A Conservative Staggered-Grid Chebyshev Multidomain Method for Compressible Flows

DAVID A. KOPRIVA* AND JOHN H. KOLIAS†

**Supercomputer Computations Research Institute and †Department of Mechanical Engineering, The Florida State University, Tallahassee, Florida 32306*

Received September 12, 1995; revised December 5, 1995

We present a new multidomain spectral collocation method that uses a staggered grid for the solution of compressible flow problems. The solution unknowns are defined at the nodes of a Gauss quadrature rule. The fluxes are evaluated at the nodes of a Gauss–Lobatto rule. The method is conservative, free-stream preserving, and exponentially accurate. A significant advantage of the method is that subdomain corners are not included in the approximation, making solutions in complex geometries easier to compute. © 1996 Academic Press, Inc.

1. INTRODUCTION

Standard Chebyshev spectral methods applied to compressible flow problems have some severe restrictions [7]. The computational domain must be simple enough to map onto a square in two space dimensions, or a cube in three. To increase spatial resolution the polynomial approximation order must be increased. For high orders, the derivative approximations must be performed with fast Fourier transform methods to be efficient. If matrix multiplication is used instead, the work grows too rapidly with the number of degrees of freedom to be practical. Finally, the time step restrictions are severe since the time step decreases asymptotically as the square of the order of the approximating polynomials.

The basic premise of a multidomain method is that these restrictions can be reduced by subdividing the computational domain into multiple zones, called subdomains, on which the spectral approximation is applied. As a result, the method can be used on more complex geometries. The use of lower order approximating polynomials in each subdomain means that matrix multiplication can be both efficient and accurate and the time step restrictions need not be as severe. A discussion of the advantages of multidomain methods over the single domain method was presented in [25] and has been updated in [20].

Less than a decade after they were first introduced [25], the bulk of the spectral multidomain methods that have been proposed for compressible flows or similar hyperbolic problems still define the solution unknowns at the nodes of the Gauss–Lobatto quadrature, just as in a single domain method. Examples include [35, 26, 31], and [3] for general hyperbolic problems and [27] for the Euler gas-dynamics equations. Methods for the compressible Navier–Stokes equations were presented in [19, 20, 28]. An interesting method for coupled acoustic and elastic wave interactions was proposed in [1].

The main differences between the Lobatto grid methods are whether the equations are written in conservative or non-conservative form, and the manner in which the interfaces are treated. The conservative form of the equations was used, for example, in the methods presented by [19, 4]. Non-conservative forms were considered in [35, 27, 1]. We will show below that the use of the conservative form of the equations does not guarantee that the method is globally conservative, since the interface treatment may lead to loss of conservation.

Two approaches have been used at subdomain interfaces to ensure that waves propagate properly through them. The two methods were contrasted in [26] and are considered in more detail in [3]. At least two values of the normal derivative are available an interface point, depending on the number of subdomains that have that point in common. One interface method integrates a differential compatibility equation for the points along the interface [25, 35, 19, 3]. Derivatives are chosen from appropriate subdomains so that wave components are "upwinded." The other approach uses a correction procedure [26, 27, 1, 4]. To implement the correction method, the interior point approximation is integrated everywhere, including at the boundaries. As a result, multiple solution values are available at each interface point. A characteristic combination of these solutions is then made to correct the solution for the propagation of waves across the interface.

Each interface treatment has its advantages and disadvantages. Integrating the differential equation means that the solution can be approximated to any order of accuracy in time, depending on the choice of the time integration scheme. Implicit time integration schemes can also be used [3]. The serious disadvantage is that the tangential space derivatives must be continuous across subdomain interfaces in more than one space dimension. This requirement severely restricts the types of geometries on which solutions can be computed, since it means that the Jacobians of the transformations of the mappings between the subdomains and the unit square must be continuous across subdomain interfaces. The correction scheme, on the other hand, does not require smoothness of the grids, since only the solution values are required at the interfaces. However, the temporal accuracy of the correction method is limited to first order (cf. [7, p. 245].)

A disadvantage shared by the two interface treatments is their complexity. Either method is simple to apply in one space dimension. In two space dimensions a choice must be made at corners to determine from which subdomains the solution bicharacteristics must be computed. Special algorithms can be developed for the approximations at the corners of subdomains [27], but if more than four subdomains meet at a single point, the choice can be even more complex.

A very different multidomain approximation is based on the Chebyshev cell-averaged grid originally proposed by Cai *et al.* [5]. In this method, ''cell'' averaged quantities are defined on the Gauss–Chebyshev grid, while fluxes are defined at the more usual Lobatto points [37, 16, 17]. The cell-averaged method avoids many of the disadvantages of the methods just described. It is fully conservative, and it can be approximated to any temporal order of accuracy. It is also geometrically flexible because it does not require continuity of the transformations across interfaces. In more than one space dimension, the method does require special attention at the corners of the subdomains. Currently, a simple average of the multiple solutions is computed and broadcast to all contributing subdomains [17].

In this paper, we present a new multidomain spectral collocation method for the solution of compressible flow problems. The new method is based on a staggered grid, analogous to fully staggered grids often used with finite difference methods. The solutions are defined at the nodes of a Gauss quadrature rule, and the fluxes are evaluated at the nodes of a Gauss–Lobatto rule. Staggered-grid spectral approximations were first proposed for the solution of the incompressible Navier–Stokes equations (cf. [7, p. 234]). Our grid will be identical to the fully staggered grid of Bernardi and Maday [2].

The staggered grid multidomain method for compressible flow problems has all the desirable features found in the methods discussed above. First, like the cell-averaged method it is conservative. Thus, it should be possible to apply shock capturing techniques to the approximation. Subdomains can be defined independently of their neighbors, so the method is geometrically flexible. The interface condition can be computed to the same temporal accuracy as the interiors. However, in multiple space dimensions the method does not include (the Gauss rules being open) the corners of the subdomains. Thus, the coding of the method does not require special cases at corners and any number of subdomains can meet at a point without difficulty.

The paper is divided as follows. The algorithm is presented in the next section for problems in one space dimension, along with definitions of the notation used throughout the paper. We show that the staggered grid method is conservative, while methods that upwind derivatives are not. A scalar problem and a linear system will be used as examples to show that the method is exponentially convergent for smooth problems. Although we will be concerned in this paper primarily with steady problems, an example is included to show that high order temporal accuracy can also be obtained. In Section 3, we describe the algorithm in two space dimensions. We show that the method remains conservative and is also free-stream preserving. Section 4 provides three examples of the use of the method for two-dimensional problems. The first problem is that of a point source flow, for which there is an exact solution. We show that exponential accuracy is obtained for this problem. The second problem is a subsonic flow over a circular bump in a channel, and we show that the entropy errors decay exponentially fast. Finally, we solve a transonic flow in an axisymmetric converging–diverging nozzle and compare the results to experimental data. Concluding remarks are then made in the Section 5.

2. THE STAGGERED GRID APPROXIMATION IN ONE SPACE DIMENSION

2.1 *Notation*

The staggered grid approximation uses two grids to compute the solution values and advective fluxes. Unlike the common Chebyshev approximation [7], which uses only the nodes of the Gauss–Lobatto quadrature as collocation points, the new method uses both the Gauss and the Gauss–Lobatto points. We denote the points on the two grids by the Lobatto points, X_j, and the Gauss points, $\overline{X}_{j+1/2}$,

$$
\begin{aligned}
X_j &= \frac{1}{2}\left(1 - \cos\left(\frac{j\pi}{N}\right)\right), \quad j = 0, 1, ..., N, \\
\overline{X}_{j+1/2} &= \frac{1}{2}\left(1 - \cos\left(\frac{2j+1}{2N+2}\pi\right)\right), \quad j = 0, 1, ..., N-1.
\end{aligned} \tag{1}
$$

In (1), we have mapped the usual collocation points defined on $[-1, 1]$ to the more convenient unit interval. The overbar and half point notations for the Gauss points are used only for their value as an analogy to staggered grid finite difference methods. It must be understood that the Gauss points do not lie halfway between the Lobatto points [7].

Two polynomial approximations are defined, one for each grid. Let the space of polynomials of degree less than or equal to N be denoted $\mathbf{P}_N$. Let $l_j(\xi) \in \mathbf{P}_N$ be the Lagrange interpolating polynomial

$$l_j(\xi) = \prod_{\substack{i=0\\ i\neq j}}^{N} \left(\frac{\xi - X_i}{X_j - X_i}\right) \tag{2a}$$

defined on the Lobatto grid. On the Gauss grid, we define $h_{j+1/2} \in \mathbf{P}_{N-1}$ to be the polynomial

$$h_{j+1/2}(\xi) = \prod_{\substack{i=0\\ i\neq j}}^{N-1} \left(\frac{\xi - \overline{X}_{i+1/2}}{\overline{X}_{j+1/2} - \overline{X}_{i+1/2}}\right)\cdot \tag{2b}$$

Finally, let Q_j be a grid point value on the Lobatto grid and let $\overline{Q}_{j+1/2}$ be a value defined on the Gauss grid. Then we write the polynomials that interpolate these values as

$$Q(X) = \sum_{j=0}^{N} Q_j l_j(X) \tag{3a}$$

$$\overline{Q}(X) = \sum_{j=0}^{N-1} \overline{Q}_{j+1/2} h_{j+1/2}(X). \tag{3b}$$

2.2. *The One-Dimensional Staggered Grid Approximation for Scalar Equations*

To motivate the staggered grid approximation, we consider the approximation of scalar problems of the form

$$\begin{aligned} &u_t + f_x(u) = 0, \quad \partial f/\partial u > 0, x \in [a, b], t > 0,\\ &u(x, 0) = u_0(x)\\ &u(a, t) = g(t). \end{aligned} \tag{4}$$

The interval $[a, b]$ is subdivided into multiple, non-overlapping subdomains, $\Omega^k = [a_k, b_k]$, $k = 1, 2, ..., K$, which are ordered left to right. A simple linear transformation can be made to the unit interval, so that on each subdomain we solve the problem

$$u_t + \frac{1}{x_X} f_X(u) = 0, \quad X \in [0, 1], t > 0. \tag{5}$$

On each subdomain is placed the staggered grid defined by (1). For convenience, we will assume that the same number of points is used in each subdomain, but this is not required by the method. We then let $\overline{U}^k(X) \in \mathbf{P}_{N-1}$, defined by (3b), approximate the exact solution, u on Ω^k. Similarly, the flux is approximated by the polynomial $F^k(X) \in \mathbf{P}_N$, defined by (3a). Substitution of these approximations into (5) gives

$$\overline{U}_t^k + \frac{1}{x_X} \frac{\partial F^k(X)}{\partial X} = R^k(X), \quad k = 1, 2, ..., K. \tag{6}$$

To obtain the equations that define the solution unknowns at the Gauss points, we require that the residual, R, be zero at the Gauss points of the subdomain. This leads to the collocation approximation

$$\frac{d\overline{U}_{j+1/2}^k}{dt} + \frac{1}{x_X} \frac{\partial F^k(\overline{X}_{j+1/2})}{\partial X} = 0, \quad j = 0, 1, ..., N-1. \tag{7}$$

The spatial derivative operation in (7) can be evaluated as the multiplication of the vector of flux values by a derivative matrix, $\mathbf{D}$. From (3a), we see that

$$\frac{\partial F^k(\overline{X}_{j+1/2})}{\partial X} = \sum_{n=0}^{N} l_n'(\overline{X}_{j+1/2}) F_n^k = \sum_{n=0}^{N} d_{jn} F_n^k \tag{8}$$

so we write

$$\left.\frac{\partial F^k}{\partial X}\right|_{j+1/2} = (\mathbf{D}\mathbf{F}^k)_{j+1/2} = \sum_{n=0}^{N} d_{jn} F_n^k. \tag{9}$$

Thus, (7) can be written in vector form as

$$\frac{d\overline{\mathbf{U}}^k}{dt} + \frac{1}{x_X} \mathbf{D}\mathbf{F}^k = 0, \quad k = 1, 2, ..., K, \tag{10}$$

where $\overline{\mathbf{U}}^k = [U_{1/2}^k U_{3/2}^k \ldots U_{N-1/2}^k]^T$, $\mathbf{F}^k = [F_0^k F_1^k \ldots F_N^k]^T$.

