NASA Technical Memorandum 106928

p- 57

# An Approach to the Development of Numerical Algorithms for First Order Linear Hyperbolic Systems in Multiple Space Dimensions: The Constant Coefficient Case

John W. Goodrich
*Lewis Research Center*
*Cleveland, Ohio*

September 1995




National Aeronautics and
Space Administration

# An Approach to the Development of Numerical Algorithms For First Order Linear Hyperbolic Systems In Multiple Space Dimensions: The Constant Coefficient Case

**John W. Goodrich**
NASA Lewis Research Center
Cleveland, Ohio

## Abstract

Two methods for developing high order single step explicit algorithms on symmetric stencils with data on only one time level are presented. Examples are given for the convection and linearized Euler equations with up to the eighth order accuracy in both space and time in one space dimension, and up to the sixth in two space dimensions. The method of characteristics is generalized to nondiagonalizable hyperbolic systems by using exact local polynomial solutions of the system, and the resulting exact propagator methods automatically incorporate the correct multidimensional wave propagation dynamics. Multivariate Taylor or Cauchy-Kowaleskaya expansions are also used to develop algorithms. Both of these methods can be applied to obtain algorithms of arbitrarily high order for hyperbolic systems in multiple space dimensions. Cross derivatives are included in the local approximations used to develop the algorithms in this paper in order to obtain high order accuracy, and improved isoptropy and stability. Efficiency in meeting global error bounds is an important criterion for evaluating algorithms, and the higher order algorithms are shown to be up to several orders of magnitude more efficient even though they are more complex. Stable high order boundary conditions for the linearized Euler equations are developed in one space dimension, and demonstrated in two space dimensions.

# I: Introduction

Hyperbolic systems include familiar examples such as the linear systems for convective transport, acoustics, and electromagnetics, and the nonlinear Euler equations of fluid mechanics ([18],[39]). These examples have very practical applications in areas such as aircraft noise ([24],[33],[35]). Numerical methods for hyperbolic systems have a broad history ([8],[20],[36]). There is an increasing interest in the use of computational fluid dynamics techniques for these systems ([26],[27],[32],[34],[42]), but with unusually severe accuracy requirements ([16],[25]). Algorithms are needed for hyperbolic systems that are able to propagate a wide range of wavelengths with high accuracy over long distances.

Compact difference methods ([5],[29],[37]) are generally viewed as being highly accurate, and can be tailored to produce both high order accuracy and high resolution [22], which is defined as the ability to propagate relatively high frequency waves with the correct velocity. Compact difference methods generally use separate treatments for space and time, either with different [40] or the same [14] order of accuracy. The dispersion relation preserving method [38] is similar to compact difference methods in its spatial treatment, but it addresses the relationship between space and time treatments by using some of the degrees of freedom of temporal data that it requires in order to reduce the dispersion or phase speed errors in each time step. The dispersion relation preserving scheme provides fourth order accuracy in space and second or third order in time on a seven point stencil with data from four time levels. High resolution is frequently stated in terms of the number of grid points required to accurately propagate a normal mode, or algorithm performance relative to a grid scale, which does not address the question of efficiency with respect to meeting a stated global error bound. Compact difference methods do not provide a general efficient high order time stepping method.

A variety of high order finite difference methods use multiple time steps. The dissipative two-four method [13] is a two step generalization of the Lax-Wendroff method, with second order accuracy in time and fourth order accuracy in space. The two-four method has versions ([1],[13]) that are similar to the MacCormack [28] operator splitting method. The third order difference method of Burstein, Mirin and Rusanov ([3],[31]) uses intermediate time steps and a correction to obtain third order accuracy in space and time on a five point stencil. The modified equation approach [6] uses data on three time levels to obtain fourth order accuracy in space and time. A five-six finite difference scheme has been developed [43] on a seven point stencil in one dimension with a six stage time marching method, providing fifth order accuracy in time and sixth order in space. This method has an optimized variation which relaxes its order of accuracy in order to increase its resolution. The particular approach of directional splitting in multiple space dimensions violates the physics of nondiagonalizable hyperbolic sytems in a fundamental way, and in general, the use of multiple time steps raises issues such as intermediate time level boundary conditions, artifical dissipation, significant time step constraints, starting values, and efficiency.

Many classical approaches to developing numerical methods can be viewed as simultaneously treating the spatial representation of data and the temporal evolution of the system. Lax Wendroff [21] methods can be viewed as using a second order Taylor series expansion in time,

with time and space derivatives related by the partial differential equation. Semi Lagrangian methods use the method of characteristics [41] and incorporate the geometric behavior of the solution in space and time. Godunov [10] methods use the solution of a Riemann problem to approximate the physics of shocks within a control volume in space and time. Finite volume methods use integral forms of conservation laws in space and time, and are close in spirit to the general method of control volume analysis. The finite analytical method [4] also combines the treatments of space and time by using a local Fourier decomposition in space and a separation of variables analytical treatment for time evolution. These approaches generally are limited in either accuracy or applicability.

This paper presents and compares two different approaches to algorithm development for linear hyperbolic systems in multiple space dimensions, the use of local exact polynomial solutions in space and time, or exact propagators, and the use of multivariate Taylor series, or Cauchy-Kowaleskaya expansions. Local exact polynomial solutions can be obtained for diagonalizable hyperbolic systems by the method of characteristics, and for nondiagonalizable systems in multiple space dimensions by requiring that a general polynomial expansion in space and time exactly solves the partial differential equations, with all of the unknown expansion coefficients that involve time expressed in terms of spatial coefficients. The Cauchy-Kowaleskaya procedure ([9],[17]) is an equivalent method for obtaining time expansion coefficients. Local exact solutions which incorporate the multidimensional wave dynamics of the hyperbolic system can be viewed as a correct way to extend the method of characteristics to nondiagonalizable systems. If the Taylor series methods are considered only at the spatial center of the local expansions, then they can be viewed as a Taylor series expansion in time alone, similar to the second order Lax Wendroff method [21]. The exact propagator and multivariate Taylor series approaches produce single step explicit algorithms that have the same order of accuracy in both space and time, while using data from only one time level, and that can be extended to arbitrarily high order and multiple space dimensions for nondiagonalizable systems. These two approaches to algorithm development produce the same methods in one space dimension, and different methods in multiple dimensions.

The exact propagator and multivariate Taylor series approaches to algorithm development can both be viewed as direct applications of the general use of multivariate approximating polynomials in space and time ([2], [8]). As a related example of this technique, in a discussion from the Taylor series perspective about extending Leith's method from one to two space dimensions, Roache [30] makes the point that a scheme in two space dimensions would have to include cross derivative terms in order to obtain high order accuracy. The use of cross derivative spatial terms has been shown to also improve isotropy and stability, for example in the development of finite volume methods for multidimensional advection [23], and in the development of a nearly exact second order algorithm on three time levels for the wave equation [7]. Including sufficient cross derivative terms in a multidimensional algorithm is required for high order accuracy, and improved isotropy and stability, but it does not necessarily produce a locally exact solution with the correct multidimensional wave dynamics. A fundamental viewpoint in this paper is to approximate the solution of a system of partial differential equations as a whole, instead of approximating separate derivative terms in particular equations. This viewpoint is expressed in the two general approaches to algorithm development that are used throughout this paper, which in turn are realized in a sequence of particular algorithms

for linear hyperbolic systems in one and two space dimensions.

Development of high order algorithms for the two dimensional linearized Euler equations is a prominant goal of this paper, but various issues are addressed in the simplest possible context by developing algorithm for other systems. Examples for the convection and linearized Euler equations will be shown in one space dimension up to eighth order in both space and time, and in two space dimensions up to sixth order in both space and time. In the second section of this paper, a fourth order algorithm is presented for the scalar first order linear wave equation in one space dimension, and is used to relate several viewpoints of algorithm development. In the third section, algorithms for the one dimensional linearized Euler equations are presented, with second, fourth, sixth and eighth order accuracy in space and time. The accuracy and relative efficiency of these algorithms is examined for short and long time calculations, with high and low error bounds [19]. The problem of creating stable high order boundary algorithms for this system is also treated. In the fourth section, algorithms are developed on symmetric stencils for the convection equation in two space dimensions with second, fourth and sixth order accuracy in space and time. Relative efficiency, stencil choice, and the differences between exact propagator and Taylor expansion algorithms are examined. In the fifth section, exact propagator and Taylor series algorithms with second, fourth, and sixth order accurcy in both space and time are developed for the linearized Euler equations in two space dimensions. A method for developing local exact polynomial solutions to the two dimensional linearized Euler equations is described and used for developing exact propagator algorithms. This method for developing local exact polynomial solutions is a departure from the method of characteristics which is used throughout the rest of the paper. The relative effect of algorithm type and order is compared, and a high order implementation of a new unobtrusive outflow boundary condition by Hagstrom [15] is demonstrated. The sixth section is a brief concluding summary. Several longer formulas have been included in appendices.

## II: A Fourth Order Algorithm For The 1D Convection Equation

Consider the scalar first order linear wave equation in one space dimension for $u(x,t)$,

$$\frac{\partial u}{\partial t} + M\frac{\partial u}{\partial x} = 0, \tag{1}$$

where $M$ is the constant mean convection speed. The initial value problem for this equation with $u(x,0) = ui(x)$ for $x \in \Re$ can be immediately solved by the method of characteristics, and $u(x,t) = ui(x - Mt)$ for $x \in \Re$ and $0 \leq t$. With this well understood example it is possible to quickly develop high order algorithms, and to readily present an algorithm from various viewpoints.

A fourth order numerical algorithm can be developed for this problem by the method of charateristics with a local quartic interpolation to $u$ at time $t_n$ on a five point central stencil. A uniform grid is used, with $(x_i, t_n) = (ih, nk)$ for integer $i$ and $n$, where $h = \triangle x$ and $k = \triangle t$ are the uniform mesh spacings in $x$ and $t$. The quartic spatial approximation to $u$ around $x_i$ at $t_n$ can be written in local coordinates as

$$u(x_i + x, t_n) \approx ua(x) = u_0 + u_1 x + u_2 x^2 + u_3 x^3 + u_4 x^4,$$

where the coefficients are not indexed with respect to the mesh point. The method of undetermined coefficients and the known data on the five point stencil immediately yield the unknown coefficient solutions

$$\begin{aligned}
u_0 &= u_i^n, \\
u_1 &= \frac{1}{12h}(u_{i-2}^n - 8u_{i-1}^n + 8u_{i+1}^n - u_{i+2}^n), \\
u_2 &= \frac{1}{24h^2}(-u_{i-2}^n + 16u_{i-1}^n - 30u_i^n + 16u_{i+1}^n - u_{i+2}^n), \\
u_3 &= \frac{1}{12h^3}(-u_{i-2}^n + 2u_{i-1}^n - 2u_{i+1}^n + u_{i+2}^n), \\
u_4 &= \frac{1}{24h^4}(u_{i-2}^n - 4u_{i-1}^n + 6u_i^n - 4u_{i+1}^n + u_{i+2}^n).
\end{aligned} \tag{2}$$

Notice that $u_\alpha = \frac{1}{\alpha!}\frac{\partial^\alpha ua}{\partial x^\alpha} \approx \frac{1}{\alpha!}\frac{\partial^\alpha u}{\partial x^\alpha}$, so that the spatial approximation to $u$ about $x_i$ at $t_n$ is essentially a truncated Taylor series in $x$. The method of characteristics and the local solution approximation at time $t_n$ now give

$$u(x_i, t_{n+1}) = u(x_i - Mk, t_n) \approx ua(-Mk) = \sum_{\alpha=0}^{4} u_\alpha(-Mk)^\alpha = u_i^{n+1},$$

where $u_i^{n+1}$ is implicitly defined. This algorithm correctly incorporates the wave dynamics of equation (1) by using the method of charateristics. Notice that the time propagation is exact,

so that the error at each time step is due to the local spatial interpolation $ua$. This algorithm can be rewritten in the familiar form of a conventional single step explicit finite difference method,

$$u_i^{n+1} = \sum_{r=-2}^{2} a_r u_{i+r}^n,$$

where if the CFL number is $\lambda = \frac{Mk}{h}$, then

$$\begin{aligned}
a_{+2} &= \frac{\lambda}{24}(+2 - \lambda - 2\lambda^2 + \lambda^3), \\
a_{+1} &= \frac{\lambda}{6}(-4 + 4\lambda + \lambda^2 - \lambda^3), \\
a_0 &= \frac{1}{4}(4 - 5\lambda^2 + \lambda^4), \\
a_{-1} &= \frac{\lambda}{6}(+4 + 4\lambda - \lambda^2 - \lambda^3), \\
a_{-2} &= \frac{\lambda}{24}(-2 - \lambda + 2\lambda^2 + \lambda^3).
\end{aligned}$$

In this form the algorithm requires five multiplies and four adds at each grid point for each time step. Standard analysis shows that this method is fourth order accurate in both space and time, with truncation error

$$\begin{aligned}
\frac{1}{k}(u(x_i, t_{n+1}) - u_i^{n+1}) =& h^4(\frac{-M}{120})(1-\lambda^2)(4-\lambda^2)\frac{\partial^5 u}{\partial x^5} \\
&+ h^5(\frac{+M}{720})\lambda(1-\lambda^2)(4-\lambda^2)\frac{\partial^6 u}{\partial x^6} \\
&+ O[h^6].
\end{aligned}$$

The leading order error term is dispersive, which is natural for algorithms with symmetric central stencils. Numerical experiments suggest that the method is stable for $|\lambda| \leq 1$.

If the local expansion coefficients are interpreted in terms of space derivatives, and if $u$ is an exact solution to equation (1), then

$$u_i^{n+1} = \sum_{\alpha=0}^{4} \frac{(-Mk)^\alpha}{\alpha!}\frac{\partial^\alpha ua}{\partial x^\alpha}(0) \approx \sum_{\alpha=0}^{4} \frac{(-Mk)^\alpha}{\alpha!}\frac{\partial^\alpha u}{\partial x^\alpha}(x_i, t_n) = \sum_{\alpha=0}^{4} \frac{k^\alpha}{\alpha!}\frac{\partial^\alpha u}{\partial t^\alpha}(x_i, t_n).$$

This form of the algorithm is a fourth order expansion in the time step size $k$, and is similar to the standard second order Lax-Wendroff method [21] expressed as a truncated Taylor series in time. If the general form of the solution to equation (1) is used, then $u$ can be approximated in both local coordinates $x$ and $t$ near $(x_i, t_n)$, with $u(x_i + x, t_n + t) \approx ua(x - Mt)$. If the interpretation of the expansion coefficients are used, then this local approximation to $u$ in $x$

and $t$ can be written as

$$\begin{aligned} u(x_i + x, t_n + t) \approx & ua(x - Mt) \\ & = \sum_{\alpha=0}^{4} u_\alpha (x - Mt)^\alpha \\ & \approx \sum_{\alpha=0}^{4} \frac{1}{\alpha!}(x - Mt)^\alpha \frac{\partial^\alpha u}{\partial x^\alpha}(x_i, t_n) \\ & = \sum_{\alpha=0}^{4} \sum_{\beta=0}^{\alpha} \frac{1}{(\alpha-\beta)!\beta!} x^{\alpha-\beta} t^\beta \frac{\partial^\alpha u}{\partial x^{\alpha-\beta} \partial t^\beta}(x_i, t_n), \end{aligned}$$

where ${u_i}^{n+1} = ua(-Mk)$. This form of the solution is actually just a truncated Taylor series in both space and time simultaneously. This truncated Taylor polynomial is also an exact polynomial solution to equation (1). This particular exact local solution approximates the solution $u$ for points $(x, t)$ with a domain of dependence contained in the interval $[x_{i-2}, x_{i+2}]$.

There are four related interpretations of this fourth order algorithm: a locally exact solution derived from the geometric method of characteristics with a local truncated Taylor series approximation in space for its initial data; a conventional finite difference method; a truncated Taylor series expansion in time; and a locally exact polynomial solution derived analytically from a truncated Cauchy-Kowaleskaya or bivariate Taylor series expansion in space and time. The first interpretation ensures that the numerical method properly represents the wave dynamics of the partial differential equation, since the characteristic behavior of the equation is incorporated into the numerical solution. The second and third interpretations ground this algorithm in the mature tradition of finite difference methods. The fourth interpretation provides an avenue for generalization and extension, since a locally valid exact analytic solution is possible for hyperbolic systems in higher space dimensions. Notice that no consideration has been directly given to the problem of finite difference approximation to any derivative, but rather to the interpolation of the known data on a stencil, and to an exact local solution to the partial differential equation. A key shift in perspective is away from the details of approximating separate terms in an equation, and toward the approximation of a solution to the equation.

## III: Numerical Algorithms For 1D Linearized Euler Equations

Consider now the linearized Euler equations in one space dimension, in particular the isentropic case in the form of the nondimensionalized system

$$\begin{aligned} \frac{\partial u}{\partial t} + M\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} &= 0, \\ \frac{\partial p}{\partial t} + M\frac{\partial p}{\partial x} + \frac{\partial u}{\partial x} &= 0, \end{aligned} \tag{3}$$

where $M$ is the constant mean convection speed in terms of Mach number, and where $p$ and $u$ are the pressure and velocity of the disturbance. A general discussion of the linearized Euler equations can be found in Kreiss and Lorenz [18]. System (3) can be diagonalized and written with Riemann variables in the equivalent decoupled form

$$\begin{aligned} \frac{\partial \omega_1}{\partial t} + (M-1)\frac{\partial \omega_1}{\partial x} &= 0, \\ \frac{\partial \omega_2}{\partial t} + (M+1)\frac{\partial \omega_2}{\partial x} &= 0, \end{aligned} \tag{4}$$

where $\omega_1 = \frac{1}{2}(u-p)$ and $\omega_2 = \frac{1}{2}(u+p)$. The initial value problem for $u$ and $p$ with $u(x,0) = ui(x)$ and $p(x,0) = pi(x)$ for $x \in \Re$ is equivalent to a corresponding problem for $\omega_1$ and $\omega_2$ with $\omega_1(x,0) = \frac{1}{2}(ui(x) - pi(x))$ and $\omega_2(x,0) = \frac{1}{2}(ui(x) + pi(x))$. Each equation in the Riemann variable system (4) can be solved separately by the method of characteristics, and these solutions can be used to obtain the general solution for system (3),

$$\begin{aligned} u(x,t) =& \frac{1}{2}(ui(x-(M+1)t) + pi(x-(M+1)t)) \\ &+ \frac{1}{2}(ui(x-(M-1)t) - pi(x-(M-1)t)), \\ p(x,t) =& \frac{1}{2}(ui(x-(M+1)t) + pi(x-(M+1)t)) \\ &- \frac{1}{2}(ui(x-(M-1)t) - pi(x-(M-1)t)). \end{aligned} \tag{5}$$

Note that this system has two characteristics with separate unidirectional solutions that are constant along their characteristics, and that are travelling with the characteristic velocities $M-1$ and $M+1$, respectively.

