamma = -\alpha$.

from [4] and [48] that have the form

$$\begin{aligned}
\frac{\partial u}{\partial t} &= D\delta\Delta_S u + \alpha u\left(1 - \tau_1 v^2\right) + v(1 - \tau_2 u), \\
\frac{\partial v}{\partial t} &= \delta\Delta_S v + \beta v\left(1 + \frac{\alpha\tau_1}{\beta} uv\right) + u(\gamma + \tau_2 v),
\end{aligned} \tag{8.8}$$

where Δ_S denotes the Laplace–Beltrami operator. Here, $(u, v) = (0, 0)$ is a spatially uniform steady state, which is unique for $\gamma = -\alpha$. Turing showed that under certain conditions of the parameter values the steady state could be linearly stable in the absence of diffusion but unstable in the presence of diffusion. This diffusion-driven instability leads to the formation of different patterns. The interaction parameter τ_1 associated with cubic coupling favors the formation of a striped pattern. The quadratic coupling τ_2 produces spot patterns. The parameter δ describes the size of the domain in terms of the chosen wavelength presented in the pattern. Initially, u and v take random values between $-1/2$ and $1/2$. In Figure 8.10 we present plots from a numerical simulation for two different sets of parameter values leading to a striped and a spotted pattern, respectively.

In addition to the reaction-diffusion process modeled in (8.8), the species u and v could also propagate on the sphere due to advection processes. In the example provided on the webpage a rotation of u and v on the sphere is included, leading to the same pattern which rotates on the sphere.

REFERENCES

[1] A. Adcroft, J.-M. Campin, C. Hill, and J. Marshall, *Implementation of an atmosphere ocean general circulation model on the expanded spherical cube*, Monthly Weather Rev., 132 (2004), pp. 2845–2863.

[2] A. S. Almgren, J. B. Bell, P. Colella, and T. Marthaler, *A Cartesian grid projection method for the incompressible Euler equations in complex geometries*, SIAM J. Sci. Comput., 18 (1997), pp. 1289–1309.

[3] A. Asenov, A. R. Brown, S. Roy, and J. R. Barker, *Topologically rectangular grids in the parallel simulation of semiconductor devices*, VLSI Design, 6 (1998), pp. 91–95.

[4] R. Barrio, C. Varea, J. Aragón, and P. Maini, *A two-dimensional numerical study of spatial pattern formation in interacting Turing systems*, Bull. Math. Biol., 61 (1999), pp. 483–505.

[5] M. Berger and R. J. LeVeque, *An Adaptive Cartesian Mesh Algorithm for the Euler Equations in Arbitrary Geometries*, AIAA paper AIAA-89-1930, AIAA, Reston, VA, 1989, pp. 1–7.

[6] M. Berger and R. J. LeVeque, *Stable boundary conditions for Cartesian grid calculations*, Comput. Systems Engrg., 1 (1990), pp. 305–311.

[7] M. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[8] M. J. Berger and P. Colella, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[9] M. J. Berger, C. Helzel, and R. J. LeVeque, *k-box methods for the approximation of hyperbolic conservation laws on irregular grids*, SIAM J. Numer. Anal., 41 (2003), pp. 893–918.

[10] M. J. Berger and R. J. LeVeque, *Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems*, SIAM J. Numer. Anal., 35 (1998), pp. 2298–2316.

[11] D. Calhoun, *A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions*, J. Comput. Phys., 176 (2002), pp. 231–275.

[12] D. Calhoun, C. Helzel, and R. LeVeque, *A finite volume grid for solving hyperbolic problems on the sphere*, in Hyperbolic Problems: Theory, Numerics, Applications, Proceedings of the Eleventh International Conference on Hyperbolic Problems (Lyon, 2006), Springer, Berlin, pp. 355–362.

[13] D. Calhoun and R. J. LeVeque, *Solving the advection-diffusion equation in irregular geometries*, J. Comput. Phys., 156 (2000), pp. 1–38.

[14] D. A. Calhoun, C. Helzel, and R. J. LeVeque, *Webpage with supplemental material for this paper*, http://www/amath.washington.edu/~rjl/pubs/circles, 2006.

[15] J. E. Castillo, ed., *Mathematical Aspects of Numerical Grid Generation*, Frontiers Appl. Math. 8, SIAM, Philadelphia, 1991.

[16] P. Colella, D. T. Graves, B. J. Keen, and D. Modiano, *A Cartesian grid embedded boundary method for hyperbolic conservation laws*, J. Comput. Phys., 211 (2006), pp. 347–366.

[17] P. Colella, D. T. Graves, T. J. Ligocki, D. Modiano, and B. Van Straalen, *EBChombo software package for Cartesian grid, embedded boundary applications*, https://apdec.org/designdocuments/ebchombo.pdf, 2003.

[18] J. Côté, *A Lagrange multiplier approach for the metric terms of semi–Lagrangian models on the sphere*, Quart. J. R. Meteorol. Soc., 114 (1988), pp. 1347–1352.