To compute the flux values on the Lobatto grid, we use the following reconstruction procedure. We first evaluate the interpolant $\overline{U}^k(X) \in \mathbf{P}_{N-1}$ at the Lobatto points by multiplying the vector of solution values by an interpolation matrix, $\mathbf{I}$:

$$\overline{U}(X_j) = \sum_{n=0}^{N-1} \overline{U}_{n+1/2} h_{n+1/2}(X_j) = \sum_{n=0}^{N-1} i_{j,n+1/2} \overline{U}_{n+1/2}. \tag{11}$$

The family of characteristics of (5) runs left to right. Thus, we expect the use of the solution extrapolated to the left subdomain boundary to lead to an unstable procedure. To provide the proper characteristic domain of dependence, we use the boundary condition to define the $j = 0$ value on the furthest left subdomain. At subdomain interfaces, where two values $\overline{U}^{k-1}(1)$, $\overline{U}^k(0)$ are available, we choose the value computed from the left side of the interface. The result is an upwind evaluated approximation at both the left boundary and the interfaces. The fluxes, F_j, are then computed from the solution values on the Lobatto grid.

The method imposes the boundary conditions, weakly, through the definition of the flux, since the discrete solution values are not used directly at the boundary or interfaces. To see this, consider the single domain approximation of (4) for $f = u$. Then we can write the flux $F(X) = U(X) \in \mathbf{P}_N$ in terms of the interpolant $\overline{U}(X)$ and the boundary condition as

$$U(X) = \overline{U}(X) + [g - \overline{U}(a)]l_0(X), \tag{12}$$

so that the polynomial $U(X)$ satisfies

$$U(X_j) = \begin{cases} \overline{U}(X_j), & j = 1, 2, ..., N, \\ g, & j = 0. \end{cases} \tag{13}$$

Then (7) can be written as

$$\frac{d\overline{U}_{j+1/2}}{dt} + \frac{1}{x_X}\overline{U}'(X_{j+1/2}) = \frac{1}{x_X}[\overline{U}(X_0) - g]l_0'(X_{j+1/2}), \quad j = 0, 1, ..., N-1. \tag{14}$$

Thus, the boundary condition is imposed indirectly at each collocation point through the penalty term on the right.

Equation (10) is a system of ordinary differential equations that must be integrated in time to get the approximate solution values at the Gauss points. In principle, any common integration procedure can be used. We have chosen to use low storage Runge–Kutta methods that require only 2-N storage locations. For some of the computation of steady-state problems, for which the time discretization is only an iterative procedure, we have used a mid-point rule,

$$\begin{aligned} \overline{U}^{k,n+1/2}_{j+1/2} &= \overline{U}^{k,n}_{j+1/2} - \frac{\Delta t}{2}\frac{1}{x_X}\frac{\partial F^{k,n}}{\partial X}\bigg|_{j+1/2}, \quad j = 0, 1, ..., N-1, \\ \overline{U}^{k,n+1}_{j+1/2} &= \overline{U}^{k,n}_{j+1/2} - \Delta t\frac{1}{x_X}\frac{\partial F^{k,n+1/2}}{\partial X}\bigg|_{j+1/2}, \quad j = 0, 1, ..., N-1. \end{aligned} \tag{15}$$

This method appears to have a good balance between the time step that can be used and temporal damping introduced by the scheme. With additional knowledge of the eigenvalue structure of the differentiation matrices, other choices might include schemes optimized for rapid convergence to steady state, such as those discussed in [11, 10, 9].

More generally, we have used low-storage methods from the first through fourth orders. For time dependent problems, we have used the third-order 2-N storage method of Williamson [41], and the more recent fourth-order scheme of Carpenter and Kennedy [8]. We note that it should also be advantageous to use the new Runge–Kutta methods derived by Hu *et al.* [22], which are optimized to minimize the phase and dissipation error introduced by the temporal approximation.

TABLE I

Time Step Coefficient, $\Delta t = C/N^2$

Method	Coefficient
Standard Lobatto	35
Staggered	36

The time step restriction for the staggered grid approximation is comparable to the restriction required by the standard Lobatto approximation for the model equation $u_t + u_x = 0$, $-1 \le x \le 1$. As expected, the time step goes as $\Delta t \propto 1/N^2$. In Table I we show the coefficients of proportionality using the Carpenter–Kennedy fourth-order Runge–Kutta formula.

To summarize the staggered grid procedure, we present the following algorithm for the scalar problem described above.

ALGORITHM I (`Staggered grid, scalar, 1D`).

```
1. Interpolate $\overline{U} = [\overline{U}_{1/2}, \overline{U}_{3/2}, ..., \overline{U}_{N-1/2}]^T$ to the Lobatto points:
   Compute the matrix-vector product $\mathbf{U}^k = \mathbf{I}^k\overline{\mathbf{U}}^k$ defined by (11) for each subdomain
2. Compute the flux values at internal points on Lobatto Grid:
   $F^k_j = f(U^k_j)$,  $j = 1, 2, ..., N$, $k = 1, 2, ..., K$
3. Apply boundary and interface conditions:
   $F^1_0 = f(g)$
   $F^k_0 = f(U^{k-1}_N), k = 2, 3, ..., K$
4. Differentiate Flux and evaluate on Gauss grid by subdomain:
   Compute the matrix-vector product $\mathbf{D}^k\mathbf{F}^k$, $k = 1, 2, ..., N$ by Eq. (9).
5. Update the solution by subdomain:
   Integrate (10) by the chosen ODE solver, repeating Steps 1–4, as necessary.
6. Repeat 1–5 until done.
```

We note that the method requires two matrix–vector products per Runge–Kutta stage. This is twice the work of a Lobatto grid method, or of the cell-averaged method.

Thus, there is no speed advantage for the method in one space dimension.

A desirable feature of the staggered grid approximation, (7), is that it is conservative. To show conservation, we define the quadrature

$$\int_0^1 F(X)\,dX = \sum_{j=0}^{N-1} F_{j+1/2}w_{j+1/2} \quad \forall F \in \mathbf{P}_{N-1}$$
$$w_{j+1/2} = \int_0^1 h_{j+1/2}(X)\,dX. \tag{16}$$

For each j, we multiply (7) by $x_X w_{j+1/2}$. The sum over all points and all subdomains is

$$\sum_{k=1}^{K}\sum_{j=0}^{N-1} w_{j+1/2}\left(x_X^k \frac{d\overline{U}_{j+1/2}}{dt} + F'^k_{j+1/2}\right) = 0. \tag{17}$$

Now, $\overline{U}^k(X)$, $F'^k(X) \in \mathbf{P}_{N-1}$, and x_X is a polynomial of degree zero, so we can replace the sum over j by integrals to get

$$\sum_{k=1}^{K}\int_{\Omega^k}\left(x_X^k \frac{d\overline{U}(X)}{dt} + F_X^k(X)\right)dX = 0. \tag{18}$$

Upon integrating the flux derivatives, the interface contributions cancel, and

$$\frac{d}{dt}\left\{\sum_{k=1}^{K}\int_{\Omega^k}\overline{U}^k(X)x_X^k\,dX\right\} = F^1(0) - F^K(1). \tag{19}$$

In contrast, a multidomain method defined on the Lobatto points, where the upwind value of the flux *derivative* is used at the interface (e.g., [19]) is not truly conservative. To show this, it is sufficient to consider two subdomains, Ω^L and Ω^R for which $x_X = 1$. Using the upwind derivatives at the interface, one integrates the following system of equations:

$$\begin{aligned}
&U_0^L = g\\
&\frac{dU_j^L}{dt} = -F'^L_j, \quad j = 1, 2, ..., N-1,\\
&\frac{dU_j^R}{dt} = -F'^R_j, \quad j = 1, 2, ..., N-1,\\
&\frac{dU_N^L}{dt} = \frac{dU_0^R}{dt} = -F'^L_N\\
&\frac{dU_N^R}{dt} = -F'^R_N.
\end{aligned} \tag{20}$$

We then multiply each equation by its associated Clenshaw–Curtis weight [7] and sum to get

$$\begin{aligned}
&\sum_{j=0}^{N}\frac{dU_j^L}{dt}w_j^L + \sum_{j=0}^{N}\frac{dU_j^R}{dt}w_j^R\\
&\quad = w_0^L\left(F'^L_0 + \frac{dg}{dt}\right) - \sum_{j=0}^{N} w_j^L F'^L_j - \sum_{j=0}^{N} w_j^R F'^R_j\\
&\qquad - w_0^R[F'^L_N - F'^R_0]\\
&\quad = F^L(0) - F^R(1) - w_0^R[F'^L_N - F'^R_0]\\
&\qquad + w_0^L\left(F'^L_0 + \frac{dg}{dt}\right).
\end{aligned} \tag{21}$$

Equations (21) shows that there is a contribution at the interface proportional to the jump in the derivative across the interface. For smooth enough functions, this is not expected to be a problem, since the difference between the derivatives should go to zero exponentially fast, while the coefficient decays as $O(1/N^2)$.

As an example of solutions computed using Algorithm I, we compute a steady solution of the equation $u_t + u_x = f, x \in [0, 2], t > 0$. Scalar examples of one-dimensional time dependent problems can be found in [29]. Comparisons to a variety of finite difference methods can be found in [39]. The initial and boundary conditions were chosen so that the exact steady solution is $u(x) = \tanh((x - 1.5)/2)$. Figure 1 shows the solution computed using three subdomains and eight points per subdomain.