### III-A: A General Algorithm Development

A general development can simultaneously provide algorithms of the second, fourth, sixth and eighth orders. A local polynomial interpolation to $u$ and $p$ in $x$ about $(x_i, t_n)$ can be

written in the general form

$$u(x_i + x, t_n) \approx ua(x) = \sum_{\alpha=0}^{Or} u_\alpha x^\alpha,$$
$$p(x_i + x, t_n) \approx pa(x) = \sum_{\alpha=0}^{Or} p_\alpha x^\alpha, \tag{6}$$

where indexing with respect to the mesh point is suppressed, and where $Or = 2, 4, 6$, or 8 is the order of the interpolation, on central stencils of from three to nine points. The method of undetermined coefficients using the data on the stencils immediately yields the unknown coefficients in terms of the known data. The coefficients have standard central finite difference forms, such as for the quartic case in (2). The expansion coefficients can be interpreted in terms of spatial derivatives, with

$$u_\alpha = \frac{1}{\alpha!}\frac{\partial^\alpha ua}{\partial x^\alpha}(0) \approx \frac{1}{\alpha!}\frac{\partial^\alpha u}{\partial x^\alpha}(x_i, t_n),$$
$$p_\alpha = \frac{1}{\alpha!}\frac{\partial^\alpha pa}{\partial x^\alpha}(0) \approx \frac{1}{\alpha!}\frac{\partial^\alpha p}{\partial x^\alpha}(x_i, t_n).$$

A new solution value at $(x_i, t_{n+1})$ is obtained from the general solution form (5) with the local spatial interpolation (6) as initial data,

$$\begin{aligned} u(x_i, t_{n+1}) \approx u_i^{n+1} =& \frac{1}{2}(ua(-(M+1)k) + pa(-(M+1)k)) \\ &+ \frac{1}{2}(ua(-(M-1)k) - pa(-(M-1)k)), \\ p(x_i, t_{n+1}) \approx p_i^{n+1} =& \frac{1}{2}(ua(-(M+1)k) + pa(-(M+1)k)) \\ &- \frac{1}{2}(ua(-(M-1)k) - pa(-(M-1)k)). \end{aligned} \tag{7}$$

Algorithm form (7) is obtained by applying an exact propagator to a local polynomial interpolant, so that form (7) correctly incorporates the multiple characteristic wave dynamics of equation (3). Algorithm (7) can also be arranged as a truncated time series expansion in $k$. On the nine point central stencil the algorithm for $u$ can be rewritten as

$$\begin{aligned} u_i^{n+1} = & u_0 - k(p_1 + Mu_1) + k^2(2Mp_2 + (M^2+1)u_2) \\ & - k^3((1+3M^2)p_3 + M(3+M^2)u_3) \\ & + k^4(4M(1+M^2)p_4 + (1+6M^2+M^4)u_4) \\ & + k^5(-(1+10M^2+5M^4)p_5 - M(5+10M^2+M^4)u_5) \\ & + k^6(M(6+20M^2+6M^4)p_6 + (1+15M^2+15M^4+M^6)u_6) \\ & + k^7(-(1+21M^2+35M^4+7M^6)p_7 - M(7+35M^2+21M^4+M^6)u_7) \\ & + k^8(M(8+56M^2+56M^4+8M^6)p_8 + (1+28M^2+70M^4+28M^6+M^8)u_8), \end{aligned} \tag{8a}$$

and for $p$ as

$$
\begin{aligned}
p_i^{n+1} &= p_0 - k(u_1 + Mp_1) + k^2(2Mu_2 + (M^2+1)p_2) \\
&\quad - k^3((1+3M^2)u_3 + M(3+M^2)p_3) \\
&\quad + k^4(4M(1+M^2)u_4 + (1+6M^2+M^4)p_4) \\
&\quad + k^5(-(1+10M^2+5M^4)u_5 - M(5+10M^2+M^4)p_5) \\
&\quad + k^6(M(6+20M^2+6M^4)u_6 + (1+15M^2+15M^4+M^6)p_6) \\
&\quad + k^7(-(1+21M^2+35M^4+7M^6)u_7 - M(7+35M^2+21M^4+M^6)p_7) \\
&\quad + k^8(M(8+56M^2+56M^4+8M^6)u_8 + (1+28M^2+70M^4+28M^6+M^8)p_8).
\end{aligned}
\tag{8b}
$$

Algorithm form (8) can be used to obtain the time series expansion forms of the second, fourth and sixth order algorithms, by simply taking the terms up to the appropriate order in $k$. Note that the symmetry of (3) in $u$ and $p$ is reflected in (8). The familiar grid ratio $\lambda = \frac{k}{h}$ is implicit in algorithm form (8), since each expansion coefficient $u_\alpha$ or $p_\alpha$ is a finite difference form with $h^\alpha$ in its denominator. A general finite difference form can be derived from (8), with

$$
\begin{aligned}
u_i^{n+1} &= \sum_{r=-R}^{R} (a_r u_{i+r}^n + b_r p_{i+r}^n), \\
p_i^{n+1} &= \sum_{r=-R}^{R} (a_r p_{i+r}^n + b_r u_{i+r}^n),
\end{aligned}
\tag{9}
$$

where the coefficients are polynomials in $\lambda$ and $M$, and where $R = 1, 2, 3$, or 4 depending upon which stencil is being used. The coefficients $a_r$ and $b_r$ are used symmetrically with respect to $u$ and $p$ in algorithm form (9), and reflect the symmetry of the equation (3) in $u$ and $p$.

Algorithm (7-9) on a three point central stencil has $Or = 2$ in (6) and $R = 1$ in (9), and represents just the first line of equations (8a) and (8b). The truncation error for this algorithm is

$$
\begin{aligned}
\frac{1}{k}(u(x_i, t_{n+1}) - u_i^{n+1}) &= \frac{\partial u}{\partial t} + M\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} \\
&\quad + h^2(\frac{M}{6})(1-(3+M^2)\lambda^2)\frac{\partial^3 u}{\partial x^3} + h^2(\frac{1}{6})(1-(1+3M^2)\lambda^2)\frac{\partial^3 p}{\partial x^3} \\
&\quad + O[h^3], \\
\frac{1}{k}(p(x_i, t_{n+1}) - p_i^{n+1}) &= \frac{\partial p}{\partial t} + M\frac{\partial p}{\partial x} + \frac{\partial u}{\partial x} \\
&\quad + h^2(\frac{M}{6})(1-(3+M^2)\lambda^2)\frac{\partial^3 p}{\partial x^3} + h^2(\frac{1}{6})(1-(1+3M^2)\lambda^2)\frac{\partial^3 u}{\partial x^3} \\
&\quad + O[h^3].
\end{aligned}
$$

Clearly, this method is consistent with equations (3), it is second order accurate in space and time, and it is dispersive. Algorithm (7-9) on a five point central stencil has $Or = 4$ in (6)

and $R = 2$ in (9), and represents just the first three lines of equations (8a) and (8b). This algorithm has the truncation error

$$\begin{aligned}
\frac{1}{k}(u(x_i,t_{n+1}) - u_i^{n+1}) &= \frac{\partial u}{\partial t} + M\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} \\
&+ h^4(\frac{-M}{30} + M(3+M^2)\frac{\lambda^2}{24} - M(5+10M^2+M^4)\frac{\lambda^4}{120})\frac{\partial^5 u}{\partial x^5} \\
&+ h^4(\frac{-1}{30} + (1+3M^2)\frac{\lambda^2}{24} - (1+10M^2+5M^4)\frac{\lambda^4}{120})\frac{\partial^5 p}{\partial x^5} \\
&+ O[h^5], \\
\frac{1}{k}(p(x_i,t_{n+1}) - p_i^{n+1}) &= \frac{\partial p}{\partial t} + M\frac{\partial p}{\partial x} + \frac{\partial u}{\partial x} \\
&+ h^4(\frac{-M}{30} + M(3+M^2)\frac{\lambda^2}{24} - M(5+10M^2+M^4)\frac{\lambda^4}{120})\frac{\partial^5 p}{\partial x^5} \\
&+ h^4(\frac{-1}{30} + (1+3M^2)\frac{\lambda^2}{24} - (1+10M^2+5M^4)\frac{\lambda^4}{120})\frac{\partial^5 u}{\partial x^5} \\
&+ O[h^5].
\end{aligned}$$

This fourth order accurate method is clearly consistent with (3) and dispersive. The truncation errors for the seven and nine point central stencil methods also verify that these methods are sixth and eighth order accurate, respectively.

The algorithms in this section can be viewed as characteristic or semi Lagrangian methods with a local polynomial spatial interpolation, as truncated Taylor series in time, or as conventional explicit finite difference methods. The exact solution form and the interpretation of the local polynomial expansion coefficients can be used to provide a local polynomial solution to the differential equation that can be interpreted as a multivariate Taylor series or Cauchy-Kowaleskaya expansion in space and time. The algorithms in this section are derived from an exact solution to the partial differential equation, and find a new solution value by using the correct wave dynamics of a local spatial interpolant.

### III-B: Numerical Comparisons

Numerical tests of these four algorithms for equation (3) are obtained from an initial value problem with data $u(x,0) = 0$ and $p(x,0) = \mathrm{Sin}(\pi x)$, for $-1 \le x \le 1$, and with periodic boundaries at $x = \pm 1$. The exact solution for this problem can be obtained from (5). Table A gives data for this problem with the four algorithms at $M = 0$ and $\lambda = \frac{k}{h} = 0.8$, on a series of mesh sizes, and calculating out to the fixed time $t = 10$, which corresponds to five periods of wave propagation for the given initial data. In Table A, $\frac{2}{h}$ is the number of grid points in $[-1,1]$, or per wavelength for the initial data, $n_{10}$ is the number of time steps required to reach $t = 10$ with $\lambda = 0.8$ at the given grid resolution, and the columns are identified by the order $Or$ of the algorithm which produced the data in the column. The error data in Table A is the maximum absolute accumulated error in $u$ or $p$ for $x \in [-1,1]$ at $t = 10$, so it reflects

the error of the algorithms in both space and time. The mesh sizes that are used are for mesh refinements by successive halving of the grid size.

Table A: Data From The Four Algorithms For The 1D LEE
$u(x,0) = 0$ and $p(x,0) = \mathrm{Sin}(\pi x)$, $\lambda = 0.8, M = 0$
Maximum Error in $u$ or $p$ at $t = 10$

| $\frac{2}{h}$ | $n_{10}$ | $Or = 2$ | $Or = 4$ | $Or = 6$ | $Or = 8$ |
|---|---|---|---|---|---|
| 4 | 25 | 1.035D+0 | 7.885D-01 | 4.253D-01 | 2.078D-01 |
| 8 | 50 | 6.760D-01 | 9.141D-02 | 1.110D-02 | 1.387D-03 |
| 16 | 100 | 2.586D-01 | 7.122D-03 | 2.158D-04 | 7.000D-06 |
| 32 | 200 | 7.133D-02 | 4.644D-04 | 3.551D-06 | 2.910D-08 |
| 64 | 400 | 1.811D-02 | 2.932D-05 | 5.620D-08 | 1.157D-10 |
| 128 | 800 | 4.539D-03 | 1.837D-06 | 8.808D-10 | 3.438D-13 |
| 256 | 1600 | 1.135D-03 | 1.149D-07 | 1.283D-11 | 9.484D-13 |
| 512 | 3200 | 2.839D-04 | 7.182D-09 | 6.303D-13 | |
| 1024 | 6400 | 7.097D-05 | 4.527D-10 | 3.776D-12 | |

The orders of accuracy of the algorithms can be verified by calculating the factor by which the error decreases with the grid resolution. The data from the method with $Or = 2$ at grid sizes $\frac{2}{h} = 512$ and $\frac{2}{h} = 1024$ gives,

$$\frac{1}{Log[2]} Log[\frac{2.839 \times 10^{-4}}{7.097 \times 10^{-5}}] = 2.00,$$

so that the error decreases by a factor of $2^{2.00}$ when the mesh size is halved, and the method as implemented actually is second order in both space and time. For the method with $Or = 6$, the error decrease from $\frac{2}{h} = 128$ to $\frac{2}{h} = 256$ is by a factor of $2^{6.10}$, but the decrease from $\frac{2}{h} = 256$ to $\frac{2}{h} = 512$ is only by a factor of $2^{4.35}$, while the error actually increases from $\frac{2}{h} = 512$ to $\frac{2}{h} = 1024$. The differences between these estimates of the order of accuracy for this method is due to the contamination of the finer grid results by roundoff error at the resolution limits for these calculations in double precision. The data from $\frac{2}{h} = 128$ and $\frac{2}{h} = 256$ can be accepted as showing sixth order accuracy for this method. A similar effect can be seen in the data for the method with $Or = 8$, but the order of the method can be obtained from the data at $\frac{2}{h} = 64$ and $\frac{2}{h} = 128$, which shows an error decrease by a factor of $2^{8.39}$ and eighth order accuracy. The order of accuracy of a finite difference method is an asymptotic concept that applies in the limit as the mesh size converges to zero, but due to roundoff errors, numerical computations in a given precision to a fixed simulation time with a particular algorithm have an effective lower bound on usable grid resolution that is finite, even though the bound can be affected by programming practices.

The second order algorithm with $\frac{2}{h} = 4$ has a maximum absolute error in $u$ and $p$ at $t = 10$ of 1.03451. At $t = 10$, the numerical solution from this simulation shows a maximum absolute value of $1.55885 \times 10^{-2}$ for $u$, and $3.45145 \times 10^{-2}$ for $p$, while the exact solution has $u$ identically 0, and extreme values of $\pm 1$ for $p$. The larger than one absolute error arises

because the $p$ solution is a half wavelength out of phase, which is possible even at a short time with a very coarse resolution and a dispersive method. Notice in Table A that the maximum errors at $\frac{2}{h} = 4$ range from $O[1]$ to $O[0.1]$, decreasing slightly as the order of accuracy of the method increases. This shows no outstanding advantage for the higher order methods at this coarse grid resolution. The actual advantages of higher order methods only become apparent when the qualities of the methods are not confounded with marginal data resolution.

Table B presents data from the same periodic problem as in Table A, but at $t = 1000$. In Table B the number of time steps is now given as $n_{1000}$, the number required to compute to $t = 1000$ with $\lambda = 0.8$ and the given grid resolution. The errors in Table B from the second order method at $\frac{2}{h} = 16$ and the fourth order method at $\frac{2}{h} = 8$ are produced by a very large phase error, just as the error in Table A for the second order method at $\frac{2}{h} = 4$. In general, the errors in Table B tend to be larger than the corresponding errors in Table A by a factor of 100, which is the ratio of $n_{1000}$ to $n_{10}$ at any given grid resolution. These algorithms are derived with an exact propagator for equation (3), so that the discrete approximation of the time evolution can be expected to have $O[1]$ eigenvalues and a linear growth trend in error. But notice that the maximum absolute errors on the coarsest grids never become much larger than one, or that the error growth has an asymptotic limit as the simulation time increases, since the errors eventually become of the order of the solution. Also notice that the noise level where accumulated roundoff error begins to significantly affect the solution has increased from $O[10^{-13}]$ at $t = 10$ to $O[10^{-9}]$ at $t = 1000$.

Table B: Data From The Four Algorithms For The 1D LEE
$u(x,0) = 0$ and $p(x,0) = \mathrm{Sin}(\pi x)$, $\lambda = 0.8, M = 0$
Maximum Error in $u$ or $p$ at $t = 1000$

| $\frac{2}{h}$ | $n_{1000}$ | $Or = 2$ | $Or = 4$ | $Or = 6$ | $Or = 8$ |
|---|---|---|---|---|---|
| 4 | 2500 | 1.000D+0 | 1.000D+0 | 1.000D+0 | 1.000D+0 |
| 8 | 5000 | 1.000D+0 | 1.004D+0 | 7.787D-01 | 1.265D-01 |
| 16 | 10000 | 1.001D+0 | 5.407D-01 | 2.145D-02 | 6.998D-04 |
| 32 | 20000 | 7.401D-01 | 4.614D-02 | 3.552D-04 | 2.909D-06 |
| 64 | 40000 | 1.215D+0 | 2.931D-03 | 5.618D-06 | 9.548D-09 |
| 128 | 80000 | 4.327D-01 | 1.837D-04 | 9.263D-08 | 4.590D-09 |
| 256 | 160000 | 1.131D-01 | 1.150D-05 | 9.364D-09 | 7.987D-09 |
| 512 | 320000 | 2.838D-02 | 7.000D-07 | 1.816D-08 | |
| 1024 | 640000 | 7.096D-03 | 1.294D-08 | | |

If interpolation is used with the data in Table A, then this data suggests that in order to achieve a maximum absolute error of less than $5.0 \times 10^{-4}$ at $t = 10$, the second order method requires $\frac{1}{h} \geq 215$, the fourth requires $\frac{1}{h} \geq 16$, the sixth $\frac{1}{h} \geq 8$, and the eighth $\frac{1}{h} \geq 7$. For a grid ratio of $\lambda = \frac{\Delta t}{\Delta x}$, the number of time steps $n_{10}$ required to compute to $t = 10$ is $n_{10} = \frac{10h}{\lambda}$. The total number of multiplications required for each computation on the domain $-1 \leq x \leq 1$, with $\frac{1}{h}$ mesh points per unit interval, using a stencil with $n_s$ grid points, with

two equations using data from two functions, and for $n_{10}$ time steps is

$$\text{total multiplications} = n_{10} \times 2h \times n_s \times 4 = \frac{80h^2 n_s}{\lambda}.$$

With modern RISC processors, a single cycle multiply and add instruction is possible, so that additions can be neglected in floating point operation counts. In order to meet the $5 \times 10^{-4}$ error bound at the time $t = 10$, the ratio between the number of multiplications required by the second and fourth order methods is

$$\frac{215^2 \times 3}{16^2 \times 5} \approx 108.34$$

it is 2.86 for the fourth to the sixth order, and 1.02 for the sixth to the eighth order. If the error bound is decreased to $5 \times 10^{-5}$ at $t = 10$, then the second order method requires $\frac{1}{h} \geq 673$, the fourth $\frac{1}{h} \geq 31$, the sixth $\frac{1}{h} \geq 14$, and the eighth $\frac{1}{h} \geq 8$. In this case, the ratio of the number of multiplications required by the different methods is approximately 282.79 for the second to the fourth order, 3.50 for the fourth to the sixth order, and 2.38 for the sixth to the eighth order. If the error bound is kept at $5 \times 10^{-4}$ but the time is increased to $t = 1000$, then the second order method requires $\frac{1}{h} \geq 1989$, the fourth $\frac{1}{h} \geq 60$, the sixth $\frac{1}{h} \geq 16$, and the eighth $\frac{1}{h} \geq 10$. In this case, the ratio of the number of multiplications required by the different methods is approximately 659.4 for the second to the fourth order, 10.04 for the fourth to the sixth order, and 1.99 for the sixth to the eighth order.

The difference between low and high order algorithms is not very significant if the data is not resolved well enough for any of them to be able to accurately propagate meaningful information to a useful time. It is significant that small errors can be obtained at relatively long times, even though there is a limit to what can be achieved with double precision arithmetic. For example, if the periodic problem is computed with the eighth order method at $\frac{1}{h} = 16$ out to $t = 100000$, or 50000 periods, then the maximum absolute error 3.022D-04 is invisible. The higher order methods are clearly more efficient in terms of the total number of floating point operations required to compute to a fixed simulation time with a given error bound, and the comparative efficiency of the higher order methods increases as the simulation time is increased or as the error bound is decreased. There is a particularly great advantage shown by the fourth order method compared to the second. These observations should be expected, since they are predicted by the analysis of Kreiss and Oliger [19]. Higher order methods can be more complex, requiring additional research and development time as well as extra care in implementation, but the value of the efficiency of higher order methods increases either if they are incorporated in codes that are frequently used, or if specific computations are made possible only with the use of a highly efficient algorithm.

### III-C: High Order Boundary Conditions

Propagation algorithms must be complemented by high order boundary algorithms in order to be useful. Stable high order boundary algorithms can be developed by using the exact propagator approach. The general algorithm development for equation (3) uses a local

spatial interpolant (6) at time $t_n$ as initial data, and the general solution form (7), to obtain an exact solution to the linearized Euler equations which correctly incorporates the wave dynamics of the local interpolant. This exact solution can be written in local coordinates as

$$
\begin{aligned}
u(x_i + x, t_n + t) &\approx ua(x,t) \\
&= \frac{1}{2}\Big(\sum_{\alpha=0}^{Or} u_\alpha (x-(M+1)t)^\alpha + \sum_{\alpha=0}^{Or} p_\alpha (x-(M+1)t)^\alpha\Big) \\
&+ \frac{1}{2}\Big(\sum_{\alpha=0}^{Or} u_\alpha (x-(M-1)t)^\alpha - \sum_{\alpha=0}^{Or} p_\alpha (x-(M-1)t)^\alpha\Big), \\
p(x_i + x, t_n + t) &\approx pa(x,t) \\
&= \frac{1}{2}\Big(\sum_{\alpha=0}^{Or} u_\alpha (x-(M+1)t)^\alpha + \sum_{\alpha=0}^{Or} p_\alpha (x-(M+1)t)^\alpha\Big) \\
&- \frac{1}{2}\Big(\sum_{\alpha=0}^{Or} u_\alpha (x-(M-1)t)^\alpha - \sum_{\alpha=0}^{Or} p_\alpha (x-(M-1)t)^\alpha\Big),
\end{aligned}
\tag{10}
$$

where $Or$ is the order of the interpolant, and where $u_\alpha$ and $p_\alpha$ are interpolation coefficients obtained by the method of undetermined coefficients. Note in (10) that an extra free variable has been added to the domain of $ua$ and $pa$ from (6), and that the general numerical algorithm in either form (7) or (8) can be obtained from (10) as $u_i^{n+1} = ua(0,k)$ and $p_i^{n+1} = pa(0,k)$. The local exact solution (10) can be interpreted as a multivariate Taylor series or Cauchy-Kowaleskaya expansion in space and time. The local exact solution (10) represents more than just an approach to obtaining a numerical approximation to $u$ and $p$ at $(x_i, t_{n+1})$, it is also a valid representation of the evolution in space and time of the local spatial interpolant (6) at all points which have their domain of dependence entirely within the interpolation stencil. The Riemann variables $\omega_1 = \frac{1}{2}(u-p)$ and $\omega_2 = \frac{1}{2}(u+p)$ can also be locally approximated in space and time, with $\omega a_1 = \frac{1}{2}(ua - pa)$ and $\omega a_2 = \frac{1}{2}(ua + pa)$. For subsonic mean flow with $M < 1$, the left going Riemann variable $\omega_1$ will have a valid representation up to the left end of the stencil, and the right going Riemann variable $\omega_2$ will have a valid representation up to the right end of the stencil. The general boundary algorithm idea is that at the boundaries of the computational domain, the outgoing Riemann variables are calculated, and appropriate boundary conditions are used to provide enough degrees of freedom of information to determine the primitive variable solutions at the boundary. If there are any points between the center and boundary points of a boundary stencil, then solution values are computed at these intermediate points with (10). All of these different calculations use a single spatial interpolation on one boundary stencil, and a consistent representation of the local evolution of the interpolant over the interval from the stencil center to the boundary of the computational domain.