[19] D. De Zeeuw and K. Powell, *An adaptively-refined Cartesian mesh solver for the Euler equations*, J. Comput. Phys., 104 (1993), pp. 56–68.

[20] H. Forrer and R. Jeltsch, *A higher-order boundary treatment for Cartesian-grid methods*, J. Comput. Phys., 140 (1998), pp. 259–277.

[21] F. Giraldo, *A spectral element shallow water model on spherical geodesic grids*, Internat. J. Numer. Methods Fluids, 35 (2001), pp. 869–901.

[22] F. Giraldo, J. Hesthaven, and T. Warburton, *Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations*, J. Comput. Phys., 181 (2002), pp. 499–525.

[23] F. Giraldo and T. Warburton, *A nodal triangle-based spectral element method for the shallow water equations on the sphere*, J. Comput. Phys., 207 (2005), pp. 129–150.

[24] B. Haurwitz, *The motion of atmospheric disturbances on the spherical earth*, J. Mar. Res., 3 (1940), pp. 254–267.

[25] C. Helzel, M. J. Berger, and R. J. LeVeque, *A high-resolution rotated grid method for conservation laws with embedded geometries*, SIAM J. Sci. Comput., 26 (2005), pp. 785–809.

[26] J. M. Hyman, S. Li, P. Knupp, and M. Shashkov, *An algorithm for aligning a quadrilateral grid with internal boundaries*, J. Comput. Phys., 163 (2000), pp. 133–149.

[27] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys., 147 (1998), pp. 60–85.

[28] A. Kageyama and T. Sato, *The "Yin-Yang grid": An overset grid in spherical geometry*, Geochem. Geophys. Geosystems, 5 (2004), article Q 09005.

[29] D. Kröner, *Numerical Schemes for Conservation Laws*, Wiley, Chichester, UK, 1997.

[30] J. O. Langseth and R. J. LeVeque, *A wave-propagation method for three-dimensional hyperbolic conservation laws*, J. Comput. Phys., 165 (2000), pp. 126–166.

[31] D. Lanser, J. Blom, and J. Verwer, *Spatial discretization of the shallow water equations in spherical geometries*, J. Comput. Phys., 165 (2000), pp. 542–565.

[32] L. Lee and R. J. LeVeque, *An immersed interface method for incompressible Navier–Stokes equations*, SIAM J. Sci. Comput., 25 (2003), pp. 832–856.

[33] R. J. LeVeque, *CLAWPACK software*, http://www.amath.washington.edu/~claw.

[34] R. J. LeVeque, *CLAWPACK User's Guide*, http://www.amath.washington.edu/~claw/doc.html.

[35] R. J. LeVeque, *High resolution finite volume methods on arbitrary grids via wave propagation*, J. Comput. Phys., 78 (1988), pp. 36–63.

[36] R. J. LeVeque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge, UK, 2002.
[37] V. D. Liseikin, *Grid Generation Methods*, Sci. Comput., Springer-Verlag, Berlin, 1999.
[38] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.
[39] A. Mayo and A. Greenbaum, *Fast parallel iterative solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 101–118.
[40] M. Rancic, R. J. Purser, and F. Messinger, *A global shallow-water model using an expanded spherical cube: Gnonomic versus conformal coordinates*, Quart. J. R. Meteorol. Soc., 122 (1996), pp. 959–982.
[41] C. Ronchi, R. Iacono, and P. S. Paolucci, *The "cubed sphere": A new method for the solution of partial differential equations in spherical geometry*, J. Comput. Phys., 124 (1996), pp. 93–114.
[42] J. A. Rossmanith, *A Wave Propagation Method with Constrained Transport for Ideal and Shallow Water Magnetohydrodynamics*, Ph.D. thesis, University of Washington, Seattle, 2002.
[43] J. A. Rossmanith, *A wave propagation method for hyperbolic systems on the sphere*, J. Comput. Phys., 213 (2006), pp. 629–658.
[44] R. Sadourny, *Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids*, Monthly Weather Rev., 100 (1972), pp. 211–224.
[45] R. Sadourny, A. Arakawa, and Y. Mintz, *Integration of a non-divergent barotropic vorticity equation with an icosahedral-hexagonal grid for the sphere*, Monthly Weather Rev., 96 (1968), pp. 351–356.
[46] B. Sommeijer, L. Shampine, and J. Verwer, *RKC: An explicit solver for parabolic PDEs*, J. Comput. Appl. Math., 88 (1997), pp. 315–326.
[47] J. Thuburn and Y. Li, *Numerical simulations of Rossby-Haurwitz waves*, Tellus, 52A (2000), pp. 181–189.
[48] C. Varea, J. Aragon, and R. Barrio, *Turing patterns on a sphere*, Phys. Rev. E, 60 (1999), pp. 4588–4592.
[49] D. Williamson, J. Drake, J. Hack, R. Jakob, and P. Swarztrauber, *A standard test set for numerical approximations to the shallow water equations in spherical geometry*, J. Comput. Phys., 102 (1992), pp. 211–224.