Convergence of the error is exponential for this problem. Figure 2 shows the error plotted as a function of the Gauss polynomial order for the subdivision shown in Fig. 1. For comparison, we have also plotted the error of the single grid multidomain method (20). We see that the staggered grid approximation is at least as accurate as the non-staggered grid approximation, and sometimes it is more accu-

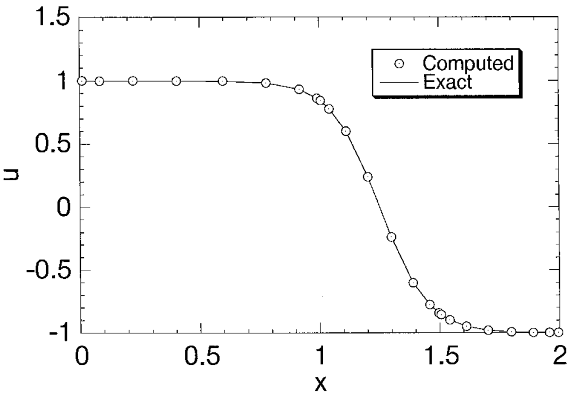

FIG. 1. Steady solution of a scalar wave equation.

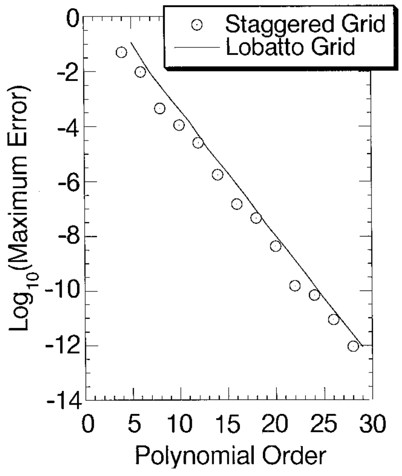

FIG. 2. Convergence of the error for the staggered grid multidomain method compared to a non-staggered multidomain approximation.

rate by a factor of four. This increase in accuracy is consistent with the observations of [15]. As argued in [15], enforcing the boundary condition weakly through a penalty term (see Eq. (14)) is typically more accurate than enforcing it exactly.

2.3. *The One-Dimensional Staggered Grid Approximation for Systems*

Algorithm I can be easily extended to systems of hyperbolic equations of the form

$$\begin{aligned}&\mathbf{Q}_t + \mathbf{F}_x(\mathbf{Q}) = 0, \quad x \in [a,b], t > 0,\\&\mathbf{Q}(x,0) = \mathbf{Q}_0(x),\end{aligned} \tag{22}$$

where $\mathbf{Q}$ and $\mathbf{F}$ are m-vectors. We assume that the system is hyperbolic, that is, the Jacobian matrix $\mathbf{A} = \partial\mathbf{F}/\partial\mathbf{Q} = \mathbf{Z}\Lambda\mathbf{Z}^{-1}$, where Λ is a real diagonal matrix. We further assume, as is the case for the Euler gas-dynamics equations, that the flux can be written as $\mathbf{F} = \mathbf{AQ}$. To complete (22), we assume that appropriate dissipative boundary conditions are applied.

The approximation of the system follows that of the scalar equation, except for the treatment of boundary and interface conditions. At an interface between subdomains $k-1$ and k, there are two vector values of the interpolated solution available, $\mathbf{Q}_N^{k-1}$ and $\mathbf{Q}_0^k$. The computed flux must use these two values to allow waves to propagate through the interfaces. For constant coefficient linear problems, we can write

$$\mathbf{F} = \mathbf{AQ} = \mathbf{Z}\Lambda\mathbf{Z}^{-1}\mathbf{Q} = \mathbf{Z}\Lambda^{+}\mathbf{Z}^{-1}\mathbf{Q} + \mathbf{Z}\Lambda^{-}\mathbf{Z}^{-1}\mathbf{Q}, \tag{23}$$

where $\Lambda^{\pm} = \Lambda \pm |\Lambda|$. The first term represents waves moving left to right, and the second represents waves moving right to left. An upwind approximation chooses $\mathbf{Q}_N^{k-1}$ for the right going components, and $\mathbf{Q}_0^k$ for the left going components to give

$$F_N^{k-1} = F_0^k = \mathscr{F}(\mathbf{Q}_N^{k-1}, \mathbf{Q}_0^k) \equiv \mathbf{Z}\Lambda^{+}\mathbf{Z}^{-1}\mathbf{Q}_N^{k-1} + \mathbf{Z}\Lambda^{-}\mathbf{Z}^{-1}\mathbf{Q}_0^k. \tag{24}$$

Characteristic decompositions for nonlinear flux vectors have been addressed extensively in the finite difference community (Ref. [21]). We have considered flux vector splitting and flux difference splitting.

The resolution of the jump at the subdomain interfaces can be easily viewed using flux vector splitting. The flux is decomposed into a right going and a left going flux, $\mathbf{F}(\mathbf{Q}) = \mathbf{F}^{+}(\mathbf{Q}) + \mathbf{F}^{-}(\mathbf{Q})$. The splitting is done so that the Jacobian matrix of $\mathbf{F}^{+}$ has only positive eigenvalues and the Jacobian of $\mathbf{F}^{-}$ has only negative eigenvalues. Examples are the Van Leer [40] splitting and the more recent splitting of Liou and Steffen [32]. Using flux vector splitting, the positive flux is evaluted using the solution from the left, the negative flux is computed using the value from the right:

$$F_N^{k-1} = F_0^k = \mathscr{F}(\mathbf{Q}_N^{k-1}, \mathbf{Q}_0^k) \equiv \mathbf{F}^{+}(\mathbf{Q}_N^{k-1}) + \mathbf{F}^{-}(\mathbf{Q}_0^k). \tag{25}$$

Van Leer's flux vector splitting was used in [19] to compute the flux derivatives at interfaces for the advective part of the Navier–Stokes equations.

As an interface treatment, flux vector splitting has the desirable feature that the positive and negative fluxes can be computed within a subdomain without regard to the neighbors. The final flux computation, (25), requires only a simple sum of the boundary fluxes. However, we found that the Van Leer splitting applied to the staggered grid scheme was unstable for some long time integrations.

As an alternative method to compute the interface flux, we have chosen to use an approximate Riemann solver. This approach was also taken by Giannakouros and Karniadakis [16]. Several solver choices are possible, but we have used Roe's [36] solver with the entropy fix. Formally, given the two states $\mathbf{Q}_N^{k-1}$ and $\mathbf{Q}_0^k$, we write

$$\begin{aligned}\mathscr{F}(\mathbf{Q}_N^{k-1}, \mathbf{Q}_0^k) &= \tfrac{1}{2}(\mathbf{F}(\mathbf{Q}_N^{k-1}) + \mathbf{F}(\mathbf{Q}_0^k))\\&\quad - \tfrac{1}{2}\mathbf{R}|\Lambda|\mathbf{R}^{-1}(\mathbf{Q}_0^k - \mathbf{Q}_N^{k-1}),\end{aligned} \tag{26}$$

where $\mathbf{R}$ is the matrix of the right eigenvectors of the Jacobian of $\mathbf{F}$, computed using the Roe-average of $\mathbf{Q}_N^{k-1}$ and $\mathbf{Q}_0^k$. Equation (26) is modified to correct the entropy across sonic points [21].

Boundaries can be considered as interfaces between the computed solution and the solution assumed to exist outside the computational region, if fully known. Thus, we can compute the boundary flux by

$$F_0^1 = \mathscr{F}(\mathbf{Q}(a,t), \mathbf{Q}_0^1) \tag{27a}$$

on the left, and

$$F_N^K = \mathscr{F}(\mathbf{Q}_N^K, \mathbf{Q}(b,t)) \tag{27b}$$

on the right, where $\mathbf{Q}(a, t)$ and $\mathbf{Q}(b, t)$ represent the exterior solution at the boundaries. Other ways to compute the boundary flux when the full exterior solution is not known will be described in regard to specific problems in Section 4.

In summary, for systems of equations, we have the following algorithm.

```
ALGORITHM II (Staggered grid, system, 1D).
1. Interpolate the Gauss-point solution values to the Lobatto points:
   Compute the matrix–vector product $\mathbf{Q}^k = \mathbf{I}^k\overline{\mathbf{Q}}^k$ by Eq. (11) for each subdomain
2. Compute the interior point fluxes:
   $\mathbf{F}_j^k = \mathbf{F}(\mathbf{Q}_j^k), \quad j = 1, 2, ..., N, k = 1, 2, ..., K$
3. Apply the interface conditions:
   $\mathbf{F}_N^{k-1} = \mathbf{F}_0^k = \mathscr{F}(\mathbf{Q}_N^{k-1}, \mathbf{Q}_0^k), k = 1, 2, ..., K$
4. Apply boundary conditions at left and right
5. Compute spatial derivatives at Gauss points:
   Compute the matrix–vector product by Eq. (9).
6. Update the solution at the Gauss points
   $\frac{d\overline{\mathbf{Q}}_{j+1/2}^k}{dt} + \frac{1}{x_X}\mathbf{F}'^k_{j+1/2} = 0, j = 0, 1, ... N-1, k = 1, 2, ..., K,$
   repeating Steps 1–4 for each Runge–Kutta stage.
7. Repeat Steps 1–6 until done.
```

As an example of the application of Algorithm II, we solve the problem

$$\mathbf{Q}_t + \mathbf{F}_x = 0, \quad x \in [-1, 4], t > 0$$
$$\mathbf{Q} = \begin{bmatrix} u \\ v \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\mathbf{Q}. \tag{28}$$

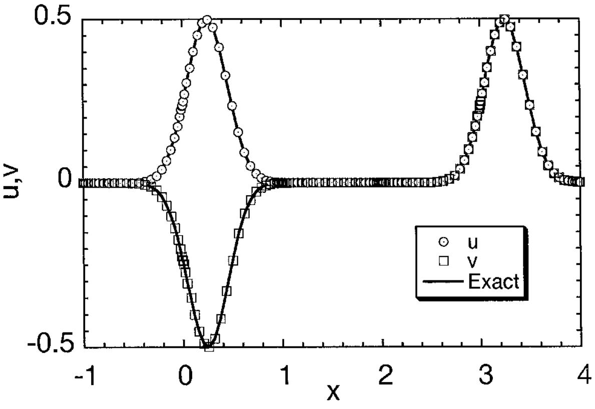

FIG. 3. Solution of the system (27) at $t = 0.75$ using four subdomains.