In the case of $Or = 2$ on a three point stencil, in addition to the new values at the stencil center the only calculation that is needed is the outgoing Riemann variable. At the stencil center, either (8) or (10) give

$$
\begin{aligned}
u_i^{n+1} &= u_0 - k(p_1 + Mu_1) + k^2(2Mp_2 + (M^2+1)u_2), \\
p_i^{n+1} &= p_0 - k(u_1 + Mp_1) + k^2(2Mu_2 + (M^2+1)p_2).
\end{aligned}
\tag{11a}
$$

The left going Riemann variable is obtained from (10) at $(x_{i-1}, t_{n+1})$ as

$$\begin{aligned}\omega_{1\,i-1}^{n+1} =& \frac{1}{2}((u_0 - p_0) - h(u_1 - p_1) + h^2(u_2 - p_2)) \\ &+ \frac{1-M}{2}((u_1 - p_1) - 2h(u_2 - p_2)) \\ &+ \frac{(1-M)^2}{2}(u_2 - p_2),\end{aligned} \tag{11b}$$

and the right going Riemann variable at $(x_{i+1}, t_{n+1})$ as

$$\begin{aligned}\omega_{2\,i+1}^{n+1} =& \frac{1}{2}((u_0 + p_0) + h(u_1 + p_1) + h^2(u_2 + p_2)) \\ &- \frac{1+M}{2}((u_1 + p_1) + 2h(u_2 + p_2)) \\ &+ \frac{(1+M)^2}{2}(u_2 + p_2),\end{aligned} \tag{11c}$$

where the interpolation coefficients for $u$ are

$$u_0 = u_i^n, \;\; u_1 = \frac{1}{2h}(u_{i+1}^n - u_{i-1}^n), \;\; \text{and} \;\; u_2 = \frac{1}{2h^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n),$$

with similar forms for $p$. If the three point stencil is at the left of the computational domain, then (11a) and (11b) are used, and if the three point stencil is at the right of the computational domain, then (11a) and (11c) are used. In either case, a single interpolation stencil is used to provide the coefficients $u_\alpha$ and $p_\alpha$ for all three calculations, and an additional degree of freedom of information is required to obtain the primitive variable solutions at the boundary point.

In the case of $Or = 4$ on a five point stencil, the required calculations are for $u$ and $p$ values at the stencil center and one intermediate point, and for one outgoing Riemann variable at a stencil edge. At the stencil center, either (8) or (10) give

$$\begin{aligned}u_i^{n+1} &= u_0 - k(p_1 + Mu_1) + k^2(2Mp_2 + (M^2+1)u_2) \\ &\quad - k^3((1+3M^2)p_3 + M(3+M^2)u_3) \\ &\quad + k^4(4M(1+M^2)p_4 + (1+6M^2+M^4)u_4), \\ p_i^{n+1} &= p_0 - k(u_1 + Mp_1) + k^2(2Mu_2 + (M^2+1)p_2) \\ &\quad - k^3((1+3M^2)u_3 + M(3+M^2)p_3) \\ &\quad + k^4(4M(1+M^2)u_4 + (1+6M^2+M^4)p_4),\end{aligned}$$

where the coefficients for $u$ are given by (2), with similar forms for $p$. In a boundary stencil at the left of the computational domain, the intermediate solution values between the cell center

and the boundary point are given by (10) as

$$\begin{aligned}
u_{i-1}^{n+1} =& u_0 - hu_1 + h^2u_2 - h^3u_3 + h^4u_4 \\
&+k(M(-u_1 + 2hu_2 - 3h^2u_3 + 4h^3u_4) \\
&\quad + (-p_1 + 2hp_2 - 3h^2p_3 + 4h^3p_4)) \\
&+k^2((1+M^2)(u_2 - 3hu_3 + 6h^2u_4) \\
&\quad + 2M(p_2 - 3hp_3 + 6h^2p_4)) \\
&+k^3(M(3+M^2)(-u_3 + 4hu_4) \\
&\quad + (1+3M^2)(-p_3 + 4hp_4)),
\end{aligned}$$

$$\begin{aligned}
p_{i-1}^{n+1} =& p_0 - hp_1 + h^2p_2 - h^3p_3 + h^4p_4 \\
&+k(M(-p_1 + 2hp_2 - 3h^2p_3 + 4h^3p_4) \\
&\quad + (-u_1 + 2hu_2 - 3h^2u_3 + 4h^3u_4)) \\
&+k^2((1+M^2)(p_2 - 3hp_3 + 6h^2p_4) \\
&\quad + 2M(u_2 - 3hu_3 + 6h^2u_4)) \\
&+k^3(M(3+M^2)(-p_3 + 4hp_4) \\
&\quad + (1+3M^2)(-u_3 + 4hu_4)),
\end{aligned}$$

and the boundary point value for the outgoing Riemann variable is given by (10) as

$$\begin{aligned}
{\omega_1}_{i-2}^{n+1} =& \frac{1}{2}\omega_{1,0} - h\omega_{1,1} + 2h^2\omega_{1,2} - 4h^3\omega_{1,3} + 8h^4\omega_{1,4} \\
&+k(1-M)(\frac{1}{2}\omega_{1,1} - 2h\omega_{1,2} + 6h^2\omega_{1,3} - 16h^3\omega_{1,4}) \\
&+k^2(1-M)^2(\frac{1}{2}\omega_{1,2} - 3h\omega_{1,3} + 12h^2\omega_{1,4}) \\
&+k^3(1-M)^3(\frac{1}{2}\omega_{1,3} - 4h\omega_{1,4}) \\
&+k^4(1-M)^4\frac{1}{2}\omega_{1,4},
\end{aligned}$$

where $\omega_{1,\alpha} = \frac{1}{2}(u_\alpha - p_\alpha)$, with the coefficients $u_\alpha$ and $p_\alpha$ given by (2).

In the case of $Or = 6$ on a seven point stencil, $u$ and $p$ values are needed at the stencil center and two intermediate points, and one outgoing Riemann variable is needed at a stencil edge. If the three boundary treatments for $Or = 2$, 4 or 6 are used with Dirichlet boundary data for $u$ and $p$, then they maintain the order of accuracy of the propagation algorithm in both space and time, and they are stable if $|M\frac{\Delta t}{\Delta x}| \leq 1$. In the case of $Or = 8$ on a nine point stencil, if this general procedure is followed, with $u$ and $p$ values calculated at the stencil center and three intermediate points, with one outgoing Riemann variable calculated at a stencil edge, and with boundary data for $u$ and $p$, then this boundary treatment is unstable. A stable boundary treatment is obtained if a nonstandard interpolation is introduced on an

eight point stencil with four points on the interior side of the stencil center, and three points on the exterior side. The extra degree of freedom of information that is required for eighth order accuracy is obtained by adding spatial derivative variables for both $u$ and $p$ on the boundaries. The method of undetermined coefficients easily accomodates this information by using the derivatives of (6) evaluated at the boundary in addition to the values of (6) on the eight stencil points. For example, in an eight point stencil at the left edge of the conputational domain, the solution values on the stencil must be suplemented by the derivative data $u_{x_{i-3}}^n$ and $p_{x_{i-3}}^n$ at the stencil edge. In this case, the method of undetermined coefficients produces the estimate

$$u_1 = \frac{1}{14700h}(-420hu_{x_{i-3}}^n - 13230u_{i-1}^n + 4410u_{i-2}^n - 1229u_{i-3}^n \\ +1225u_i^n + 11025u_{i+1}^n - 2646u_{i+2}^n + 490u_{i+3}^n - 45u_{i+4}^n),$$

with similar estimates for the other $u_\alpha$ and $p_\alpha$. On this stencil, both the outgoing Riemann variable and its derivative are calculated at the boundary point, and two degrees of freedom of boundary information must be supplied. The derivative of the outgoing Riemann variable is obtained by combining the spatial derivatives of (10). An eight point stencil at the right edge of the computational domain is handled in a similar manner. If data is provided for one of the variables and its spatial derivative, then this eight point stencil boundary treatment is stable and eighth order accurate in both space and time.

Numerical results are presented in Table C for the intial boundary value problem with initial data $u(x,0) = 0$ and $p(x,0) = \mathrm{Sin}(\pi x)$, for $-1 < x < 1$, and with boundary data $u(-1,t) = \frac{1}{2}(\mathrm{Sin}(\pi(-1-t)) - \mathrm{Sin}(\pi(-1+t)))$ and $p(1,t) = \frac{1}{2}(\mathrm{Sin}(\pi(1-t)) - \mathrm{Sin}(\pi(1+t)))$, for $0 \leq t$. This problem is the periodic initial value problem from Section III-B in a truncated domain with the solutions for $u(-1,t)$ and $p(1,t)$ given at opposite boundaries of the domain. This type of boundary data specification is common in computational fluid dynamics. The boundary algorithms are as specified in this section.

Table C: Data From The Four Algorithms For The 1D LEE
BC - Outgoing Riemann Variable, $u(-1,t)$, and $p(1,t)$
$u(x,0) = 0$ and $p(x,0) = \mathrm{Sin}(\pi x)$, $\lambda = 0.8, M = 0$
Maximum Error in $u$ or $p$ at $t = 10$

| $\frac{2}{h}$ | $n_{10}$ | $Or = 2$ | $Or = 4$ | $Or = 6$ | $Or = 8$ |
|---|---|---|---|---|---|
| 8 | 50 | 1.870D-01 | 1.170D-02 | 1.012D-02 | 8.653D-05 |
| 16 | 100 | 4.569D-02 | 9.980D-04 | 5.260D-05 | 8.499D-07 |
| 32 | 200 | 1.323D-02 | 8.109D-05 | 8.786D-07 | 4.690D-09 |
| 64 | 400 | 3.500D-03 | 5.540D-06 | 1.290D-08 | 2.087D-11 |
| 128 | 800 | 8.937D-04 | 3.580D-07 | 1.894D-10 | 3.752D-13 |
| 256 | 1600 | 2.254D-04 | 2.270D-08 | 3.346D-12 | 4.910D-13 |
| 512 | 3200 | 5.657D-05 | 1.430D-09 | 1.889D-12 | |
| 1024 | 6400 | 1.417D-05 | 9.115D-11 | | |

The data in Table C confirms that the propagation algorithms with the boundary algorithms have the same order of accuracy as the propagation algorithms with periodic boundaries. This can also be confirmed by truncation error analysis. The parameters used for the computations in Tables A and C are the same, so that the results can be directly compared. The overall error data in Table C is slightly smaller than in Table A, by method and by grid size, most notably for the eighth order method, particularly at $\frac{2}{h} = 8$. At $\frac{2}{h} = 8$, the eighth order method has a nine point domain, and uses only three stencils for the entire calculation. The entire nine point domain is used for computing the values at the central grid point, the leftmost eight points are used to calculate everything to the left of the central point, and the rightmost eight points everything to the right. Specification of Dirichlet data and computation of the outgoing Riemann variables provides slightly more accurate results throughout Table C than comparable results in Table A with periodic boundary conditions. The computations for this initial boundary value problem can be carried out to longer times. For example, in computing out to $t = 1000$ with the eighth order method, if $\frac{2}{h} = 8$ is used, then the maximum absolute error is 5.405D-05, while if $\frac{2}{h} = 64$ is used, then the maximum absolute error is 6.522D-12. These boundary algorithms appear stable, and they clearly have the same order of accuracy as the propagation algorithms, from second to eighth. The boundary algorithms can be viewed as using interior differencing at and near the boundaries. From this viewpoint the stencils become heavily weighted towards the interior, much more so than appears possible with conventional upwind algorithms. A central feature of these boundary algorithms is the simultaneous and consistent calculation of the correct wave dynamics of the solution over an interval next to the boundary.

## IV: Numerical Algorithms For The Convection Equation In Two Dimensions

The simplest hyperbolic problem in two space dimensions is the scalar first order linear wave equation for $u(x,y,t)$,

$$\frac{\partial u}{\partial t} + M_x \frac{\partial u}{\partial x} + M_y \frac{\partial u}{\partial y} = 0, \tag{12}$$

where $M_x$ and $M_y$ are the constant mean convection velocities in the $x$ and $y$ directions, respectively. The initial value problem $u(x,y,0) = ui(x,y)$ for $(x,y) \in \Re \times \Re$ has the general solution $u(x,y,t) = ui(x - M_x t, y - M_y t)$ for $(x,y) \in \Re \times \Re$ and $0 \leq t$. This solution form is developed by the method of characteristics, it incorporates the geometry of the solution's behavior, and it can be used to develop exact propagator algorithms for equation (12). In one space dimension with a common stencil the exact propagator and Taylor series approaches to algorithm development lead to the same algorithm, but for two dimensional problems with a common stencil they lead to algorithms that are distinctly different. Both approaches are used and compared in this section with examples of second, fourth and sixth order algorithms. A significant new issue for two dimensional problems is the choice of stencil and interpolant.

### IV-A: A General Algorithm Development

A uniform mesh is used with grid points $(x_i, y_j) = (ih, jh)$ for integers $i$ and $j$, where $h = \Delta x = \Delta y$ is the uniform mesh spacing in space, and with discrete times $t_n = nk$ for integer $n$, where $k = \Delta t$ is the uniform time step size. The numerical solution at the mesh point $(x_i, y_j, t_n)$ is denoted by $u^n_{i,j} \approx u(x_i, y_j, t_n)$. In two space dimensions, a polynomial spatial interpolation to $u$ around $(x_i, y_j)$ at $t_n$ can be written in local coordinates as

$$u(x_i + x, y_j + y, t_n) \approx ua(x,y) = \sum_{\{\alpha,\beta\} \in AS} u_{\alpha,\beta} x^\alpha y^\beta, \tag{13}$$

where $AS$ is the index set for the expansion, and where the coefficients are not indexed with respect to the mesh point. All of the undetermined coefficients are simultaneously obtained by interpolating a data surface to the solution values on a given stencil. The coefficients can be interpreted as spatial derivatives, with

$$u_{\alpha,\beta} = \frac{1}{\alpha!\beta!} \frac{\partial^{\alpha+\beta} ua}{\partial x^\alpha \partial y^\beta}(x_i, y_j, t_n) \approx \frac{1}{\alpha!\beta!} \frac{\partial^{\alpha+\beta} u}{\partial x^\alpha \partial y^\beta}(x_i, y_j, t_n). \tag{14}$$

The general form of the solution to equation (12) and the interpretation of the expansion coefficients can be used to develop a truncated Cauchy-Kowaleskaya series expansion that locally approximates the exact solution in space and time, with

$$u(x_i + x, y_j + y, t_n + t) \approx ua(x - M_x t, y - M_y t) = \sum_{\{\alpha,\beta\} \in AS} u_{\alpha,\beta} (x - M_x t)^\alpha (y - M_y t)^\beta. \tag{15}$$

This local polynomial approximation of the exact solution to the global problem is an exact solution to the local approximate problem which uses the local spatial interpolant as initial data, and it incorporates the correct wave propagation dynamics for (12).

The exact propagator algorithm form is obtained from the method of characteristics by using the locally exact solution form (15), with

$$u_{i,j}^{n+1} = ua(-M_x k, -M_y k) = \sum_{\{\alpha,\beta\} \in AS} u_{\alpha,\beta} (-M_x k)^\alpha (-M_y k)^\beta \approx u(x_i, y_j, t_n + k). \qquad (16)$$

This algorithm form can also be viewed as a time expansion, with powers of $k$ up to the maximum possible $\alpha + \beta$. A truncated Taylor series expansion in time produces the algorithm form

$$\begin{aligned} u_{i,j}^{n+1} &= \sum_{\gamma=0}^{Or} (-k)^\gamma \sum_{\beta=0}^{\gamma} u_{\gamma-\beta,\beta} M_x^{\gamma-\beta} M_y^\beta \\ &= \sum_{\gamma=0}^{Or} (-k)^\gamma \sum_{\beta=0}^{\gamma} \frac{1}{(\gamma-\beta)!\beta!} \frac{\partial^\gamma ua}{\partial x^{\gamma-\beta} \partial y^\beta}(x_i, y_j, t_n) \\ &\approx \sum_{\gamma=0}^{Or} (-k)^\gamma \sum_{\beta=0}^{\gamma} \frac{1}{(\gamma-\beta)!\beta!} \frac{\partial^\gamma u}{\partial x^{\gamma-\beta} \partial y^\beta}(x_i, y_j, t_n) \\ &= \sum_{\gamma=0}^{Or} \frac{k^\gamma}{\gamma!} \frac{\partial^\gamma u}{\partial t^\gamma}(x_i, y_j, t_n) \approx u(x_i, y_j, t_n + k), \end{aligned} \qquad (17)$$

where $Or$ is the order of the expansion. Both algorithm forms can be rewritten as conventional single step explicit finite difference methods, with

$$u_{i,j}^{n+1} = \sum_{\{r,s\} \in IS} c_{r,s} u_{i+r,j+s}^n,$$

where $IS$ is the appropriate index set for the stencil that is being used, and where the constant coefficients $c_{r,s}$ are polynomials in the convection velocities $M_x$ and $M_y$, and in the grid ratio $\lambda = \frac{k}{h}$. For any given order of accuracy $Or$, if the expansion coefficients $u_{\alpha,\beta}$ of (13) from a fixed stencil are used in both the characteristic based (16) and the Taylor series time expansion (17) algorithms, then both algorithms will have the same explicit finite diference index set $IS$, and they both require the same number of floating point operations per grid point per time step. In general, the time expansion form of the characteristic based algorithm will have time terms that are higher than $Or$, and the finite different coefficients $c_{r,s}$ for the two types of algorithm will differ.

The order of the algorithms of either type will depend upon the stencil and interpolant that are used for local approximation of the known data surface. Figure 1 presents information for stencils and interpolants that can be used for second, fourth and sixth order algorithms in two space dimensions. In Figure 1, information for second, fourth and sixth order algorithms is presented in the lefthand, center, and righthand columns, respectively. Stencil schematics

are presented in the top row. The stencils are all symmetric in space, with a stencil choice for fourth and sixth order algorithms. The points at the stencil centers are marked with a '+', other necessary points are marked with an 'o', and optional points are marked with an 'x'. The only second order stencil is the familiar $3 \times 3$ square. The two fourth order stencils are the $5 \times 5$ square, with twenty five points, and this square minus its four corner points, with twenty one points. The three sixth order stencils are the $7 \times 7$ square, with forty nine points, this square minus its four corner points, with forty five points, and this square minus the three points in each corner that are either in the corner or next to it on an edge, with thirty seven points. Spatial interpolation coefficient index sets are presented in the second and third rows of Figure 1, and represent the index set $AS$ that is used in (13). Each index pair represents an expansion coefficient that can be interpreted as a mixed partial derivative in $x$ and $y$, as in (14). The expansion coefficients in the second row are for use with the complete square stencil in each case, and the expansion coefficients in the third row are for use with the smallest stencil in each case. The index sets in the second and third rows of Figure 1 are used in (16) for exact propagator algorithms on either the full stencil or the smallest stencil for each order. The fourth row of Figure 1 presents the index sets that are retained in the Taylor series time expansion algorithms (17). The coefficients in the fourth row for the Taylor series algorithms can be obtained on any of the symmetric stencils that are possible for each order, or by other methods. The algorithms in this section are completely specified by the stencils and expansion coefficient index sets in Figure 1, and the general algorithm forms (16) and (17).

It can be shown that if the data for a simulation is independent from one of the coordinate variables, and if the mean convection velocity is perpendicular to that coordinate axis, then for a given order, algorithm forms (16) and (17) both reduce to the same one dimensional algorithm along the mean convection direction. Under these assumptions, the second order methods both reduce to the Lax-Wendroff scheme in one dimension, and the fourth order methods reduce to the algorithm in Section II. In multiple dimensions with general data and symmetric interpolation, algorithms for equation (12) that are derived as truncated Taylor series time expansions cannot be interpreted as projection backwards along a characteristic to its point of intersection with the local data surface at time $t_n$. In effect, Taylor series time expansion algorithms introduce an error in the time evolution of the local interpolated data surface.

### IV-B: Second Order Algorithms

The second order algorithms both use the $3 \times 3$ square stencil on the left of the top row in Figure 1. The expansion coefficient index set for the exact propagator algorithm is given on the left of the second row in Figure 1, and the expansion coefficient forms are given in Appendix A. The local biquadratic spatial interpolant for $u$ near $(x_i, y_j)$ at $t_n$ can be written as

$$u(x_i + x, y_j + y, t_n) \approx ua(x, y) = \sum_{\alpha,\beta=0}^{2} u_{\alpha,\beta} x^\alpha y^\beta. \tag{18}$$

With this interpolant, the locally exact algorithm (16) is

$$\begin{aligned} u_{i,j}^{n+1} =& u_{00} - k(u_{10}M_x + u_{01}M_y) + k^2(u_{20}{M_x}^2 + u_{11}M_xM_y + u_{02}{M_y}^2) \\ & - k^3(u_{21}{M_x}^2M_y + u_{12}M_x{M_y}^2) + k^4(u_{22}{M_x}^2{M_y}^2). \end{aligned} \tag{19}$$

Notice that algorithm (19) has time expansion terms up to $k^4$. The truncation error for algorithm (19) is

$$\begin{aligned} \frac{1}{k}(u(x_i, y_j, t_{n+1}) - u_{i,j}^{n+1}) = & \frac{\partial u}{\partial t} + M_x\frac{\partial u}{\partial x} + M_y\frac{\partial u}{\partial y} \\ & + h^2(\frac{M_x}{6})(1 - {M_x}^2\lambda^2)\frac{\partial^3 u}{\partial x^3} + h^2(\frac{M_y}{6})(1 - {M_y}^2\lambda^2)\frac{\partial^3 u}{\partial y^3} \\ & + O[h^3]. \end{aligned}$$

Algorithm (19) is consistent with equation (12), it is second order accurate in space and time, and it is dispersive with no cross derivative terms in its $O[h^2]$ truncation error.