Further examples that model unsteady acoustic propagation, including acoustic propagation in a quasi-one-dimensional nozzle, can be found in [29]. The initial condition for (28) was chosen to be a Gaussian pulse with the peak at $x = 1$,

$$\mathbf{Q}(x, 0) = \begin{bmatrix} e^{-12(x-1)^2} \\ 0 \end{bmatrix}. \tag{29}$$

We specify boundary conditions so that the waves pass through the boundaries without reflection,

$$\begin{aligned} u(-1, t) - v(-1, t) &= e^{-12(t-2)^2} \\ u(4, t) + v(4, t) &= e^{-12(3-3t)^2}. \end{aligned} \tag{30}$$

The solution was computed using four subdomains of equal length and with the same number of points within each subdomain. For the time discretization, we have used both the Williamson [41] third-order and the Carpenter and Kennedy [8] fourth-order schemes. Figure 3 shows the solution at time $t = 0.75$. For a small enough time step, spectral decay of the error is observed, as shown in Fig. 4.

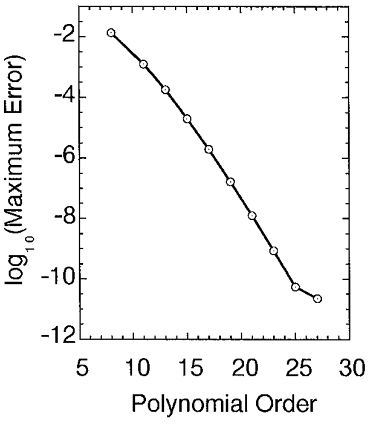

FIG. 4. Decay of the spatial error of the system (28).

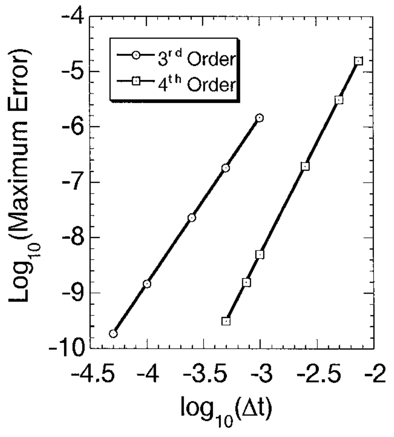

FIG. 5. Decay of the temporal error of the system (28).

To study the temporal accuracy, we computed the solution on the finest grid, $N = 25$, so that the spatial accuracy was close to rounding error. A plot of the errors as a function of Δt for the third- and fourth-order methods is shown in Fig. 5. A least squares fit to the errors indicates a slope of 2.995 for the third order and 3.998 for the fourth-order method, so the expected high order temporal accuracy is obtained. We also see that over the Δt range shown, the fourth-order method is about two orders of magnitude more accurate than the third. This is consistent with the observations of [8].

The conclusion that the temporal accuracy is determined by the time differencing approximation carries over to the non-linear case. As an example, we considered the solution of $u_t + 1/2(u^2)_x = 0$ with the initial condition $u(x, 0) = \tanh(10x)$. In this case, a least squares fit to the logarithm of the error as a function of the logarithm of the time step gives a slope of 4.014.

3. THE TWO-DIMENSIONAL APPROXIMATION

We now describe the approximation of the Euler equations of gas-dynamics in conservative form,

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{0}, \tag{31a}$$

where $\mathbf{Q}$ is the vector of solution unknowns and $\mathbf{F(Q)}$ and $\mathbf{G(Q)}$ are the advective flux vectors

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho u \\ p + \rho u^2 \\ \rho u v \\ u(\rho e + p) \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho v \\ \rho u v \\ p + \rho v^2 \\ v(\rho e + p) \end{bmatrix}. \tag{31b}$$

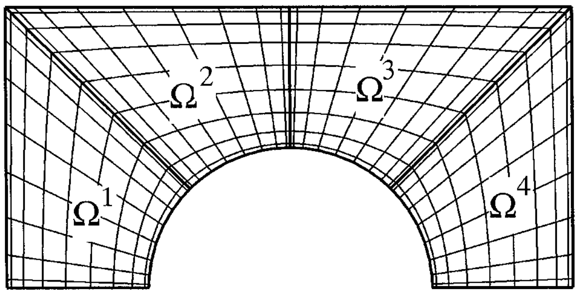

FIG. 6. Diagram of a subdomain decomposition in 2D.

We assume $\gamma = 1.4$ and that $\rho e = p/(\gamma - 1) + \rho(u^2 + v^2)/2$. For axisymmetric problems, such as the transonic flow in the converging–diverging nozzle discussed later, we interpret x as the axial coordinate and y as the radial coordinate. We then add to the right-hand side of (31a) the vector

$$\mathbf{H} = \frac{1}{y} \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 \\ v(\rho e + p) \end{bmatrix}. \tag{32}$$

3.1. *Mapping in Two Space Dimensions*

In two space dimensions, we subdivide a computational domain, Ω, into quadrilateral subdomains, Ω^k, $k = 1, 2, \ldots, K$. Figure 6 shows an example of a division of a region into four subdomains. We make three assumptions about the subdivision in this paper. First, we allow subdomains to intersect only at a point or along an entire side. Second, we assume that the approximation is conforming, so that grid lines coincide across subdomain interfaces. Finally, we assume that the subdomain boundaries do not move in time. In the discussion that follows, we will make the assumption that the same polynomial order is used in each space direction and for each subdomain. In practice, the number of grid points can vary, as long as the approximation remains conforming at subdomain interfaces.

Subdomains are mapped onto the unit square by an isoparametric mapping. Let the vector function $\mathbf{g}(s)$, $0 \leq s \leq 1$, define a parametric curve. Define also the polynomial of degree N that interpolates $\mathbf{g}$ at the Gauss–Lobatto points to be

$$\Gamma(s) = \sum_{j=0}^{N} \mathbf{g}(s_j) l_j(s). \tag{33}$$

Four such polynomial curves, $\Gamma_m(s)$, $m = 1, 2, 3, 4$, counted counterclockwise, bound each subdomain. We map each

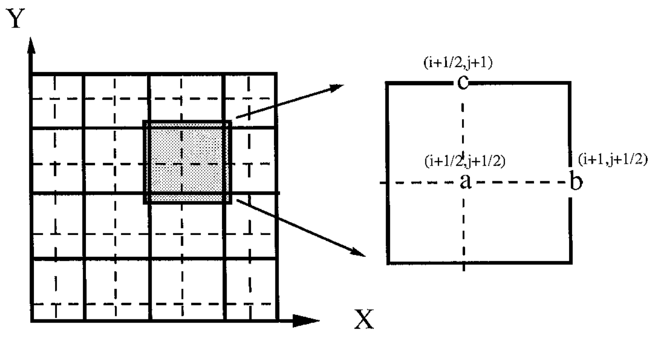

FIG. 7. Diagram of the fully staggered grid in two space dimensions.

subdomain onto the unit square by the linear blending formula

$$\begin{aligned}\mathbf{x}^N(X, Y) = (1 - Y)\Gamma_1(X) + Y\Gamma_3(X) + (1 - X)\Gamma_4(Y) \\ + X\Gamma_2(Y) - \mathbf{x}_1(1 - X)(1 - Y) - \mathbf{x}_2X(1 - Y) \\ - \mathbf{x}_3XY - \mathbf{x}_4(1 - X)Y,\end{aligned} \tag{34}$$

where the $\mathbf{x}_j$'s represent the locations of the corners of the subdomain, counted counterclockwise.

Under the mapping $\Omega^k \leftrightarrow [0, 1] \times [0, 1]$ given by (34), the Euler equations (31) become

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{J}\left[\frac{\partial \tilde{\mathbf{F}}}{\partial X} + \frac{\partial \tilde{\mathbf{G}}}{\partial Y}\right] = 0, \tag{35a}$$

where

$$\begin{aligned}\tilde{\mathbf{F}} = y_Y^N\mathbf{F} - x_Y^N\mathbf{G}, \quad \tilde{\mathbf{G}} = -y_X^N\mathbf{F} + x_X^N\mathbf{G} \\ J(X, Y) = x_X^Ny_Y^N - x_Y^Ny_X^N.\end{aligned} \tag{35b}$$

Since we assume here that the subdomain boundaries do not move in time, we can write (35a) as

$$\frac{\partial \tilde{\mathbf{Q}}}{\partial t} + \frac{\partial \tilde{\mathbf{F}}(\mathbf{Q})}{\partial X} + \frac{\partial \tilde{\mathbf{G}}(\mathbf{Q})}{\partial Y} = 0, \tag{36}$$

where $\tilde{\mathbf{Q}} = J\mathbf{Q}$ and the fluxes are still defined by (35b).

3.2 *The Staggered Grid*

A fully staggered grid is used in two space dimensions. A schematic of the grid on a single subdomain is shown in Fig. 7. The grid is the same as the staggered grid proposed by Bernardi and Maday [2] for the solution of the incompressible Navier–Stokes equations. In what follows, we will ignore superscripts that denote which subdomain is being considered, unless necessary.

Points of type "a" in Fig. 7 represent the Gauss/Gauss points $(\overline{X}_{i+1/2}, \overline{Y}_{j+1/2})$, $i, j = 0, 1, ..., N - 1$. The grid that results from these points is the tensor product of the one-dimensional Gauss grid defined in (1). We approximate the solution and the transformation Jacobian at the Gauss/Gauss points and denote them by $\overline{\mathbf{Q}}_{i+1/2,j+1/2}$ and $\overline{J}_{i+1/2,j+1/2} = J(\overline{X}_{i+1/2,j+1/2}, \overline{Y}_{i+1/2,j+1/2})$. From these, we compute the Gauss point values $\tilde{\mathbf{Q}}_{i+1/2,j+1/2} = \overline{J}_{i+1/2,j+1/2}\overline{\mathbf{Q}}_{i+1/2,j+1/2}$. Finally, the interpolant of the solution through the Gauss points is a polynomial in $\mathbf{P}_{N-1,N-1} = \mathbf{P}_{N-1} \otimes \mathbf{P}_{N-1}$:

$$\tilde{\mathbf{Q}}(X, Y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \tilde{\mathbf{Q}}_{i+1/2,j+1/2}h_{i+1/2}(X)h_{j+1/2}(Y). \tag{37}$$

The points of type "b" in Fig. 7 form the Lobatto/Gauss grid $(X_i, \overline{Y}_{j+1/2})$, $i, j = 0, 1, ..., N$. On this grid are evaluated the horizontal flux vector, $\tilde{\mathbf{F}}$ and the metric terms y_Y and x_Y. The metric terms are computed as $\partial y^N(X_i, \overline{Y}_{j+1/2})/\partial Y$ and $\partial x^N(X_i, \overline{Y}_{j+1/2})/\partial Y$. At points interior to a subdomain, the horizontal flux is computed by

$$\begin{aligned}\tilde{\mathbf{F}}_{i,j+1/2} = y_Y^N(X_i, \overline{Y}_{j+1/2})\mathbf{F}(\overline{Q}(X_i, \overline{Y}_{j+1/2})) \\ -x_Y^N(X_i, \overline{Y}_{j+1/2})\mathbf{G}(\overline{Q}(X_i, \overline{Y}_{j+1/2})),\end{aligned} \tag{38}$$

where $\overline{\mathbf{Q}}$ is a polynomial of the type (37) that passes through the values $\tilde{\mathbf{Q}}_{i+1/2,j+1/2}/\overline{J}_{i+1/2,j+1/2}$. The computation of the flux at boundary and interface points is described in the next subsection.

The vertical flux and the derivatives y_X and x_X are computed on the Gauss/Lobatto grid, marked by "c" on Fig. 7. The points on this grid are $(\overline{X}_{i+1/2}, Y_j)$, $i, j = 0, 1, ..., N - 1$. The metric terms are computed as $\partial y^N(\overline{X}_{i+1/2}, Y_j)/\partial X$ and $\partial x^N(\overline{X}_{i+1/2}, Y_j)/\partial X$. The vertical flux is computed at interior points by

$$\begin{aligned}\tilde{\mathbf{G}}_{i+1/2,j} = -y_X^N(\overline{X}_{i+1/2}, Y_j)\mathbf{F}(\overline{Q}(\overline{X}_{i+1/2}, Y_j)) \\ + x_X^N(\overline{X}_{i+1/2}, Y_j)\mathbf{G}(\overline{Q}(\overline{X}_{i+1/2}, Y_j))\end{aligned} \tag{39}$$

and at boundary points as described in Section 3.3.

It may appear that to define quantities on three different grids would lead to a significantly more complicated method than a single grid Lobatto approximation. This turns out not to be the case. The definitions of the fluxes on the staggered grid by (38) and (39) mean that the reconstruction procedure, i.e., the interpolation of the solution needed to compute the fluxes at the Lobatto points, is not a two-dimensional operation. Rather, it is the less expensive sequence of one-dimensional interpolations, given by (11).

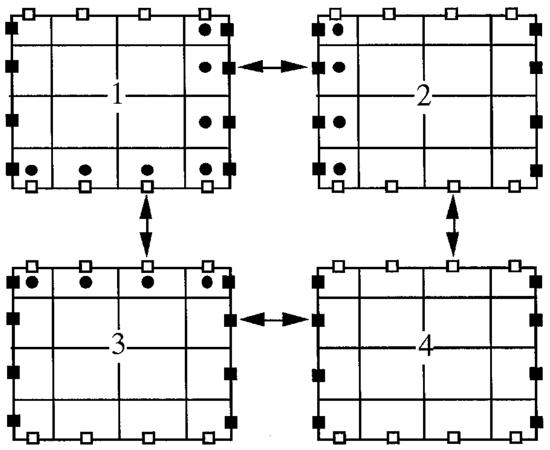

FIG. 8. Diagram of four subdomains showing locations near interfaces where solutions and fluxes are computed. Symbols: ● solution; ■, F; □, G.

The values of the solution vector required to compute the flux vectors are actually

$$\overline{\mathbf{Q}}(X_i, \overline{Y}_{j+1/2}) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \overline{\mathbf{Q}}_{i+1/2,j+1/2} h_{i+1/2}(X_i) h_{j+1/2}(\overline{Y}_{j+1/2})$$
$$= \sum_{i=0}^{N-1} \overline{\mathbf{Q}}_{i+1/2,j+1/2} h_{i+1/2}(X_i) \tag{40a}$$

and

$$\overline{\mathbf{Q}}(\overline{X}_{i+1/2}, Y_j) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \overline{\mathbf{Q}}_{i+1/2,j+1/2} h_{i+1/2}(\overline{X}_{i+1/2}) h_{j+1/2}(Y_j)$$
$$= \sum_{j=0}^{N-1} \overline{\mathbf{Q}}_{i+1/2,j+1/2} h_{j+1/2}(Y_j) \tag{40b}$$

since, by construction,

$$h_{m+1/2}(\overline{Y}_{j+1/2}) = \delta_{m,j}$$
$$h_{n+1/2}(\overline{X}_{i+1/2}) = \delta_{n,i}$$

3.3. *Interface and Boundary Treatment*

To describe how we compute the interface and boundary conditions using the staggered grid approximation, we will refer to Fig. 8, which schematically represents four subdomains and the locations at which solution and flux values are computed. Only the collocation points near the boundaries are marked. The circles represent the solution values, which are located on the Gauss/Gauss grid. The locations of the horizontal flux values, $\tilde{\mathbf{F}}_{i,j+1/2}$, are represented by solid squares. The locations of the vertical flux values, $\tilde{\mathbf{G}}_{i+1/2,j}$, are marked by hollow squares. From the diagram, we see that along the interfaces between subdomains 1 and 2 and between subdomains 3 and 4, only the horizontal fluxes need to be computed. Along horizontal interfaces, like those between subdomains 1 and 3, only the vertical flux needs to be computed. Because the grid is fully staggered, the coupling is through subdomain faces only, not through the corners.

Figure 8 indicates a significant advantage of the fully staggered grid over an unstaggered grid. In the unstaggered approximation, for example, as described in [27], special corner algorithms must be devised to ensure correct propagation of waves through the corners. Each special case must be coded separately. Also, the choice of bicharacteristics that determines the domains of dependence becomes more complex as the number of subdomains/boundaries that come together at a point increases, making the derivation of these special cases more difficult. The staggered approximation does not include subdomain corners, so conditions do not have to be specified at corner points. Any number of subdomains can come together at a point without the need for special point approximations. No special code is required even for very complex subdomain topologies.

The interpolation of the solution by (40) produces two solution values at an interface point, one from each of the two contributing subdomains. As in the one-dimensional case, we do not expect these two values to coincide, except in the limit of infinite resolution. A single flux is calculated, as described for the one-dimensional problem, except that we only consider waves propagating normal to the interface. This normal wave approximation is common for finite difference approximations [21] and has been used in [19, 17, 4] for spectral approximations. We note, however, that other two-dimensional wave decompositions are possible, like those surveyed in [34].

Physical boundaries can be viewed as interfaces between the external flow and the computational region. Wall boundaries can be computed by imposing an opposing flow that enforces zero normal momentum flux across the interface. Subsonic inflow and outflow boundaries can be computed by replacing the solution that would have come from a neighboring subdomain by the free-stream values, if they are known. If the full state of the exterior flow is not known, the known quantities can be specified and the remaining quantities can be computed by a characteristic method. Once all solution quantities are known on the boundary, the flux can be computed. An example of this approach is provided in Section 4.3. Supersonic outflow boundaries require no extra conditions.

3.4. *Discretization of the Equations*

Once the fluxes are computed, the spatial discretization can be made. From the discrete flux values are defined the polynomials

$$\tilde{\mathbf{F}}(X,Y) = \sum_{i=0}^{N}\sum_{j=0}^{N-1} \tilde{\mathbf{F}}_{i,j+1/2} l_i(X) h_{j+1/2}(Y)$$
$$\tilde{\mathbf{G}}(X,Y) = \sum_{i=0}^{N-1}\sum_{j=0}^{N} \tilde{\mathbf{G}}_{i+1/2,j} h_{i+1/2}(X) l_j(Y). \tag{41}$$

Derivatives of the interpolating polynomials are then evaluated at the Gauss/Gauss grid points. Like the reconstruction procedure, the differentiation of (41) can also be done as a sequence of one-dimensional operations:

$$\left.\frac{\partial \tilde{\mathbf{F}}}{\partial X}\right|_{i+1/2,j+1/2} = \sum_{n=0}^{N} \tilde{\mathbf{F}}_{n,j+1/2} l_n'(\overline{X}_{i+1/2})$$
$$\left.\frac{\partial \tilde{\mathbf{G}}}{\partial Y}\right|_{i+1/2,j+1/2} = \sum_{m=0}^{N} \tilde{\mathbf{G}}_{i+1/2,m} l_m'(\overline{Y}_{j+1/2}). \tag{42}$$

Because both interpolation and differentiation operations must be performed at each step, the total work of the staggered grid method is twice that of a method that only uses the Lobatto grid. The new method requires the same amount of work, however, as the cell-averaged method [17]. The reconstruction procedure in two space dimensions for the cell-averaged method is more complex than in one and requires the same amount of work as both the interpolation and differentiation operations here.

Finally, from the definitions (37)–(42), the semi-discrete approximation for the solution unknowns can be written as

$$\left.\frac{d\tilde{\mathbf{Q}}}{dt}\right|_{i+1/2,j+1/2} + \left[\frac{\partial \tilde{\mathbf{F}}}{\partial X} + \frac{\partial \tilde{\mathbf{G}}}{\partial Y}\right]_{i+1/2,j+1/2} = \mathbf{0}, \qquad \begin{cases} i = 0,1,\dots,N-1, \\ j = 0,1,\dots,N-1. \end{cases} \tag{43}$$

Equation 43 can be integrated in time as described in Section 2.2.

3.5. *Properties of the Staggered Grid Approximation*

The staggered grid approximation is both conservative and free-stream preserving. A net gain or loss of $\tilde{\mathbf{Q}}$ is determined only by the flux through the exterior boundaries. If the solution is constant in space, then the solution must remain constant in time, even in the presence of a spatially varying mapping.