The truncated Taylor series time expansion algorithm (17) for this stencil is

$$u_{i,j}^{n+1} = u_{00} - k(u_{10}M_x + u_{01}M_y) + k^2(u_{20}{M_x}^2 + u_{11}M_xM_y + u_{02}{M_y}^2), \tag{20}$$

which is just the second order Lax-Wendroff method for (12). Notice that algorithm (20) includes the terms of algorithm (19) up through $k^2$, but does not have $k^3$ and $k^4$ terms. Algorithm (20) requires the cross difference term $u_{11}$, and the simplest approximation of $u_{11}$ with a symmetric stencil requires the use of the four corner points in the $3 \times 3$ nine point central stencil. The truncation error for algorithm (20) is

$$\begin{aligned} \frac{1}{k}(u(x_i, y_j, t_{n+1}) - u_{i,j}^{n+1}) = & \frac{\partial u}{\partial t} + M_x\frac{\partial u}{\partial x} + M_y\frac{\partial u}{\partial y} \\ & + h^2(\frac{M_x}{6})(1 - {M_x}^2\lambda^2)\frac{\partial^3 u}{\partial x^3} + h^2(\frac{M_y}{6})(1 - {M_y}^2\lambda^2)\frac{\partial^3 u}{\partial y^3} \\ & - h^2(\frac{{M_x}^2M_y}{2})\lambda^2\frac{\partial^3 u}{\partial x^2\partial y} - h^2(\frac{M_x{M_y}^2}{2})\lambda^2\frac{\partial^3 u}{\partial x\partial y^2} \\ & + O[h^3]. \end{aligned}$$

Algorithm (20) is consistent with equation (12), it is second order accurate in space and time, it is dispersive, and it has every possible $O[h^2]$ cross derivative term in its truncation error.

### IV-C: Fourth Order Algorithms

The square $5 \times 5$ central stencil can be used for fourth order algorithms, with the biquartic interpolation

$$u(x_i + x, y_j + y, t_n) \approx ua(x, y) = \sum_{\{\alpha,\beta\} \in A_{25}} u_{\alpha,\beta}x^\alpha y^\beta = \sum_{\alpha,\beta=0}^{4} u_{\alpha,\beta}x^\alpha y^\beta. \tag{21}$$

The locally exact algorithm form (16) with this interpolant is

$$\begin{aligned} u_{i,j}^{n+1} =& u_{00} - k(u_{10}M_x + u_{01}M_y) + k^2(u_{20}{M_x}^2 + u_{11}M_xM_y + u_{02}{M_y}^2) \\ & - k^3(u_{30}{M_x}^3 + u_{21}{M_x}^2M_y + u_{12}M_x{M_y}^2 + u_{03}{M_y}^3) \\ & + k^4(u_{40}{M_x}^4 + u_{31}{M_x}^3M_y + u_{22}{M_x}^2{M_y}^2 + u_{13}M_x{M_y}^3 + u_{04}{M_y}^4) \\ & - k^5(u_{41}{M_x}^4M_y + u_{32}{M_x}^3{M_y}^2 + u_{23}{M_x}^2{M_y}^3 + u_{14}M_x{M_y}^4) \\ & + k^6(u_{42}{M_x}^4{M_y}^2 + u_{33}{M_x}^3{M_y}^3 + u_{24}{M_x}^2{M_y}^4) \\ & - k^7(u_{43}{M_x}^4{M_y}^3 + u_{34}{M_x}^3{M_y}^4) \\ & + k^8(u_{44}{M_x}^4{M_y}^4). \end{aligned} \tag{22}$$

Notice that this algorithm has time expansion terms up to $k^8$.

The second fourth order algorithm uses the reduced stencil with twenty one grid points and the modified biquartic interpolation

$$u(x_i + x, y_j + y, t_n) \approx ua(x, y) = \sum_{\{\alpha,\beta\} \in A_{21}} u_{\alpha,\beta} x^\alpha y^\beta, \tag{23}$$

where $A_{21}$ is set of indexes in the center of the third row in Figure 1. The coefficients for $A_{21}$ are given in Appendix B. With interpolant (23), the locally exact algorithm form (16) is

$$\begin{aligned} u_{i,j}^{n+1} =& u_{00} - k(u_{10}M_x + u_{01}M_y) + k^2(u_{20}{M_x}^2 + u_{11}M_xM_y + u_{02}{M_y}^2) \\ & - k^3(u_{30}{M_x}^3 + u_{21}{M_x}^2M_y + u_{12}M_x{M_y}^2 + u_{03}{M_y}^3) \\ & + k^4(u_{40}{M_x}^4 + u_{31}{M_x}^3M_y + u_{22}{M_x}^2{M_y}^2 + u_{13}M_x{M_y}^3 + u_{04}{M_y}^4) \\ & - k^5(u_{41}{M_x}^4M_y + u_{32}{M_x}^3{M_y}^2 + u_{23}{M_x}^2{M_y}^3 + u_{14}M_x{M_y}^4) \\ & + k^6(u_{42}{M_x}^4{M_y}^2 + u_{24}{M_x}^2{M_y}^4). \end{aligned} \tag{24}$$

Note that algorithm (24) has time terms up to $k^6$.

The leading order truncation error for both algorithms (22) and (24) is

$$\begin{aligned} \frac{1}{k}(u(x_i, y_j, t_{n+1}) - u_{i,j}^{n+1}) =& \frac{\partial u}{\partial t} + M_x\frac{\partial u}{\partial x} + M_y\frac{\partial u}{\partial y} \\ & - h^4(\frac{M_x}{120})(1 - {M_x}^2\lambda^2)(4 - {M_x}^2\lambda^2)\frac{\partial^5 u}{\partial x^5} \\ & - h^4(\frac{M_y}{120})(1 - {M_y}^2\lambda^2)(4 - {M_y}^2\lambda^2)\frac{\partial^5 u}{\partial y^5} \\ & + O[h^5]. \end{aligned}$$

Algorithms (22) and (24) have the same $O[h^5]$ truncation errors, except that algorithm (24) has an additional mixed $\{3,3\}$ cross derivative term. The $\{3,3\}$ index pair is not in $A_{21}$. Algorithms (22) and (24) are consistent with equation (12), they are fourth order accurate

in space and time, and they have leading order truncation error that are dispersive with no cross derivative terms. In the form of conventional finite difference methods, algorithm (22) requires twenty five multiplications and twenty four additions at each grid point for each time step, and algorithm (24) requires twenty one multiplications and twenty additions.

The third fourth order algorithm uses the truncated Taylor series in time (17), with

$$\begin{aligned} u_{i,j}^{n+1} =& u_{00} - k(u_{10}M_x + u_{01}M_y) + k^2(u_{20}{M_x}^2 + u_{11}M_xM_y + u_{02}{M_y}^2) \\ & - k^3(u_{30}{M_x}^3 + u_{21}{M_x}^2M_y + u_{12}M_x{M_y}^2 + u_{03}{M_y}^3) \\ & + k^4(u_{40}{M_x}^4 + u_{31}{M_x}^3M_y + u_{22}{M_x}^2{M_y}^2 + u_{13}M_x{M_y}^3 + u_{04}{M_y}^4). \end{aligned} \tag{25}$$

Notice that algorithm (25) has the terms up to $k^4$ which are in both algorithm (22) and (24), but that it does not have the higher order terms that are used for the accurate evolution of their local data interpolants. The coefficients in (25) can be obtained from interpolations (21) or (23), or elsewhere. If the twenty one point stencil from algorithm (24) is used, then the leading order truncation error for algorithm (25) is

$$\begin{aligned} \frac{1}{k}(u(x_i,y_j,t_{n+1}) - u_{i,j}^{n+1}) =& \frac{\partial u}{\partial t} + M_x\frac{\partial u}{\partial x} + M_y\frac{\partial u}{\partial y} \\ & - h^4(\frac{M_x}{120})(1-{M_x}^2\lambda^2)(4-{M_x}^2\lambda^2)\frac{\partial^5 u}{\partial x^5} \\ & - h^4(\frac{M_y}{120})(1-{M_y}^2\lambda^2)(4-{M_y}^2\lambda^2)\frac{\partial^5 u}{\partial y^5} \\ & - h^4(\frac{{M_x}^4M_y}{24})\lambda^4\frac{\partial^5 u}{\partial x^4\partial y} - h^4(\frac{{M_x}^3{M_y}^2}{12})\lambda^4\frac{\partial^5 u}{\partial x^3\partial y^2} \\ & - h^4(\frac{M_x{M_y}^4}{24})\lambda^4\frac{\partial^5 u}{\partial x\partial y^4} - h^4(\frac{{M_x}^2{M_y}^3}{12})\lambda^4\frac{\partial^5 u}{\partial x^2\partial y^3} \\ & + O[h^5]. \end{aligned}$$

If the twenty five point stencil from algorithm (22) is used to estimate the coefficients for algorithm (25), then the $O[h^4]$ truncation error terms remain unchanged, and there is a change in only the $\{3,3\}$ mixed $O[h^5]$ term. Algorithm (25) is consistent with equation (12), it is fourth order accurate in space and time, it is dispersive, and it has every possible $O[h^4]$ cross derivative term in its truncation error.

### IV-D: Sixth Order Algorithms

Within the $7\times7$ square stencil there are three symmetric options for developing algorithms that are sixth order, but only two will be considered. Sixth order algorithms will use either a $7 \times 7$ central stencil and the bisextic interpolation

$$u(x_i+x, y_j+y, t_n) \approx ua(x,y) = \sum_{\{\alpha,\beta\}\in A_{49}} u_{\alpha,\beta}x^\alpha y^\beta = \sum_{\alpha,\beta=0}^{6} u_{\alpha,\beta}x^\alpha y^\beta, \tag{26}$$

or the reduced stencil with thirty seven grid points and the modified sixth order interpolation

$$u(x_i + x, y_j + y, t_n) \approx ua(x, y) = \sum_{\{\alpha,\beta\} \in A_{37}} u_{\alpha,\beta} x^\alpha y^\beta. \tag{27}$$

The index sets $A_{49}$ and $A_{37}$ are given in the righthand column of the second and third rows in Figure 1, respectively. The locally exact solution form (16) produces the algorithms

$$u_{i,j}^{n+1} = \sum_{\{\alpha,\beta\} \in A_{49}} u_{\alpha,\beta} (-M_x k)^\alpha (-M_y k)^\beta, \tag{28}$$

with the forty nine point square stencil and interpolant (26), and

$$u_{i,j}^{n+1} = \sum_{\{\alpha,\beta\} \in A_{37}} u_{\alpha,\beta} (-M_x k)^\alpha (-M_y k)^\beta, \tag{29}$$

with the reduced stencil and interpolant (27). The leading order truncation error terms for both algorithms (28) and (29) are

$$\begin{aligned}
\frac{1}{k}(u(x_i, y_j, t_{n+1}) - u_{i,j}^{n+1}) &= \frac{\partial u}{\partial t} + M_x \frac{\partial u}{\partial x} + M_y \frac{\partial u}{\partial y} \\
&+ h^6 (\frac{M_x}{5040})(1 - {M_x}^2 \lambda^2)(4 - {M_x}^2 \lambda^2)(12 - {M_x}^2 \lambda^2) \frac{\partial^7 u}{\partial x^7} \\
&+ h^6 (\frac{M_y}{5040})(1 - {M_y}^2 \lambda^2)(4 - {M_y}^2 \lambda^2)(12 - {M_y}^2 \lambda^2) \frac{\partial^7 u}{\partial y^7} \\
&+ O[h^7].
\end{aligned}$$

These algorithms are both consistent with equation (12), and sixth order accurate in space and time, and they both have dispersive leading order truncation error with no cross derivative terms. Notice that algorithm (28) has terms up to $k^{12}$, while algorithm (29) has terms up to $k^8$. There appears to be little accuracy advantage for algorithm (28) with the larger stencil, which has approximately a third more grid points. The thirty seven point stencil does not have corners, so that special care has to be exercised when treating boundaries and corners for problems with nonperiodic boundaries.

The sixth order truncated Taylor series algorithm (16) has the form

$$u_{i,j}^{n+1} = \sum_{\gamma=0}^{6} \frac{k^\gamma}{\gamma!} \frac{\partial^\gamma u}{\partial t^\gamma}(x_i, y_j, t_n), \tag{30}$$

with terms up to $k^6$. If the thirty seven point stencil is used to obtain the coefficients for

algorithm (30), then the leading order truncation error terms are

$$\frac{1}{k}(u(x_i,y_j,t_{n+1})-u_{i,j}^{n+1}) = \frac{\partial u}{\partial t}+M_x\frac{\partial u}{\partial x}+M_y\frac{\partial u}{\partial y}$$
$$+h^6(\frac{M_x}{5040})(1-{M_x}^2\lambda^2)(4-{M_x}^2\lambda^2)(12-{M_x}^2\lambda^2)\frac{\partial^7 u}{\partial x^7}$$
$$+h^6(\frac{M_y}{5040})(1-{M_y}^2\lambda^2)(4-{M_y}^2\lambda^2)(12-{M_y}^2\lambda^2)\frac{\partial^7 u}{\partial y^7}$$
$$-h^6(\frac{{M_x}^6M_y}{720})\lambda^6\frac{\partial^7 u}{\partial x^6\partial y}-h^6(\frac{M_x{M_y}^6}{720})\lambda^6\frac{\partial^7 u}{\partial x\partial y^6}$$
$$-h^6(\frac{{M_x}^5{M_y}^2}{240})\lambda^6\frac{\partial^7 u}{\partial x^5\partial y^2}-h^6(\frac{{M_x}^2{M_y}^5}{240})\lambda^6\frac{\partial^7 u}{\partial x^2\partial y^5}$$
$$-h^6(\frac{{M_x}^4{M_y}^3}{144})\lambda^6\frac{\partial^7 u}{\partial x^4\partial y^3}-h^6(\frac{{M_x}^3{M_y}^4}{144})\lambda^6\frac{\partial^7 u}{\partial x^3\partial y^4}$$
$$+O[h^7].$$

Algorithm (30) is consistent with equation (12), it is sixth order accurate in space and time, and it has a leading order truncation error term that is dispersive with every possible cross derivative term.

### IV-E: Numerical Comparisons

As a numerical test of the algorithms in this section, consider equation (12) with the initial data

$$u(x,y,0)=\mathrm{Sin}(\pi x)\mathrm{Sin}(\pi y),$$

for $(x,y)\in[-1,1]\times[-1,1]$, and with periodic boundaries. Computations will be done with $M_x=1=M_y$ and $\lambda=\frac{k}{h}=\frac{8}{10}$, for a sequence of grid sizes, and out to the fixed time $t=10$. Numerical data for this problem is presented in Table D. In Table D, $\frac{2}{h}$ is the number of grid points in $[-1,1]$, or per wavelength, $n_{10}$ is the number of time steps required to compute to $t=10$ at a given grid resolution with $\lambda=\frac{8}{10}$, and the data is the maximum absolute error in $u$ over the entire grid at $t=10$. In the column headings of Table D, $Or$ is the order of the method, $EX_9$ and $TS_9$ represent the second order exact propagator and Taylor series algorithms (19) and (20) on the nine point $3\times3$ stencil, $EX_{21}$ and $TS_{21}$ represent algorithms (24) and (25) on the twenty one point stencil, $EX_{49}$ and $TS_{49}$ represent algorithms (28) and (30) on the forty nine point $7\times7$ stencil, and $EX_{37}$ and $TS_{37}$ represent algorithms (29) and (30) on the thirty seven point stencil. Note that the square twenty five point $5\times5$ stencil is not used, and that the sixth order Taylor series algorithm (30) is used with both the thirty seven and forty nine point stencils. The second order Taylor series method $TS_9$ is unstable at all grid resolutions with $\lambda=0.8$ for $M_x=1$ and $M_y=1$, so that $\lambda=0.4$ is used to obtain it's data. The large error for the second order Taylor series algorithm at $\frac{2}{h}=8$ is due to dispersion errors from poorly resolved data, just as for similar cases among the data in Table

A. Calculation of error reduction exponents similar to those conducted for the data in Table A confirms the order of accuracy for each of the methods used to produce Table D.

Table D: Data For The Scalar First Order Wave Equation In 2D
$u(x,y,0) = \mathrm{Sin}(\pi x)\mathrm{Sin}(\pi y)$, $\lambda = 0.8, M_x = 1 = M_y$
Maximum Error In $u$ at $t = 10$

| $\frac{2}{h}$ | $n_{10}$ | $Or = 2$ $EX_9$ | $Or = 2$ $TS_9$* | $Or = 4$ $EX_{21}$ | $Or = 4$ $TS_{21}$ | $Or = 6$ $EX_{49}$ | $Or = 6$ $TS_{49}$ | $Or = 6$ $EX_{37}$ | $Or = 6$ $TS_{37}$ |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 25 | 0.999 | 0.959 | 0.978 | 0.675 | 0.670 | | 0.733 | 0.841 |
| 8 | 50 | 0.895 | 1.414 | 0.156 | 0.365 | 1.8D-02 | 2.0D-02 | 2.2D-02 | 2.2D-02 |
| 16 | 100 | 0.322 | 0.387 | 9.6D-03 | 4.2D-02 | 2.8D-04 | 2.3D-04 | 3.1D-04 | 2.6D-04 |
| 32 | 200 | 8.0D-02 | 8.6D-02 | 5.4D-04 | 2.8D-03 | 4.1D-06 | 2.9D-06 | 4.3D-06 | 3.2D-06 |
| 64 | 400 | 1.9D-02 | 2.0D-02 | 3.2D-05 | 1.8D-04 | 6.0D-08 | 4.0D-08 | 6.2D-08 | 4.2D-08 |
| 128 | 800 | 4.7D-03 | 4.7D-03 | 1.9D-06 | 1.1D-05 | 9.1D-10 | 6.0D-10 | 9.2D-10 | 6.1D-10 |
| 256 | 1600 | 1.2D-03 | 1.2D-03 | 1.2D-07 | 7.0D-07 | 1.4D-11 | 2.8D-06 | 1.4D-11 | 9.2D-12 |
| 512 | 3200 | 2.9D-04 | 2.9D-04 | 7.3D-09 | 4.4D-08 | 2.2D-13 | 2.9D-05 | 2.2D-13 | 4.0D-11 |

* Note that $\lambda = 0.4$ for the data in this column.

The sixth order Taylor series methods become inaccurate at fine grid resolutions with $\lambda = 0.8$, near $\frac{2}{h} = 256$ for $TS_{49}$, and near $\frac{2}{h} = 512$ for $TS_{37}$. If the grid ratio is reduced, then these Taylor series methods produce accurate results at these grid resolutions. The fine grid errors with the $TS_{49}$ and $TS_{37}$ methods appear to be due to the excitation of parasitic steady state periodic solutions of equation (12). Notice that the Taylor series algorithm on the larger square stencil shows signs of inaccuracy at a coarser grid resolution than the algorithm on the reduced stencil. Numerical experiments show that exact propagator methods are stable if both one dimensional CFL numbers are less than one in absolute value, but that the Taylor series methods have more restrictive stability constraints. At each level of grid refinement, the data in Table D shows errors from the exact propagator and Taylor series methods on the same stencil that differ by at most a factor of about 5. The largest differences are in the fourth order results, with smaller errors from the exact propagator method. Recall that both types of method can be recast as conventional finite difference algorithms, and that in this form the number of operations depends only upon the stencil. The exact propagator methods appear to be at least comparable in accuracy to the Taylor series methods, if not more accurate, and they appear to be more robustly stable, without requiring more operations. The choice of largest or smallest possible symmetric stencil does not appear to be significant for accuracy.

The data in Table A for the one dimensional linearized Euler equations, and the data in Table D for the two dimensional convection equation both come from periodic initial value problems with pure frequency sine wave initial data at the same wavelength. The two dimensional convection calculations are run with the velocity $(M_x, M_y) = (1, 1)$, which is at a $45^o$ angle with the grid lines. The close agreement between the error levels of the comparable one and two dimensional results suggests that convection skew to the grid does not introduce significant errors if these algorithms are used. Note that both algorithm types use time evolution that reflects the genuinely multidimensional nature of the convection equation. The Taylor

series in time methods include the cross derivative terms required for higher than first order accuracy in time, and the exact propagator methods directly incorporate the characteristic behaviour of the solution. Both method types use genuinely multidimensional interpolation for the local approximation of the solution surface in order to obtain the expansion coefficients that are needed for accurate time evolution.

If interpolation is used with the data in Table D, then in order to achieve a maximum absolute error of less than $5.0 \times 10^{-4}$ at $t = 10$, the second order methods require $\frac{1}{h} \geq 216$, the exact propagator fourth order method requires $\frac{1}{h} \geq 17$, the Taylor series fourth order method requires $\frac{1}{h} \geq 30$, and the sixth order methods require $\frac{1}{h} \geq 8$. For a grid ratio of $\lambda = \frac{\Delta t}{\Delta x}$, the number of time steps $n_{10}$ required to compute to $t = 10$ is $n_{10} = \frac{10h}{\lambda}$. Consequently, the total number of multiplications required for each computation on the domain $(x, y) \in [-1, 1] \times [-1, 1]$ with $\frac{1}{h}$ mesh points per unit interval, using a stencil with $n_s$ grid points, and for $n_{10}$ time steps is

$$\text{total multiplications} = n_{10} 4h^2 n_s = \frac{40h^3 n_s}{\lambda}.$$

In order to meet the $5.0 \times 10^{-4}$ error bound at $t = 10$, the ratio of the number of multiplications required by the different order methods is approximately

$$\frac{40 \times 216^3 \times 9}{40 \times 17^3 \times 21} \approx 879.1,$$

for either of the second order methods to the exact fourth order method, and

$$\frac{40 \times 17^3 \times 21}{40 \times 8^3 \times 37} \approx 5.4,$$

for the exact fourth order method to either of the sixth order methods. If the Taylor series fourth order method is used, the ratio of the number of multiplications required by the different order methods becomes approximately 160.0 for the second order methods to the fourth order method, and approximately 29.9 for the fourth order method to the sixth order methods. These comparisons show that the efficiency advantage of higher order methods is increased for increased spatial dimension, as well as for increased absolute simulation time and decreased maximum error limit. The greatest payoff by far is in the relative efficiency shown by the exact propagator fourth order method when compared to the second order methods.

## V: Numerical Algorithms For Linearized Euler Equations In Two Dimensions

The linearized Euler equations in two space dimensions can be written as

$$\begin{aligned}
&\frac{\partial u}{\partial t} + M_x \frac{\partial u}{\partial x} + M_y \frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} = 0, \\
&\frac{\partial v}{\partial t} + M_x \frac{\partial v}{\partial x} + M_y \frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} = 0, \\
&\frac{\partial p}{\partial t} + M_x \frac{\partial p}{\partial x} + M_y \frac{\partial p}{\partial y} + \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0,
\end{aligned} \tag{31}$$

where $p$ is the pressure, $(u, v)$ the disturbance velocity, and $(M_x, M_y)$ the constant mean convection velocity. This formulation is for the isentropic case, and is nondimensionalised in terms of the Mach number. The linearized Euler equations are essentially multidimensional because they cannot be diagonalized and transformed into a simpler set of decoupled equations, and wave propagation is along characteristic surfaces instead of characteristic curves.

A uniform mesh is used, with mesh sizes $h$ and $k$ in space and time, and with numerical solutions denoted by $u^n_{i,j}$, $v^n_{i,j}$, and $p^n_{i,j}$. Polynomial spatial interpolations to $u$, $v$, and $p$ are written in local coordinates around $(x_i, y_j)$ at $t_n$ with the form (13). Second, fourth and sixth order methods will use the nine, twenty one, and thirty seven point stencils in Figure 1. Interpolation coefficients are obtained by the method of undetermined coefficients, and can be interpreted in terms of spatial derivatives as in (14). Exact polynomial solution forms for the linearized Euler equations can be derived by substituting the expansion forms

$$\begin{aligned}
u(x_i + x, y_j + y, t_n + t) &\approx ua(x, y, t) = \sum_{\{\alpha,\beta\} \in AS} \sum_{\gamma=0}^{Or} u_{\alpha,\beta,\gamma} x^\alpha y^\beta t^\gamma, \\
v(x_i + x, y_j + y, t_n + t) &\approx va(x, y, t) = \sum_{\{\alpha,\beta\} \in AS} \sum_{\gamma=0}^{Or} v_{\alpha,\beta,\gamma} x^\alpha y^\beta t^\gamma, \\
p(x_i + x, y_j + y, t_n + t) &\approx pa(x, y, t) = \sum_{\{\alpha,\beta\} \in AS} \sum_{\gamma=0}^{Or} p_{\alpha,\beta,\gamma} x^\alpha y^\beta t^\gamma,
\end{aligned} \tag{32}$$

into system (31), and obtaining all the terms with $\gamma \neq 0$ by requiring system (31) to be satisfied for all $x$, $y$ and $t$. Coefficients with $\gamma \neq 0$ are equivalent to time derivatives, and the resulting polynomial solutions are expressed entirely in terms of the spatial expansion coefficients. The exact polynomial solution forms with the local spatial interpolants as initial data give exact propagator algorithms, and automatically incorporate into the solutions the correct local multidimensional wave propagation dynamics for the local spatial interpolants.

### V-A: Second Order Algorithms

The second order methods approximate $u$, $v$ and $p$ with the biquadratic spatial interpolation (18) on the $3 \times 3$ stencil. Formulas for the $u$ expansion coefficients are given in Appendix

A. The second order exact propagator method has $Or = 4$ in (32), with the solutions given in Appendix C. The exact propagator algorithm is

$$\begin{aligned}
u_{i,j}^{n+1} =& ua(0,0,k) \\
=& u_{00} \\
&+ k(-p_{10} - M_y u_{01} - M_x u_{10}) \\
&+ k^2(M_y p_{11} + 2M_x p_{20} + {M_y}^2 u_{02} + M_x M_y u_{11} + (1 + {M_x}^2)u_{20} + \frac{1}{2}v_{11}) \\
&+ k^3(-(\frac{1}{3} + {M_y}^2)p_{12} - 2M_x M_y p_{21} \\
&\quad - M_x {M_y}^2 u_{12} - (M_y + {M_x}^2 M_y)u_{21} - M_y v_{12} - M_x v_{21}) \\
&+ k^4((\frac{2}{3}M_x + 2M_x {M_y}^2)p_{22} + (\frac{1}{6} + {M_y}^2 + {M_x}^2 {M_y}^2)u_{22} + 2M_x M_y v_{22}),
\end{aligned} \tag{33a}$$

$$\begin{aligned}
v_{i,j}^{n+1} =& va(0,0,k) \\
=& v_{00} \\
&+ k(-p_{01} - M_y v_{01} - M_x v_{10}) \\
&+ k^2(2M_y p_{02} + M_x p_{11} + \frac{1}{2}u_{11} + (1 + M_y^2)v_{02} + M_x M_y v_{11} + M_x^2 v_{20}) \\
&+ k^3(-2M_x M_y p_{12} - (\frac{1}{3} + M_x^2)p_{21} \\
&\quad - M_y u_{12} - M_x u_{21} - (M_x + M_x M_y^2)v_{12} - M_x^2 M_y v_{21}) \\
&+ k^4((\frac{2}{3}M_y + 2M_x^2 M_y)p_{22} + 2M_x M_y u_{22} + (\frac{1}{6} + M_x^2 + M_x^2 M_y^2)v_{22}),
\end{aligned} \tag{33b}$$

and

$$\begin{aligned}
p_{i,j}^{n+1} =& pa(0,0,k) \\
=& p_{00} \\
&+ k(-M_y p_{01} - M_x p_{10} - u_{10} - v_{01}) \\
&+ k^2((1 + M_y^2)p_{02} + M_x M_y p_{11} + (1 + M_x^2)p_{20} \\
&\quad + M_y u_{11} + 2M_x u_{20} + 2M_y v_{02} + M_x v_{11}) \\
&+ k^3(-(M_x + M_x M_y^2)p_{12} - (M_y + M_x^2 M_y)p_{21} \\
&\quad - (\frac{1}{3} + M_y^2)u_{12} - 2M_x M_y u_{21} - 2M_x M_y v_{12} - (\frac{1}{3} + M_x^2)v_{21}) \\
&+ k^4((\frac{1}{3} + M_x^2 + M_y^2 + M_x^2 M_y^2)p_{22} \\
&\quad + (2M_x/3 + 2M_x M_y^2)u_{22} + (2M_y/3 + 2M_x^2 M_y)v_{22}).
\end{aligned} \tag{33c}$$

Notice that algorithm (33) has the form of a time expansion with terms up to $k^4$. If algorithm (33) is written in the form of a conventional explicit finite difference method, then it requires twenty seven constant coefficients for each of the three equations. The time expansion form

is easier to develop and program, and is more flexible, but the finite difference form is more computationally efficient.

The truncation errors for algorithm (33) are

$$\begin{aligned}\frac{1}{k}(u(x_i,y_j,t_{n+1})-u_{i,j}^{n+1}) = &+h^2\frac{\partial^3 u}{\partial x^3}(\frac{M_x}{6})(1-(3+M_x^2)\lambda^2)\\ &+h^2\frac{\partial^3 u}{\partial y^3}(\frac{M_y}{6})(1-M_y^2\lambda^2)\\ &+h^2\frac{\partial^3 p}{\partial x^3}(\frac{1}{6})(1-(1+3M_x^2)\lambda^2)\\ &+O[h^3],\end{aligned} \tag{34a}$$

$$\begin{aligned}\frac{1}{k}(v(x_i,y_j,t_{n+1})-v_{i,j}^{n+1}) = &+h^2\frac{\partial^3 v}{\partial x^3}(\frac{M_x}{6})(1-M_x^2\lambda^2)\\ &+h^2\frac{\partial^3 v}{\partial y^3}(\frac{M_y}{6})(1-(3+M_y^2)\lambda^2)\\ &+h^2\frac{\partial^3 p}{\partial y^3}(\frac{1}{6})(1-(1+3M_y^2)\lambda^2)\\ &+O[h^3],\end{aligned} \tag{34b}$$

$$\begin{aligned}\frac{1}{k}(p(x_i,y_j,t_{n+1})-p_{i,j}^{n+1}) = &+h^2\frac{\partial^3 p}{\partial x^3}(\frac{M_x}{6})(1-(3+M_x^2)\lambda^2)\\ &+h^2\frac{\partial^3 p}{\partial y^3}(\frac{M_y}{6})(1-(3+M_y^2)\lambda^2)\\ &+h^2\frac{\partial^3 u}{\partial x^3}(\frac{1}{6})(1-(1+3M_x^2)\lambda^2)\\ &+h^2\frac{\partial^3 v}{\partial y^3}(\frac{1}{6})(1-(1+3M_y^2)\lambda^2)\\ &+O[h^3].\end{aligned} \tag{34c}$$

Note that the truncation errors (34a) and (34c) for $u$ and $p$ are the lowest order errors of the second order method from Section III plus one additional third derivative term each. The truncation errors (34) clearly show that algorithm (33) is second order accurate and dispersive.

The second order Taylor series time expansion algorithm on the $3 \times 3$ central stencil can be obtained by taking the time terms up through $k^2$ in (33). If symmetric crossdifferencing is used to obtain $u_{11}$, $v_{11}$ and $p_{11}$, then the second order Taylor series time expansion algorithm requires the same number of floating point operations per grid point per time step as algorithm (33). With this differencing, the truncation error for $u$ from the second order Taylor series

time expansion algorithm is

$$\begin{aligned}\frac{1}{k}(u(x_i,y_j,t_{n+1})-u_{i,j}^{n+1}) =& h^2\frac{\partial^3 u}{\partial x^3}(\frac{M_x}{6})(1-(3+M_x^2)\lambda^2)\\ &+h^2\frac{\partial^3 u}{\partial y^3}(\frac{M_y}{6})(1-M_y^2)\lambda^2)\\ &+h^2\frac{\partial^3 p}{\partial x^3}(\frac{M_x}{6})(1-(1+3M_x^2)\lambda^2)\\ &+h^2\frac{\partial^3 u}{\partial x^2\partial y}(\frac{-1}{2})(1+M_x^2)M_y\lambda^2+h^2\frac{\partial^3 u}{\partial x\partial y^2}(\frac{-1}{2})M_xM_y^2\lambda^2\\ &+h^2\frac{\partial^3 v}{\partial x^2\partial y}(\frac{-1}{2})M_x\lambda^2+h^2\frac{\partial^3 v}{\partial x\partial y^2}(\frac{-1}{2})M_y\lambda^2\\ &+h^2\frac{\partial^3 p}{\partial x^2\partial y}(-1)M_xM_y\lambda^2+h^2\frac{\partial^3 p}{\partial x\partial y^2}(\frac{-1}{6})(1+3M_y^2)\lambda^2\\ &+O[h^3].\end{aligned}$$

This truncation error for $u$ from the time expansion method contains the error (34a) for $u$ from the exact propagator algorithm (33) plus an additional six cross derivative terms. The truncation errors for $v$ and $p$ have a similar complexity with cross derivative terms that are absent from (34b) or (34c). Note that algorithm (33) correctly incorporates the local wave dynamics for its spatial interpolant, but that the Taylor expansion in time does not.

### V-B: Higher Order Algorithms

For the sake of brevity, only an outline will be given of higher order methods, but with sufficient detail to ensure their specification. The local approximation of $u$, $v$ and $p$ is done for the fourth order algorithms with the twenty one point stencil and interpolation (23), and for the sixth order algorithms with the thirty seven point stencil and interpolation (27). With these approximations the fourth order exact propagator method has $Or = 6$ in (32), and the sixth order has $Or = 8$. The exact solution forms are used to obtain the numerical algorithms with $u_{i,j}^{n+1} = ua(0,0,k)$, $v_{i,j}^{n+1} = va(0,0,k)$, and $p_{i,j}^{n+1} = pa(0,0,k)$. The time expansion forms of the fourth order algorithm are given in Appendix D. Notice in Appendix D that the algorithms for $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$ are symmetric in $u$ and $v$, if the role of $x$ and $y$ are interchanged. The truncation errors for the fourth order exact propagator algorithm are

$$\begin{aligned}\frac{1}{k}(u(x_i,y_j,t_{n+1})-u_{i,j}^{n+1}) =& -\frac{h^4}{120}\frac{\partial^5 u}{\partial x^5}M_x(4-5\lambda^2(3+M_x^2)+\lambda^4(5+10M_x^2+M_x^4))\\ &-\frac{h^4}{120}\frac{\partial^5 u}{\partial y^5}M_y(4-5\lambda^2M_y^2+\lambda^4M_y^4)\\ &-\frac{h^4}{120}\frac{\partial^5 p}{\partial x^5}(4-5\lambda^2(3+M_x^2)+\lambda^4(1+10M_x^2+5M_x^4))\\ &+O[h^5],\end{aligned} \qquad (35a)$$

$$\frac{1}{k}(v(x_i,y_j,t_{n+1})-v_{i,j}^{n+1}) = -\frac{h^4}{120}\frac{\partial^5 v}{\partial y^5}M_y(4-5\lambda^2(3+M_y^2)+\lambda^4(5+10M_y^2+M_y^4))$$
$$-\frac{h^4}{120}\frac{\partial^5 v}{\partial x^5}M_x(4-5\lambda^2M_x^2+\lambda^4M_x^4)$$
$$-\frac{h^4}{120}\frac{\partial^5 p}{\partial y^5}(4-5\lambda^2(3+M_y^2)+\lambda^4(1+10M_y^2+5M_y^4)) \qquad (35b)$$
$$+O[h^5],$$

$$\frac{1}{k}(p(x_i,y_j,t_{n+1})-p_{i,j}^{n+1}) = -\frac{h^4}{120}\frac{\partial^5 p}{\partial x^5}M_x(4-5\lambda^2(3+M_x^2)+\lambda^4(5+10M_x^2+M_x^4))$$
$$-\frac{h^4}{120}\frac{\partial^5 u}{\partial x^5}(4-5\lambda^2(3+M_x^2)+\lambda^4(1+10M_x^2+5M_x^4))$$
$$-\frac{h^4}{120}\frac{\partial^5 p}{\partial y^5}M_y(4-5\lambda^2(3+M_y^2)+\lambda^4(5+10M_y^2+M_y^4)) \qquad (35c)$$
$$-\frac{h^4}{120}\frac{\partial^5 v}{\partial y^5}(4-5\lambda^2(3+M_y^2)+\lambda^4(1+10M_y^2+5M_y^4))$$
$$+O[h^5].$$

Note that the truncation errors (35a) and (35c) for $u$ and $p$ in two dimensions are the lowest order error terms of the fourth order method from Section III plus an additional fifth derivative term. The truncation errors for both methods are dispersive and completely lacking in cross derivatives, and they show that the methods are fourth and sixth order accurate in space and time, respectively.

The fourth order Taylor series time expansion algorithm can be obtained by dropping the $k^5$ and $k^6$ terms from the exact propagator algorithm in Appendix D. Similarly, the sixth order Taylor series method can be obtained by dropping the $k^7$ and $k^8$ terms from the sixth order exact propagator algorithm. The truncation errors for these methods confirm their order of accuracy in both space and time. None of the Taylor series algorithms can be interpreted as correctly incorporating the wave propagation dynamics of their interpolants, and they exhibit a plethora of cross derivative terms in their lowest order truncation errors because these derivatives from the spatial interpolation are not incorporated in the local time evolution. The number of error terms that the Taylor series algorithms add to the lowest order truncation errors from the exact propagator algorithms increases with the order of the algorithm.

### V-C: Numerical Comparisons

As a numerical test of the algorithms that are discussed in this section, consider system (31) with the initial data

$$p(x,y,0) = \mathrm{Sin}(\pi x)\mathrm{Sin}(\pi y),$$
$$u(x,y,0) = 0,$$
$$v(x,y,0) = 0,$$

for $(x,y) \in [-1,1] \times [-1,1]$, and with periodic boundaries. The exact solution is

$$p(x,y,t) = \mathrm{Cos}(\sqrt{2}\pi t)\mathrm{Sin}(\pi(x - M_x t))\mathrm{Sin}(\pi(y - M_x t)),$$
$$u(x,y,t) = \frac{-1}{\sqrt{2}}\mathrm{Sin}(\sqrt{2}\pi t)\mathrm{Cos}(\pi(x - M_x t))\mathrm{Sin}(\pi(y - M_x t)),$$
$$v(x,y,t) = \frac{1}{\sqrt{2}}\mathrm{Sin}(\sqrt{2}\pi t)\mathrm{Sin}(\pi(x - M_x t))\mathrm{Cos}(\pi(y - M_x t)),$$

for $(x,y) \in [-1,1] \times [-1,1]$ and $t \geq 0$. This problem is the direct two dimensional extension of the problem used in Section III-B for the numerical comparison of algorithms for the one dimensional linearized Euler equations, and similar data is used for the numerical tests in Section IV-D. The algorithms in Section III for the linearized Euler equations in one space dimension have the stability constraint $\lambda = \frac{k}{h} \leq \frac{1}{1+|M|}$, and in that section $\lambda = 0.8$ is used with mean convection velocity $M = 0$. Two dimensional analogs of the one dimensional stability constraint include replacing $1 + |M|$ by either $1 + (M_x^2 + M_y^2)^{\frac{1}{2}}$ or by $1 + \mathrm{Max}\{|M_x|, |M_y|\}$. For $M_x = 1 = M_y$, this becomes either $\lambda \leq 1/(1+\sqrt{2}) \approx 0.414$, or $\lambda \leq 1/(1+1) = 0.5$. With $M_x = 1 = M_y$, the exact propagator algorithms for equation (31) are stable with $\lambda = 0.5$, the second order Taylor series algorithm is not stable with $\lambda = 0.4$, but it is stable with $\lambda = 0.25$, and the fourth and sixth order Taylor series algorithms are not stable with $\lambda = 0.5$, but they are stable with $\lambda = 0.4$. The computations in this section will be on a sequence of grid sizes out to the fixed time $t = 10$, generally with $\lambda = \frac{k}{h} = 0.4$, except with $\lambda = 0.2$ for the second order Taylor series algorithm. Numerical data for this problem is presented in Table E.

Table E: Data For The Linearized Euler Equations In 2D
$p(x,y,0) = \mathrm{Sin}(\pi x)\mathrm{Sin}(\pi y)$, $u(x,y,0) = 0 = v(x,y,0)$, $\lambda = 0.4, M_x = 1 = M_y$
Maximum Error In $p$ at $t = 10$

| $\frac{2}{h}$ | $n_{10}$ | $Or = 2$ $EX_9$ | $Or = 2$ $TS_9$* | $Or = 4$ $EX_{21}$ | $Or = 4$ $TS_{21}$ | $Or = 6$ $EX_{37}$ | $Or = 6$ $TS_{37}$ |
|---|---|---|---|---|---|---|---|
| 4 | 50 | 0.885 | 0.851 | 0.931 | 0.917 | 0.735 | 0.763 |
| 8 | 100 | 0.888 | 1.054 | 0.168 | 0.262 | 2.2D-02 | 1.7D-02 |
| 16 | 200 | 0.344 | 0.524 | 9.9D-03 | 2.5D-02 | 3.0D-04 | 1.9D-04 |
| 32 | 400 | 0.102 | 0.145 | 6.9D-04 | 1.7D-03 | 5.3D-06 | 3.4D-06 |
| 64 | 800 | 2.7D-02 | 4.0D-02 | 4.5D-05 | 1.1D-04 | 8.8D-08 | 5.9D-08 |
| 128 | 1600 | 6.9D-03 | 1.0D-02 | 2.9D-06 | 6.8D-06 | 1.4D-09 | 9.7D-10 |
| 256 | 3200 | 1.7D-03 | 2.6D-03 | 1.8D-07 | 4.3D-07 | 2.3D-11 | 5.1D-12 |
| 512 | 6400 | 4.4D-04 | 6.6D-04 | 1.2D-08 | 2.7D-08 | 1.6D-11 | 5.0D-12 |

* Note that $\lambda = 0.2$ for the data in this column.