We first show that the staggered grid approximation is conservative. It is sufficient to consider the four subdomains shown in Fig. 8. Let the quadrature weights $w_{i+1/2}$, $\eta_{j+1/2}$ be defined so that

$$\int_0^1\int_0^1 P\,dX\,dY = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} P_{i+1/2,j+1/2} w_{i+1/2}\eta_{j+1/2} \qquad \forall P \in \mathbf{P}_{N-1,N-1}. \tag{44}$$

By exactness of the quadrature, the sum of Eq. (43) times $w_{i+1/2}\eta_{j+1/2}$ over all the points within a subdomain is

$$\int_0^1\int_0^1 \frac{d\tilde{\mathbf{Q}}}{dt}\,dX\,dY = \sum_{i,j=0}^{N-1} \left.\frac{d\tilde{\mathbf{Q}}}{dt}\right|_{i+1/2,j+1/2} w_{i+1/2}\eta_{j+1/2}$$
$$= \sum_{i,j=0}^{N-1}\left[\frac{\partial \tilde{\mathbf{F}}}{\partial X} + \frac{\partial \tilde{\mathbf{G}}}{\partial Y}\right]_{i+1/2,j+1/2} w_{i+1/2}\eta_{j+1/2}$$
$$= \int_0^1\int_0^1 \left[\frac{\partial \tilde{\mathbf{F}}}{\partial X} + \frac{\partial \tilde{\mathbf{G}}}{\partial Y}\right] dX\,dY. \tag{45}$$

Thus, for each subdomain,

$$\frac{d}{dt}\int_0^1\int_0^1 \tilde{\mathbf{Q}}\,dX\,dY = -\int_0^1 \tilde{\mathbf{F}}(1,Y)\,dY + \int_0^1 \tilde{\mathbf{F}}(0,Y)\,dY - \int_0^1 \tilde{\mathbf{G}}(X,1)\,dX + \int_0^1 \tilde{\mathbf{G}}(X,0)\,dX. \tag{46}$$

When (46) is summed over all subdomains, the interior integrals cancel so that only the boundary contributions remain:

$$\frac{d}{dt}\sum_{k=1}^{4}\int_0^1\int_0^1 \tilde{\mathbf{Q}}^k\,dX\,dY = \int_0^1 (\tilde{\mathbf{F}}^1(0,Y) + \tilde{\mathbf{F}}^3(0,Y))\,dY - \int_0^1 (\tilde{\mathbf{F}}^2(1,Y) + \tilde{\mathbf{F}}^4(1,Y))\,dY + \int_0^1 (\tilde{\mathbf{G}}^3(X,0) + \tilde{\mathbf{G}}^4(X,0))\,dX - \int_0^1 (\tilde{\mathbf{G}}^1(X,1) + \tilde{\mathbf{G}}^2(X,1))\,dX. \tag{47}$$

The staggered grid approximation is also free-stream preserving, which means that the isoparametric spatial mappings do not introduce false source terms. It is sufficient to consider the approximation within one subdomain, since all derivatives are computed locally by subdomain. If we take $\mathbf{F}(\mathbf{Q}) = \mathbf{G}(\mathbf{Q}) = \mathbf{1}$, then the approximation (43) becomes

$$\left.\frac{d\tilde{\mathbf{Q}}}{dt}\right|_{i+1/2,j+1/2} + \mathbf{1}\left[\frac{\partial}{\partial X}(y_Y^N - x_Y^N) + \frac{\partial}{\partial Y}(-y_X^N + x_X^N)\right]_{i+1/2,j+1/2} = \mathbf{0}. \tag{48}$$

Since $\mathbf{x}^N \in P_{N,N}$,

$$\frac{\partial}{\partial X}\left(\frac{\partial \mathbf{x}^N}{\partial Y}\right)_{i+1/2,j+1/2} = \sum_{k,l=0}^{N} x^N_{k,l} l'_k(\overline{X}_{i+1/2}) l'_l(\overline{Y}_{j+1/2}) = \frac{\partial}{\partial Y}\left(\frac{\partial \mathbf{x}^N}{\partial X}\right)_{i+1/2,j+1/2} \tag{49}$$

so that,

$$\left.\frac{d\tilde{\mathbf{Q}}}{dt}\right|_{i+1/2,j+1/2} = \mathbf{0}, \quad \begin{cases} i = 0, 1, ..., N-1, \\ j = 0, 1, ..., N-1. \end{cases} \tag{50}$$

4. EXAMPLES

In this section, we use the staggered grid approximation to compute three steady flow problems. The first problem is subsonic flow from a point source, which has an exact, analytic solution. We use this solution to show that exponential convergence is obtained. The second problem is a subsonic flow over a circular bump in a channel. Although there is no exact solution for this problem, we show that the errors due to entropy generated along the curved wall decay exponentially fast. The final problem computes a transonic flow in an axisymmetric converging–diverging nozzle. That solution is compared to experimental data.

4.1. *Subsonic Point Source Flow*

As our first example, we consider the flow of a steady, irrotational flow exiting from a point. This flow can be solved exactly by a hodograph transformation [12]. The streamlines are radial, and the level curves of the Mach number, pressure, and density are circles centered on the source. We will compute this flow in two geometries. The first represents a flow in an expanding duct, where two streamlines are chosen as walls of the duct. The second geometry, a square with five circles cut out of its interior, is included to show that the method can be used to compute a flow in a complex, multiply connected geometry.

The first geometry represents steady flow in an expanding two-dimensional duct with straight walls. The lower wall was chosen to be the line $y = 0$ and the upper wall was the line $y = x \tan(\pi/6)$, for x between 1 and 1.5. The exact solution chosen sets the Mach number at the lower left corner of the domain to $M = 0.6$.

We examine solutions for three subdomain decompositions, each having four subdomains. Figure 9 shows the three decompositions. In the first (Grid I), the subdomain boundaries are straight lines so that the mappings defined by (34) become bilinear transformations. The second and third decompositions are included to study the effect of curved subdomains. Both perturb Grid I by a sine wave of amplitude 0.1 into "bulging" (Grid II) and "wedging" (Grid III) decompositions, named so in [38].

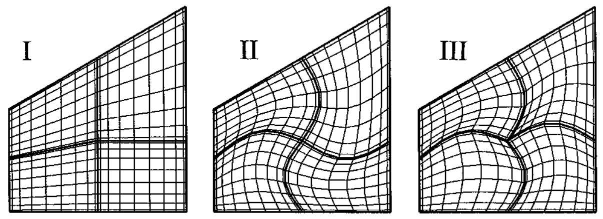

FIG. 9. Three subdomain decompositions for the diverging duct problem.

Wall conditions, applied as described in the previous section, are specified on the top and bottom boundaries. The left boundary is a subsonic inflow boundary. For that, we specify the exact solution as the incoming condition for the Riemann solver. The right boundary is a subsonic outflow boundary, and again the exact solution is used to specify the external flow. A perturbation of the exact solution was used as the initial condition.

Figure 10 shows the computed, steady Mach number contours for Grid I. In that figure, and in those following, contour lines are plotted using solution values interpolated from the Gauss points to the Lobatto points using (37). The grids in Fig. 9 show those Lobatto points. The solutions are represented interior to each "cell" bounded by the grid lines. The interpolation is done for display reasons, since a plot using the Gauss points would show gaps between the subdomains, a result of the fact that the solution

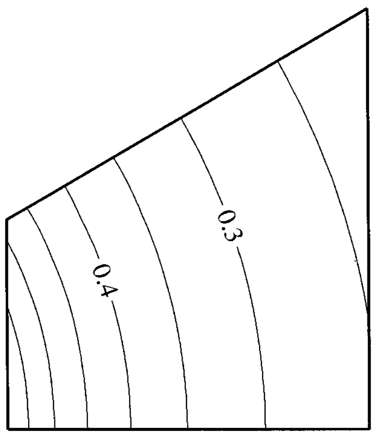

FIG. 10. Mach contours for flow in a diverging duct.

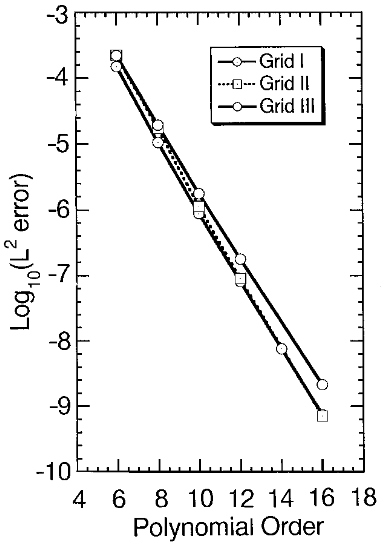

FIG. 11. Convergence of the error for the three grids of Fig. 9.

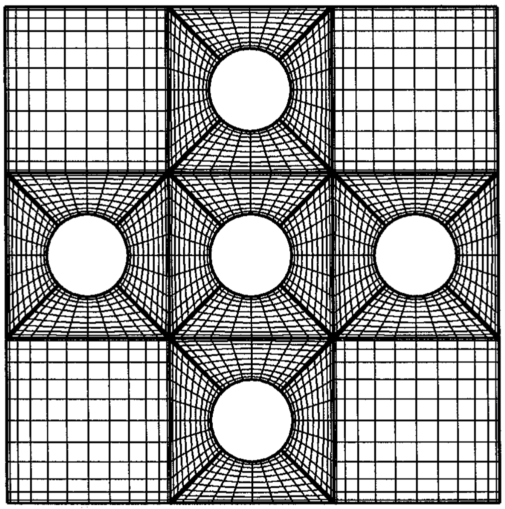

FIG. 12. Grid for point source problem.

is not defined at the interfaces. Plotting the interpolant does give some visual indication of the size of the solution jumps at the interfaces.

The staggered grid approximation is exponentially convergent for the point source flow in the duct. For Grids I–III, Fig. 11 shows the weighted L^2 error in the density as a function of the polynomial order used in each subdomain. The most marked observation is that for this problem, which is non-linear, the error and the convergence rate are not sensitive to the presence of curved interfaces. In fact, the convergence rate for the "bulged" decomposition is slightly higher than for the straight-sided subdomains. This contrasts strongly with the observations of [38], which considered the approximation of second-order linear problems. There, the presence of even slightly curved interfaces increased the error by orders of magnitude. The reason for the difference is that for the linear problem, the presence of curved boundaries introduces variable coefficients into the equation. These variable coefficients are represented as polynomials of the same degree as the approximation order. The product of the two gives a polynomial order twice as large, which must be projected back down onto the original polynomial space. Thus, under a curved mapping, twice as many collocation points were required to get the same error as under a linear mapping. For the non-linear problem, however, the projection of a high order polynomial onto the original polynomial space is required even if the mapping is linear. This is particularly evident in the calculation of the pressure, where the ratio of two polynomials is squared. The additional error introduced by a curved mapping, then, is masked by the errors introduced by approximating the nonlinear terms on the grid.

As our last example of the point source flow, we use the grid shown in Fig. 12. The geometry, a square with five circles cut out of its interior, was chosen to show that the method can be used to compute on a complex, multiply connected region. Twenty-four subdomains were used to cover the computational domain. Up to seven subdomains share a common corner point without difficulty, because such points are not included in the discrete approximation.

The boundary conditions were chosen so that the exact steady solution was a radial flow with the point source at the center of the middle circle of Fig. 12. The center cutout circle was specified as an inflow boundary, with the conditions chosen so that the Mach number of the incoming flow was $M = 0.6$. The boundary conditions along the remaining cutout circles were either inflow or outflow, depending on the direction of the normal velocity. The square outer boundary was an outflow boundary. For all inflow/outflow boundaries, the exact solution was used to

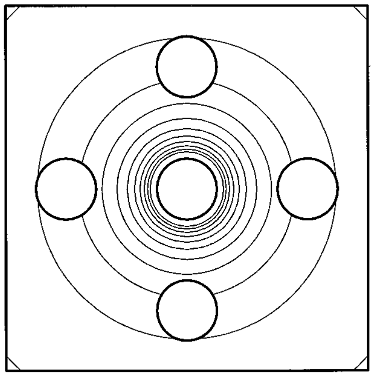

FIG. 13. Solution of the point source flow for the geometry shown in Fig. 12. The exact solution is plotted with dashed lines, the computed with solid lines.

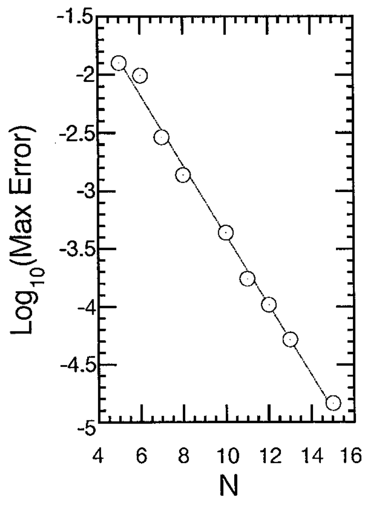

FIG. 14. Convergence of the density error for the solution shown in Fig. 13.

provide the external flow values required by the Riemann solver.

In Fig. 13, we plot the exact and computed Mach number contours for the solution of the point source flow. For the grid shown in Fig. 12, the contour lines of the exact solution, which are plotted with dashed lines, are coincident with the solid contour lines of the computed solution.

The approximation on the grid of Fig. 12 converges exponentially. Figure 14 shows the maximum error in the density as a function of the polynomial order in each subdomain. We see that doubling the number of points per subdomain causes the error to decay by approximately two orders of magnitude.

4.2. *Subsonic Flow over a Circular Bump in a Channel*

The second example is that of a Mach 0.3 subsonic flow over a circular bump in a channel. The geometry and grid with $N = 9$ is shown in Fig. 15. Wall boundaries were specified at the top and the bottom. At the left and right boundaries, the uniform flow free stream solution was specified as input to the Riemann solver. Initially, the free stream solution was specified everywhere, and then the boundary conditions were imposed. This problem does not have an exact analytic solution. However, since the incoming flow was chosen to be irrotational and isentropic, the entropy should be zero everywhere. The fact that this is not the case can be the result of the spatial approximation and to the normal wave model used at the interfaces and boundaries for calculating the flux [34].

Solution contours of the Mach number for the grid shown in Fig. 15 are presented in Fig. 16. The wall pressure along the bottom, plotted as the pressure coefficient $C_p = (p - 1)/\gamma$, is shown in Fig. 17. Finally, a convergence study of the entropy errors as a function of N is shown in Fig. 18. In that figure, we plot the maximum value of the quantity $\Sigma = p/\rho^\gamma - 1$, which should be zero everywhere. We see that the error due to entropy generation converges exponentially fast.

4.3. *Transonic Flow in a Converging–Diverging Nozzle*

As an example of a transonic problem, we compute the flow in an axisymmetric converging–diverging nozzle. We have chosen the nozzle used in the experimental investigation of Cuffel *et al.* [13], which was designed to show significant two-dimensional effects. The nozzle consists of a converging section with a half angle of 45° and a diverging section with a half angle of 15°. The experimental tests were done in air with a stagnation temperature of 540 R and stagnation pressure of 70 psia. The nozzle geometry and the grid that were used in our computations are shown in Fig. 19. Note that we have varied the number of points per subdomain in this problem.

To match the experimental conditions, we scaled Eqs. (31) and (32) by $\rho = \rho^*/\rho_{\text{tot}}$, $p = p^*/p_{\text{tot}}$, where the "*" represents the dimensional quantity. Under this scaling, the temperature and entropy are $T = T^*/T_{\text{tot}}$, $S_{\text{tot}} = 0$. The initial condition for the computation was the exact solution of the quasi-one-dimensional nozzle that has the same area as the two-dimensional nozzle. For the inflow condition at the left boundary, we specified that the tangen-

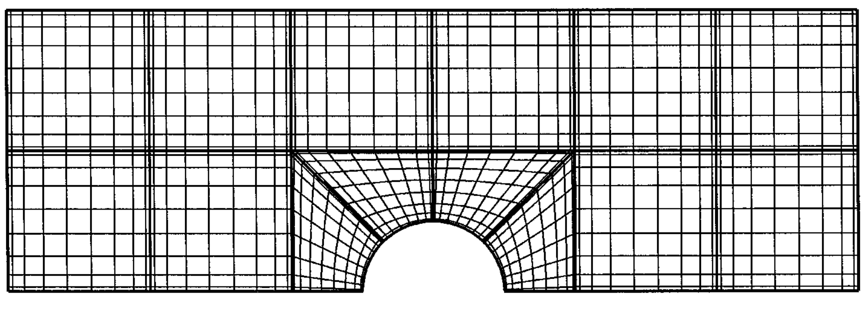

FIG. 15. Geometry and grid for N = 9 for the flow over a circular bump in a channel.

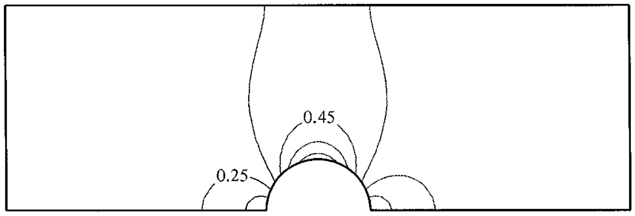

FIG. 16. Mach number contours for the flow over a circular bump in a channel.

tial velocity be zero, the entropy be zero, and the temperature be unity. At the right boundary, the outflow is supersonic, so that no boundary condition is necessary.

Since not all of the external flow values are known at the left boundary, particularly the inflow velocity, it is not convenient to use (26) to impose the boundary condition. Instead, we use the following characteristic-like method that allows us to specify only the parameters that are known. The fact that the inflow condition sets $v = 0$ means that the flow is essentially one-dimensional. Then we can write a left-going Riemann invariant for the flow. In terms of the Mach number, M, and the sound speed, a, that invariant must satisfy

$$a\left(\mathrm{M} - \frac{2}{\gamma - 1}\right) = R^{-}_{\text{computed}} \equiv a_{\text{comp.}}\left(\mathrm{M}_{\text{comp.}} - \frac{2}{\gamma - 1}\right), \tag{51}$$

where the computed quantities represent values interpolated to the boundary by the reconstruction procedure. The boundary conditions fix the total temperature and the entropy. With the scaling given above, this fixes the total sound speed at $a_{\text{tot}} = \sqrt{\gamma}$. Then the scaled sound speed and the Mach number are related by

$$a = \frac{\sqrt{\gamma}}{\sqrt{1 + ((\gamma - 1)/2)\,\mathrm{M}^2}}. \tag{52}$$

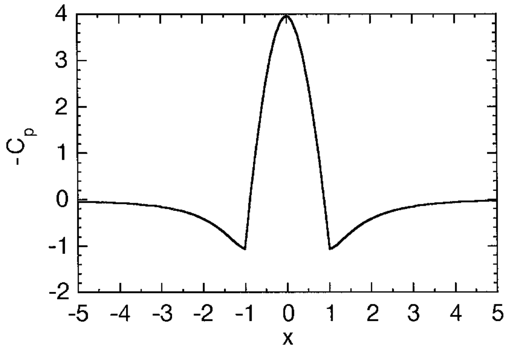

FIG. 17. Graph of the wall pressure along the bottom boundary of Fig. 16.

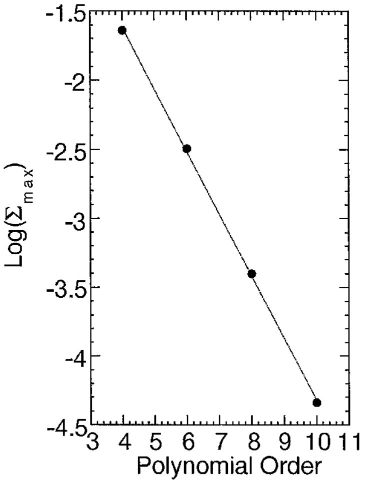

FIG. 18. Convergence of the entropy generated by the staggered grid approximation.

Combining (51) and (52), an equation for the Mach number at the boundary can be written

$$\frac{\sqrt{\gamma}}{\sqrt{1 + ((\gamma - 1)/2)\,\mathrm{M}^2}}\left(\mathrm{M} - \frac{2}{\gamma - 1}\right) = R^{-}_{\text{computed}}. \tag{53}$$

Equation (53) can be written as a quadratic equation in the Mach number and solved directly. Once the inflow Mach number is known, the sound speed can be computed using (52). From the Mach number, the sound speed, tangential velocity, and the entropy, all the remaining variables can be computed. From the full state on the boundary, the boundary flux can be evaluated.

Results computed for the nozzle are shown in Figs. 20–23. Contours for the pressure are shown in Fig. 20. A comparison of the Mach contours and measured Mach number in the neighborhood of the nozzle throat is shown

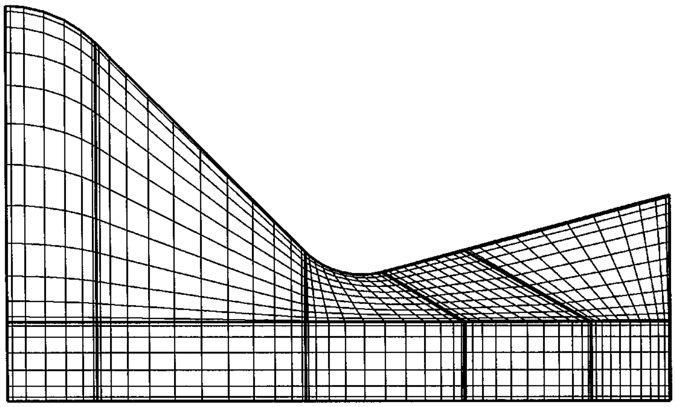

FIG. 19. Grid for the 45°–15° converging–diverging nozzle.

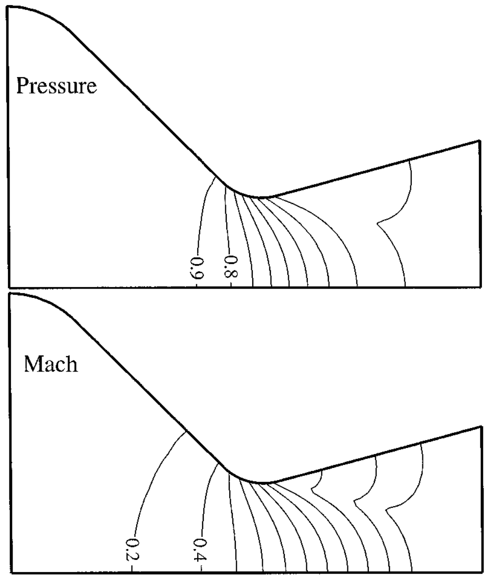

FIG. 20. Pressure and Mach contours for the nozzle flow.