In Table E, $\frac{2}{h}$ is the number of grid points in the interval $[-1,1]$ or per wavelength, $n_{10}$ is the number of time steps required to compute to $t = 10$ at a given grid resolution with $\lambda = \frac{4}{10}$, and the data is the maximum absolute error in $p$ over the entire grid at $t = 10$. In the column headings of Table E, $Or$ is the order of the method, and $EX_m$ and $TS_m$ represent the

exact propagator and Taylor series algorithms on the $m$ point stencil. Calculations of error reduction exponents similar to those conducted for the data in Table A confirm the order of accuracy for each of the methods used to produce Table E. The error levels in Table E are similar to those in Table D when compared by method and by grid size. The large error for the second order Taylor series algorithm at $\frac{2}{h} = 8$ is due to dispersion errors from poorly resolved data, just as for similar cases among the data in Tables A and C. If these methods are compared by order, then the exact propagator and Taylor series methods are similar both in the computational effort that they require, and in the errors that they produce, but the exact propagator methods have less severe stability constraints. For these computations the total number of multiplications can be calculated as in Section IV, with

$$\text{total multiplications} = n_{10} 4h^2 n_s n_e n_v = \frac{360 h^3 n_s}{\lambda},$$

where this problem involves $n_e = 3$ equations using data from $n_v = 3$ variables in each equation. Interpolation of the data from the exact propagator methods in Table E implies that in order to achieve a maximum allowable error of $5.0 \times 10^{-4}$ at $t = 10$, a grid resolution of $\frac{1}{h} = 248$ is required for the second order method, $\frac{1}{h} = 20$ for the fourth order method, and $\frac{1}{h} = 8$ for the sixth order method. The ratio of the total number of multiplictions required to meet this error constraint at this time is approximately 817.1 for the second to the fourth order methods, and 8.9 for the fourth to the sixth order methods. These results are similar to the comparisons obtained in Section IV, showing greater efficiency for higher order methods.

### V-D: Numerical Computations With Boundaries

Algorithms require boundary conditions in order to be useful. This subsection presents the results of computations with both real and artifical outflow boundaries using the fourth order exact propagator algorithm. Details on boundary treatments are given in [15] and [12]. Consider the linearized Euler equations (31) with the initial data

$$p(x, y, 0) = \exp[-ln(2)(\frac{x^2 + (y-25)^2}{25})],$$
$$u(x, y, 0) = 0 = v(x, y, 0),$$

with $M_x = 0.5$ and $M_y = 0$, and on the computational domain $(x, y) \in [-100, 100] \times [0, 200]$, where there is a wall at $y = 0$, a computational inflow boundary at $x = -100$, a computational outflow boundary at $x = +100$, and a far field boundary at $y = 200$. This problem represents a Gaussian pressure pulse near a wall in a Mach 0.5 flow parallel to the wall. Calculations will be done with $h = 1$ and $k = 0.25$ out to $t = 150$.

The wall boundary conditions are

$$v = 0,$$
$$\frac{\partial p}{\partial y} = 0.$$

The normal derivative of $p$ is evaluated on the wall with fourth order interior differencing on a one sided five point stencil. In a wall boundary cell, the fourth order method is used to produces values for the three variables at the cell center and at one intermediate point between the cell center and the wall, and for $u$ on the wall. All of these solution values are found in essentially the same way as for the one dimensional equations in Section III-C, using the algorithm in the form of a Cauchy-Kowaleskaya expanion in space and time, with expansion coefficients that are obtained from one interpolation on the boundary cell. The form of the algorithm as an expansion in space and time is evaluated at $(0, 0, k)$ to obtain the three solution values for the cell center, at $(0, -h, k)$ for the intermediate point values, and at $(0, -2h, k)$ for the value of $u$ on the wall.

New outflow conditions that have been derived by Hagstrom [15] are used at the outflow boundary $x = +100$. If system (31) is diagonalized in the $x$ direction, or normal to the boundary, then the primitive variables $p$, $u$ and $v$ can be replaced with

$$\begin{aligned} r_1 &= u - p, \\ r_2 &= v, \\ r_3 &= u + p. \end{aligned}$$

For subsonic flows, $r_1$ is conventionally viewed as coming into the computational domain from the $+x$ direction, and both $r_2$ and $r_3$ as going out. The variable $r_1$ is obtained on the outflow boundary as the solution of the system

$$\begin{aligned} &\frac{\partial r_1}{\partial t} - M_x \frac{\partial v}{\partial y} - (f_1 + f_2 + g_1 + g_2) = 0, \\ &\frac{\partial f_k}{\partial t} - \alpha_k \sqrt{(1 - {M_x}^2)} \frac{\partial f_k}{\partial y} = -\frac{\beta_k}{2}(1 - {M_x}^2) \frac{\partial^2 p}{\partial {y}^2}, \\ &\frac{\partial g_k}{\partial t} + \alpha_k \sqrt{(1 - {M_x}^2)} \frac{\partial g_k}{\partial y} = -\frac{\beta_k}{2}(1 - {M_x}^2) \frac{\partial^2 p}{\partial {y}^2}, \end{aligned}$$

for $k = 2$, where $\alpha_1 = \mathrm{Cos}(\frac{\pi}{5})$, $\alpha_2 = \mathrm{Cos}(\frac{2\pi}{5})$, $\beta_1 = \frac{2}{5}\mathrm{Sin}(\frac{\pi}{5})$, and $\beta_2 = \frac{2}{5}\mathrm{Sin}(\frac{2\pi}{5})$. Notice that this boundary condition requires no geometric information about the disturbance, neither globally nor locally. Details are given in [15]. The algorithmic implementation of this boundary condition is similar to the algorithm for the wall conditions. In an outflow boundary cell, the space and time expansion is evaluated at $(0, 0, k)$ for the three solution values that are needed at the cell center, at $(h, 0, k)$ for the three values at the intermediate point, and in combination at $(2h, 0, k)$ for the values of $r_2$ and $r_3$ that are needed on the outflow. Separate expansion forms are developed on the boundary for $r_1$, $f_1$, $f_2$, $g_1$, and $g_2$, using the expansion forms for $p$ and $v$. These expansion forms use a stencil of width five, and are designed to be fourth order accurate in both space and time. The system for $r_1$ and the $f_k$ and $g_k$ is one dimensional on the boundary line, forced by the evolution of $p$ and $v$. The values of $r_1$ and the $f_k$ and $g_k$ are all initialized at 0. At the intersection of the wall $y = 0$ and the outflow $x = +100$, $f_k$ is obtained by solving its PDE using interior differencing, while $g_k$ is obtained from

$$\frac{\partial g_k}{\partial y} + \frac{\partial f_k}{\partial y} = 0, \tag{36}$$

for $k = 1$ and 2. At the far field $y = +200$ on the outflow $x = +100$, $f_k = 0$ and $g_k$ is obtained by solving its PDE using interior differencing, for $k = 1$ and 2. Details are given in [12].

Characteristic boundary conditions are used on the inflow boundary at $x = -100$, and on the far field boundary at $y = +200$, with outgoing Riemann variables solved using interior differencing and the appropriate momentum equation, and incoming Riemann variables set equal to 0. For these two boundaries, the Riemann variables are determined by a one dimensional diagonalization of the system in the direction normal to the boundary. The propagation algorithm, the wall conditions, and the outflow boundary algorithm are all implemented as fourth order methods, in both space and time, with central stencils having a stencil width of five grid points, unless they are offset by interior differencing at a boundary. The inflow and far field boundary algorithms are implemented with a second order method using a stencil with a three point width, since nothing ever happens at these two boundaries for $t \leq 150$.

The results for this wall pulse problem are presented in Figure 2, which shows pressure contours at $t = 15$, 45, 75, and 150. These results have been briefly described in [11]. In Figure 2a at $t = 15$ the expanding pressure wave just begins to touch the wall. In Figure 2b at $t = 45$ there is a very evident reflecting wave expanding behind the expansion front, with interferance patterns where they interact near the wall. The expanding wave reaches the artifical outflow boundry at about $t = 60$, and shows no disturbance at this moment of first contact. In Figure 2c at $t = 75$ the two waves have already passed through the artifical outflow, and have progressed from being parallel to the outflow boundary at their first contact, to being at approximately $45^o$ with the boundary. Notice that there are no evident disturbances in the wave front contours near the boundary at $t = 75$, even though the expanding waves have just passed through the intersection between the wall and the outflow. This type of intersection can create a transient error that is propagated with the waves as a reflection trailing backwards from the intersection between the solution and the boundary. In Figure 2d at $t = 150$ the two wave fronts have become nearly perpendicular to the outflow boundary, and the pulse center has convected to $(x, y) = (75, 25)$. The solution downstream from the artificial outflow boundary at $x = 100$ continuously effects the analytic solution for $x \leq 100$ over the time interval $60 \leq t \leq 150$ during which there is significant downstream solution structure. The plot sequence shows a correct evolution on the computational domain, with solutions that are symmetric in $x$ and have round wavefronts, with no visible perturbation of the solution right up to the boundary, and with no evident reflection from the boundary or the wall, either as a transient or cumulative effect. This data reflects both the accuracy with which the propagation algorithm simulates the fully multidimensional wave dynamics of the linearized Euler equations, and the unobtrusive quality of the artifical outflow boundary condition.

It could be argued that the structure of the wall pulse solution is so simple near and downstream from the outflow boundary that the quality of the outflow boundary is not seriously tested. A more severe test of the algorithm and outflow condition is given by placing the initial Gaussian pressure pulse in a duct. Consider the same initial data for the Gaussian pressure pulse but on the computational domain $(x, y) \in [-100, 100] \times [0, 50]$, with two walls at $y = 0$ and $y = 50$, with an inflow at $x = -100$, and an outflow at $x = +100$. The fourth order exact propagator algorithm is used, with $h = 1$ and $k = 0.25$ on a uniform grid. The only change in the outflow boundary is that at the intersection of the wall $y = 50$ and the

outflow $x = +100$, $g_k$ is obtained by solving its PDE using interior differencing, while $f_k$ is obtained from (36) for $k = 1$ and 2. The results for this duct pulse problem are presented in Figure 3, which shows pressure contours at $t = 15$, 45, 75, and 150. Notice that the wave front hits the two walls and is reflected back and forth several times. Each time this happens, a more complex structure of interference patterns is visible in the pressure contours. The same story is visible in these calculations as from the wall pulse problem, but now the pressure structures are very much more complex, both as they are successfully passed through the numerical boundary, and as they continue to effect the evolution of the pressure structures within the numerical domain because of their virtual representation by means of the boundary condition. In Figure 3a at $t = 15$ the expanding pressure wavefront begins to touch both walls. In Figure 3b at $t = 45$ there are two reflecting wavefronts that have almost reached each other in the center of the duct. In Figure 3c at $t = 75$ the two reflecting wavefronts have passed through each other while propagating completely across the duct, and have been reflected once again to pass through each other a second time and reach the wall from which they were originally reflected. The complex interference pattern at the upstream end of the disturbance has a rich structure created by the crossing and recrossing of the reflecting waves. Notice at $t = 75$ that approximately half of the complex structure at the downstream end of the disturbance has already passed out of the computational domain. In Figure 3d at $t = 150$ the two wavefronts have gone through further reflections and interactions, creating a very complex upstream interference pattern. Notice that the entire interference pattern is missing from the downstream end of this plot, since it has passed entirely out from the computational domain. If a transperancy is made of this plot, it can be turned over and the contour lines in the center will allign perfectly, with the transparency contours near the center of the domain exactly overlaying the plot contours near the boundary. This symmetry in the plot shows that even though nearly half of the solution structure is outside of the compuational domain, there is no visible effect on the solution inside the domain. The lack of discernable effect from the outflow boundary occurs in spite of the fact that the propagating wavefronts in the computational domain are virtually perpendicular to the outflow boundary. This computation shows the unobtrusive and robust quality of the artificial outflow boundary condition [15].

## VI: Summary And Discussion

This paper presents two techniques for developing algorithms for hyperbolic systems in multiple space dimensions, the use of local exact polynomial solutions, or exact propagators, and the use of multivariate Cauchy-Kowaleskaya expansions, or truncated Taylor series. Both techniques of algorithm development can be viewed as approximating the solution of a hyperbolic system as a whole, rather than individual derivative terms in separate equations. The multivariate polynomial solution approximations that are used by these techniques include various cross derivative terms, which are needed for high order accuracy, and improved isotropy and stability. Both of these two techniques are used to develop algorithms for the linear convection equation and the linearized Euler equations, in one and two space dimensions. The explicit single step algorithms developed by these two methods use symmetric stencils and are dispersive, they use data from only one time level, they have the same order of accuracy in both space and time, and they can be extended to arbitrarily high orders of accuracy. Algorithm examples are given with up to eighth order accuracy in one space dimension, and sixth in two dimensions. For each order of method, and for each type of algorithm, the choice of large or small symmetric stencil in two space dimensions does not appear to be significant for accuracy. The higher order methods are more efficient in terms of the total number of floating point operations required to compute to a fixed simulation time with a stated error bound, and they can require less operations than a low order method by factors of up to several orders of magnitude. The relative efficiency of the higher order methods increases with either the simulation time or the dimension of the system domain, and with decreases in the error bound. The exact propagator methods appear to be at least comparable in accuracy to the Taylor series methods, if not more accurate, and they appear to be more robustly stable, without requiring more operations. Exact propagator algorithms incorporate the correct local multidimensional wave propagation dynamics for their polynomial spatial interpolants, and they can be viewed as a generalization of the method of characteristics to nondiagonalizable hyperbolic systems in multiple space dimensions. In particular, exact local polynomial solutions are shown for the linearized Euler equations in two space dimensions. Stable high order boundary conditions are possible using a consistant calculation of the correct wave dynamics of the solution simultaneously from the center to the exterior edge of a grid cell on the boundary. High order boundary conditions are developed for the linearized Euler equations in one space dimension, and demonstrated in two.

There are several evident extensions, and issues that are not addressed in this paper. Analysis of the algorithms is being done, particularly for stability, with an investigation of the different stability constraints for the exact propagator and Taylor series methods in two space dimensions. The form of the algorithms are independent from the spatial interpolation that is used, so that various methods can be used for estimating the spatial expansion coefficients. In particular, an implementation on a nonuniform or unstructured grid is being prepared with an appropriate spatial interpolation, and it should not incur any degredation in the order of accuracy. Implementation in other coordinate systems is being explored. Complete details of the outflow boundary algorithm are being prepared for publication [12]. In particular, compatibility conditions for the juncture of two artifical boundaries are being developed, and

implementations for an artifical inflow are being prepared. These two approaches to algorithm development are being extended to other first order linear systems, such as for Maxwell's equations. The Taylor series method is being extended to variable coefficient cases.

## Acknowledgments

The development of the algorithms in this paper was done with the help of Mathematica, the calculations were all done on an IBM RS6000 model 560, and this paper was prepared with AMS TeX on an IBM PS/2 model 80.

I would like to thank H. T. Huynh for getting me interested in hyperbolic systems, Eli Turkel for pointing me toward central methods, and Reda Mankbadi and Jay Hardin for getting me interested in acoustics. I would like to thank Steve Zalesak and Phil Roe for encouragement and discussions. I would like to thank Tom Hagstrom for a fine outflow boundary condition.

## Bibliography

[1] A. Bayliss, K. E. Jordan, B. J. LeMesurier and E. Turkel, "A Fourth-Order Accurate Finite-Difference Scheme For The Computation Of Elastic Waves," *Bull, Seism. Soc. Am.*, **76**, 1115 (1986).

[2] J. F. Botha and G. F. Pinder, *Fundamental Concepts in the Numerical Solution of Differential Equations*, (Wiley, New York, 1983).

[3] S. Z. Burstein and A. A. Mirin, "Third Order Difference Methods for Hyperbolic Equations," *J. Comput. Phys.*, **5**, 547 (1970).

[4] C-J Chen, H. Naseri-Neshat and P. Li, "The Finite Analytic Method," Iowa Institute of Hydrolic Research Report No 232, vol 1-4, The University of Iowa, (1980).

[5] M. Ciment and S. H. Leventhal, "High Order Compact Implicit Schemes for the Wave Equation," *Math. Comp.*, **29**, 985 (1975).

[6] G. Cohen and P. Joly, "Fourth Order Schemes For The Heterogeneous Acoustics Equation," *Compt. Meth. Appl. Math. Eng.*, **80**, 397 (1990).

[7] J. B. Cole, "A nearly exact second-order finite-difference time-domain wave propagation algorithm on a coarse grid," *Compt. in Phys.*, **8**, 730 (1994).

[8] L. Collatz, *The Numerical Treatment of Differential Equations*, (Springer-Verlag, New York, 1966).

[9] P. Garabedian, *Partial Differential Equations*, (Wiley, New York, 1964).

[10] S. K. Godunov, "Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equation of Fluid Dynamics," *Mat. Sbornik.*, **47**, 271 (1959).

[11] J. W. Goodrich, "Application of a New Finite Difference Algorithm for Computational Aeroacoustics," to appear in the proceedings of *The ICASE/LaRC Workshop on Computationl Aeroacoustics*, (Hampton, Va., October 24-26, 1994).

[12] J. W. Goodrich, and T. Hagstrom, in preparation.

[13] D. Gottlieb and E. Turkel, "Dissipative Two-Four Methods for Time-Dependent Problems," *Math. Comp.*, **30**, 703 (1976).

[14] B. Gustafsson and P. Olsson, "Fourth Order Difference Methods for Hyperbolic IBVP's," RIACS Tech. Report 94-04, (March, 1994).

[15] T. Hagstrom, "On High Order Radiation Boundary Conditions," to appear in the proceedings of *The IMA Workshop on Computational Wave Propagation*, (Minneapolis, Minn., September, 1994).

[16] J. Hardin, "Numerical Considerations For Computational Aeroacoustics," to appear in "Computational Aeroacoustics," edited by J. Hardin and J. Hussaini, (Springer Verlag, New York, 1993).

[17] A. Harten, B, Engquist, S. Osher and S. R. Chakravarthy, "Uniformly High Order Accurate Essentially Non-oscillatory Schemes, III," *J. Comput. Phys.*, **71**, 231 (1987).

[18] H. O. Kreiss and J. Lorenz, *Initial Boundary Value Problems and the Navier Stokes Equations*, (Academic Press, New York, 1989).

[19] H. O. Kreiss and J. Oliger, "Comparison of accurate methods for the integration of hyperbolic equations," *Tellus*, **24**, 199 (1972).

[20] H. O. Kreiss and J. Oliger, "Methods for the Approximate Solution of Time Dependent Problems," GARP Publications Series No. 10, (February, 1973).

[21] P. D. Lax and B. Wendroff, "Systems of Conservation Laws," *Comm. Pure Appl. Math*, **13**, 217 (1960).

[22] S. K. Lele, "Compact Finite Difference Schemes with Spectral like Resolution", *J. Comput. Phys.* **103**, pp16-42 (1992).

[23] B. P. Leonard, M. K. MacVean and A. P. Lock, "Positivity-Preserving Numerical Schemes for Multidimensional Advection," NASA TM 106055, ICOMP Report No. ICOMP-93-05, (1993).

[24] J. Lighthill, "A General Introduction To Aeroacoustics And Atmospheric Sound," NASA Contractor Report 189717, ICASE Report No. 92-52 (October 1992).

[25] J. Lighthill, "Report On The Final Panel Discussion On Computational Aeroacoustics," NASA Contractor Report 189718, ICASE Report No. 92-53 (October 1992).

[26] Yen Liu, AIAA-93-0368, presented at *The 31th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 11-14 January, 1993.

[27] D. P. Lockard, K. S. Brentner, and H. L. Atkins, AIAA-94-0460, presented at *The 32th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 10-13 January, 1994.

[28] R. W. MacCormack, "Numerical Solution of the Interaction of a Shock Wave with a Laminar Boundary Layer," Proc. 2nd Internat. Conf. on Numerical Methods in Fluid Dynamics, *Lec. Notes in Phys.*, **8** M. Holt, Editor, (Springer-Verlag, New York, 1974).

[29] S. M. Orszag and M. Israeli, "Numerical Simulation of Viscous Incompressible Flows," *Ann. Rev. Fluid Mech.*, **5**, (1974).