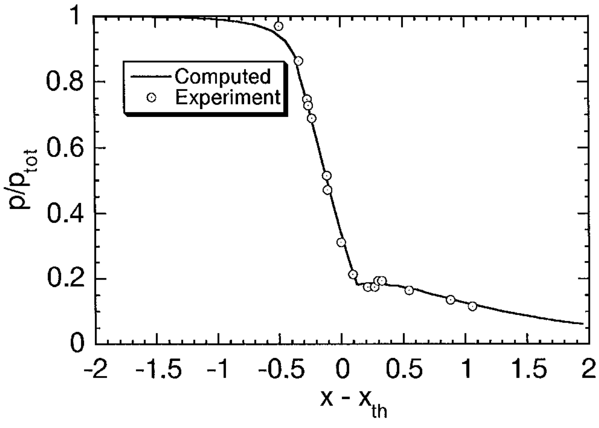

FIG. 22. Comparison of computed and measure wall pressure as a function of distance from the nozzle throat.

in Fig. 21. We see good agreement between the computed Mach contours and the measured values for Mach numbers up to about 1.6. We note that the discrepancies between the computed and measured Mach numbers are consistent with the discrepancies observed with the solutions of the inviscid flow solvers reported in [13]. Finally, in Figs. 22 and 23, we show a comparison between the computed and measured values of the pressure and the Mach number along the upper wall of the nozzle.

5. CONCLUDING REMARKS

We have presented a new, staggered-grid Chebyshev spectral multidomain method for the solution of inviscid compressible flow problems. The solutions are defined at the nodes of a Gauss quadrature rule, while the fluxes are evaluated at the nodes of a Gauss–Lobatto rule. We have applied the method to one- and two-dimensional problems, but it should extend directly to three dimensions.

The staggered grid multidomain method for compressible flow problems has many desirable features. These features include

- *Conservation.* Mass, momentum, and energy are conserved globally.
- *Free-stream preservation.* A uniform, steady flow stays uniform and steady, even for complex subdomain shapes.
- *Temporal accuracy.*
- *Geometric flexibility.* The domain need only be decomposable into quadrilaterals.
- *Programming simplicity.* Corners of subdomains are not included as part of the approximation. Special cases do not need to be coded. Subdomains have at most four neighbors in two space dimensions and six in three space dimensions. Boundary conditions require no special corner treatments.

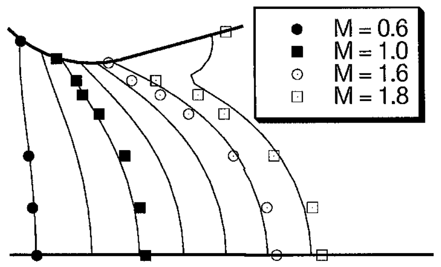

FIG. 21. Comparison of computed and measured Mach contours.

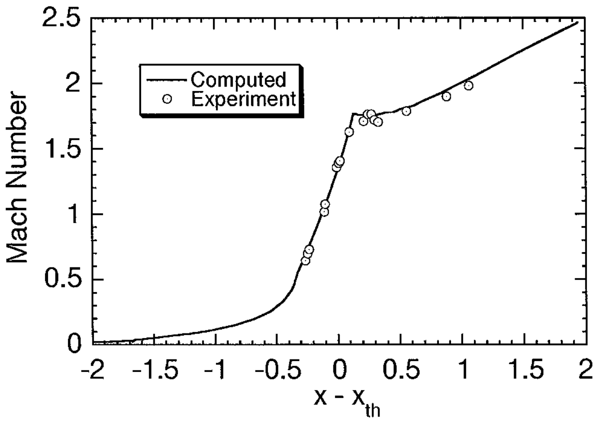

FIG. 23. Comparison of computed and measured Mach numbers as a function of distance from the nozzle throat.

While the new method is flexible, there remain a few limitations. We have assumed that the approximation is conforming. Thus, subdomains must meet at a point or along a full side. Along a side, the points at which the fluxes are computed must coincide with their neighbors. Also, the domain must be decomposable into quadrilaterals. The first restriction can be eliminated by considering non-conforming approximations [33].

We have not considered the approximation of shocks in this paper. Methods for shock capturing with spectral methods that have been proposed to date include direct filtering [23], removal of the discontinuity plus filtering [6], flux corrected transport methods [16, 17], and weak filtering with postprocessing [14, 18]. Shock fitting should also be possible [24, 30]. A thorough study of the best options is beyond the scope of this paper.

We have emphasized steady-state computations in this paper. However, the method is applicable to time dependent problems. Some one-dimensional examples are included here and in [29].

The new method is a factor of 2 more expensive than methods that compute both the solution and the fluxes on the Lobatto grid. This is due to the extra interpolation from the Gauss points to the Gauss–Lobatto points. FFT techniques cannot be used to compute either the interpolation or differentiation operations with the new method. Thus, the approximation order within the subdomains must be kept low enough to where matrix multiplication is more efficient than FFTs, a value that varies from machine to machine. However, while the Lobatto grid methods require point operations at subdomain corners, the new method requires vector operations only. We believe that the flexibility and programming ease of the new method compensates for the additional work.

The new method differs from the cell-averaged method proposed in [37, 16, 17]. The cell-averaged method requires a pointwise procedure to be performed at subdoman corners that is not required by this method. The reconstruction and differentiation operations performed are different. In one space dimension the work required by the staggered grid method is twice that of the cell-averaged method. However, the reconstruction procedure for the cell-averaged method in two space dimensions requires two matrix multiplication operations per line in each direction, as does the staggered-grid method, so the work is equivalent in two space dimensions. In three space dimensions, the staggered-grid method will require only two-thirds the work of the cell-averaged method, making it more efficient in that case.

ACKNOWLEDGMENTS

The first author thanks Professor P. L. Roe, Professor C.-W. Shu, Professor D. Gottlieb, and Dr. D. Sidilkover for many helpful discussions. This research was supported in part by the U.S. Department of Energy through Contract DE-FC05-85ER250000, by the National Science Foundation through Grant DMS-9404322, and by NAS1-19480 while the first author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

REFERENCES

1. Ø. Andreassen, and I. Lie, *J. Acoust. Soc. Am.* **95,** 171 (1994).
2. C. Bernardi, and Y. Maday, *Int. J. Numer. Methods Fluids* **8,** 537 (1988).
3. M. Bjørhus, *SIAM J. Sci. Comput.* **16,** 542 (1995).
4. W. Cai, *AIAA J.* **33,** 1248 (1995).
5. W. Cai, D. Gottlieb, and A. Harten, *Comput. Math. Appl.* **24** (1992).
6. W. Cai, and C. W. Shu, *J. Comput. Phys.* **104,** 427 (1993).
7. C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral Methods in Fluid Mechanics* (Springer-Verlag, New York, 1987).
8. M. H. Carpenter, and C. A. Kennedy, NASA T.M. 109112, 1994; *SIAM J. Sci. Comput.,* in press.
9. M-Q. Chen and C. Chiu, *Numer. Methods Partial Differential Equations* **9,** 643 (1993).
10. C. Chiu, *Appl. Numer. Math.* **11,** 475 (1993).
11. C. Chiu and D. A. Kopriva, *SIAM J. Numer. Anal.* **29,** 425 (1992).
12. R. Courant, and K. O. Friedrichs, *Supersonic Flow and Shock Waves* (Springer-Verlag, New York, 1976).
13. R. F. Cuffel, L. F. Back, and P. F. Massier, *AIAA J.* **7,** 1364 (1969).
14. W. S. Don, *J. Comput. Phys.* **110,** 103 (1994).
15. D. Funaro, and D. Gottlieb, *Math. Comput.* **51,** 599 (1988).
16. J. Giannakourous, and G. E. Karniadakis, *Int. J. Numer. Methods Fluids* **14,** 707 (1992).
17. J. Giannakouros, and G. E. Karniadakis, *J. Comput. Phys.* **115,** 65 (1994).
18. D. Gottlieb and C.-W. Shu, ICASE Report 94-61, 1994 (unpublished).
19. P. Hanley, *J. Comput. Phys.* **108,** 153 (1993).
20. J. S. Hesthaven, *SIAM J. Sci. Comput.,* in press.
21. C. Hirsch, *Numerical Computation of Internal and External Flows,* Vol Two (Wiley, Chichester, 1990).
22. F. W. Hu, M. Y. Hussaini, and J. Manthey, ICASE Report 94-102, 1994 (unpublished).
23. M. Y. Hussaini, D. A. Kopriva, M. D. Salas, and T. A. Zang, *AIAA J.* **23,** 234 (1985).
24. M. Y. Hussaini, D. A. Kopriva, M. D. Salas, and T. A. Zang, *AIAA J.* **23,** 64 (1985).
25. D. A. Kopriva, *Appl. Numer. Math.* **2,** 221 (1986).
26. D. A. Kopriva, *SIAM J. Sci. Statist. Comput.* **10,** 120 (1989).
27. D. A. Kopriva, *J. Comput. Phys.* **96,** 428 (1991).
28. D. A. Kopriva, *J. Comput. Phys.* **115,** 184 (1994).
29. D. A. Kopriva, and J. H. Kolias, "Solution of Acoustic Workshop Problems by a Spectral Multidomain Method," in *Proceedings, ICASE/LaRC Workshop on Benchmark Problems in Computational Aeroacoustics,* edited by J. C. Hardin, J. R. Risorcelli, and C. K. W. Tam (NASA CP 3300, May 1995), p. 117, (NASA, Langley Field, VA, 1995).
30. D. A. Kopriva, T. A. Zang, and M. Y. Hussaini, *AIAA J.* **9,** 1458 (1991).
31. I. Lie, *J. Sci. Comput.,* **9,** 39 (1994).
32. M.-S. Liou, and C. J. Steffen Jr., *J. Comput. Phys.* **107,** 23 (1993).

33. C. A. Mavriplis, Ph.D. dissertation, MIT, Feb. 1989 (unpublished).

34. H. Paillere, H. Deconinck, R. Struijs, P. Roe, L. Mesaros, and J-D. Muller, AIAA Paper 93-3301, 1993 (unpublished).

35. A. Quarteroni, *SIAM J. Sci. Statist. Comput.* **11,** 1029 (1990).

36. Roe, P. L., *J. Comp. Phys.* **43,** 357 (1981).

37. D. Sidilkover and G. E. Karniadakis, *J. Comput. Phys.* **107,** 10 (1993).

38. C. R. Schneidesch and M. O. Deville, *J. Comput. Phys.* **106,** 234 (1993).

39. C. K. W. Tam, ''Overview of Computed Results,'' in *Proceedings, ICASE/LaRC Workshop on Benchmark Problems in Computational Aeroacoustics,* edited by J. C. Hardin, J. R. Risorcelli, and C. K. W. Tam (NASA CP 3300, May 1995), p. 313, (NASA, Langley Field, VA, 1995).

40. B. Van Leer, ''Flux Vector Splitting for the Euler Equations,'' in *Proceedings, 8th International Conference on Numerical Methods in Fluid Dynamics,* (Springer-Verlag, Berlin, 1982).

41. J. H. Williamson, *J. Comput. Phys.* **35,** 48 (1980).