[30] P. Roache, *Computational Fluid Dynamics*, (Hermosa Publishers, Albuquerque, 1976).

[31] V. V. Rusanov, "Difference Schemes of the Third Order Accuracy for Continuous Computation of Discontinuous Solutions," *Sov. Math. Dokl.*, **9**, (1968).

[32] J. N. Scott, AIAA-92-0506, presented at *The 30th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 6-9 January, 1992.

[33] J. M. Seiner and E. A. Krejsa, AIAA-89-2358, presented at *The AIAA / ASME / SAE / ASEE 25th Joint Propulsion Conference*, Monterey, CA, 10-12 July, 1989.

[34] M. J. Smith and R. W. Stoker, AIAA-93-0150, presented at *The 31th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 11-14 January, 1993.

[35] M. J. T. Smith, *Aircraft Noise*, (Cambridge University Press, Cambridge, 1989).

[36] G. A. Sod, *Numerical Methods In Fluid Dynamics: Initial And Initial Boundary-Value Problems*, (Cambridge University Press, Cambridge, 1985).

[37] B. Swartz and B. Wendroff, "The relative efficiency of finite difference methods. I: Hyperbolic problems and solines," *SIAM J. Numer. Anal.*, **11**, 979 (1974).

[38] C. K. W. Tam and J. C. Webb, "Dispersion Relation Preserving Finite Difference Schemes For Computational Acoustics," to appear in *J. Comput. Phys.*.

[39] G. B. Whitham, *Linear and Nonlinear Waves*, (Wiley, New York, 1974).

[40] S. T. Yu, AIAA-93-0148, presented at *The 31th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 11-14 January, 1993.

[41] E. Zauderer, *Partial Differential Equations of Applied Mathematics*, (Wiley, New York, 1983).

[42] D. W. Zingg, E. M. Epstein, and H. M. Jurgens, AIAA-95-0162, presented at *The 33th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 9-12 January, 1995.

[43] D. W. Zingg, H. Lomax, and H. M. Jurgens, AIAA-93-0459, presented at *The 31th Aerospace Sciences Meeting and Exhibit*, Reno, NV, 11-14 January, 1993.

## Appendix A: Biquadratic Expansion Coefficients For A $3 \times 3$ Stencil

The spatial approximation to $u$ around $(x_i, y_j)$ at $t_n$ is

$$u(x_i + x, y_j + y, t_n) \approx ua(x, y) = \sum_{\alpha,\beta=0}^{2} u_{\alpha,\beta} x^\alpha y^\beta,$$

with difference coefficients

$$\begin{aligned}
u_{00} &= u^n_{i,j}, \\
u_{10} &= \frac{1}{2h}(u^n_{i+1,j} - u^n_{i-1,j}), \\
u_{20} &= \frac{1}{2h^2}(u^n_{i+1,j} - 2u^n_{i,j} + u^n_{i-1,j}), \\
u_{01} &= \frac{1}{2h}(u^n_{i,j+1} - u^n_{i,j-1}), \\
u_{11} &= \frac{1}{4h^2}(u^n_{i+1,j+1} - u^n_{i+1,j-1} - u^n_{i-1,j+1} + u^n_{i-1,j-1}), \\
u_{21} &= \frac{1}{4h^3}(u^n_{i+1,j+1} - 2u^n_{i,j+1} + u^n_{i-1,j+1} - u^n_{i+1,j-1} + 2u^n_{i,j-1} - u^n_{i-1,j-1}), \\
u_{02} &= \frac{1}{2h^2}(u^n_{i,j+1} - 2u^n_{i,j} + u^n_{i,j-1}), \\
u_{12} &= \frac{1}{4h^3}(u^n_{i+1,j+1} - 2u^n_{i+1,j} + u^n_{i+1,j-1} - u^n_{i-1,j+1} + 2u^n_{i-1,j} - u^n_{i-1,j-1}), \\
u_{22} &= \frac{1}{4h^4}(u^n_{i+1,j+1} - 2u^n_{i+1,j} + u^n_{i+1,j-1} - 2u^n_{i,j+1} + 4u^n_{i,j} - 2u^n_{i,j-1} \\
&\quad + u^n_{i-1,j+1} - 2u^n_{i-1,j} + u^n_{i-1,j-1}).
\end{aligned}$$

## Appendix B: Expansion Coefficients For A Modified Biquartic On A Twenty One Point Stencil

The spatial approximation to $u$ around $(x_i, y_j)$ at $t_n$ is

$$\begin{aligned} ua(x,y) =& u_{00} + u_{10}x + u_{20}x^2 + u_{30}x^3 + u_{40}x^4 \\ &+(u_{01} + u_{11}x + u_{21}x^2 + u_{31}x^3 + u_{41}x^4)y \\ &+(u_{02} + u_{12}x + u_{22}x^2 + u_{32}x^3 + u_{42}x^4)y^2 \\ &+(u_{03} + u_{13}x + u_{23}x^2)y^3 \\ &+(u_{04} + u_{14}x + u_{24}x^2)y^4 \\ &= \sum_{\{\alpha,\beta\}\in A_{21}} u_{\alpha,\beta}x^\alpha y^\beta, \end{aligned}$$

with difference coefficients

$$\begin{aligned} u_{00} &= u^n_{i,j}, \\ u_{10} &= \frac{1}{12h}(u^n_{i-2,j} - 8u^n_{i-1,j} + 8u^n_{i+1,j} - u^n_{i+2,j}), \\ u_{20} &= \frac{1}{24h^2}(-u^n_{i-2,j} + 16u^n_{i-1,j} - 30u^n_{i,j} + 16u^n_{i+1,j} - u^n_{i+2,j}), \\ u_{30} &= \frac{1}{12h^3}(-u^n_{i-2,j} + 2u^n_{i-1,j} - 2u^n_{i+1,j} + u^n_{i+2,j}), \\ u_{40} &= \frac{1}{24h^4}(u^n_{i-2,j} - 4u^n_{i-1,j} + 6u^n_{i,j} - 4u^n_{i+1,j} + u^n_{i+2,j}), \\ u_{01} &= \frac{1}{12h}(u^n_{i,j-2} - 8u^n_{i,j-1} + 8u^n_{i,j+1} - u^n_{i,j+2}), \\ u_{11} &= \frac{1}{24h^2}(-u^n_{i-2,j-1} + u^n_{i-2,j+1} \\ &\quad - u^n_{i-1,j-2} + 10u^n_{i-1,j-1} - 10u^n_{i-1,j+1} + u^n_{i-1,j+2} \\ &\quad + u^n_{i+1,j-2} - 10u^n_{i+1,j-1} + 10u^n_{i+1,j+1} - u^n_{i+1,j+2} \\ &\quad + u^n_{i+2,j-1} - u^n_{i+2,j+1}), \\ u_{21} &= \frac{1}{48h^3}(u^n_{i-2,j-1} - u^n_{i-2,j+1} \\ &\quad + 2u^n_{i-1,j-2} - 20u^n_{i-1,j-1} + 20u^n_{i-1,j+1} - 2u^n_{i-1,j+2} \\ &\quad - 4u^n_{i,j-2} + 38u^n_{i,j-1} - 38u^n_{i,j+1} + 4u^n_{i,j+2} \\ &\quad + 2u^n_{i+1,j-2} - 20u^n_{i+1,j-1} + 20u^n_{i+1,j+1} - 2u^n_{i+1,j+2} \\ &\quad + u^n_{i+2,j-1} - u^n_{i+2,j+1}), \end{aligned}$$

$$
\begin{aligned}
u_{31} &= \frac{1}{24h^4}(u^n_{i-2,j-1} - u^n_{i-2,j+1} - 2u^n_{i-1,j-1} + 2u^n_{i-1,j+1} \\
&\quad + 2u^n_{i+1,j-1} - 2u^n_{i+1,j+1} - u^n_{i+2,j-1} + u^n_{i+2,j+1}), \\
u_{41} &= \frac{1}{48h^5}(-u^n_{i-2,j-1} + u^n_{i-2,j+1} + 4u^n_{i-1,j-1} - 4u^n_{i-1,j+1} \\
&\quad - 6u^n_{i,j-1} + 6u^n_{i,j+1} \\
&\quad + 4u^n_{i+1,j-1} - 4u^n_{i+1,j+1} - u^n_{i+2,j-1} + u^n_{i+2,j+1}), \\
u_{02} &= \frac{1}{24h^2}(-u^n_{i,j-2} + 16u^n_{i,j-1} - 30u^n_{i,j} + 16u^n_{i,j+1} - u^n_{i,j+2}), \\
u_{12} &= \frac{1}{48h^3}(2u^n_{i-2,j-1} - 4u^n_{i-2,j} + 2u^n_{i-2,j+1} \\
&\quad + u^n_{i-1,j-2} - 20u^n_{i-1,j-1} + 38u^n_{i-1,j} - 20u^n_{i-1,j+1} + u^n_{i-1,j+2} \\
&\quad - u^n_{i+1,j-2} + 20u^n_{i+1,j-1} - 38u^n_{i+1,j} + 20u^n_{i+1,j+1} - u^n_{i+1,j+2} \\
&\quad - 2u^n_{i+2,j-1} + 4u^n_{i+2,j} - 2u^n_{i+2,j+1}), \\
u_{22} &= \frac{1}{48h^4}(-u^n_{i-2,j-1} + 2u^n_{i-2,j} - u^n_{i-2,j+1} \\
&\quad - u^n_{i-1,j-2} + 20u^n_{i-1,j-1} - 38u^n_{i-1,j} + 20u^n_{i-1,j+1} - u^n_{i-1,j+2} \\
&\quad + 2u^n_{i,j-2} - 38u^n_{i,j-1} + 72u^n_{i,j} - 38u^n_{i,j+1} + 2u^n_{i,j+2} \\
&\quad - u^n_{i+1,j-2} + 20u^n_{i+1,j-1} - 38u^n_{i+1,j} + 20u^n_{i+1,j+1} - u^n_{i+1,j+2} \\
&\quad - u^n_{i+2,j-1} + 2u^n_{i+2,j} - u^n_{i+2,j+1}), \\
u_{32} &= \frac{1}{24h^5}(-u^n_{i-2,j-1} + 2u^n_{i-2,j} - u^n_{i-2,j+1} \\
&\quad + 2u^n_{i-1,j-1} - 4u^n_{i-1,j} + 2u^n_{i-1,j+1} \\
&\quad - 2u^n_{i+1,j-1} + 4u^n_{i+1,j} - 2u^n_{i+1,j+1} \\
&\quad + u^n_{i+2,j-1} - 2u^n_{i+2,j} + u^n_{i+2,j+1}), \\
u_{42} &= \frac{1}{48h^6}(u^n_{i-2,j-1} - 2u^n_{i-2,j} + u^n_{i-2,j+1} \\
&\quad - 4u^n_{i-1,j-1} + 8u^n_{i-1,j} - 4u^n_{i-1,j+1} \\
&\quad + 6u^n_{i,j-1} - 12u^n_{i,j} + 6u^n_{i,j+1} \\
&\quad - 4u^n_{i+1,j-1} + 8u^n_{i+1,j} - 4u^n_{i+1,j+1} \\
&\quad + u^n_{i+2,j-1} - 2u^n_{i+2,j} + u^n_{i+2,j+1}), \\
u_{03} &= \frac{1}{12h^3}(-u^n_{i,j-2} + 2u^n_{i,j-1} - 2u^n_{i,j+1} + u^n_{i,j+2}), \\
u_{13} &= \frac{1}{24h^4}(u^n_{i-1,j-2} - 2u^n_{i-1,j-1} + 2u^n_{i-1,j+1} - u^n_{i-1,j+2} \\
&\quad - u^n_{i+1,j-2} + 2u^n_{i+1,j-1} - 2u^n_{i+1,j+1} + u^n_{i+1,j+2}),
\end{aligned}
$$

$$
\begin{aligned}
u_{23} = \frac{1}{24h^5} - ((&u^n_{i-1,j-2} - 2u^n_{i-1,j-1} + 2u^n_{i-1,j+1} - u^n_{i-1,j+2} \\
&- 2u^n_{i,j-2} + 4u^n_{i,j-1} - 4u^n_{i,j+1} + 2u^n_{i,j+2} \\
&+ u^n_{i+1,j-2} - 2u^n_{i+1,j-1} + 2u^n_{i+1,j+1} - u^n_{i+1,j+2}), \\
u_{04} = \frac{1}{24h^4}(&u^n_{i,j-2} - 4u^n_{i,j-1} + 6u^n_{i,j} - 4u^n_{i,j+1} + u^n_{i,j+2}), \\
u_{14} = \frac{1}{48h^5} - ((&u^n_{i-1,j-2} - 4u^n_{i-1,j-1} + 6u^n_{i-1,j} - 4u^n_{i-1,j+1} + u^n_{i-1,j+2} \\
&- u^n_{i+1,j-2} + 4u^n_{i+1,j-1} - 6u^n_{i+1,j} + 4u^n_{i+1,j+1} - u^n_{i+1,j+2}), \\
u_{24} = \frac{1}{48h^6}(&u^n_{i-1,j-2} - 4u^n_{i-1,j-1} + 6u^n_{i-1,j} - 4u^n_{i-1,j+1} + u^n_{i-1,j+2} \\
&- 2u^n_{i,j-2} + 8u^n_{i,j-1} - 12u^n_{i,j} + 8u^n_{i,j+1} - 2u^n_{i,j+2} \\
&+ u^n_{i+1,j-2} - 4u^n_{i+1,j-1} + 6u^n_{i+1,j} - 4u^n_{i+1,j+1} + u^n_{i+1,j+2}).
\end{aligned}
$$

## Appendix C: Exact Second Order Polynomial Solution For Equation (31)

The local exact solution to the linearized Euler equations that uses a biquadratic spatial expansion for initial data is given for $u$ by

$$\begin{aligned}
ua(x,y,t) = & \; u_{00} \\
& + u_{10}(x - M_x t) + u_{01}(y - M_y t) \\
& + u_{20}((x - M_x t)^2 + t^2) + u_{11}(x - M_x t)(y - M_y t) + u_{02}(y - M_y t)^2 \\
& + u_{21}((x - M_x t)^2 + t^2)(y - M_y t) + u_{12}(x - M_x t)(y - M_y t)^2 \\
& + u_{22}(((x - M_x t)^2 + t^2)(y - M_y t)^2 + t^4/6) \\
& + v_{11}(t^2/2) \\
& + v_{21}(x - M_x t)t^2 + v_{12}(y - M_y t)t^2 \\
& + v_{22}(2(x - M_x t)(y - M_y t)t^2) \\
& + p_{10}(-t) \\
& + p_{20}(-2(x - M_x t)t) + p_{11}(-(y - M_y t)t) \\
& + p_{21}(-2(x - M_x t)(y - M_y t)t) + p_{12}(-(y - M_y t)^2 t - t^3/3) \\
& + p_{22}(-2(x - M_x t)((y - M_y t)^2 + t^2/3)t),
\end{aligned}$$

for $v$ by

$$\begin{aligned}
va(x,y,t) = & \; v_{00} \\
& + v_{10}(x - M_x t) + v_{01}(y - M_y t) \\
& + v_{20}(x - M_x t)^2 + v_{11}(x - M_x t)(y - M_y t) + v_{02}((y - M_y t)^2 + t^2) \\
& + v_{21}(x - M_x t)^2(y - M_y t) + v_{12}(x - M_x t)((y - M_y t)^2 + t^2) \\
& + v_{22}((x - M_x t)^2((y - M_y t)^2 + t^2) + t^4/6) \\
& + u_{11}(t^2/2) \\
& + u_{21}(x - M_x t)t^2 + u_{12}(y - M_y t)t^2 \\
& + u_{22}(2(x - M_x t)(y - M_y t)t^2) \\
& + p_{01}(-t) \\
& + p_{11}(-(x - M_x t)t) + p_{02}(-2(y - M_y t)t) \\
& + p_{21}(-(x - M_x t)^2 t - t^3/3) + p_{12}(-2(x - M_x t)(y - M_y t)t) \\
& + p_{22}(-2((x - M_x t)^2 + t^2/3)(y - M_y t)t),
\end{aligned}$$

and for $p$ by

$$
\begin{aligned}
pa(x,y,t) = {} & p_{00} \\
& + p_{10}(x - M_x t) + p_{01}(y - M_y t) \\
& + p_{20}((x - M_x t)^2 + t^2) + p_{11}(x - M_x t)(y - M_y t) + p_{02}((y - M_y t)^2 + t^2) \\
& + p_{21}((x - M_x t)^2 + t^2)(y - M_y t) + p_{12}(x - M_x t)((y - M_y t)^2 + t^2) \\
& + p_{22}(((x - M_x t)^2 + t^2)((y - M_y t)^2 + t^2) - 2t^4/3) \\
& + u_{10}(-t) \\
& + u_{20}(-2(x - M_x t)t) + u_{11}(-(y - M_y t)t) \\
& + u_{21}(-2(x - M_x t)(y - M_y t)t) + u_{12}(-(y - M_y t)^2 t - t^3/3) \\
& + u_{22}(-2(x - M_x t)((y - M_y t)^2 + \frac{1}{3}t^2)t) \\
& + v_{01}(-t) \\
& + v_{11}(-(x - M_x t)t) + v_{02}(-2(y - M_y t)t) \\
& + v_{21}(-(x - M_x t)^2 t - t^3/3) + v_{12}(-2(x - M_x t)(y - M_y t)t) \\
& + v_{22}(-2((x - M_x t)^2 + t^2/3)(y - M_y t)t).
\end{aligned}
$$

## Appendix D: A Fourth Order Algorithm For Equation (31)

(Appendix D-U) $u$ solution for the linearized Euler equations

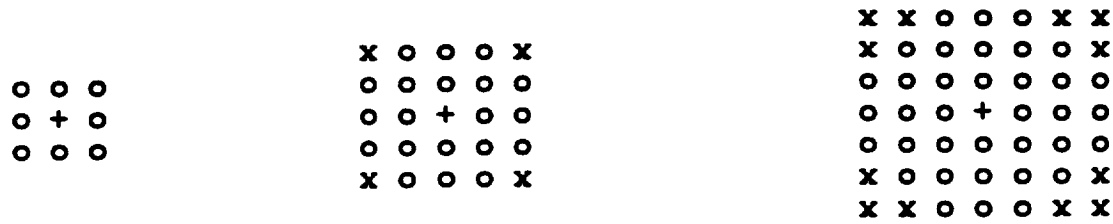
$$
\begin{aligned}
u_{i,j}^{n+1} =& u_{00} \\
&+k(-p_{10} - M_y u_{01} - M_x u_{10}) \\
&+k^2(M_y p_{11} + 2M_x p_{20} + M_y^2 u_{02} + M_x M_y u_{11} + (1+M_x^2)u_{20} + v_{11}/2) \\
&+k^3(-(1+3M_y^2)p_{12}/3 - 2M_x M_y p_{21} - (1+3M_x^2)p_{30} \\
&\quad - M_y^3 u_{03} - M_x M_y^2 u_{12} - (1+M_x^2)M_y u_{21} - M_x(3+M_x^2)u_{30} \\
&\quad - M_y v_{12} - M_x v_{21}) \\
&+k^4(M_y(1+M_y^2)p_{13} + 2M_x(1/3+M_y^2)p_{22} \\
&\quad + (1+3M_x^2)M_y p_{31} + 4M_x(1+M_x^2)p_{40} \\
&\quad + M_y^4 u_{04} + M_x M_y^3 u_{13} + (1/6 + M_y^2 + M_x^2 M_y^2)u_{22} \\
&\quad + M_x(3+M_x^2)M_y u_{31} + (1+6M_x^2+M_x^4)u_{40} \\
&\quad + (1+6M_y^2)v_{13}/4 + 2M_x M_y v_{22} + (1+6M_x^2)v_{31}/4) \\
&+k^5(-(1+10M_y^2+5M_y^4)p_{14}/5 - 2M_x M_y(1+M_y^2)p_{23} \\
&\quad - (1+5M_x^2+5M_y^2+15M_x^2M_y^2)p_{32}/5 - 4M_x(1+M_x^2)M_y p_{41} \\
&\quad - M_x M_y^4 u_{14} - M_y(1/2+M_y^2+M_x^2M_y^2)u_{23} \\
&\quad - M_x(1/2+3M_y^2+M_x^2M_y^2)u_{32} - (1+6M_x^2+M_x^4)M_y u_{41} \\
&\quad - M_y(1+2M_y^2)v_{14} - M_x(1/2+3M_y^2)v_{23} \\
&\quad - (1/2+3M_x^2)M_y v_{32} - M_x(1+2M_x^2)v_{41}) \\
&+k^6((2M_x/5 + 4M_x M_y^2 + 2M_x M_y^4)p_{24} \\
&\quad + (4M_x/5 + 4M_x^3/3 + 4M_x M_y^2 + 4M_x^3 M_y^2)p_{42} \\
&\quad + (1/15 + M_y^2 + M_y^4 + M_x^2 M_y^4)u_{24} \\
&\quad + (2/15 + M_x^2 + M_y^2 + 6M_x^2M_y^2 + M_x^4M_y^2)u_{42} \\
&\quad + (2M_x M_y + 4M_x M_y^3)v_{24} \\
&\quad + (2M_x M_y + 4M_x^3 M_y)v_{42})
\end{aligned}
$$

(Appendix D-V) $v$ solution for the linearized Euler equations

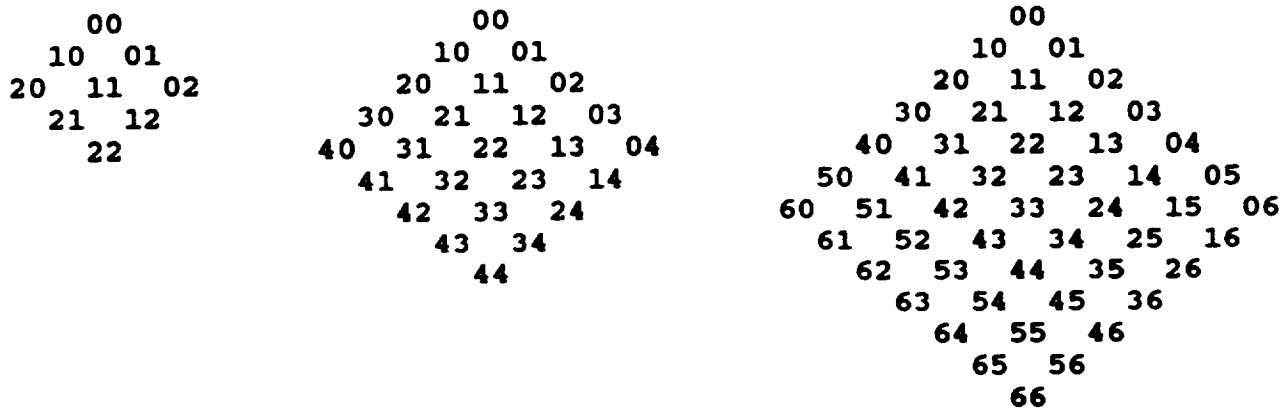
$$
\begin{aligned}
v_{i,j}^{n+1} =& v_{00} \\
&+k(-p_{01} - M_y v_{01} - M_x v_{10}) \\
&+k^2(2M_y p_{02} + M_x p_{11} + u_{11}/2 + (1+M_y^2)v_{02} + M_x M_y v_{11} + M_x^2 v_{20}) \\
&+k^3(-(1+3M_y^2)p_{03} - 2M_x M_y p_{12} - (1+3M_x^2)p_{21}/3 \\
&\quad - M_y u_{12} - M_x u_{21} \\
&\quad - M_y(3+M_y^2)v_{03} - M_x(1+M_y^2)v_{12} - M_x^2 M_y v_{21} - M_x^3 v_{30}) \\
&+k^4(4M_y(1+M_y^2)p_{04} + M_x(1+3M_y^2)p_{13} \\
&\quad + 2(1/3+M_x^2)M_y p_{22} + M_x(1+M_x^2)p_{31} \\
&\quad + (1+6M_y^2)u_{13}/4 + 2M_x M_y u_{22} + (1+6M_x^2)u_{31}/4 \\
&\quad + (1+6M_y^2+M_y^4)v_{04} + M_x M_y(3+M_y^2)v_{13} \\
&\quad + (1/6+M_x^2+M_x^2 M_y^2)v_{22} + M_x^3 M_y v_{31} + M_x^4 v_{40}) \\
&+k^5(-4M_x M_y(1+M_y^2)p_{14} - (1+5M_x^2+5M_y^2+15M_x^2M_y^2)p_{23}/5 \\
&\quad - 2M_x(1+M_x^2)M_y p_{32} - (1+10M_x^2+5M_x^4)p_{41}/5 \\
&\quad - M_y(1+2M_y^2)u_{14} - M_x(1/2+3M_y^2)u_{23} \\
&\quad - (1/2+3M_x^2)M_y u_{32} - M_x(1+2M_x^2)u_{41} \\
&\quad - M_x(1+6M_y^2+M_y^4)v_{14} - M_y(1/2+3M_x^2+M_x^2M_y^2)v_{23} \\
&\quad - M_x(1/2+M_x^2+M_x^2M_y^2)v_{32} - M_x^4 M_y v_{41}) \\
&+k^6((4M_y/5+4M_x^2M_y+4M_y^3/3+4M_x^2M_y^3)p_{24} \\
&\quad + (2M_y/5+4M_x^2M_y+2M_x^4M_y)p_{42} \\
&\quad + (2M_x M_y+4M_x M_y^3)u_{24} \\
&\quad + (2M_x M_y+4M_x^3 M_y)u_{42} \\
&\quad + (2/15+M_x^2+M_y^2+6M_x^2M_y^2+M_x^2M_y^4)v_{24} \\
&\quad + (1/15+M_x^2+M_x^4+M_x^4M_y^2)v_{42})
\end{aligned}
$$

(Appendix D-P) $p$ solution for the linearized Euler equations

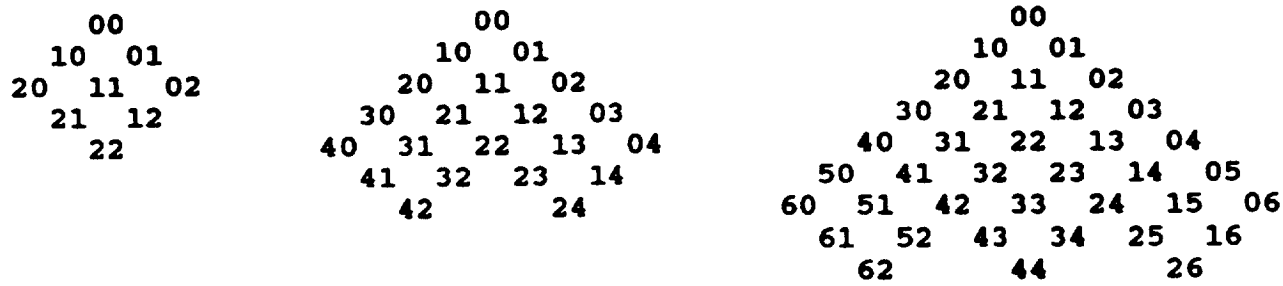
$$
\begin{aligned}
p_{i,j}^{n+1} =& p_{00} \\
&+k(-(M_y p_{01}) - M_x p_{10} - u_{10} - v_{01}) \\
&+k^2((1+M_y^2)p_{02} + M_x M_y p_{11} + (1+M_x^2)p_{20} \\
&\quad + M_y u_{11} + 2M_x u_{20} + 2M_y v_{02} + M_x v_{11}) \\
&+k^3(-M_y(3+M_y^2)p_{03} - M_x(1+M_y^2)p_{12} \\
&\quad - (1+M_x^2)M_y p_{21} - M_x(3+M_x^2)p_{30} \\
&\quad - (1+3M_y^2)u_{12}/3 - 2M_x M_y u_{21} - (1+3M_x^2)u_{30} \\
&\quad - (1+3M_y^2)v_{03} - 2M_x M_y v_{12} - (1+3M_x^2)v_{21}/3) \\
&+k^4((1+6M_y^2+M_y^4)p_{04} + M_x M_y(3+M_y^2)p_{13} \\
&\quad + (1/3+M_x^2+M_y^2+M_x^2M_y^2)p_{22} + M_x(3+M_x^2)M_y p_{31} \\
&\quad + (1+6M_x^2+M_x^4)p_{40} \\
&\quad + M_y(1+M_y^2)u_{13} + 2M_x(1/3+M_y^2)u_{22} \\
&\quad + (1+3M_x^2)M_y u_{31} + 4M_x(1+M_x^2)u_{40} \\
&\quad + 4M_y(1+M_y^2)v_{04} + M_x(1+3M_y^2)v_{13} \\
&\quad + 2(1/3+M_x^2)M_y v_{22} + M_x(1+M_x^2)v_{31}) \\
&+k^5(-M_x(1+6M_y^2+M_y^4)p_{14} - M_y(1+3M_x^2+M_y^2+M_x^2M_y^2)p_{23} \\
&\quad - M_x(1+M_x^2+3M_y^2+M_x^2M_y^2)p_{32} - (1+6M_x^2+M_x^4)M_y p_{41} \\
&\quad - (1+10M_y^2+5M_y^4)u_{14}/5 - 2M_x M_y(1+M_y^2)u_{23} \\
&\quad - (1+5M_x^2+5M_y^2+15M_x^2M_y^2)u_{32}/5 - 4M_x(1+M_x^2)M_y u_{41} \\
&\quad - 4M_x M_y(1+M_y^2)v_{14} - (1+5M_x^2+5M_y^2+15M_x^2M_y^2)v_{23}/5 \\
&\quad - 2M_x(1+M_x^2)M_y v_{32} - (1+10M_x^2+5M_x^4)v_{41}/5) \\
&+k^6((1/5+M_x^2+2M_y^2+6M_x^2M_y^2+M_y^4+M_x^2M_y^4)p_{24} \\
&\quad + (1/5+2M_x^2+M_x^4+M_y^2+6M_x^2M_y^2+M_x^4M_y^2)p_{42} \\
&\quad + (2M_x/5+4M_xM_y^2+2M_xM_y^4)u_{24} \\
&\quad + (4M_x/5+4M_x^3/3+4M_xM_y^2+4M_x^3M_y^2)u_{42} \\
&\quad + (4M_y/5+4M_x^2M_y+4M_y^3/3+4M_x^2M_y^3)v_{24} \\
&\quad + (2M_y/5+4M_x^2M_y+2M_x^4M_y)v_{42})
\end{aligned}
$$



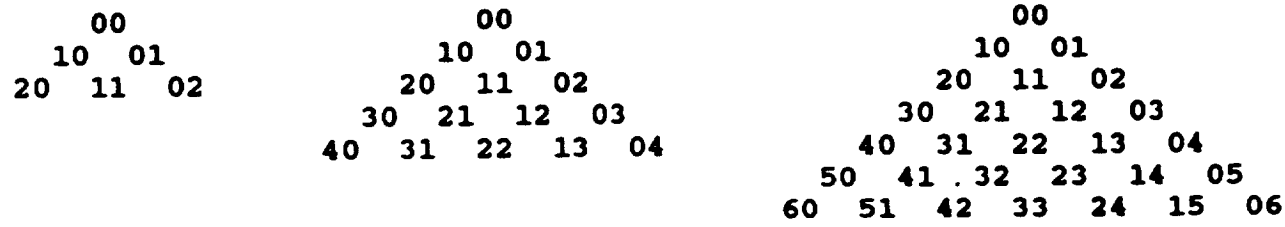

(a) Stencils for second, fourth and sixth order interpolation.




(b) Square stencil (x,y) expansion coefficients.




(c) Minimal stencil (x,y) expansion coefficients.




(d) Taylor series (x,y) expansion coefficients.

Figure 1.—Two dimensional interpolation information.

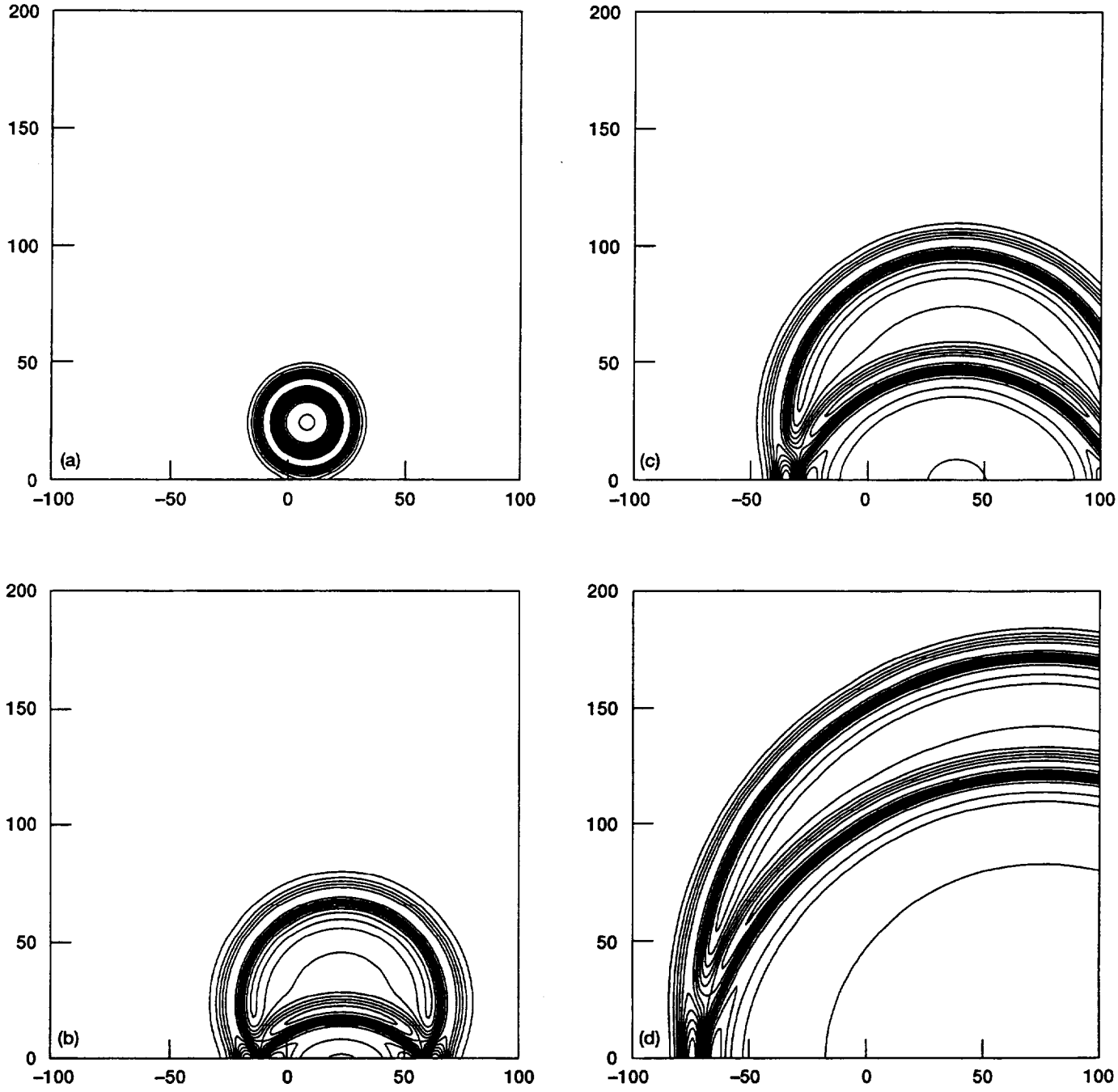



Figure 2.—Plots for the linearized Euler equations with $M_x = 0.5$ and $M_y = 0$, using the fourth order exact propagator algorithm along a wall. (a) Pressure at t = 15. (b) Pressure at t = 45. (c) Pressure at t = 75. (d) Pressure at t = 150.

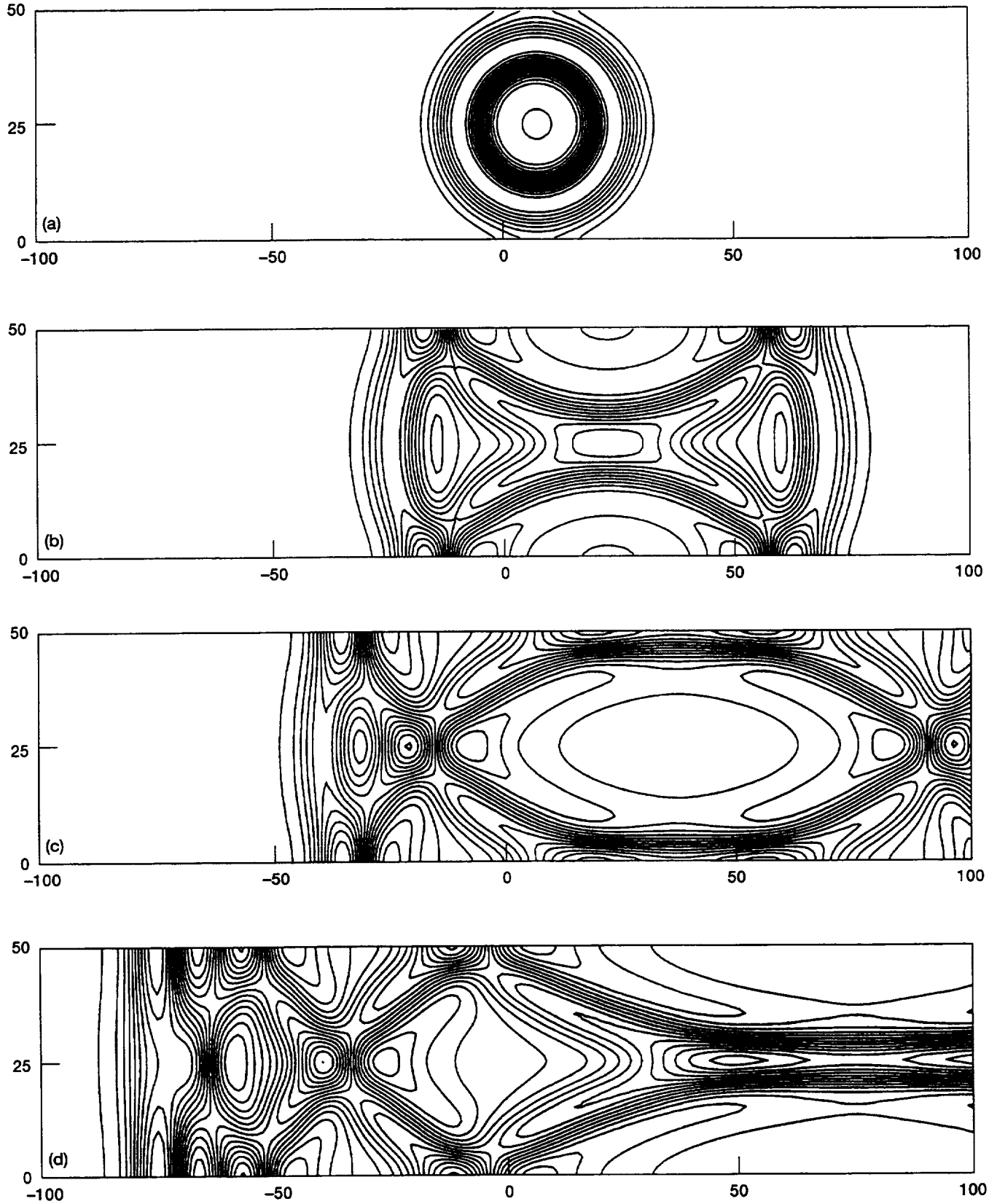



Figure 3.—Plots for the linearized Euler equations with $M_x = 0.5$ and $M_y = 0$, using the fourth order exact propagator algorithm in a duct. (a) Pressure at t = 15. (b) Pressure at t = 45. (c) Pressure at t = 75. (d) Pressure at t = 150.

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (*Leave blank*) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | September 1995 | Technical Memorandum |

**4. TITLE AND SUBTITLE**
An Approach to the Development of Numerical Algorithms for First Order Linear Hyperbolic Systems in Multiple Space Dimensions: The Constant Coefficient Case

**5. FUNDING NUMBERS**
WU-505-62-52

**6. AUTHOR(S)**
John W. Goodrich

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

**8. PERFORMING ORGANIZATION REPORT NUMBER**
E-9649

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
NASA TM-106928

**11. SUPPLEMENTARY NOTES**
Responsible person, John W. Goodrich, organization code 2610, (216) 433-5922.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Unclassified - Unlimited
Subject Categories 64 and 71

This publication is available from the NASA Center for Aerospace Information, (301) 621-0390.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** (*Maximum 200 words*)

Two methods for developing high order single step explicit algorithms on symmetric stencils with data on only one time level are presented. Examples are given for the convection and linearized Euler equations with up to the eighth order accuracy in both space and time in one space dimension, and up to the sixth in two space dimensions. The method of characteristics is generalized to nondiagonalizable hyperbolic systems by using exact local polynominal solutions of the system, and the resulting exact propagator methods automatically incorporate the correct multidimensional wave propagation dynamics. Multivariate Taylor or Cauchy-Kowaleskaya expansions are also used to develop algorithms. Both of these methods can be applied to obtain algorithms of arbitrarily high order for hyperbolic systems in multiple space dimensions. Cross derivatives are included in the local approximations used to develop the algorithms in this paper in order to obtain high order accuracy, and improved isoptropy and stability. Efficiency in meeting global error bounds is an important criterion for evaluating algorithms, and the higher order algorithms are shown to be up to several orders of magnitude more efficient even though they are more complex. Stable high order boundary conditions for the linearized Euler equations are developed in one space dimension, and demonstrated in two space dimensions.

**14. SUBJECT TERMS**
Finite difference methods; Hyperbolic systems; Acoustics; Multidimensional; Outflow boundary conditions; High order accuracy

**15. NUMBER OF PAGES**
56

**16. PRICE CODE**
A04

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | |