A Thesis

entitled

Verification of a Computational Aeroacoustics Code Using External Verification Analysis (EVA)

by

Daniel Ingraham

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Mechanical Engineering

---

Dr. Ray Hixon, Committee Chair

Dr. Douglas Oliver, Committee Member

Dr. Chunhua Sheng, Committee Member

Dr. Patricia Komuniecki, Dean
College of Graduate Studies

The University of Toledo
May 2010

An Abstract of

Verification of a Computational Aeroacoustics Code Using External Verification Analysis (EVA)

by

Daniel Ingraham

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Mechanical Engineering

The University of Toledo
May 2010

As Computational Aeroacoustics (CAA) codes become more complex and widely used, robust Verification of such codes becomes more and more important. Recently, Hixon et al. proposed a variation of the Method of Manufactured Solutions of Roache especially suited for Verifying unsteady CFD and CAA codes that does not require the generation of source terms or any modification of the code being Verified. This work will present the development of the External Verification Analysis (EVA) method and the results of its application to some popular model equations of CFD/CAA and a high-order nonlinear CAA code.

To Melissa, my everything.

Acknowledgments

This work was supported by the Subsonic Fixed Wing Project of the NASA Fundamental Aeronautics Program, under Task NNC07E125T-0. The technical monitor for this work was Dr. Edmane Envia of the NASA Glenn Research Center.

I am grateful for Dr. Oliver and Dr. Sheng for agreeing to serve on my committee.

I am deeply appreciative of Dr. Hixon's help and guidance, and for allowing me to work on such an interesting and challenging project.

My thanks to my parents, Jim and Becky, and my sisters, Cassie and Trisha. Their love and constant encouragement are responsible for any success I may have in the future.

And finally, my heartfelt thanks to my wonderful wife Melissa, who endured more than a few lonely nights and did not receive the attention she deserves during the creation of this work.

Contents

List of Figures

List of Abbreviations

BASS	The NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulator CAA Code
CAA	Computational Aeroacoustics
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy number
DEE	Discretization Error Estimation, Salari and Knupp's term for Solution Verification
DRP	Tam and Webb's Dispersion Relation Preserving spatial differencing scheme
EVA	External Verification Analysis
GCI	Grid Convergence Index
HALE_RK	High-accuracy large-step explicit Runge-Kutta schemes of Allampalli et al.
MMS	Method of Manufactured Solutions
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
RANS	Reynolds-Averaged Navier-Stokes Equations
RK	Runge-Kutta schemes, a class of algorithms for numerically integrating ordinary differential equations and pseudo-ODEs

List of Symbols

α_n ..... Local measure of grid volume
Δ ..... Spatial discretization measure (grid spacing)
Δt ..... Time step size
Δx ..... Grid spacing in the x-direction
Δy ..... Grid spacing in the y-direction
Δz ..... Grid spacing in the z-direction
ε ..... Error magnitude
γ ..... Ratio of specific heats
μ ..... Viscous term coefficient of the viscous Burgers' equation
ρ ..... Density
ρu ..... Momentum in the x-direction
ρv ..... Momentum in the y-direction
ρw ..... Momentum in the z-direction
ω^* ..... Numerical frequency of disturbances
c ..... Propagation speed of disturbances
E_{total} .... Total energy
$\vec{E}$ ..... Cartesian flux vector in x-direction
$\vec{F}$ ..... Cartesian flux vector in y-direction
$\vec{G}$ ..... Cartesian flux vector in z-direction
j ..... Grid point index
k ..... Wavenumber of disturbances
$(k\Delta x)^*$ .. Numerical wavenumber of disturbances
n ..... Time step index
N ..... Number of grid points
p ..... Order-of-accuracy of a numerical scheme
p ..... Pressure
$\vec{P}$ ..... Taylor series polynomial
q ..... Number of spatial dimensions
$\vec{Q}$ ..... Solution vector of the Euler Equations

$\bar{Q}$ ..... Mean value of $\vec{Q}$
$\tilde{Q}$ ..... Magnitude of fluctuating component of $\vec{Q}$
$\vec{\mathcal{Q}}$ ..... Exact solution of a hypothetical PDE
$\vec{R}$ ..... Residual error of truncated Taylor Series
T ..... Number of time steps
U ..... Reference solution used to calculate an error norm
u ..... Dependent variable of Burgers' equation
u ..... Velocity in the x-direction
v ..... Velocity in the z-direction
w ..... Velocity in the y-direction

Chapter 1

Overview of Code Verification

The development of high-speed computers in the last half-century has resulted in the creation of many sophisticated tools intended to aid engineers in the design process. The usefulness of these tools (or codes, as they are more commonly called, and of which Computation Fluid Dynamics (CFD) and Computation Aeroacoustics (CAA) codes are a member) lie in their ability to simulate complex phenomena for which analytic solutions are not known and reproduction in a laboratory may be difficult, impossible or prohibitively expensive. No one can deny that these codes are very useful, but what assurance does one have that they perform correctly? The question is made even more pertinent by the amount of time, effort and money that is required to develop these tools, and by the importance of the decisions that are informed by them. The fact that a code may compare well with a handful of experiments or is widely used in industry is unfortunately not a guarantee of its accuracy. Abanto et al. [1] show how popular CFD codes can return surprising results for simple-looking test cases. Of course, experimental or developing codes in academia may have their problems, too.

To address this problem, the field of Verification and Validation has developed in the past few decades. One of the more popular techniques that has emerged from

the field is the Method of Manufactured Solutions, or MMS. First introduced by Roache [2], MMS provides a way to rigorously demonstrate that a CFD or CAA code is solving its governing equations correctly. Its main disadvantage, however, is that it requires one to modify or add a source term to the solver in the code, which may be undesirable, impractical or (perhaps in the case of a proprietary code) impossible. Recently, Hixon and Anderson [3] have presented a variant of MMS called "External Verification Analysis" (EVA) that does not require the addition of source terms to the governing equations of a code, or modification of the code at all. The focus of this work will be the development and application of the EVA method to code Verification with special emphasis on Verifying the NASA Glenn Research Center's Broadband Aeroacoustic Stator Simulator (BASS) code.

1.1 Code Verification vs. Solution Verification vs. Validation

To the general public the terms "Code Verification," "Solution Verification" and "Validation" may all appear to be synonymous; however, each of these three terms have a precise definition in the literature and describe very different processes. The popular distinction between "Verification" and "Validation" is attributed to Boehm [4] and Blottner [5] and is emphasized repeatedly by Roache [6] [7] [8] and Oberkampf and Trucano [9]:

Verification Solving the equations right

Validation Solving the right equations

Verification consists of convincingly demonstrating that a code solves its governing equations to the expected order-of-accuracy, while Validation is ensuring that the equations and models used by the code adequately represent the physics of the prob-

lem. Verification, as emphasized by Roache, is purely a mathematical exercise with the goal of ensuring that the numerical techniques of a code are functioning properly, while Validation is an exercise in science or engineering to ensure that the solution obtained with the code actually corresponds with reality. A code and/or solution could be Verified yet fail Validation. For instance, using an incompressible flow solver for a supersonic configuration will always fail Validation no matter how convincingly the solver is Verified. On the other hand, Verification is a necessary prerequisite for Validation; there is no point in attempting to determine if one is "solving the right equations" if one isn't "solving the equations right" in the first place.

There also exists a distinction between "Code Verification" and "Solution Verification" in the literature. Roache does an excellent job discussing this, as do Salari and Knupp [10] — although they prefer the term "Discretization Error Estimation (DEE)" to Solution Verification. Solution Verification, as defined by Roache, is the process of "banding" the numerical error of the solution by performing grid convergence studies. Solution Verification should be preformed for each code run, and (according to Roache) should accompany any results in scholarly publications. Roache has also developed a "Grid Convergence Index (GCI)" [6] based on Richardson Extrapolation with the intention of promoting uniform reporting of grid convergence in the literature. Code Verification, on the other hand, is performed to Verify that the numerical methods used in the code are functioning properly by calculating the code's observed order-of-accuracy and comparing it to the formal order-of-accuracy of the numerical scheme. In contrast to Solution Verification, Code Verification need only be preformed once for each combination of code options. This work is focused on describing the External Verification Analysis (EVA) method, a new technique of *Code* Verification.

1.2 Review of Methods of Code Verification

The most widely-used method of Code Verification is the Method of Manufactured Solutions (MMS) first developed by Steinberg and Roache [2], and explained in detail in [11, 10, 6]. In short, the Method of Manufactured Solutions consists of specifying and inserting a "test" (manufactured) solution into the governing equation(s) of interest. Because the "test" solution is not an actual solution of the equations, a residual error will remain; however, if the governing equations are modified to include a source term equal to the residual error, then the "test solution" is an *exact* — or "manufactured" — solution to the modified governing equations. The computer code to be Verified must then be modified to include the source term found during the "manufacturing" process. The code is then run using different grid spacings or time steps and compared to the manufactured solution so that the rate of convergence may be observed. As an example, consider the nonlinear inviscid Burgers' equation:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{1.1}$$

If a "test solution" $\tilde{u}(x,t)$ is assumed and substituted into Equation (1.1)

$$\frac{\partial \tilde{u}}{\partial t} + \tilde{u}\frac{\partial \tilde{u}}{\partial x} = \tilde{S}(x,t) \tag{1.2}$$

where $\tilde{S}(x,t)$ is the residual error from $\tilde{u}$ not satisfying the inviscid Burgers' equation. If $\tilde{S}(x,t)$ is added to the governing equation solver as a source term, however, $\tilde{u}$ becomes an exact solution to the new, modified governing equation. This "manufactured" solution can now be used to Verify the order-of-accuracy of the spatial or temporal discretization schemes used to solve Equation (1.1), provided that the code supports or can be made to support arbitrary source terms.

Numerous examples of the use of the Method of Manufactured Solutions can be

found in the literature. Roy applies MMS to two steady-state two-dimensional Euler Equation solvers in [12]. Roy et al. use MMS to Verify the same codes for both the two-dimensional Euler and Navier-Stokes equations in [13]. Bond, Knupp and Ober used MMS to Verify both the interior stencil and various boundary conditions of a Reynolds-Averaged Navier-Stokes (RANS) solver in [14, 15, 16]. The two textbooks published on Code Verification — one by Roache [6] and another by Salari and Knupp [10] — contain excellent examples of and advice on the MMS process (though some of the content of the textbooks is duplicated in other papers by the authors — see [8, 11]). Finally, Eça et al. [17] used MMS to Verify the implementation of an incompressible RANS solver with numerous turbulence models.

1.3 Linear Analysis of Numerical Schemes

A familiarity with the numerical schemes used in a CFD or CAA code is not required in order to Verify it using External Verification Analysis (or any other method); however, experience has shown it to be immensely helpful in interpreting results and troubleshooting difficult test cases.

Generally, the governing equations that a CFD or CAA code will be designed to solve can be written in a form such as

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{F}(\vec{Q}, \vec{Q}_x, \vec{Q}_y, \vec{Q}_z)}{\partial x, y, z} = 0 \tag{1.3}$$

where $\vec{Q}$ is a vector of the solution variables and $\vec{F}$ is a flux, which is a function of the solution variables. An initial condition is also provided

$$\vec{Q}(x, y, z, t = t_0) = \vec{f}(x, y, z) \tag{1.4}$$

One may then rewrite

$$\frac{\partial \vec{Q}}{\partial t} = -\frac{\partial \vec{F}(\vec{Q}, \vec{Q}_x, \vec{Q}_y, \vec{Q}_z)}{\partial x, y, z} \tag{1.5}$$

and use some type of spatial differencing scheme to approximate the flux derivatives, which in turn gives the temporal derivative of the solution vector

$$\left.\frac{\partial \vec{Q}}{\partial t}\right|_{num} = -\left.\frac{\partial \vec{F}}{\partial x, y, z}\right|_{num} + \vartheta(\Delta) \tag{1.6}$$

where Δ is some spatial discretization measure. With an initial condition and a temporal derivative of that initial condition, one can march in time:

$$\vec{Q}(t = t_0 + \Delta t) = \vec{Q}(t = t_0) + \int_{t_0}^{t_0+\Delta t} \frac{\partial \vec{Q}}{\partial t} d\, t + \vartheta(\Delta t) \tag{1.7}$$

where Δt is the time step size, then

$$\vec{Q}(t = t_0 + \Delta t) = \vec{Q}(t = t_0) + \int_{t_0}^{t_0+\Delta t} \frac{\partial \vec{F}}{\partial x, y, z} d\, t + \vartheta(\Delta) + \vartheta(\Delta t) \tag{1.8}$$

Equation (1.8) shows that there are two sources of errors to consider in our Verification: the spatial discretization scheme (Δ) and the time-marching scheme (Δt). In this section, each of the spatial differencing and time-marching schemes discussed in this work will be applied to the linear inviscid Burgers' equation (see Section 3.1 for more information on the linear and nonlinear Burgers' equations) in order to better understand the behavior of these schemes.

1.3.1 Spatial Differencing using Finite Differences

The linear inviscid Burgers' equation is

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{1.9}$$

One solution to (1.9) is

$$u(x,t) = e^{i(kx-\omega t)} \tag{1.10}$$

where $\omega \equiv ck$. Now, define

$$\begin{aligned} j &\equiv \tfrac{x-x_{\min}}{\Delta x} + 1 \\ n &\equiv \tfrac{t-t_0}{\Delta t} \end{aligned} \tag{1.11}$$

where

$$\begin{aligned} \Delta x &\equiv \tfrac{x_{\max}-x_{\min}}{N-1} \\ \Delta t &\equiv \tfrac{t_{\max}-t_{\min}}{T} \end{aligned} \tag{1.12}$$

and N is the number of points in the domain and T is the number of time steps taken. Now one can write

$$u(x,t) = u_j^n \tag{1.13}$$

In this work, most of the spatial derivative schemes are central differences that can be expressed in the general form

$$\left.\frac{\partial u}{\partial x}\right|_{j,num}^{n} = \frac{1}{\Delta x}\sum_{m=1}^{M} a_m(u_{j+m} - u_{j-m}) \tag{1.14}$$

where M is the "span" of the stencil (1 for the standard second-order central, 2 for the standard fourth-order central, etc.) and a_m are the coefficients of the stencil. The only exception in this work to Equation (1.14) is the compact 6th-order scheme of Lele [18]. If Equation (1.10) is substituted into Equation (1.14)

$$\begin{aligned} \left.\tfrac{\partial u}{\partial x}\right|_{j,num}^{n} &= \tfrac{1}{\Delta x}\textstyle\sum_{m=1}^{M} a_m \left(e^{ik(j+m)\Delta x-\omega t} - e^{ik(j-m)\Delta x-\omega t}\right) \\ &= \tfrac{1}{\Delta x}\textstyle\sum_{m=1}^{M} a_m \left(e^{ikm\Delta x} - e^{-ikm\Delta x}u_j^n\right) \\ &= \tfrac{i}{\Delta x}\textstyle\sum_{m=1}^{M} 2a_m \sin(mk\Delta x)u_j^n \end{aligned} \tag{1.15}$$

One can also find the exact derivative of the assumed solution (Equation (1.10))

$$\begin{aligned} \frac{\partial u}{\partial x}\Big|_{j,exact}^{n} &= ike^{kx-\omega t} \\ &= iku_j^n \\ &= i\frac{(k\Delta x)}{\Delta x}u_j^n \end{aligned} \tag{1.16}$$

One can now define the numerical wavenumber $(k\Delta x)^*$ such that

$$(k\Delta x)^* \equiv \sum_{m=1}^{M} 2a_m \sin(mk\Delta x) \tag{1.17}$$

so that

$$\frac{\partial u}{\partial x}\Big|_{j,num}^{n} = i\frac{(k\Delta x)^*}{\Delta x}u_j^n \tag{1.18}$$

Equations (1.16) and (1.18) are very similar — they only differ in the $(k\Delta x)$ term. Thus the accuracy of a given spatial derivative is wholly dependent on the behavior of $(k\Delta x)^*$, which itself is a function of $k\Delta x$. Figure 1-1 shows $(k\Delta x)^*$ as a function of $k\Delta x$ for each of the spatial differencing schemes considered in this work. The compact 6th-order scheme "hugs" the exact curve for the largest range of $k\Delta x$ and thus performs the best. All of the schemes agree with the exact solution very closely as $k\Delta x$ goes to 0, however.

One can calculate the error introduced by using Equation (1.14) as an approximation of the spatial derivative of $u(x,t)$ with the equation

$$\varepsilon(k\Delta x) = \left| \frac{\frac{\partial u}{\partial x}_{num} - \frac{\partial u}{\partial x}_{exact}}{\frac{\partial u}{\partial x}_{exact}} \right| \tag{1.19}$$

or, after substituting Equations (1.16) and (1.18)

$$\varepsilon(k\Delta x) = \left| \frac{(k\Delta x)^*}{k\Delta x} - 1 \right| \tag{1.20}$$

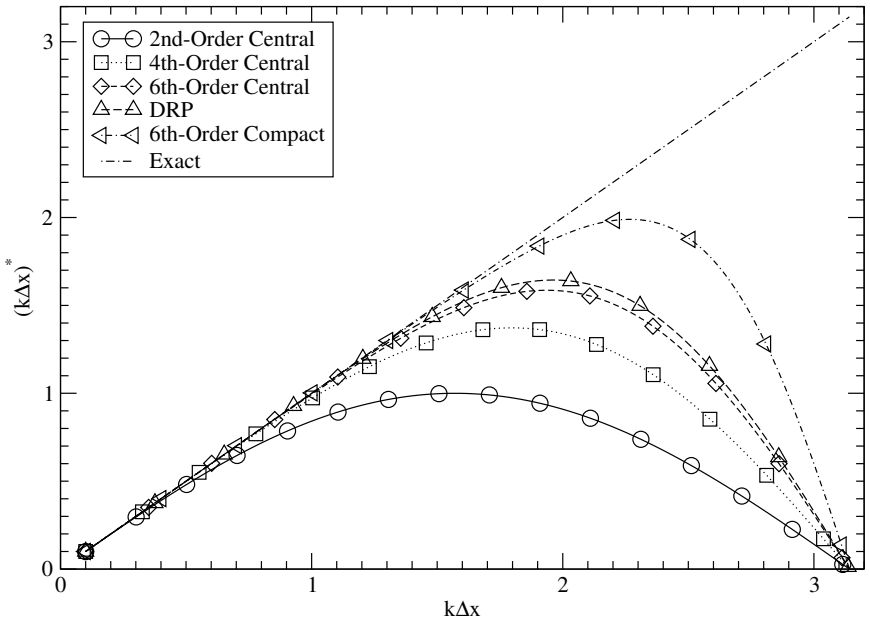


Figure 1-1: Numerical Wavenumber $(k\Delta x)^*$ for Various Spatial Differencing Schemes

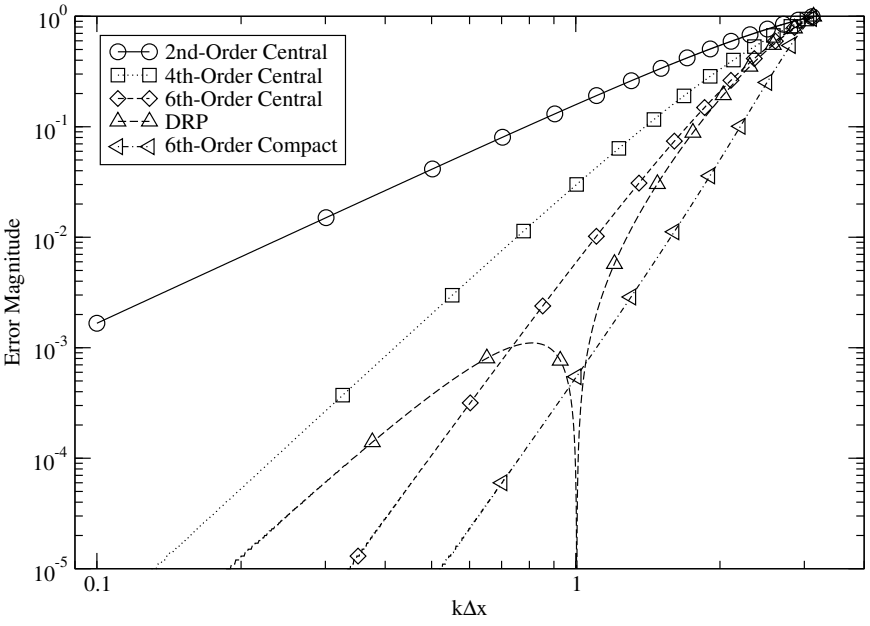


Figure 1-2: Error Magnitude for Various Spatial Differencing Schemes

Figure 1-2 shows a plot of the error magnitude for each of the spatial differencing schemes considered here. Figure 1-1 would seem to indicate that the compact 6th-order scheme's error magnitude would be the lowest, and Figure 1-2 confirms that this is true. Also as expected, the explicit 6th-order scheme is more accurate than the explicit 4th-order scheme, which is in turn more accurate than the 2nd-order scheme. The most interesting scheme is the Dispersion Relation Preserving (DRP) scheme of Tam and Webb [19]. The DRP is a 4th-order scheme but actually compares better to the exact solution than the 6th-order explicit scheme for higher $k\Delta x$ ranges and worse for lower. This is because the DRP scheme is optimized to achieve a low level of error for a wide range of grid points — a side-effect of this optimization process is that the error magnitude can actually *increase* as Δx is lowered.

When the error magnitude of a spatial differencing scheme is plotted on a log-log plot the slope of the error is the observed order-of-accuracy of the scheme. To find an expression for the observed order-of-accuracy of each scheme as a function of $k\Delta x$, one can use the chain rule

$$\frac{d\left[\log(\varepsilon(k\Delta x))\right]}{d\left[k\Delta x\right]} = \frac{d\left[\log(\varepsilon(k\Delta x))\right]}{d\left[\log(k\Delta x)\right]} \frac{d\left[\log(k\Delta x)\right]}{d\left[k\Delta x\right]} \tag{1.21}$$

and then

$$\frac{d\left[\log(\varepsilon(k\Delta x))\right]}{d\left[\log(k\Delta x)\right]} = \frac{d\left[\log(\varepsilon(k\Delta x))\right]}{d\left[k\Delta x\right]} \Big/ \frac{d\left[\log(k\Delta x)\right]}{d\left[k\Delta x\right]} \tag{1.22}$$

Equation (1.22) was used to plot the observed order-of-accuracy for each of the spatial differencing schemes — the results are shown in Figure 1-3. Each of the schemes eventually attain their formal order-of-accuracy (recall from above that the DRP is a fourth-order scheme) as $k\Delta x$ goes to zero, but none of the schemes return their expected order-of-accuracy for the entire range of $k\Delta x$. In general, the higher-order the scheme, the smaller the range of $k\Delta x$ values where it shows its formal order-of-accuracy. In particular, the DRP scheme is only briefly fourth-order accurate at very

low values of $k\Delta x$ — this indicates that it will require many more grid points (i.e., smaller Δx) to convincingly Verify it than the other spatial differencing schemes.

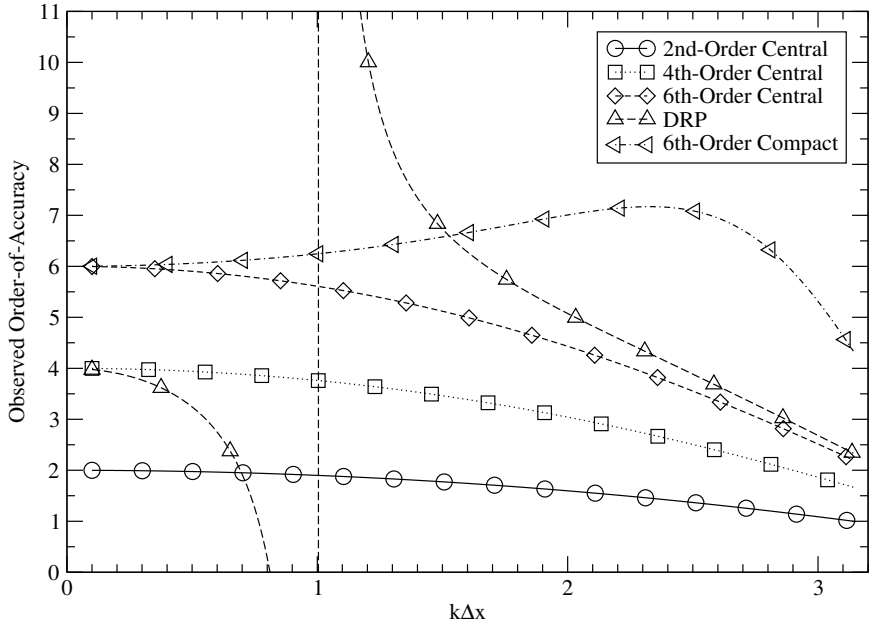


Figure 1-3: Observed Order-of-Accuracy for Various Spatial Differencing Schemes

1.3.2 Runge-Kutta Time-Marching Methods

Once an approximation of $\frac{\partial u}{\partial x}$ is obtained, an approximation of $\frac{\partial u}{\partial t}$ can also be found from the governing equation:

$$\begin{aligned} \frac{\partial u}{\partial t}\Big|_{num}^{n} &= -c\,\frac{\partial u}{\partial x}\Big|_{num}^{n} \\ &= -ic\frac{(k\Delta x)^*}{\Delta x}u^n \\ &= -i\omega^* u^n \end{aligned} \tag{1.23}$$

where ω^* is the numerical frequency, $\omega^* = c\frac{(k\Delta x)^*}{\Delta x}$. With knowledge of u^n and $\frac{\partial u}{\partial t}\Big|^n$, a time-marching scheme can be used to find the solution at the next time level u^{n+1}. Many such schemes exist, but this work will focus on explicit Runge-Kutta time-

marching methods. To use a Runge-Kutta scheme the temporal derivative is rewritten

$$\left.\frac{\partial u}{\partial t}\right|_{num}^{n} = -i\omega^* u^n = F(u^n) \tag{1.24}$$

The function $F(u^n)$ in Equation (1.24) is then used to evaluate the various stages of the Runge-Kutta scheme. For instance, the "classical" RK4 4th-order scheme can be written as

$$\begin{aligned} u_0 &= u^n \\ k_1 &= \Delta t F(u_0) \\ k_2 &= \Delta t F(u_0 + \tfrac{1}{2}k_1) \\ k_3 &= \Delta t F(u_0 + \tfrac{1}{2}k_2) \\ k_4 &= \Delta t F(u_0 + k_3) \\ u^{n+1} &= u^n + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned} \tag{1.25}$$

If the function $F(u^n)$ of Equation (1.24) is substituted into Equation (1.25)

$$u^{n+1} = \left(1 + (i\omega^*\Delta t) + \frac{1}{2}(i\omega^*\Delta t)^2 + \frac{1}{6}(i\omega^*\Delta t)^3 + \frac{1}{24}(i\omega^*\Delta t)^4\right) u^n \tag{1.26}$$

we see that the RK4 scheme is effectively using a Taylor series (in this case, 4th-order, though not all Runge-Kutta schemes are 4th-order) to integrate the equations in time — all explicit Runge-Kutta schemes can be expressed in this form, and each scheme differs only in the number of stages and the value of the leading coefficients for each term of the Taylor series in Equation (1.26). The amplitude $\left|\frac{u^{n+1}}{u^n}\right|$ is only a function of $\omega^*\Delta t$ and is shown for each of the time-marching schemes considered for this work in Figure 1-4. For stability, $\left|\frac{u^{n+1}}{u^n}\right| < 1$, so the stability limits for each scheme can be seen from Figure 1-4. Similar to the $(k\Delta x)^*$ plot in Figure 1-1, each scheme approaches the exact solution as $\omega^*\Delta t$ goes to zero. It appears from Figure 1-4 that the Hu RK56 [20] scheme is the most accurate, followed closely by the HALE_RK7 [21].

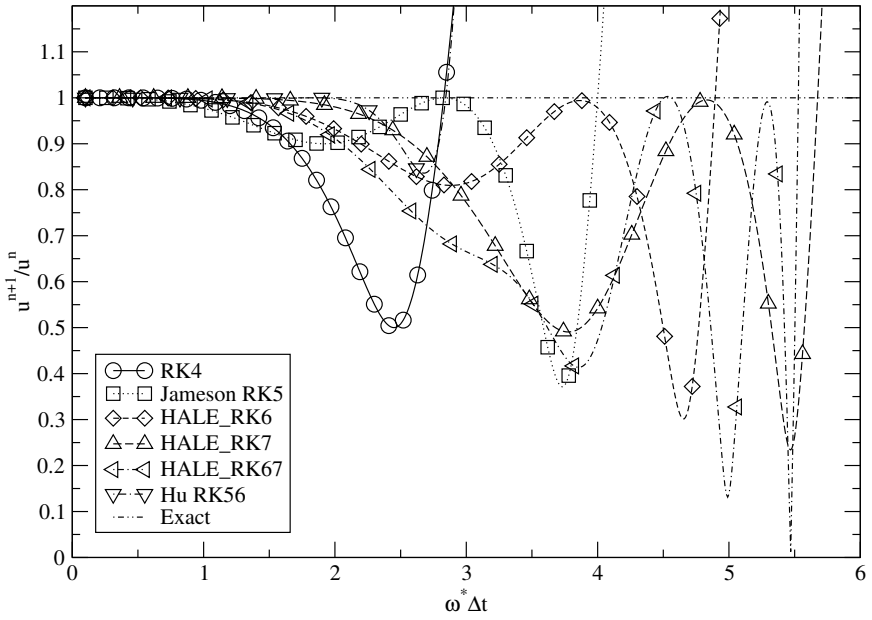


Figure 1-4: Amplification Factor of Various Runge-Kutta Time-Marching Schemes

The error magnitude of each time-marching scheme can be calculated using the expression

$$\varepsilon(\omega^*\Delta t) = \left| \frac{u^{n+1}}{u^n} - e^{i\omega^*\Delta t} \right| \tag{1.27}$$

Figure 1-5 shows the error magnitude for the various Runge-Kutta schemes considered in this work. As expected, the Hu RK56 and HALE_RK7 schemes are the most accurate. Like in the corresponding plot for the spatial differencing schemes, the slope of the error curves for each scheme indicates the observed order-of-accuracy of the scheme. After using the chain rule in a fashion similar to Equation (1.22), the observed order-of-accuracy can be plotted as a function of $\omega^*\Delta t$, as in Figure 1-6. Like in Figure 1-3, each scheme eventually achieves it's formal order-of-accuracy (4th for all schemes except the Jameson RK5 [22], which is 2nd-order) as the numerical frequency goes to zero. Note that the truncation error of a pth-order Runge-Kutta scheme for one time step is actually on the order of Δt^{p+1} — this is explained in more

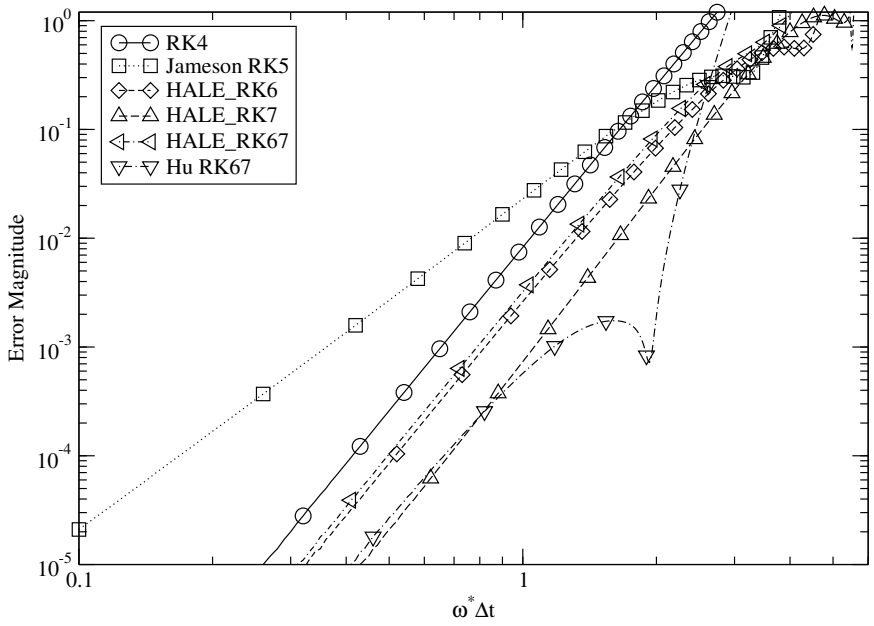


Figure 1-5: Error Magnitude of Various Runge-Kutta Time-Marching Schemes

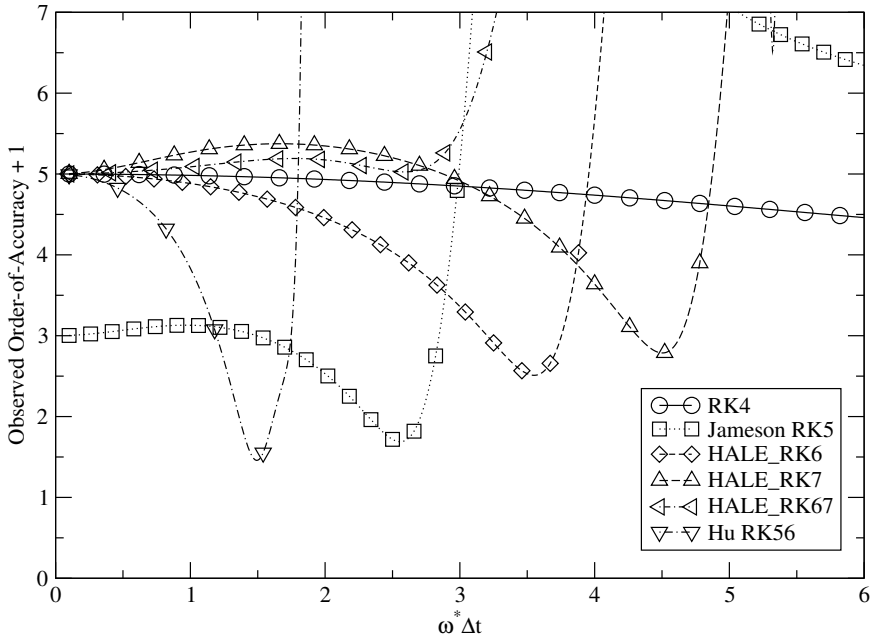


Figure 1-6: Observed Order-of-Accuracy of Various Runge-Kutta Time-Marching Schemes

detail in Section 2.2.5. The Hu RK56 scheme is similar to Tam and Webb's DRP spatial differencing scheme in that each is heavily optimized for accuracy and thus only exhibits its formal order-of-accuracy for a short range of numerical frequency or wavenumber, respectively. As with the DRP, one would expect the Hu RK56 to be the most difficult to Verify.

An additional Runge-Kutta scheme is used in this work but is not discussed here: the RK4 scheme of Jameson [22]. The Jameson RK4 scheme is, for linear problems, identical to the classical RK4 scheme, but is 2nd-order accurate for nonlinear problems. The Jameson RK4 "lost" its 4th-order accuracy when it was recast in a low-storage form by Jameson.

Chapter 2

The External Verification Analysis Method

The External Verification Analysis (EVA) method is used to strongly Verify the numerical schemes employed in codes used to solve partial differential equations, such as CFD/CAA codes. As the name implies, it is external, i.e., it does not require any access to or modification of the source code, unlike the Method of Manufactured Solutions (MMS). Like MMS, the EVA method works by providing a very accurate solution to the governing equations the simulation code solves. Grid convergence or time step studies are then preformed to Verify the theoretical order-of-accuracy of the spatial differencing or time-marching scheme, respectively. The mathematical foundation and EVA process will be described in the following sections, as well as details of the implementation of the EVA method in "EVA3," a FORTRAN 2003 code.

2.1 Mathematical Foundation of EVA

The External Verification Analysis (EVA) method will be applied to the two- and three-dimensional Euler equations in the present work, in addition to the linear

and nonlinear forms of the inviscid Burgers' equation. The three-dimensional Euler equations can be written as

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} + \frac{\partial \vec{G}}{\partial z} = 0 \tag{2.1}$$

where $\vec{Q}$ is the vector of conserved flow variables and $\vec{E}$, $\vec{F}$ and $\vec{G}$ are the Cartesian flux vectors in the x, y and z directions, respectively.

Equation (2.1) can be rewritten as:

$$\frac{\partial \vec{Q}}{\partial t} = -\left[\frac{\partial \vec{E}(\vec{Q})}{\partial x} + \frac{\partial \vec{F}(\vec{Q})}{\partial y} + \frac{\partial \vec{G}(\vec{Q})}{\partial z}\right] \tag{2.2}$$

and, if the conserved flow variables are continuous in space at a time t_0,

$$\left.\frac{\partial \vec{Q}}{\partial t}\right|_{t=t0} = -\left[\frac{\partial \vec{E}}{\partial \vec{Q}}\frac{\partial \vec{Q}}{\partial x} + \frac{\partial \vec{F}}{\partial \vec{Q}}\frac{\partial \vec{Q}}{\partial y} + \frac{\partial \vec{G}}{\partial \vec{Q}}\frac{\partial \vec{Q}}{\partial z}\right]_{t=t0} \tag{2.3}$$

where the derivatives of the flux vectors with respect to $\vec{Q}$ are the flux Jacobian matrices.

One can obtain higher temporal derivatives of $\vec{Q}$ by taking the t-derivative of Equation (2.3):

$$\left.\frac{\partial^2 \vec{Q}}{\partial t^2}\right|_{t=t_0} = -\left[\frac{\partial^2 \vec{E}}{\partial x \partial t} + \frac{\partial^2 \vec{F}}{\partial y \partial t} + \frac{\partial^2 \vec{G}}{\partial z \partial t}\right]_{t=t_0} \tag{2.4}$$

Analytical expressions for the mixed temporal derivatives can be found by using the flux Jacobian matrices and by taking spatial derivatives of (2.3). For example,

$$\left.\frac{\partial^2 \vec{E}}{\partial x \partial t}\right|_{t=t_0} = \left[\frac{\partial^2 \vec{E}}{\partial \vec{Q}^2}\frac{\partial \vec{Q}}{\partial x}\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial \vec{Q}}\frac{\partial^2 \vec{Q}}{\partial x \partial t}\right]_{t=t_0} \tag{2.5}$$

and

$$\left.\frac{\partial^2 \vec{Q}}{\partial x \partial t}\right|_{t=t_0} = -\left[\frac{\partial^2 \vec{E}}{\partial x^2} + \frac{\partial^2 \vec{F}}{\partial x \partial y} + \frac{\partial^2 \vec{G}}{\partial x \partial z}\right]_{t=t_0} \tag{2.6}$$

This process can be repeated to obtain analytical expressions for higher-order temporal derivatives of $\vec{Q}$ in terms of $\vec{Q}$ and its spatial derivatives. Once these expressions are found, they can be used to build a Taylor series centered at $t = t_0$ to obtain the flow at the time $t = t_0 + \Delta t$:

$$\vec{Q}\Big|_{x,y,z,t_0+\Delta t} = \vec{Q}\Big|_{x,y,z,t_0} + \Delta t \frac{\partial \vec{Q}}{\partial t}\bigg|_{x,y,z,t_0} + \frac{\Delta t^2}{2}\frac{\partial^2 \vec{Q}}{\partial t^2}\bigg|_{x,y,z,t_0} + \frac{\Delta t^3}{6}\frac{\partial^3 \vec{Q}}{\partial t^3}\bigg|_{x,y,z,t_0} + \frac{\Delta t^4}{24}\frac{\partial^4 \vec{Q}}{\partial t^4}\bigg|_{x,y,z,t_0} + \cdots \tag{2.7}$$

By increasing the number of terms included in Equation (2.7), a solution of arbitrary accuracy can be found at a later time level ($t = t_0 + \Delta t$). Practically, the order of Equation (2.7) is limited by computational constraints — especially the amount of memory available, as the analytic expressions of the higher-order derivatives of the two- and three-dimensional Euler equations contain many terms and are computationally intensive to evaluate. Currently, the EVA tool can calculate up to an eleventh-order Taylor series for the two-dimensional Euler equations and a tenth-order Taylor series for the three-dimensional Euler equations, and a thirteenth-order order series for the one-dimensional Euler and linear and nonlinear inviscid Burgers' equations. These limits could be increased with the availability of more memory — they are not limitations "built-in" to the code itself.

2.2 The EVA Process

Section 2.1 shows how the flow at a later time level can be found using the EVA method; however, obtaining an EVA solution is only one step in Verifying a CFD/CAA code. The actual process of Verifying a CFD/CAA code can be summarized in the following steps:

1. Construct a grid and specify an initial condition and final time level for the EVA method, and then use the EVA method to obtain the solution at the later

time level.

2. Run the code to be Verified using the same initial condition using a range of grid spacings or time steps to Verify either the spatial or temporal discretization scheme, respectively.

3. Calculate some error norm of each simulation run using the EVA solution as an "exact" reference solution.

4. Calculate the rate of convergence of the error norm found in step 3 (i.e., the observed order-of-accuracy) and compare to the theoretical order-of-accuracy of the numerical scheme used.

Each step will be described in detail in this section.

2.2.1 Step 1: Obtain a Reference Solution using EVA

The first step in the EVA process must be to specify an initial condition, grid and final time level that will be used for both the EVA method and the simulation. The grid used for the EVA method can be identical to the grid used in the simulation code; however, using a smaller, coarser grid (i.e., a grid containing less points) has been found to be much more computationally efficient and to have no discernible effect on the Verification process. If the EVA grid boundary locations are identical to the simulation grid boundary locations care must be taken to ensure that no flow gradients exist in the vicinity of the boundaries as the EVA method does not enforce boundary conditions. Since the calculation of the EVA solution at a given grid point is not dependent on the solution at adjacent grid points the grid file format need not contain connectivity information, and the use of nonuniform, curvilinear or fully-unstructured grids require no special considerations or computational techniques.

The initial condition specified must be an analytical one, as the EVA method relies on being able to find analytical expressions of the spatial derivatives of the initial

condition. The initial condition should be smooth and possess many derivatives. Again, it may be prudent to design the initial condition's gradients to approach zero at the boundaries of the simulation domain so that any boundary conditions enforced in the simulation are not exercised.

2.2.2 Step 2: Run the Code to be Verified

After the EVA solution has been found, the code to be Verified is run using the same initial condition used in the EVA method. The analytical distribution of the initial condition must be evaluated at discrete points for numerical codes. The simulation grid should contain the EVA solution grid points and, as indicated previously, will also likely be much larger and contain more grid points than the EVA grid.

When Verifying the spatial discretization scheme the simulation code is run on a series of progressively finer grids for a single time step in order to observe the grid spacing's effect on the accuracy of the solution calculated by the simulation code. The EVA method does not require integer grid refinement (i.e., dividing the grid spacing of the previous grid by some integer to obtain the next grid's grid spacing) — in practice, integer grid refinement even by a factor of two (for instance, two-dimensional grids with 101^2, 201^2 401^2, 801^2, $1601^2 \ldots$ grid points), will quickly produce very dense grids. A better solution might be to add an integer number of grid points between each grid point in the coarsest grid (i.e., two-dimensional grids with 101^2, 201^2, 301^2, 401^2, $501^2 \ldots$ grid points). This latter approach will increase the number of simulation runs that can be performed before the grids become prohibitively large, allowing one to "zoom in" on a wider range of grid spacings, which may be necessary when Verifying schemes that do not uniformly converge at their "formal" order-of-accuracy.

To insure that the numerical error of the simulation code is dominated by the spatial discretization error a very small time step should be used. It may also be

advisable to use the most accurate time-marching scheme available; however, with a sufficiently small time step this may not be necessary.

When Verifying the temporal discretization scheme the simulation code is run using progressively smaller time step sizes in order to observe the time step's effect on the accuracy of the simulation code's solution. Similar to the spatial case, integer time step refinement is not required and may actually be detrimental; by reducing the time step so aggressively the range of time steps for which the scheme converges at its "formal" rate may be missed.

Again like the spatial case, a very fine grid spacing should be used in conjunction with a high-resolution spatial scheme to keep the spatial discretization error much lower than the temporal discretization error. Care must be taken to avoid increasing the Courant number ($c\frac{\Delta t}{\Delta x}$, where c is the maximum propagation speed of a disturbance), however, beyond the stability limit of the temporal scheme being used, though the calculation may avoid becoming unstable if only a few time steps are taken (as may be the case for the larger time step sizes).

2.2.3 Step 3: Calculate the Error Norm

After the EVA solution and each of the simulation solutions have been obtained the error norm for each simulation solution is calculated using the EVA solution as the "exact" solution. Two error norms have been used with the EVA method thus far: the L_2 and $L_{\max}$ error norms. The latter error norm is simply the maximum error found in the problem domain, as indicated by Salari and Knupp [11]:

$$L_{\max} = \max_n \mid u_n - U_n \mid \tag{2.8}$$

where u and U are different solutions evaluated at n common points.

The L_2 error norm is essentially the root-mean-square of the error:

$$L_2 = \sqrt{\frac{\sum_n (u_n - U_n)^2 \alpha_n}{\sum_n \alpha_n}} \tag{2.9}$$

where α is some measure of the local volume of the grid. Salari and Knupp point out that since $u_n - U_n$ should be on the order of Δ^p, where Δ is the grid spacing or time step and p is the order of the numerical scheme used, then one can replace Equation (2.9) with

$$L_2 = \sqrt{\frac{\sum_n (u_n - U_n)^2}{N}} \tag{2.10}$$

where N is the total number of grid points, so no special efforts need to be made to accommodate non-uniform grids. This change will effect the magnitude of the error calculated for each run but will not effect the order-of-accuracy calculation, as Equation (2.10) will still be on the order of Δ^p.

The EVA3 Code contains routines to calculate both the L_2 and $L_{\max}$ error norms, but *very* little difference can be discerned when comparing the results using the two methods, and both appear to be perfectly adequate for Verification.

2.2.4 Step 4: Calculate the Observed Order-of-Accuracy

After the error for each of the simulation runs have been calculated, the order-of-accuracy is calculated and compared to the theoretical order-of-accuracy of the numerical scheme used. If the observed order-of-accuracy matches the theoretical order-of-accuracy the code is Verified. The discussion to follow can be found in sources by Roy [12], Salari and Knupp [10] or Roache [6].

The error of most numerical schemes is a function of some measure of discretization (grid spacing or time step, typically):

$$\varepsilon(\Delta) = f(\Delta) - f_{exact} \tag{2.11}$$

where ε is the error. Since many numerical schemes are based on Taylor series expansions, one can also write

$$\varepsilon(\Delta) = f(\Delta) - f_{exact} = C\Delta^p + HOT \tag{2.12}$$

where ε is again the error, p is the order of the scheme, HOT stands for higher-order terms, and C is some constant that is independent of Δ. If there are two simulation solutions ε_1 and ε_2 with two discretization Δ_1 and Δ_2, and both solutions are in the asymptotic range, then HOT can be neglected and

$$\frac{\varepsilon_1(\Delta_1)}{\varepsilon_2(\Delta_2)} = \frac{C\Delta_1^p}{C\Delta_2^p} = \left(\frac{\Delta_1}{\Delta_2}\right)^p \tag{2.13}$$

then, rearranging,

$$p = \frac{\log\left(\frac{\varepsilon_1}{\varepsilon_2}\right)}{\log\left(\frac{\Delta_1}{\Delta_2}\right)} \tag{2.14}$$

The use of Equation (2.14) is straightforward when Verifying the spatial discretization scheme of a code on grids with uniform spacing, or when Verifying temporal discretization schemes. When Verifying spatial discretization schemes on nonuniform grids, however, Δ is no longer constant and the evaluation of p using Equation (2.14) may not be clear. In this work the equation

$$p = \frac{\log\left(\frac{\varepsilon_1}{\varepsilon_2}\right)}{\log\left(\frac{\sqrt[q]{N_2-1}}{\sqrt[q]{N_1-1}}\right)} \tag{2.15}$$

was used to find the observed order-of-accuracy when Verifying spatial discretization schemes, where N is the number of grid points and q is the number of spatial dimensions of the domain. Using Equation (2.15) with logically square (for 2D) or cubic (for 3D) domains removes any ambiguity when using nonuniform grids and has been found to work well in the cases considered here.

2.2.5 An Alternative Method for Verifying Temporal Order-of-Accuracy — Using Multiple EVA Solutions

An alternative method for Verifying the temporal order-of-accuracy of a CFD/CAA code involves using the EVA method to obtain the solution to the governing equations at *multiple* time levels (this is the approach used by Hixon and Anderson in [3]). The code to be Verified is then marched one time step to each of the aforementioned time levels and the error norms and order-of-accuracy are computed as described in Sections 2.2.3 and 2.2.4. This approach has the advantage of requiring a less-accurate Taylor series (i.e., less terms in Equation (2.7)) than the single-EVA-solution method. When using the single-EVA-solution method, the user must include enough terms in the Taylor series to drive the truncation error down to less than the truncation error of the simulation code run with the greatest number of time steps (i.e., the smallest Δt). For example, if the final time level chosen for the EVA and simulation runs is Δt and the maximum number of time steps that will be used by the simulation to march to the Δt final time level is n, then the truncation error of the EVA solution is

$$\varepsilon_{EVA} \sim (\Delta t)^{p+1} \tag{2.16}$$

where p is the number of terms used in the Taylor series used to obtain the EVA solution, and the smallest truncation error of the simulation runs will be (assuming the error from the spatial discretization is negligible)

$$\varepsilon_{MinSim} \sim \left(\frac{\Delta t}{n}\right)^{q} \tag{2.17}$$

where q is the order of the time marching scheme used by the simulation code. To be able to use the EVA solution to Verify the simulation code, ε_{EVA} must be less than ε_{MinSim} — to achieve this, p must generally be much larger than q. If, instead,

multiple EVA solutions are used, then the truncation error of the EVA solution at a given Δt would be

$$\varepsilon_{EVA} \sim (\Delta t_i)^{p+1} \tag{2.18}$$

and the truncation error of a given simulation solution would be

$$\varepsilon_{Sim} \sim (\Delta t_i)^{q+1} \tag{2.19}$$

and to insure that $\varepsilon_{EVA} < \varepsilon_{Sim}$ only requires $p > q$.

Using the multiple-EVA-solution method requires calculating the observed order-of-accuracy slightly differently, as the order of the error of one time step of a Runge-Kutta method is actually one greater than the total accumulated error over multiple time steps. So, for instance, the "classical" 4th-order RK4 method will actually appear to be a 5th-order time marching scheme if Equation (2.14) is used. One can show this by expressing the approximate solution $\vec{Q}$ obtained from a pth Runge-Kutta scheme at the $(n+1)$th time level using a range of time levels Δt_i in terms of the exact solution $\vec{\mathcal{Q}}$ as

$$\vec{Q}_i^{n+1} = \vec{\mathcal{Q}}^n + \Delta t_i \left.\frac{\partial \vec{\mathcal{Q}}}{\partial t}\right|^n + \frac{\Delta t_i^2}{2} \left.\frac{\partial^2 \vec{\mathcal{Q}}}{\partial t^2}\right|^n + \frac{\Delta t_i^3}{6} \left.\frac{\partial^3 \vec{\mathcal{Q}}}{\partial t^3}\right|^n + \cdots + \frac{\Delta t_i^p}{p!} \left.\frac{\partial^p \vec{\mathcal{Q}}}{\partial t^p}\right|^n \tag{2.20}$$

and the error of the ith run can be evaluated as

$$\varepsilon_i = \left\| \vec{\mathcal{Q}}^{n+1} - \vec{Q}_i^{n+1} \right\| \tag{2.21}$$

If Equation (2.20) is substituted into Equation (2.21) and $\vec{\mathcal{Q}}^{n+1}$ is expanded into a Taylor series centered around the nth time level, then

$$\varepsilon_i = \left\| \frac{\Delta t_i^{p+1}}{(p+1)!} \left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^n \right\| + HOT \tag{2.22}$$

will be the form of the error, where HOT indicates higher-order terms. If the rate of convergence of the error is found using the ith and $(i+1)$th CFD/CAA run, then

$$\begin{aligned}
\frac{\log\left(\frac{\varepsilon_i}{\varepsilon_{i+1}}\right)}{\log\left(\frac{\Delta t_i}{\Delta t_{i+1}}\right)} &= \frac{\log\left(\frac{\frac{\Delta t_i^{p+1}}{(p+1)!}\frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}}}{\frac{\Delta t_{i+1}^{p+1}}{(p+1)!}\frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}}}\right)}{\log\left(\frac{\Delta t_i}{\Delta t_{i+1}}\right)} \\
&= \frac{\log\left(\left(\frac{\Delta t_i}{\Delta t_{i+1}}\right)^{p+1}\right)}{\log\left(\frac{\Delta t_i}{\Delta t_{i+1}}\right)} \\
&= p+1
\end{aligned} \tag{2.23}$$

and it is clear that the error will converge at a rate one order higher than the order-of-accuracy of the time-marching scheme, and that the equation that should be used to evaluate the observed order-of-accuracy is

$$p = \frac{\log\left(\frac{\varepsilon_i}{\varepsilon_{i+1}}\right)}{\log\left(\frac{\Delta t_i}{\Delta t_{i+1}}\right)} - 1. \tag{2.24}$$

As described earlier, the single-EVA-solution approach advances the solution from the nth to the $(n+1)$th time level using a varying number of time steps, and Equation (2.14) is the correct equation for evaluating the observed order-of-accuracy. Again, this can be seen by expanding the calculated solution using a Taylor series. If a pth-order Runge-Kutta scheme is used to with just one time step, then

$$\vec{Q}^{n+1} = \vec{\mathcal{Q}}^n + \Delta t \left.\frac{\partial \vec{\mathcal{Q}}}{\partial t}\right|^n + \frac{\Delta t^2}{2}\left.\frac{\partial^2 \vec{\mathcal{Q}}}{\partial t^2}\right|^n + \cdots + \frac{\Delta t^p}{p!}\left.\frac{\partial^p \vec{\mathcal{Q}}}{\partial t^p}\right|^n \tag{2.25}$$

and the error can be found

$$\varepsilon_{1,total} = \left\|\vec{\mathcal{Q}}^{n+1} - \vec{Q}^{n+1}\right\| = \left\|\frac{\Delta t^{p+1}}{(p+1)!}\left.\frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^n\right\| + HOT \tag{2.26}$$

where $\vec{\mathcal{Q}}$ has been expanded in a Taylor series centered around n and the subscript

on ε indicates the total number of time steps used to advance the solution to $(n+1)$. If two time steps are used to advance the solution to the same time level the time step size will be $\frac{\Delta t}{2}$, and

$$\vec{Q}^{n+\frac{1}{2}} = \vec{\mathcal{Q}}^{n} + \left(\frac{\Delta t}{2}\right) \left.\frac{\partial \vec{\mathcal{Q}}}{\partial t}\right|^{n} + \frac{\left(\frac{\Delta t}{2}\right)^2}{2} \left.\frac{\partial^2 \vec{\mathcal{Q}}}{\partial t^2}\right|^{n} + \cdots + \frac{\left(\frac{\Delta t}{2}\right)^p}{p!} \left.\frac{\partial^p \vec{\mathcal{Q}}}{\partial t^p}\right|^{n} \tag{2.27}$$

then the error from the first time step can be found

$$\varepsilon_2^{n+\frac{1}{2}} = \left\| \vec{\mathcal{Q}}^{n+\frac{1}{2}} - \vec{Q}^{n+\frac{1}{2}} \right\| = \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^{n} \right\| + HOT \tag{2.28}$$

where, again, $\vec{\mathcal{Q}}^{n+\frac{1}{2}}$ has been expanded using a Taylor series centered at n. The calculated solution at the $(n+\frac{1}{2})$th time level can then be written as

$$\vec{Q}^{n+\frac{1}{2}} = \vec{\mathcal{Q}}^{n+\frac{1}{2}} + \varepsilon_2^{n+\frac{1}{2}} \tag{2.29}$$

Then the solution is marched from the $(n+\frac{1}{2})$th to the $(n+1)$th time level

$$\vec{Q}^{n+1} = \vec{Q}^{n+\frac{1}{2}} + \left(\frac{\Delta t}{2}\right) \left.\frac{\partial \vec{Q}}{\partial t}\right|^{n+\frac{1}{2}} + \frac{\left(\frac{\Delta t}{2}\right)^2}{2} \left.\frac{\partial^2 \vec{Q}}{\partial t^2}\right|^{n+\frac{1}{2}} + \cdots + \frac{\left(\frac{\Delta t}{2}\right)^p}{p!} \left.\frac{\partial^p \vec{Q}}{\partial t^p}\right|^{n+\frac{1}{2}} \tag{2.30}$$

The "exact" and "error" parts of $\vec{Q}^{n+\frac{1}{2}}$ can be considered separately:

$$\widetilde{Q}^{n+1} = \vec{\mathcal{Q}}^{n+\frac{1}{2}} + \left(\frac{\Delta t}{2}\right) \left.\frac{\partial \vec{\mathcal{Q}}}{\partial t}\right|^{n+\frac{1}{2}} + \frac{\left(\frac{\Delta t}{2}\right)^2}{2} \left.\frac{\partial^2 \vec{\mathcal{Q}}}{\partial t^2}\right|^{n+\frac{1}{2}} + \cdots + \frac{\left(\frac{\Delta t}{2}\right)^p}{p!} \left.\frac{\partial^p \vec{\mathcal{Q}}}{\partial t^p}\right|^{n+\frac{1}{2}} \tag{2.31}$$

where $\widetilde{Q}$ is "solution" part of the calculated $\vec{Q}$ that matches the exact solution $\vec{\mathcal{Q}}$ through the pth Taylor series term. The error from the previous time step will be

advanced in the same manner as the solution:

$$\varepsilon_2^{n+1} = \varepsilon_2^{n+\frac{1}{2}} + \left(\frac{\Delta t}{2}\right) \left.\frac{\partial \varepsilon_2}{\partial t}\right|^{n+\frac{1}{2}} + \frac{\left(\frac{\Delta t}{2}\right)^2}{2} \left.\frac{\partial^2 \varepsilon_2}{\partial t^2}\right|^{n+\frac{1}{2}} + \cdots + \frac{\left(\frac{\Delta t}{2}\right)^p}{p!} \left.\frac{\partial^p \varepsilon_2}{\partial t^p}\right|^{n+\frac{1}{2}} \tag{2.32}$$

then the total error accumulated over the two time steps is

$$\begin{aligned} \varepsilon_{2,total} &= \left\| \vec{\mathcal{Q}}^{n+1} - \vec{Q}^{n+1} \right\| \\ &= \left\| \vec{\mathcal{Q}}^{n+1} - \left(\widetilde{Q}^{n+1} + \varepsilon_2^{n+1} \right) \right\| \\ &= \left\| \vec{\mathcal{Q}}^{n+1} - \widetilde{Q}^{n+1} \right\| + \left\| \varepsilon_2^{n+1} \right\| \end{aligned} \tag{2.33}$$

The expression $\left\| \vec{\mathcal{Q}}^{n+1} - \widetilde{Q}^{n+1} \right\|$ can be evaluated by inspecting Equation (2.31) and expanding $\vec{\mathcal{Q}}^{n+1}$ using a Taylor series centered at the $(n+\frac{1}{2})$th time level, and then expanding the result using a Taylor series centered at the nth time level:

$$\begin{aligned} \left\| \vec{\mathcal{Q}}^{n+1} - \widetilde{Q}^{n+1} \right\| &= \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^{n+\frac{1}{2}} \right\| \\ &= \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left[\left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^{n} + \Delta t \left.\frac{\partial^{p+2} \vec{\mathcal{Q}}}{\Delta t^{p+2}}\right|^{n} + \cdots \right] \right\| + HOT \\ &= \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^{n} \right\| + HOT \end{aligned} \tag{2.34}$$

where any terms multiplied by powers of Δt greater than $p+1$ have been absorbed into HOT. The ε_2^{n+1} term can be evaluated by substituting Equation (2.28) into Equation (2.32):

$$\begin{aligned} \varepsilon_2^{n+1} &= & \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^{n} + HOT \right\| \\ &+ \left(\frac{\Delta t}{2}\right) & \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left.\frac{\partial^{p+2} \vec{\mathcal{Q}}}{\partial t^{p+2}}\right|^{n} + HOT \right\| \\ &+ & \cdots \\ &= & \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left.\frac{\partial^{p+1} \vec{\mathcal{Q}}}{\partial t^{p+1}}\right|^{n} \right\| + HOT \end{aligned} \tag{2.35}$$

where, again, the higher-order Δt terms have been included in HOT. The total error

accumulated over the two time steps is then

$$\begin{aligned} \varepsilon_{2,total} &= \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left. \frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}} \right|^n \right\| + \left\| \frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left. \frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}} \right|^n \right\| + HOT \\ &= \left\| 2\frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!} \left. \frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}} \right|^n \right\| + HOT \end{aligned} \tag{2.36}$$

Now the convergence rate can be calculated using $\varepsilon_{1,total}$ and $\varepsilon_{2,total}$, giving

$$\begin{aligned} \frac{\log\left(\frac{\varepsilon_{1,total}}{\varepsilon_{2,total}}\right)}{\log\left(\frac{\Delta t}{\frac{\Delta t}{2}}\right)} &= \frac{\log\left(\frac{\frac{\Delta t^{p+1}}{(p+1)!}\frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}}}{2\frac{\left(\frac{\Delta t}{2}\right)^{p+1}}{(p+1)!}\frac{\partial^{p+1}\vec{\mathcal{Q}}}{\partial t^{p+1}}}\right)}{\log\left(\frac{\Delta t}{\frac{\Delta t}{2}}\right)} \\ &= p \end{aligned} \tag{2.37}$$

as expected.

The disadvantage of the multiple-EVA-solution approach is the added computational cost of computing the necessary multiple EVA solutions; however, the only difference between an EVA solution at two different time levels is the value of Δt in Equation (2.7). So, if an array of Δt's were specified at the beginning, the EVA solution for each time level could be calculated for a given point one after another without needing to reevaluate the many derivatives of $\vec{Q}$. Since most of the computation effort of the EVA method is involved in finding these derivatives, the gains in a less-stringent requirement for the accuracy of the EVA solution outweigh the addition cost of evaluating the EVA Taylor series at multiple Δt's.

One must also be more mindful of the stability limits of the time-marching scheme being Verified when using the multiple-EVA-solution approach. Because the solution is advanced only one time step for each simulation run, the code will likely survive even if the CFL used is well past the stability limit of the scheme and will return observed order-of-accuracy results much higher than the formal order-of-accuracy of the scheme. This superconvergent behavior is actually indicated by Figure 1-6 —

the observed order-of-accuracy for numerical frequencies past the stability limits of each scheme is large. This behavior is not seen when using the single-EVA-solution method (usually the scheme will return large error insensitive to time step size for the unstable time steps, giving very low order-of-accuracy until a stable time step is used), and may be misleading if the stability limits are not known to the user.

2.3 The EVA3 Code — Implementation of the EVA Method

The method of solving the governing equations of fluid flow described in Section 2.1 has been implemented in a FORTRAN 2003 computer code called "EVA3". Aspects of its implementation are discussed in this section.

2.3.1 Using the Flux Jacobians with the EVA Method

After exploring the alternatives, the most computationally efficient approach to implementing the EVA method was found to involve the flux Jacobians. The flux Jacobians are found by taking derivatives of the flux vectors with respect to the solution vector (as was done in Section 2.1). For example, consider the one-dimensional Euler equation

$$\vec{Q}_t + \vec{F}_x = 0 \tag{2.38}$$

where the subscripts denote differentiation. If one writes the solution vector of the one-dimensional Euler equations as

$$\vec{Q} = \left\{ \begin{array}{c} \rho \\ \rho u \\ E_{total} \end{array} \right\} = \left\{ \begin{array}{c} Q_1 \\ Q_2 \\ Q_3 \end{array} \right\} \tag{2.39}$$

the Cartesian flux vector in the x-direction is (assuming an ideal gas: $p = (\gamma - 1)\left[E_{total} - \frac{1}{2}\rho u^2\right]$)

$$\vec{F} = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ u(E_{total} + p) \end{array} \right\} = \left\{ \begin{array}{c} Q_2 \\ \frac{3-\gamma}{2}\frac{Q_2^2}{Q_1} + (\gamma - 1)Q_3 \\ \gamma\frac{Q_2 Q_3}{Q_1} + \frac{1-\gamma}{2}\frac{Q_2^3}{Q_1^2} \end{array} \right\} \tag{2.40}$$

then the Jacobian of the flux $\vec{F}$ is

$$\vec{F}_{\vec{Q}} = \left[\begin{array}{ccc} 0 & 1 & 0 \\ \frac{\gamma-3}{2}\frac{Q_2^2}{Q_1^2} & (3-\gamma)\frac{Q_2}{Q_1} & \gamma - 1 \\ -\gamma\frac{Q_2 Q_3}{Q_1^2} + (\gamma-1)\frac{Q_2^3}{Q_1^3} & \gamma\frac{Q_3}{Q_1} + \frac{3(1-\gamma)}{2}\frac{Q_2^2}{Q_1^2} & \frac{Q_2}{Q_1} \end{array} \right] \tag{2.41}$$

and the 1D Euler equation can be rewritten as

$$\vec{Q}_t + \vec{F}_{\vec{Q}}\vec{Q}_x = 0 \tag{2.42}$$

Then, using the approach described in Section 2.1, higher temporal derivatives of $\vec{Q}$ can be found

$$\vec{Q}_{tt} = -\left(\vec{F}_{\vec{Q}^2}\vec{Q}_t\vec{Q}_x + \vec{F}_{\vec{Q}}\vec{Q}_{xt}\right) \tag{2.43}$$

where

$$\vec{Q}_{xt} = -\left(\vec{F}_{\vec{Q}^2}\vec{Q}_x\vec{Q}_x + \vec{F}_{\vec{Q}}\vec{Q}_{xx}\right) \tag{2.44}$$

In Equations (2.43) and (2.44), $\vec{F}_{\vec{Q}^2}$ is a 3 by 3 by 3 array. The alternative would be to evaluate $\vec{F}_{xt}$ directly — for example, the 3rd entry of $\vec{F}_{xt}$ would be

$$\begin{aligned}\vec{F}_{xt}(3) &= \gamma\frac{Q(2)_{xt}Q(3)}{Q(1)} + \gamma\frac{Q(2)_xQ(3)_t}{Q(1)} \\ &+\gamma\frac{Q(2)_tQ(3)_x}{Q(1)} + \gamma\frac{Q(2)Q(3)_{xt}}{Q(1)} - \gamma\frac{Q(2)Q(3)_xQ(1)_t}{Q(1)^2} \\ &-\gamma\frac{Q(2)_tQ(3)_xQ(1)_x}{Q(1)^2} - \gamma\frac{Q(2)Q(3)_{xt}Q(1)_x}{Q(1)_x} + \gamma\frac{Q(2)Q(3)_xQ(1)_xQ(1)_t}{Q(1)^3} - \gamma\frac{Q(2)Q(3)_xQ(1)_{xt}}{Q(1)^2} \\ &+3(1-\gamma)\frac{Q(2)^2Q(2)_t}{Q(1)^3} + 3(1-\gamma)\frac{Q(2)^3Q(1)_t}{Q(1)^4}\end{aligned} \tag{2.45}$$

Rewriting the governing equations using the flux Jacobians when applying the EVA method turns out to be much more efficient than using the fluxes themselves — it appears that the flux Jacobian approach "factors" the governing equations in a way that requires less operations to evaluate.

2.3.2 Controlling the Accuracy of the EVA Taylor Series Using EVA3

While the procedure described in Section 2.1 can be used to obtain a solution at $t_0 + \Delta t$ of arbitrary accuracy (providing sufficient derivatives of $\vec{Q}(x, y, z, t_0)$ exist), the computational demands of doing so increase greatly with the addition of each term in Equation (2.7). Thankfully, experience has shown that the error in the solution provided by the EVA method only needs to be a few orders-of-magnitude lower than the error of the solution of the code being Verified. It would be useful to be able to control the error of the Taylor Series used in the EVA method in order to balance accuracy with computational efficiency, and the EVA3 code provides two methods. First, the EVA3 code allows the user to set a minimum and maximum order for the EVA solution that will be used throughout the problem domain. Second, the EVA3 code allows the user to specify a maximum desired error of the Taylor Series in Equation (2.7). The error in truncating the series of Equation (2.7) (or any Taylor

Series) can be estimated using Taylor's Theorem (see any introductory calculus book — for example, [23]):

Taylor's Theorem. *Suppose that* $\vec{Q}$ *has* $(n+1)$ *derivatives on the interval* $(t_0 - r, t_0 + r)$ *for some* $r > 0$*, and* $\vec{P}_n(t = t_0 + \Delta t)$ *is the Taylor Series of* $\vec{Q}$ *truncated at* n *terms. Then, for* $t \in (t_0 - r, t_0 + r)$*,* $\vec{Q}(t) \approx \vec{P}_n(t)$ *and the error in using* $\vec{P}_n(t)$ *to approximate* $\vec{Q}(t)$ *is*

$$\vec{R}_n(t) = \vec{Q}(t) - \vec{P}_n(t) = \frac{\vec{Q}_{t^{n+1}}(z)\Delta t^{n+1}}{(n+1)!}$$

for some number z *between* t_0 *and* $t = t_0 + \Delta t$*.*

One can see that $\vec{R}_n(t)$ in Taylor's Theorem is nothing more than the $(n+1)$th term of Equation (2.7), but evaluated at z. Unfortunately, only derivatives of $\vec{Q}$ evaluated at t_0 are available to the EVA method, since we obtain the derivatives of $\vec{Q}$ by using the spatial derivatives of an analytic initial condition (at t_0) and the governing equation to find $\vec{Q}(t_0)_t$. One might suggest using $\vec{Q}_{t^{n+1}}(t_0)$ in place of $\vec{Q}_{t^{n+1}}(z)$ to estimate $\vec{R}_n(t)$, but $\vec{Q}_{t^{n+1}}(t_0)$ might provide an unacceptably poor approximation to $\vec{R}(t)$ in some cases. For instance, it is well-known that alternating Taylor Series terms of $\sin(t)$ or $\cos(t)$ become zero when the Taylor Series is expanded around certain points. If one used only $\vec{Q}_{t^{n+1}}(t_0)$ to estimate $\vec{R}(t)$ in this case one would greatly underestimate the error of using $\vec{P}_n(t)$ to approximate $\vec{Q}(t)$. The natural suggestion would be to also use the $(n+2)$th term of $\vec{P}_n$ to estimate $\vec{R}_n$, and that is precisely what the EVA3 code does. Specifically, the EVA3 code estimates $\vec{R}_n$ as the sum of the absolute value of the $(n+1)$th and $(n+2)$th terms of $\vec{P}_n$ for each (x, y, z) point in the problem domain. If this value is greater than the user's desired maximum error, then the EVA3 code will use an additional term to approximate $\vec{Q}(t)$ at that point. The EVA3 code originally used some logical statements that resulted in approximating $\vec{R}_n$ as the absolute value of the $(n+2)$th term when it (the absolute value of the $(n+2)$th term) exceeded the

absolute value of the $(n+1)$th term and as the absolute value of the $n+1$th term otherwise, but it was decided that the former method was more straightforward and essentially resulted in the same thing.

Also, for systems of equations like the one-, two- or three-dimensional Euler equations, the EVA3 code will consider the user-specified desired error constraint satisfied only when the estimated error for a given Taylor Series order is less than the desired error for every flow variable. In other words, at a given (x, y, z) point the EVA3 code will use a Taylor Series of the same order to estimate each entry of $\vec{Q}$.

Since estimating the truncation error of Equation (2.7) at a given n essentially involves calculating the $(n+1)$th and $(n+2)$th terms, it seemed wasteful to not include both the $(n+1)$th and $(n+2)$th terms of Equation (2.7) in the EVA3 code's approximation of $\vec{Q}(t)$. So, when calculating $\vec{Q}(t)$ at each (x, y, z) point in the problem domain, the EVA3 code will report an estimated error that would result from using *two fewer* Taylor Series terms of Equation (2.7) — so the actual error is likely to be considerably smaller than the EVA3 code suggests.

Chapter 3

Application of the EVA Method to Some Model Equations

3.1 Equations Considered

Because of the complex nature of the governing equations of fluid flow, so-called "Model" equations are frequently used to explore the properties of numerical schemes. Model equations are PDEs that mimic the physical behavior of the actual governing equations, but are much simpler. Often some exact solutions of the model equations are known. Many different model equations are used in CFD [24], but perhaps one of the most common is

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \mu\frac{\partial^2 u}{\partial x^2} \tag{3.1}$$

which is known as Burgers' Equation. Equation (3.1) can be used as a simple analogue of the Navier-Stokes Equations, as it contains (from left to right) an unsteady, nonlinear convection and diffusion term. This work is concerned with the Euler Equations, however, which are the inviscid form of the Navier-Stokes. If μ in Equation (3.1) is

set to 0, the Nonlinear Advection Equation is obtained:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \tag{3.2}$$

It can be shown that, if some initial condition is specified

$$u(x, t = 0) = f(x) \tag{3.3}$$

then

$$u(x, t) = f(x - u(x, t)t) \tag{3.4}$$

is an *implicit* solution of the Nonlinear Advection equation. Physically, this means that an initial disturbance will be propagated at a speed u to the right if $u(x, t) > 0$ at a given x and point in time, to the left otherwise. Because the speed of the disturbance is different from point to point, a solution to (3.2) may steepen and eventually develop discontinuities. Explicit solutions to (3.2) do exist [25] for some initial conditions.

If the u multiplying $\frac{\partial u}{\partial x}$ in Equation (3.2) is replaced by some real constant c

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{3.5}$$

the Linear Advection equation is obtained. Similar to the Nonlinear Advection equation, if some initial condition is specified

$$u(x, t = 0) = f(x) \tag{3.6}$$

then

$$u(x, t) = f(x - ct) \tag{3.7}$$

is an *explicit* solution to the Linear Advection equation, so the solution to (3.5) is immediately know after the initial condition (3.6) is specified. Physically, (3.7) shows

that an initial disturbance will be propagated at a speed c to the right if $c > 0$, to the left otherwise.

The linear and nonlinear forms of the inviscid Burgers' equation are used in this work to demonstrate the capabilities of the EVA3 code.

3.2 Description of the "FD1D" Code

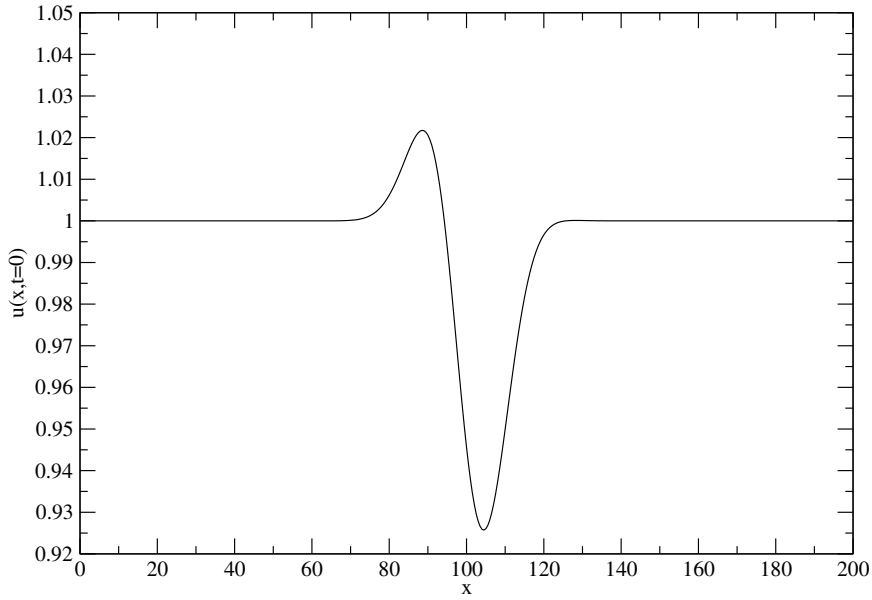
To demonstrate the capabilities of the EVA method and to gain experience before attacking more challenging equations, a simple CFD code was developed that solves the linear and nonlinear forms of the inviscid Burgers' equation. The code is named "FD1D" for Finite Difference One-Dimensional, and can solve the full nonlinear viscous Burgers' equation in addition to the inviscid forms using the finite difference method. Numerous spatial differencing schemes are implemented, including 2nd, 4th and 6th order central differencing and the Dispersion Relation Preserving stencil of Tam and Webb [19]. Explicit time marching is used; 1st order biased time marching is included, as well as Runge-Kutta-style schemes of Jameson [22], Hu [20] and Allampali et al. [21] in addition to the "classical" 4th order Runge-Kutta scheme. Periodic boundary conditions are used in the FD1D code.

For all test cases including both spatial and temporal studies the initial condition used was

$$u(x,0) = 1.0 + 0.1e^{-\ln(2)(x-100.0)^2}\sin(0.1x). \tag{3.8}$$

and is shown in Figure 3-1. The computational domain for the FD1D code extended from 0.0 to 200.0, while the EVA3 code's was restricted to 90.0 to 110.0. The large domain and "compact" initial condition was designed to prevent the periodic boundary conditions from affecting the solution in the center of the domain where the EVA solution was obtained. For each FD1D code run, the L_2 and $L_{\max}$ error norms between the FD1D solution and the EVA solution were computed. The rate of convergence of

Figure 3-1: FD1D Initial Condition



successive FD1D code runs was then found for the spatial test cases using the formula adapted from Salari and Knupp [11] as described in Section 2.2.3:

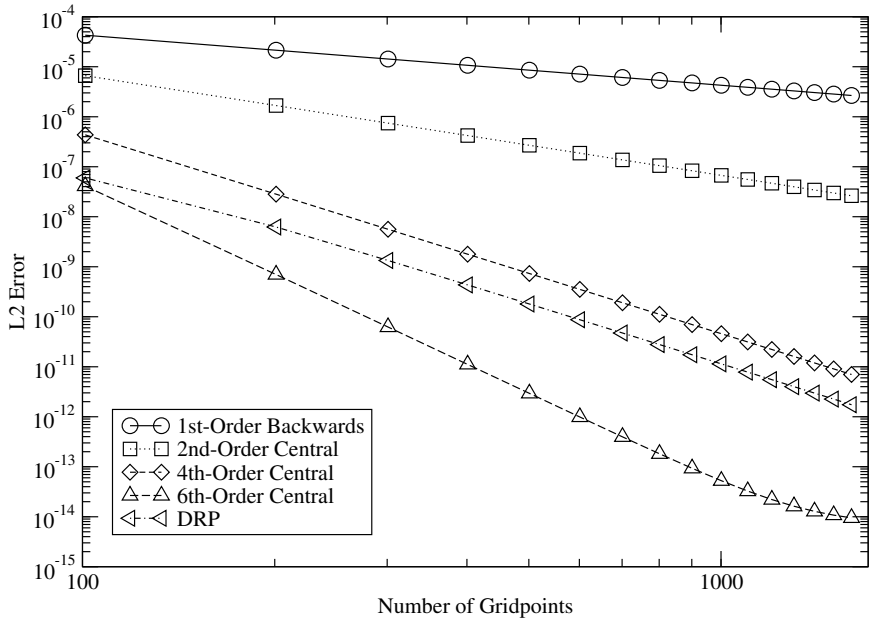
$$p = \frac{\log \frac{\varepsilon_1}{\varepsilon_2}}{\log \frac{N_2 - 1}{N_1 - 1}} \tag{3.9}$$

where p is the observed order-of-accuracy, ε is any error measure, N is the number of grid points in the domain and the subscripts 1 and 2 are used to identify successive FD1D code runs. The observed order-of-accuracy should match the theoretical order-of-accuracy of the spatial differencing scheme used. The observed order-of-accuracy for the time-marching schemes was calculated using

$$p = \frac{\log \frac{\varepsilon_1}{\varepsilon_2}}{\log \frac{\Delta t_1}{\Delta t_2}} \tag{3.10}$$

where Δt is the time step.

Figure 3-2: Linear inviscid Burger — L_2 error norm for the spatial stencils



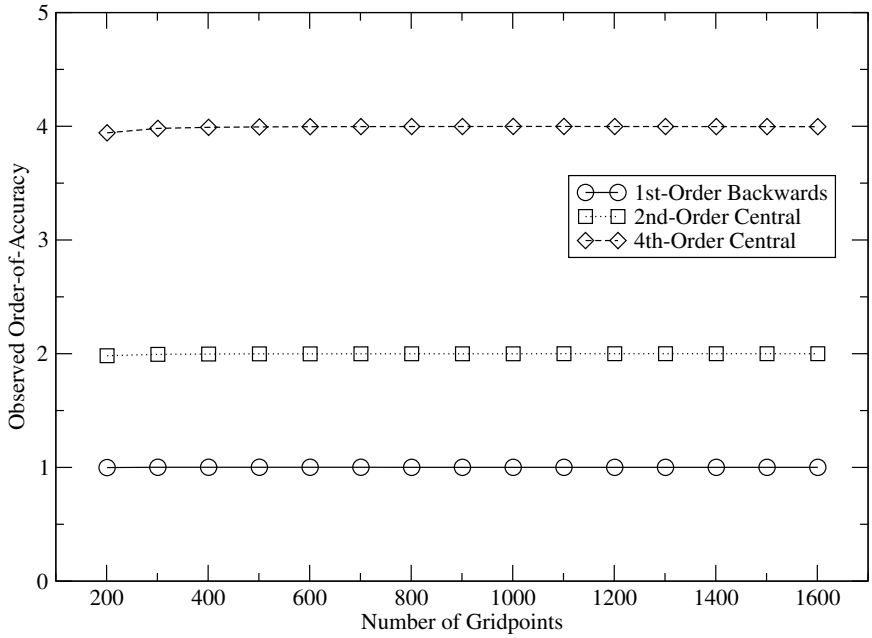
3.3 Spatial Order-of-Accuracy Results

To Verify the spatial order-of-accuracy of the FD1D code, an initial condition and final time was chosen, and the EVA3 code was used to determine the solution at the later time level. The FD1D code was marched a single time step to the final time level using the same initial condition and increasingly fine uniform grids. The final time level for all spatial cases was $t = 0.03125$ — this small time step was used to ensure that the error from the FD1D code's time marching scheme would be much less that spatial differencing's contribution to the error. The nearby final time level also makes obtaining an accurate EVA solution much less demanding. The EVA3 code was run with an 11-point grid evenly spaced from 90.0 to 110.0.

3.3.1 Linear Inviscid Burger

Figure 3-2 shows the L_2 error norms of the FD1D code results when solving the linear form of the inviscid Burgers' equation. As expected, the higher-order schemes are generally more accurate then the lower-order schemes. Note that the DRP is

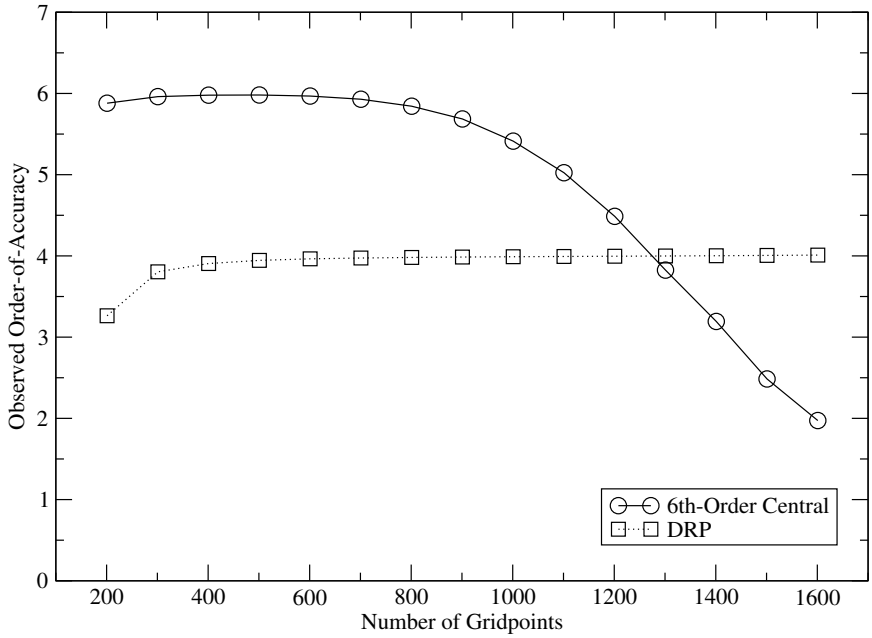
Figure 3-3: Linear inviscid Burger — observed order-of-accuracy for the 1st-Order Backward and 2nd- and 4th-Order Central Stencils



a fourth-order scheme, which explains why the DRP L_2 error curve appears nearly parallel to the 4th-order central L_2 curve. The L_2 error for the 6th-order scheme is flattening out at the very dense grid spacings, showing either that the error has reached machine precision or the accuracy of the EVA solution has been exhausted — the very low error would tend to indicate the former.

Figure 3-3 shows the observed order-of-accuracy of the FD1D code for the first-order backward and second- and fourth-order central spatial stencils, and Figure 3-4 shows the same for the sixth-order central and DRP stencils. Each scheme's observed order-of-accuracy achieves its theoretical order of accuracy (again, the DRP is a fourth-order stencil), indicating that each stencil is correctly implemented in the FD1D code for the linear Burger solver. The sharp drop-off of the 6th-order central curve is again explained by the error reaching machine precision; the 6th-order stencil is so accurate at these high grid densities that the error is too tiny to be distinguished from zero by the computer. All the schemes would eventually experience the same drop in order if the grid density was sufficiently high and the error from the time-

Figure 3-4: Linear inviscid Burger — observed order-of-accuracy for the 6th-Order Central and DRP Stencils



marching scheme was sufficiently low.

3.3.2 Nonlinear Inviscid Burger

The EVA method was also used to Verify the implementation of the spatial stencils in the nonlinear inviscid Burgers' equation solver of the FD1D code. The same initial condition, time step and grids were used as in the linear case (Section 3.3.1). Figure 3-5 shows the L_2 error norms of the FD1D code results for the nonlinear Burgers' equation. The results are nearly identical to Figure 3-2. The flattening of the 6th-order curve is again likely due to the error reaching machine precision.

Figure 3-6 shows the observed order-of-accuracy of the FD1D code for the first-order backward and second- and fourth-order central spatial stencils, and Figure 3-7 shows the same for the sixth-order central and DRP stencils. As in the linear case, each scheme's observed order-of-accuracy is achieved, and again the 6th-order central curve drops off as the error reaches machine precision. Overall, these results and the

Figure 3-5: Nonlinear inviscid Burger — L_2 error norm for the spatial stencils

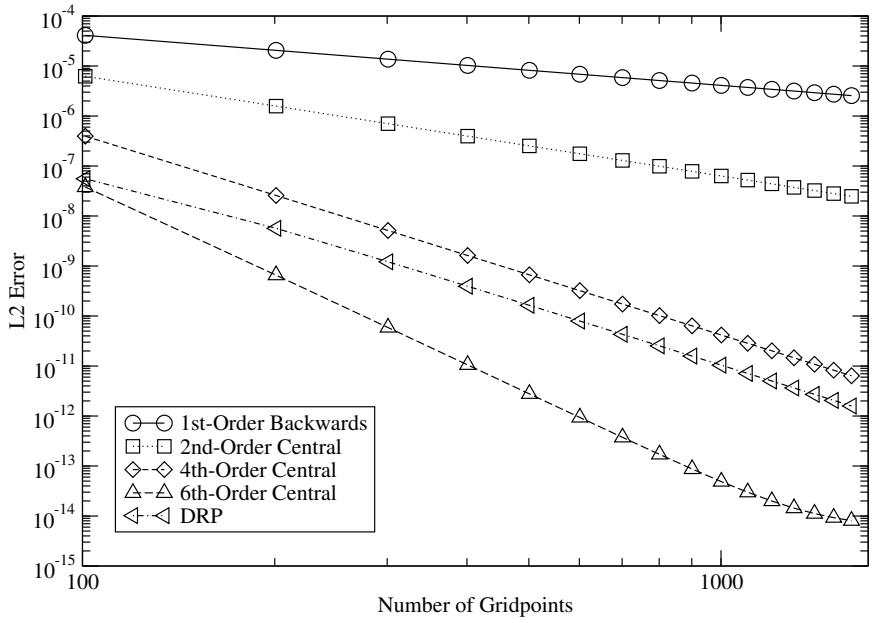


Figure 3-6: Nonlinear inviscid Burger — observed order-of-accuracy for the 1st-Order backward and 2nd- and 4th-order central stencils

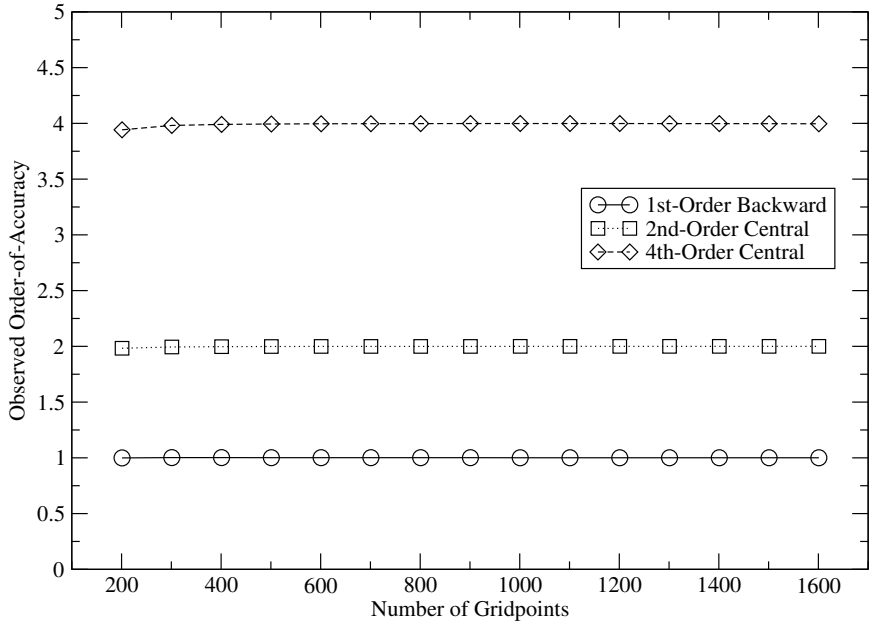
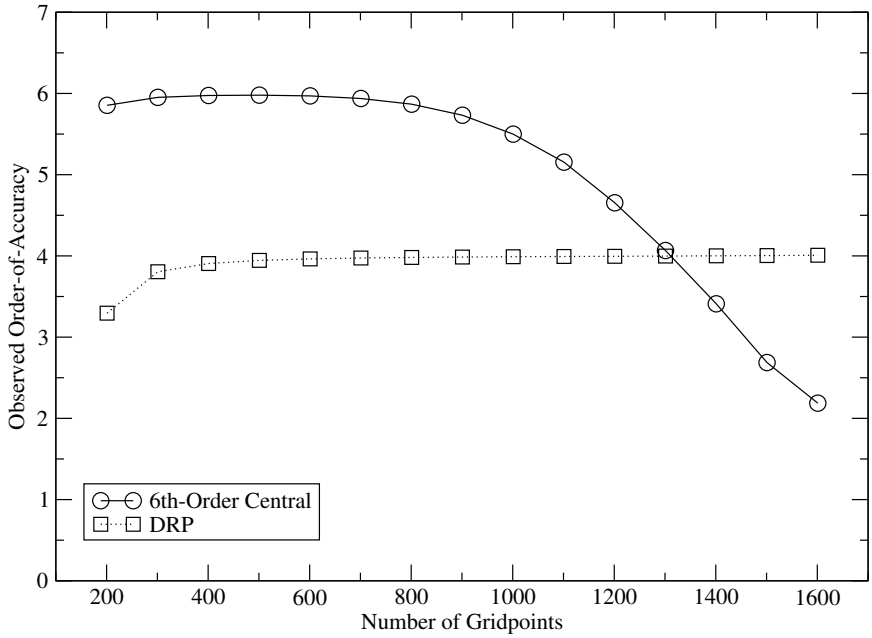


Figure 3-7: Nonlinear inviscid Burger — observed order-of-accuracy — 6th-order central and DRP stencils



results presented in Section 3.3.1 strongly Verify that the FD1D code's spatial stencils are properly implemented and are solving the linear and nonlinear forms of Burgers' equation correctly.

3.4 Temporal Order-of-Accuracy Results — Single EVA Solution

The same initial condition (Figure 3-1) and computational domain used to verify the spatial stencils of the FD1D code was used to Verify the time-marching schemes. The EVA3 code was used to find a solution at $t = 0.25$ for 21 points evenly spaced from 90.0 to 110.0. The FD1D code used a 6401-point grid for all test runs. The very fine grid and later time level was used to ensure most of the error in the FD1D solution would be due to the time-marching scheme and not the spatial stencil (6th-order central for all cases). The FD1D code was marched to the desired time level

using an increasing number of time steps — $1, 2, 3 \ldots 32$ for the single-step schemes and $2, 4, 6 \ldots 64$ for the two-step schemes (the HALE_RK67 and the Hu RK56).

3.4.1 Linear Inviscid Burger

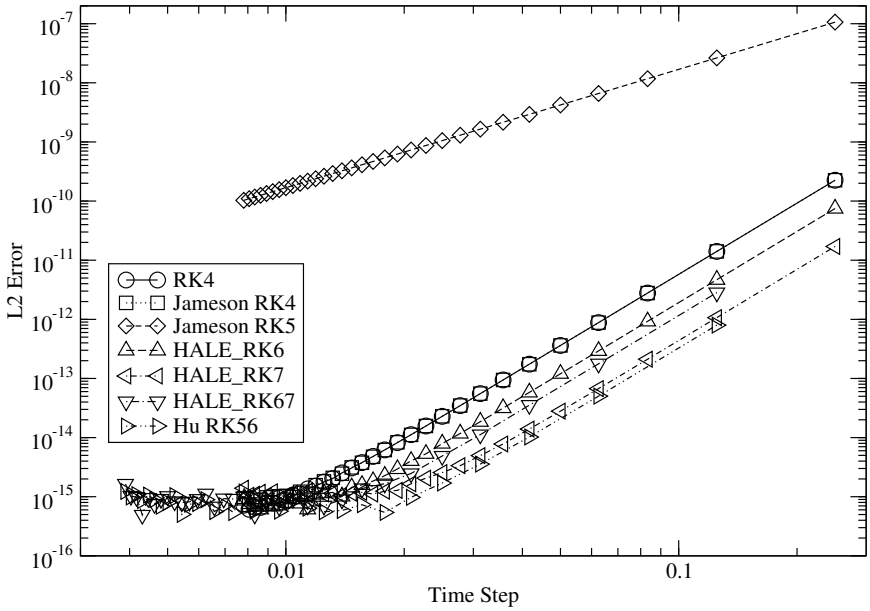


Figure 3-8: Linear inviscid Burger — L_2 error norm for the time-marching schemes

Figure 3-8 shows the L_2 error norms of the FD1D code when solving the linear form of the inviscid Burgers' equation. As expected, the error decreases with smaller time steps. For the smallest time steps the error "flattens out" for the more accurate schemes as the error reaches machine precision. The less accurate schemes would experience the same flattening of error if the time step used was sufficiently small.

Figure 3-9 shows the observed order-of-accuracy of the FD1D code for the FD1D code's time-marching schemes. The Jameson RK5 scheme is clearly a 2nd-order scheme, while all the other schemes return 4th-order, as they should. Again, the observed order-of-accuracy decreases for the more accurate schemes as the error ap-

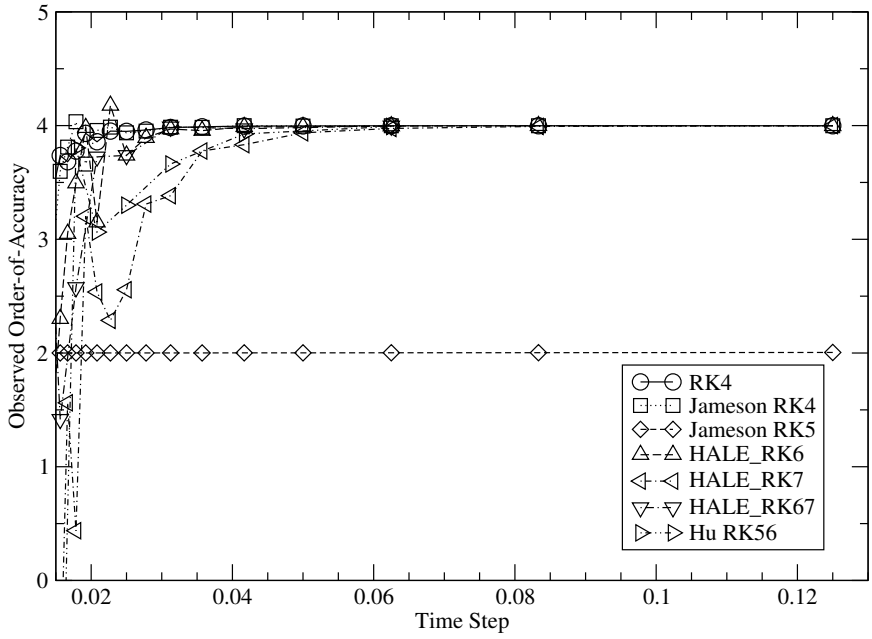


Figure 3-9: Linear inviscid Burger — observed order-of-accuracy for the time-marching schemes

proaches machine precision.

3.4.2 Nonlinear Inviscid Burger

Figure 3-10 shows the L_2 error norm of the FD1D code's time-marching scheme when solving the nonlinear form of the inviscid Burgers' equation. The results are similar to the corresponding linear results. Again, the L_2 error norm "flattens out" at the very small time steps for the more accurate schemes. Notice that the Jameson RK4 scheme is now parallel to the Jameson RK5 2nd-order scheme and not the 4th-order schemes. The Jameson RK4 is 4th-order accurate for the linear form of the inviscid Burgers' equation and 2nd-order for the nonlinear.

Figure 3-11 shows the observed order-of-accuracy of the FD1D code's time marching schemes when solving the nonlinear Burgers' equation. This figure confirms that the Jameson RK4L scheme is 2nd-order accurate when solving the nonlinear form of

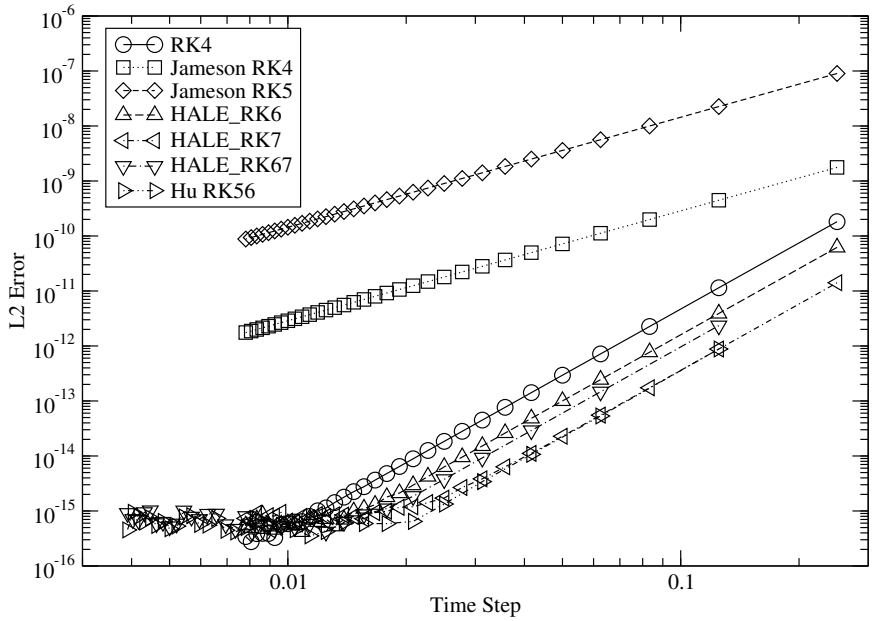


Figure 3-10: Nonlinear inviscid Burger — L_2 error norm for the time-marching schemes

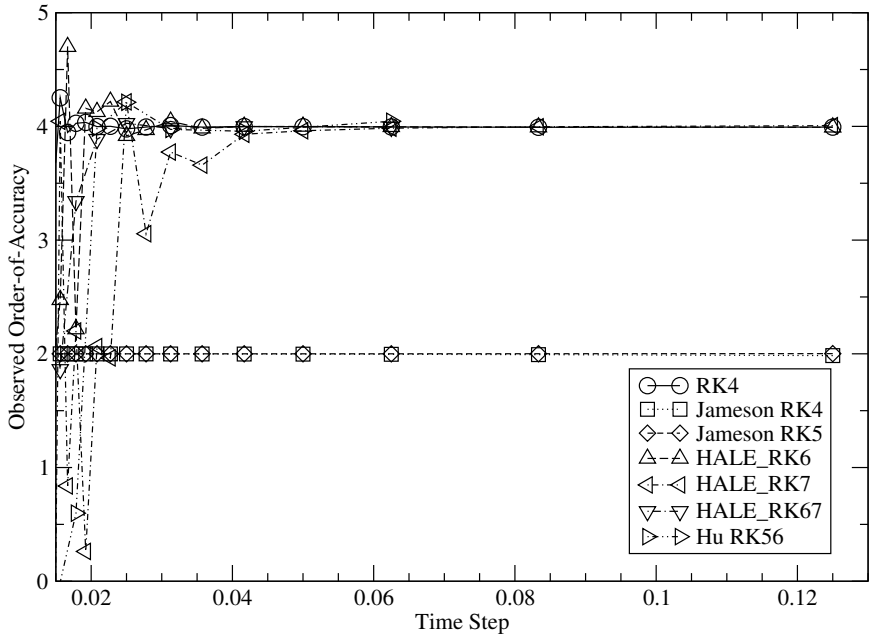


Figure 3-11: Nonlinear inviscid Burger — observed order-of-accuracy for the time-marching schemes

the equation. The observed order-of-accuracy for the remaining schemes is the same for the linear case: the Jameson RK5L is still 2nd-order, and all others (except the Jameson RK4L) are 4th-order. Again, the observed order-of-accuracy drops off for the more accurate schemes at the smallest time steps as the error approaches machine precision. These results, in conjunction with the results for the linear Burgers' equation in Section 3.4.1, strongly Verify that the FD1D code's time-marching schemes are properly implemented and solving the linear and nonlinear Burgers' equations correctly.

3.5 Temporal Order-of-Accuracy Results — Multiple EVA Solution

Another approach to Verifying the order-of-accuracy of a time-integration scheme in a CFD/CAA code using the EVA method (as discussed in Section 2.2.5) consists of using the EVA3 tool to calculate a solution at multiple time levels using a Taylor series of at least one order higher than the time-integration scheme to be Verified. The CFD code is then used to find the solution at the same time levels, and the error can be found using the same formulae that are used in the single EVA solution approach (i.e. Equation (3.10)). As discussed in Section 2.2.5, a slightly modified equation is used to find the observed order-of-accuracy. While this approach does require more EVA runs than the single EVA solution approach, the accuracy requirements for the EVA3 runs are much less restrictive, resulting in much faster EVA3 runs and less time spent using the EVA3 tool overall.

The same initial condition used in the previous sections (Figure 3-1) was used to Verify the FD1D code's time-marching schemes using the multiple EVA Solution approach. The computational domain again extended from 0.0 to 200.0 and included 3201 grid points. Thirty-two time levels were used, consisting of integer multiples of

0.0078125 (i.e., $0.0078125, 0.015625, 0.0234375 \ldots 0.25$). Again, the larger time levels and dense grid are used to insure that the error from the time-marching outweighs the error from the spatial discretization. The EVA3 code was used to solve the governing equation of interest at 21 points from 90.0 to 110.0.

3.5.1 Linear Inviscid Burger

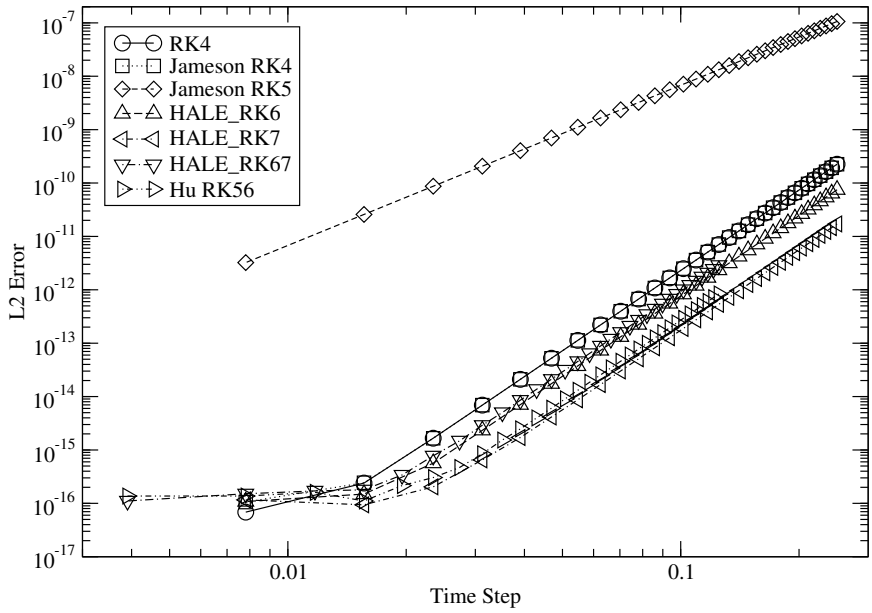


Figure 3-12: Linear inviscid Burger — L_2 error for the time-marching schemes using the multiple EVA solution approach

Figure 3-12 shows the L_2 error calculated for the FD1D code's time-marching schemes when solving the linear inviscid Burgers' equation. The results are very similar to those found using the single EVA solution method (Figure 3-8). The HALE_RK7 scheme appears to be the most accurate, followed closely by the Hu RK56. The Jameson RK5, being the only linear 2nd-order scheme is clearly much less accurate than the 4th-order schemes, while all the 4th-order schemes have comparable accuracy. As before, the L_2 error for the most accurate schemes levels off as the error

approaches machine-zero.

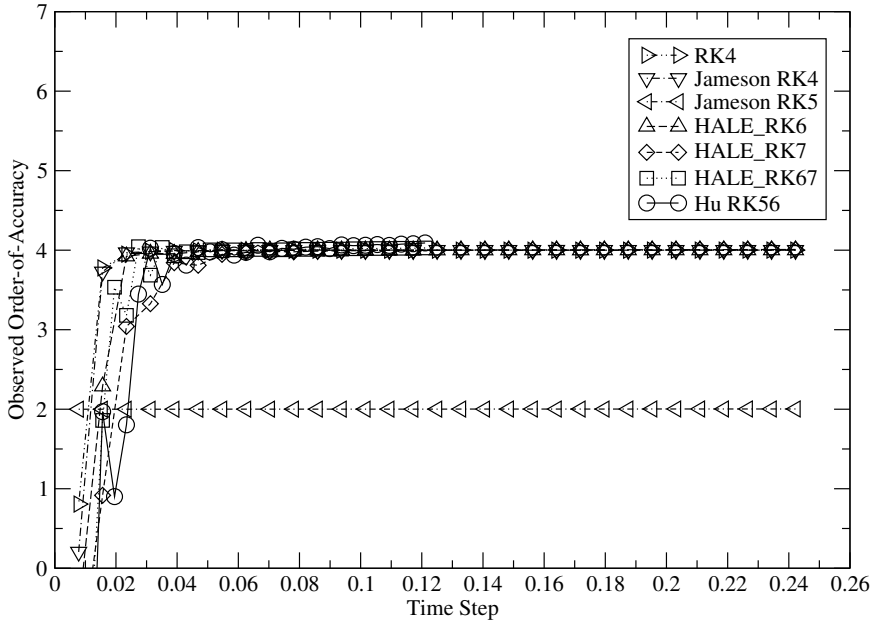


Figure 3-13: Linear inviscid Burger — observed order-of-accuracy for the time-marching schemes using the multiple EVA solution method

Figure 3-13 shows the observed order-of-accuracy based on L_2 error for the FD1D code's time-marching schemes using the multiple EVA solution method. Each scheme achieves its formal order-of-accuracy for the majority of time step sizes, and the results shown are very similar to those in Figure 3-9 — the fourth-order schemes all return the same order, and the Jameson RK5 exhibits an observed 2nd-order-accuracy. As before, the order-of-accuracy sharply decreases for the higher-order schemes for the smallest time levels as the error approaches machine precision.

3.5.2 Nonlinear Inviscid Burger

Figure 3-14 shows the L_2 error calculated for the FD1D code's time-marching schemes when solving the nonlinear form of the inviscid Burgers' equation. Again, the HALE_RK7 is the most accurate scheme with the Hu RK56 being only slightly

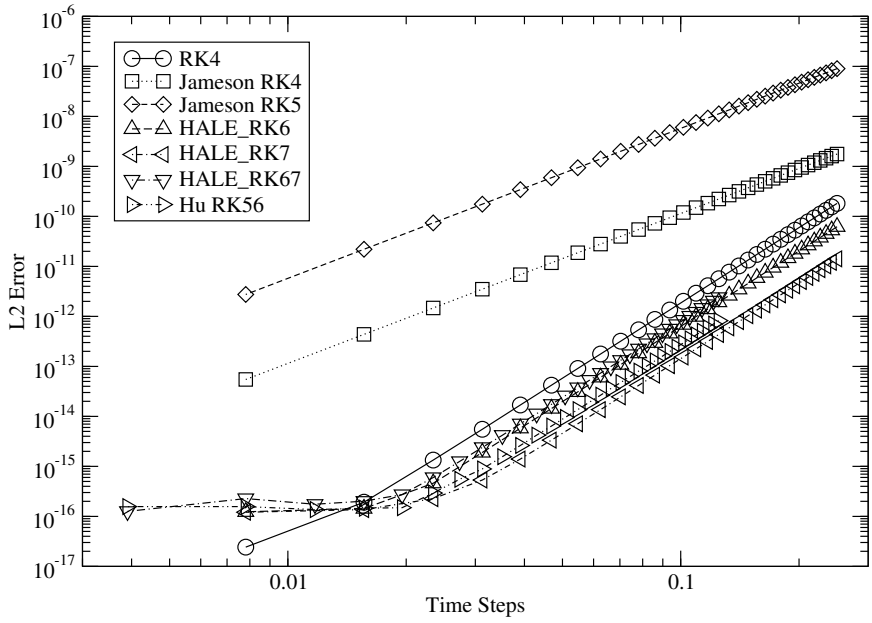


Figure 3-14: Nonlinear inviscid Burger — L_2 error for the time-marching schemes using the multiple EVA solution approach

less accurate. The Jameson RK4 can be seen to be a 2nd-order nonlinear scheme by comparing its slope with the Jameson RK5 — they are nearly identical. And yet again, the error flattens out for the more accurate schemes as the error approaches machine precision.

Figure 3-15 shows the observed order-of-accuracy for the FD1D code's time-marching schemes for the nonlinear inviscid Burger equation using the multiple EVA solution method. Again, the results are very similar to the previous linear case and to the nonlinear case using the single EVA solution. As in previous results, the Jameson RK4 is shown to be a 2nd-order nonlinear time-marching scheme. All schemes achieve their formal order-of-accuracy, indicating they are behaving as expected and implemented correctly.

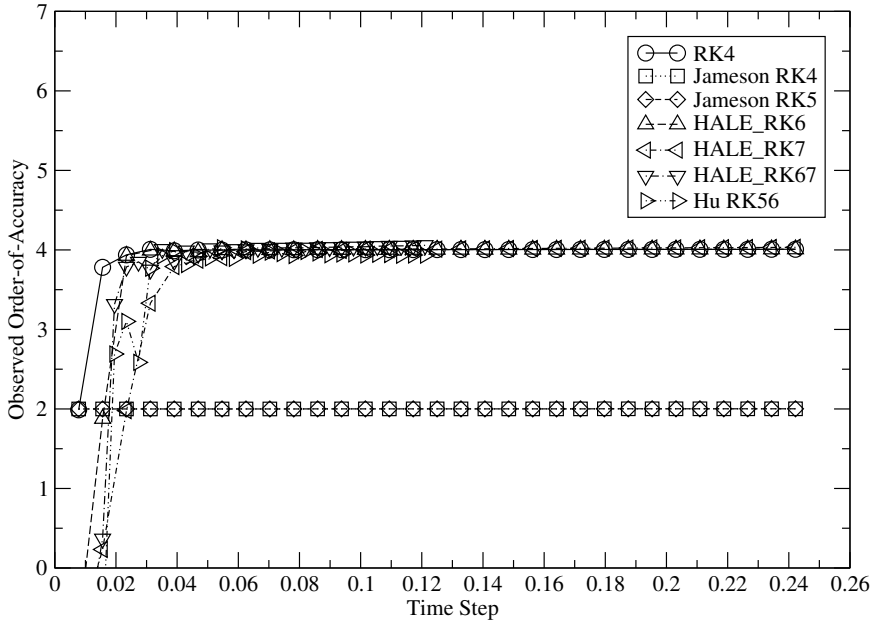


Figure 3-15: Nonlinear inviscid Burger — observed order-of-accuracy for the time-marching schemes using the multiple EVA solution method

Chapter 4

Application of the EVA Method to the Euler Equations

4.1 Equations Considered

To demonstrate the effectiveness of the EVA method in Verifying computer codes solving realistic fluid flow problems, the EVA3 code is also capable of solving the one-, two- and three-dimensional forms of the Euler equations. The "Euler equations" meant in this context are the governing equations of inviscid, compressible fluid flow, and the three-dimensional form of these equations solved by the EVA3 code were given in Equation (2.1) and are reprinted here in more detail:

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} + \frac{\partial \vec{G}}{\partial z} = 0 \tag{4.1}$$

where $\vec{Q}$ is the solution vector

$$\vec{Q} = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ \rho w \\ E_{total} \end{array} \right\} \tag{4.2}$$

which contains the density, x-, y- and z-momentum and total energy, and $\vec{E}$, $\vec{F}$ and $\vec{G}$ are the fluxes in the x, y and z directions, respectively

$$\vec{E} = \left\{ \begin{array}{c} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ u\left(E_{total} + p\right) \end{array} \right\} \tag{4.3}$$

and

$$\vec{F} = \left\{ \begin{array}{c} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ v\left(E_{total} + p\right) \end{array} \right\} \tag{4.4}$$

and

$$\vec{G} = \left\{ \begin{array}{c} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + p \\ w\left(E_{total} + p\right) \end{array} \right\} \tag{4.5}$$

The pressure (p) is related to the other solution variables in an ideal gas by the equation

$$p = (\gamma - 1)\left(E_{total} - \rho\left(\frac{u^2 + v^2 + w^2}{2}\right)\right) \tag{4.6}$$

where γ is the ratio of specific heats, commonly taken to be 1.4 for air (which is the same value used for all results throughout this work).

The two-dimensional form of the Euler Equations is very similar to the three-dimensional:

$$\frac{\partial \vec{Q}}{\partial t} + \frac{\partial \vec{E}}{\partial x} + \frac{\partial \vec{F}}{\partial y} = 0 \tag{4.7}$$

where $\vec{Q}$ again is the solution vector

$$\vec{Q} = \left\{ \begin{array}{c} \rho \\ \rho u \\ \rho v \\ E_{total} \end{array} \right\} \tag{4.8}$$

whose entries are defined like those of Equation (4.2), and $\vec{E}$ and $\vec{F}$ are the fluxes in

the x and y directions, respectively:

$$\vec{E} = \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u\left(E_{total} + p\right) \end{Bmatrix} \tag{4.9}$$

and

$$\vec{F} = \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v\left(E_{total} + p\right) \end{Bmatrix} \tag{4.10}$$

where the pressure p is again defined similarly:

$$p = (\gamma - 1)\left(E_{total} - \rho\left(\frac{u^2 + v^2}{2}\right)\right) \tag{4.11}$$

The EVA method can be used to generate a highly-accurate solution to the Euler equations for a flexible set of initial conditions that can be used to fully exercise the interior portion of the numerical scheme used by a CFD/CAA code to solve the Euler equations. In this chapter results from Verifying a high-order Computational Aeroacoustics code solving the two-dimensional and three-dimensional Euler equations will be presented.

4.2 Description of the BASS Code

To test the ability of the EVA technique to Verify high-accuracy CAA codes, the EVA method was applied to the NASA Glenn Research Center Broadband Aeroa-

coustic Stator Simulation (BASS) code [26]. The BASS code is designed to solve the full 3D Navier-Stokes equations but can also solve the 2D and 3D Euler equations with the appropriate options set. Written in FORTRAN 2003, the BASS code can be compiled with various implementations of the Message Passing Interface (MPI) and executed in parallel on a cluster of machines. Numerous spatial stencils and time-marching schemes are implemented in BASS, many of which have been discussed in Chapter 3. The spatial stencils available include the Dispersion Relation Preserving scheme of Tam and Webb and the Compact 6th-order scheme of Lele [18], as well as standard 2nd- and 6th-order explicit central differencing. The time-marching schemes include the four- and five-stage Runge-Kutta schemes of Jameson, the 5/6 stage two-step Runge-Kutta scheme of Hu and the High-Accuracy Large-step Explicit Runge-Kutta (HALE-RK) 6 and 6/7 stage two-step scheme of Allampali et al.. Various boundary conditions are implemented in the BASS code; however, periodic boundary conditions were applied for all runs in this work as the EVA3 code Verifies only the interior solver.

4.3 Spatial Order-of-Accuracy Results

The procedure for Verifying the spatial order-of-accuracy of a code described in Section 2.2 was first performed using the BASS code while solving the two-dimensional Euler equations, and then the three-dimensional equations for various grids, initial conditions and final time levels. In general, Verifying the spatial order-of-accuracy of the BASS code was found to be less challenging than Verifying the temporal order-of-accuracy, which can be explained by comparing the sources of truncation error of an EVA solution and a CFD/CAA code solution. For a CFD/CAA code the truncation error comes from both the spatial stencil used to approximate the spatial derivatives — with error on the order of some power of the grid spacing — and the

time-marching scheme used to integrate the solution in time — with error on the order of some power of the time step. For the EVA solution, the truncation error is due entirely to the method used to integrate the solution in time — namely, the Taylor series of Equation (2.7) — which will again be on the order of some power of the time step. As a result, one will use a very small time step and large range of grid spacings with the CFD/CAA code when Verifying the spatial differencing scheme. The very small time step will also drive the truncation error of the EVA solution down, requiring less terms in the Taylor series in of Equation (2.7) and thus making Verification easier. On the other hand, when Verifying the temporal order-of-accuracy of a code one will use a very fine grid with a final time level comparatively "far away" from the initial condition — this arrangement will keep error from the spatial discretization low compared to the temporal scheme's error. Unfortunately, the "far away" final time level will also tend to degrade the accuracy of the EVA solution. A solution of acceptable accuracy can always be found by increasing the number of terms in the EVA Taylor series, but by adding terms the computation effort is increased and Verification is more challenging.

4.3.1 Two-Dimensional

The initial condition used to Verify the two-dimensional Euler solver of the BASS code using a uniform Cartesian grid was

$$\vec{Q}(x, y, t = 0) = \bar{Q} + \tilde{Q} \exp\left[-(x^2 + y^2)\right] \tag{4.12}$$

where

$$\bar{Q} = \begin{Bmatrix} \bar{\rho} \\ \overline{\rho u} \\ \overline{\rho v} \\ \bar{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 1.0 \\ 0.1 \\ 0.1 \\ 1.78571425 \end{Bmatrix} \tag{4.13}$$

and

$$\tilde{Q} = \begin{Bmatrix} \tilde{\rho} \\ \widetilde{\rho u} \\ \widetilde{\rho v} \\ \tilde{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 0.1 \\ 0.01 \\ 0.01 \\ 0.1 \end{Bmatrix} \tag{4.14}$$

The final time level chosen was $t = 0.015625$. For the EVA solution a grid with 3^2 grid points was used that extended from -2.0 to 2.0 in both the x- and y-directions. Ten BASS runs were performed on grids with $25^2, 49^2, 73^2, \ldots, 241^2$ grid points. Each of the BASS grids extended from -24.0 to 24.0 in both coordinate directions. Figure 4-1 shows the $L_{\max}$ error norm of the ten BASS runs calculated using the EVA solution for each of the four spatial differencing schemes (explicit 2nd- and 6th-order central, compact 6th-order and DRP) and for each of the four solution variables. (Note that the two momentum variables are nearly on top of each other.) One can see how the DRP scheme outperforms the explicit 6th-order scheme and nearly equals the resolution of the compact 6th-order scheme for the coarser grids, but falls off for the finer grid spacings due to its "optimized" development. Figure 4-2 shows the observed order-of-accuracy for the ten BASS runs based on the $L_{\max}$ error. All four schemes reach their theoretical order-of-accuracy for the finer grid spacings, strongly indicating that each of the four schemes are implemented correctly in the BASS code for this situation (i.e., 2D Euler with uniform Cartesian grids). The DRP scheme's

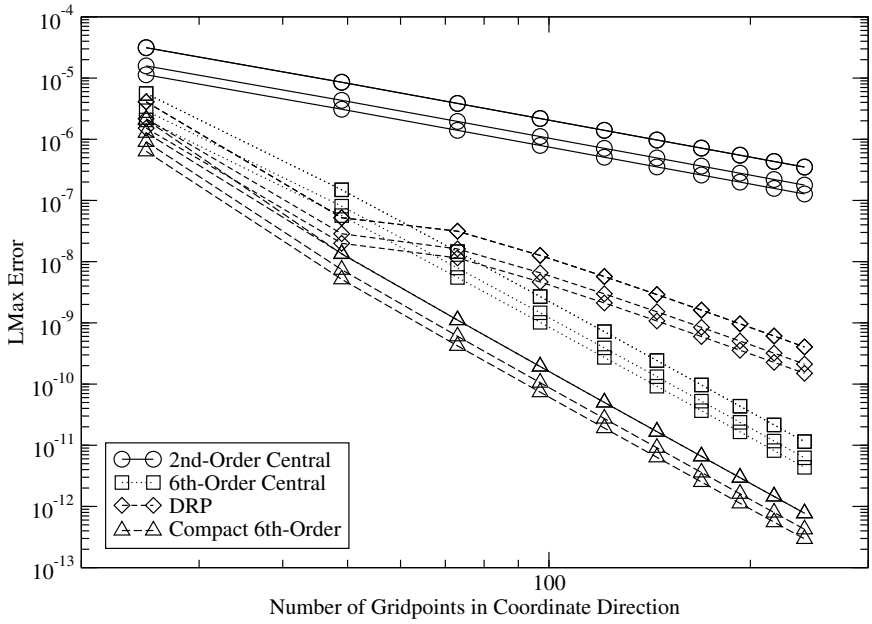


Figure 4-1: Euler 2D Spatial $L_{\max}$ Error — Uniform Grid

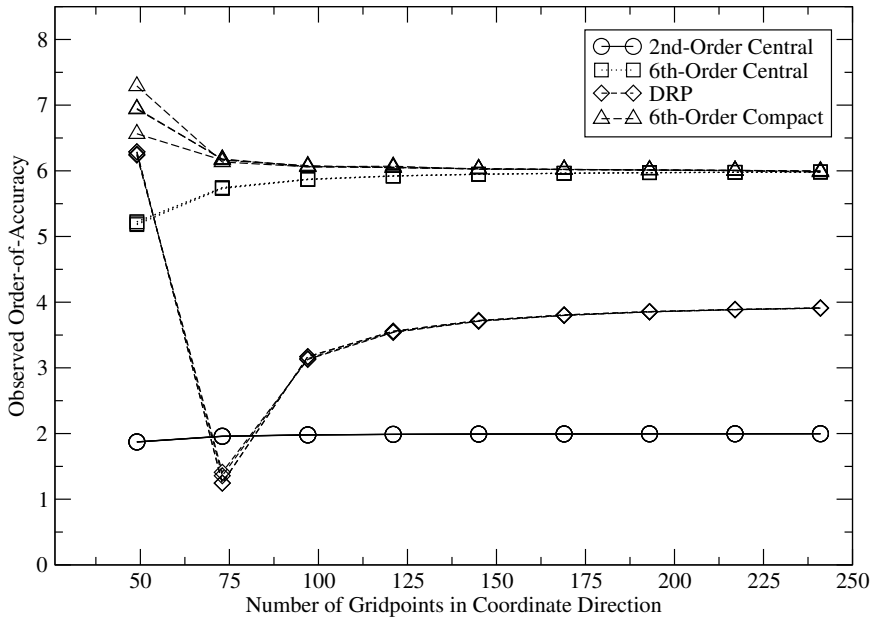


Figure 4-2: Euler 2D Spatial Observed Order-of-Accuracy — Uniform Grid

observed order-of-accuracy is very high at the coarser grid spacings, then drops off sharply and approaches its formal order-of-accuracy (4th) for the finer grid spacings. Again, this behavior is explained by the "optimized" nature of the DRP scheme.

To more rigorously test the spatial discretization schemes of the BASS 2D Euler solver the code was run on a curvilinear grid. Figure 4-3 shows the third-coarsest

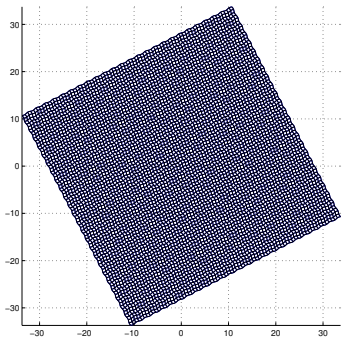


Figure 4-3: 2D Curvilinear Grid — Rotated 27.4°

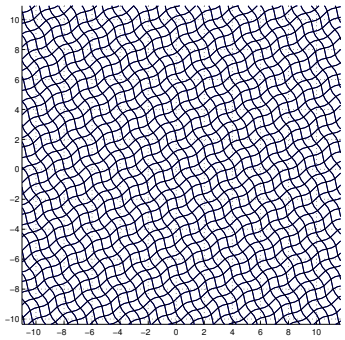


Figure 4-4: 2D Curvilinear Grid, Enlarged — Rotated 27.4°

curvilinear grid used for the BASS 2D Euler grid runs, and Figure 4-4 shows an

enlargement of the same grid. The equations used to generate the base grid were

$$x(i,j) = x_{\min} + (i-1)\Delta x + 0.1\sin\left[\pi(y_{\min} + (j-1)\Delta y)\right] \tag{4.15}$$

$$y(i,j) = y_{\min} + (j-1)\Delta y + 0.1\sin\left[\pi(x_{\min} + (i-1)\Delta x)\right] \tag{4.16}$$

The grid was then rotated counter-clockwise using elementary matrix operations. Figures 4-3 and 4-4 show the 27.4° rotation; 45.0° was also used. Each curvilinear grid used for the BASS runs extended from -25.0 to 25.0 in each coordinate direction, while the EVA grid extended from -8.0 to 8.0 with 9 grid points in each coordinate direction. Sixteen grids with $51^2, 101^2, 151^2, \ldots, 801^2$ grid points were used for the BASS runs. The final time level was set to $t = 0.03125$, and the initial condition chosen was similar to the uniform grid case:

$$\vec{Q}(x,y,t=0) = \bar{Q} + \tilde{Q}\exp\left[-(x^2+y^2)\right] \tag{4.17}$$

where

$$\bar{Q} = \begin{Bmatrix} \bar{\rho} \\ \overline{\rho u} \\ \overline{\rho v} \\ \bar{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 1.78571425 \end{Bmatrix} \tag{4.18}$$

and

$$\tilde{Q} = \begin{Bmatrix} \tilde{\rho} \\ \widetilde{\rho u} \\ \widetilde{\rho v} \\ \tilde{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.05 \end{Bmatrix} \tag{4.19}$$

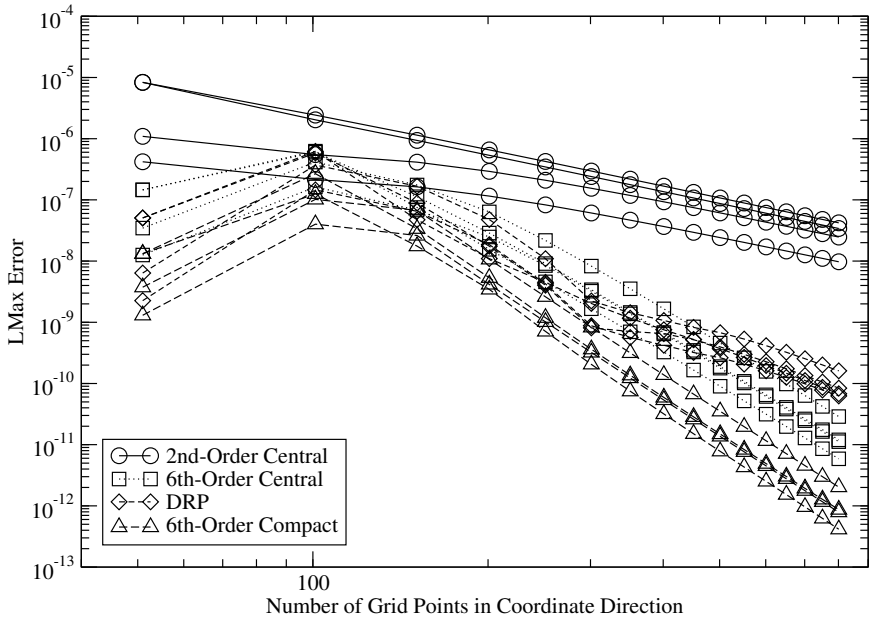


Figure 4-5: Euler 2D Spatial $L_{\max}$ Error — Curvilinear Grid Rotated 27.4°

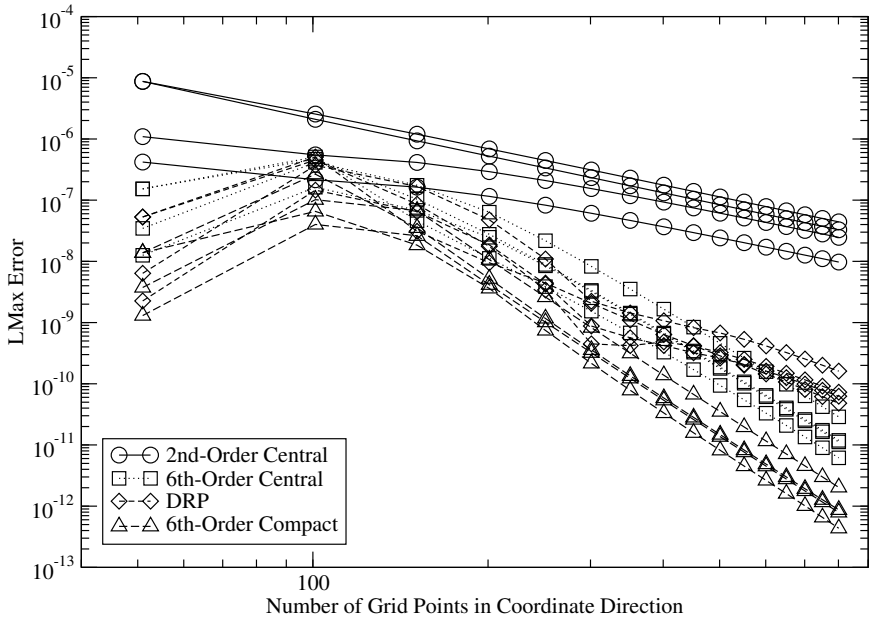


Figure 4-6: Euler 2D Spatial $L_{\max}$ Error — Curvilinear Grid Rotated 45.0°

Figures 4-5 and 4-6 show the $L_{\max}$ error norms for the curvilinear cases rotated 27.4° and 45.0°, respectively. Very little difference is seen between the two rotations, and the results overall are similar to the uniform Cartesian grid case. Figures 4-7

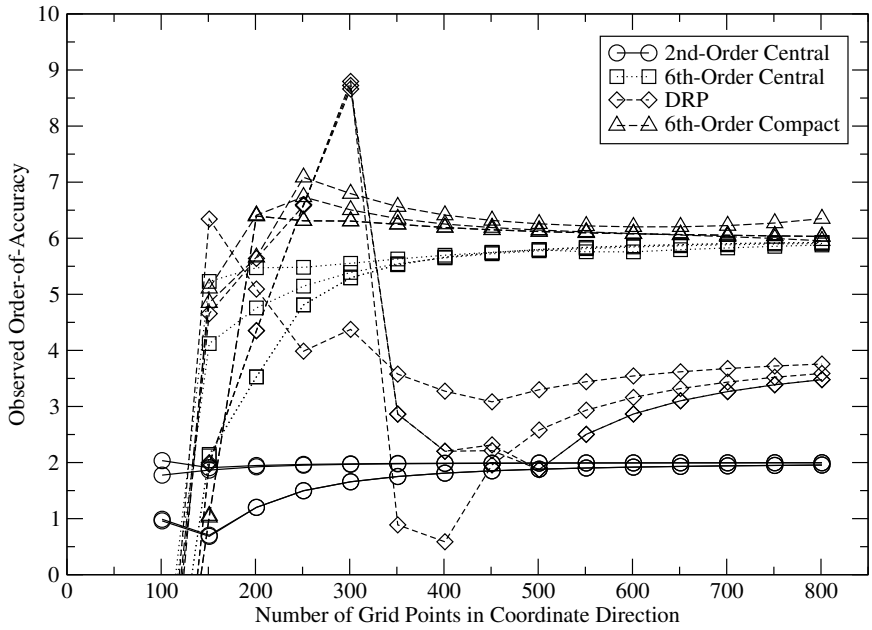


Figure 4-7: Euler 2D Spatial Observed Order-of-Accuracy — Curvilinear Grid Rotated 27.4°

and 4-8 show the observed order-of-accuracy for the two rotated curvilinear grids. For both of the rotations the four schemes eventually achieve their respective formal orders-of-accuracy with enough grid, again strongly indicating that the schemes are correctly implemented in BASS.

4.3.2 Three-Dimensional

The order-of-accuracy of the spatial schemes implemented in BASS were also Verified for the three-dimensional Euler case. In general, it is more challenging to Verify the three-dimensional Euler schemes than the two-dimensional as achieving the required grid spacing requires much more grid for the three-dimensional case.

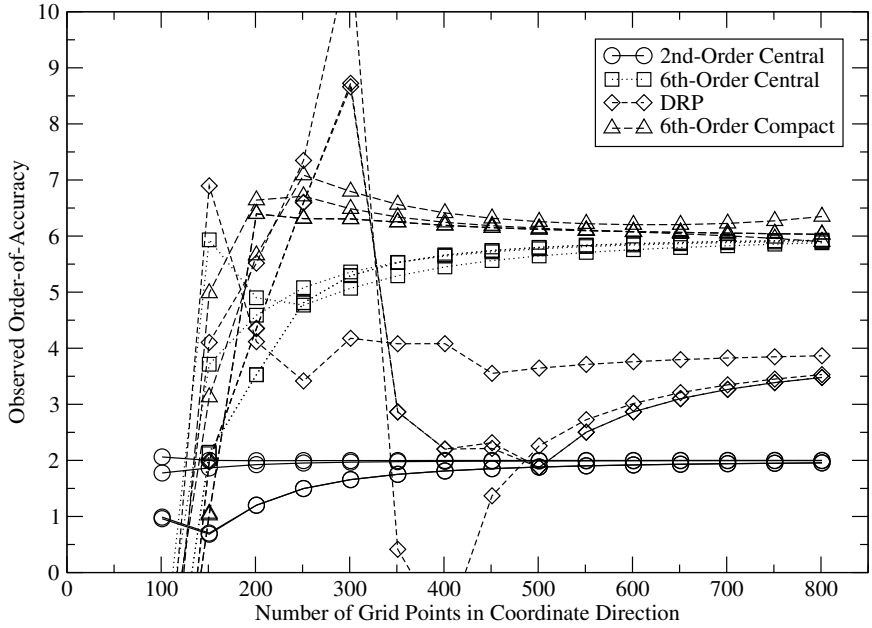


Figure 4-8: Euler 2D Spatial Observed Order-of-Accuracy — Curvilinear Grid Rotated 45.0°

For instance, if one desires an average grid spacing of $\Delta = 0.0625$ on a logically square/cubic grid that extends from -4.0 to 4.0, $129^3 = 2,146,689$ grid points are required for the three-dimensional case, while only $129^2 = 16641$ are required for the two-dimensional case.

The spatial differencing schemes implemented in the three-dimensional Euler solver of BASS are the same for the two-dimensional solver: explicit 2nd and 6th-order central differencing, the optimized 4th-order DRP scheme of Tam and Webb, and the compact 6th-order scheme of Lele. The schemes were first tested on uniform Cartesian

grids. The initial condition chosen was

$$\begin{aligned}
\rho(x,y,z) &= 1.0 + 0.1\exp\left[((x-0.5)^2 + (y-0.5)^2(x-0.5)^2)\right] \\
\rho u(x,y,z) &= 0.1 + 0.01\exp\left[((x+0.5)^2 + (y-0.5)^2 + z^2)\right] \\
\rho v(x,y,z) &= 0.1 + 0.01\exp\left[((x-0.5)^2 + (y+0.5)^2 + z^2)\right] \\
\rho w(x,y,z) &= 0.1 + 0.01\exp\left[((x+0.5)^2 + (y+0.5)^2 + z^2)\right] \\
E_{total}(x,y,z) &= 1.78571425 + 0.1\exp\left[(x^2 + y^2 + (z-0.5)^2)\right]
\end{aligned} \tag{4.20}$$

with final time level of $t = 0.001953125$. The EVA solution was found for the initial condition at the final time level on a uniform three-dimensional grid extending from -4.0 to 4.0 in all three directions with 5^3 grid points. Fourteen grids with $9^3, 17^3, 25^3, 33^3, \ldots, 113^3$ and extending from -8.0 to 8.0 in each coordinate direction were used by the BASS code. Figure 4-9 shows the L_2 error norm of the fourteen

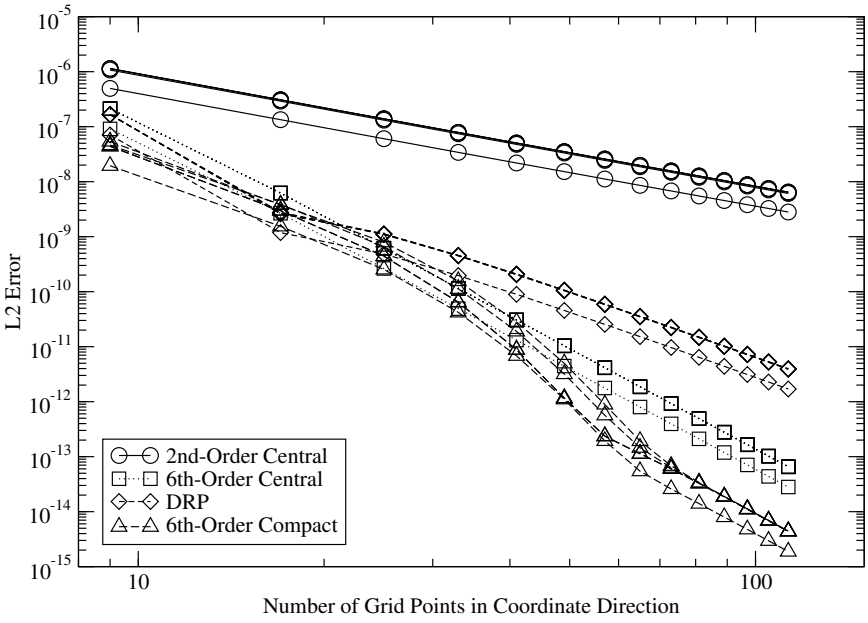


Figure 4-9: Euler 3D Spatial L_2 Error — Uniform Grid

BASS runs for each of the solution variables and spatial differencing schemes. For the 2nd- and 6th-order central and DRP schemes the error curve appears as a straight

line on the log-log plot for the denser grids, indicating that the schemes are converging at a constant order-of-accuracy in this range of grid spacings. The DRP and especially the compact 6th-order schemes do not show this same uniform convergence, as the "bends" in the error curves show. Each of these schemes do appear to "straighten out" as the grid density increases, however. Figure 4-10 shows the

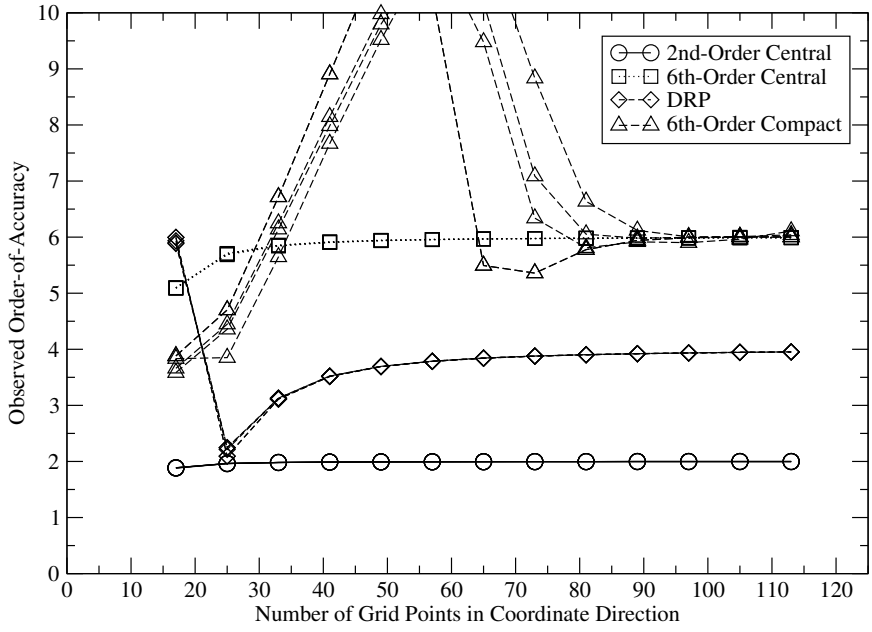


Figure 4-10: Euler 3D Spatial Observed Order-of-Accuracy — Uniform Grid

observed order-of-accuracy based on the L_2 error norm for each of the four spatial differencing schemes. The 2nd- and 6th-order central schemes attain their formal order-of-accuracy for practically the entire range of grid spacings. The DRP and compact 6th-order schemes also eventually return their formal order-of-accuracy, but the "bends" shown in the L_2 error in Figure 4-9 manifest themselves as variations in the observed order-of-accuracy. As discussed before, this behavior is a result of the optimization procedure that was used in these schemes' development and is not evidence of incorrect implementation.

Again, to more rigorously Verify the spatial discretization schemes implemented

in BASS, the EVA method was applied to BASS code on rotated curvilinear grids. A grid was constructed using the following equations

$$\begin{aligned}
x(i,j,k) &= x_{\min} + (i-1)\Delta x + 0.1250 \sin\left[\tfrac{\pi}{4}(y_{\min} + (j-1)\Delta y)\right] \\
&+ \qquad\qquad 0.0625 \sin\left[\tfrac{\pi}{4}(z_{\min} + (k-1)\Delta z)\right] \\
y(i,j,k) &= y_{\min} + (j-1)\Delta y + 0.2500 \sin\left[\tfrac{\pi}{4}(x_{\min} + (i-1)\Delta x)\right] \\
&+ \qquad\qquad 0.0625 \sin\left[\tfrac{\pi}{4}(z_{\min} + (k-1)\Delta z)\right] \\
z(i,j,k) &= z_{\min} + (k-1)\Delta z + 0.2500 \sin\left[\tfrac{\pi}{4}(x_{\min} + (i-1)\Delta x)\right] \\
&+ \qquad\qquad 0.1250 \sin\left[\tfrac{\pi}{4}(y_{\min} + (j-1)\Delta y)\right]
\end{aligned} \tag{4.21}$$

The grid was then rotated 45.0° about the x-axis, 30.0° about the y-axis and 27.4° about the z-axis using elementary matrix transformations. A "corner" of the grid is shown in Figure 4-11. The initial condition used was

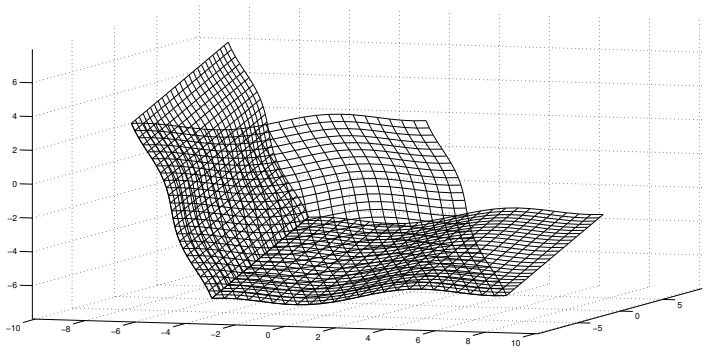


Figure 4-11: 3D Curvilinear Grid Corner with 25^3 Grid Points

$$
\begin{aligned}
\rho(x,y,z) &= 1.0 + 0.1\exp\left[((x-0.5)^2+(y-0.5)^2+(x-0.5)^2)\right]\sin(2.0x+2.0y+2.0z) \\
\rho u(x,y,z) &= 0.1 + 0.01\exp\left[((x+0.5)^2+(y-0.5)^2+z^2)\right]\sin(2.0x+2.0y+2.0z) \\
\rho v(x,y,z) &= 0.1 + 0.01\exp\left[((x-0.5)^2+(y+0.5)^2+z^2)\right]\sin(2.0x+2.0y+2.0z) \\
\rho w(x,y,z) &= 0.1 + 0.01\exp\left[((x+0.5)^2+(y+0.5)^2+z^2)\right]\sin(2.0x+2.0y+2.0z) \\
E_{total}(x,y,z) &= 1.78571425 + 0.1\exp\left[(x^2+y^2+(z-0.5)^2)\right]\sin(2.0x+2.0y+2.0z)
\end{aligned}
\tag{4.22}
$$

The HALE_RK7 scheme was used for each BASS run with a final time level of $t = 0.001953125$. For the BASS code runs, twenty-four logically-cubic grids were used with $7^3, 13^3, 19^3, 25^3, 31^3, \ldots, 145^3$ grid points extending from -6.0 to 6.0. The EVA solution was found on a uniform Cartesian grid extending from -4.0 to 4.0 with 5^3 grid points oriented identically to the grid used for the BASS code runs. Figure 4-12

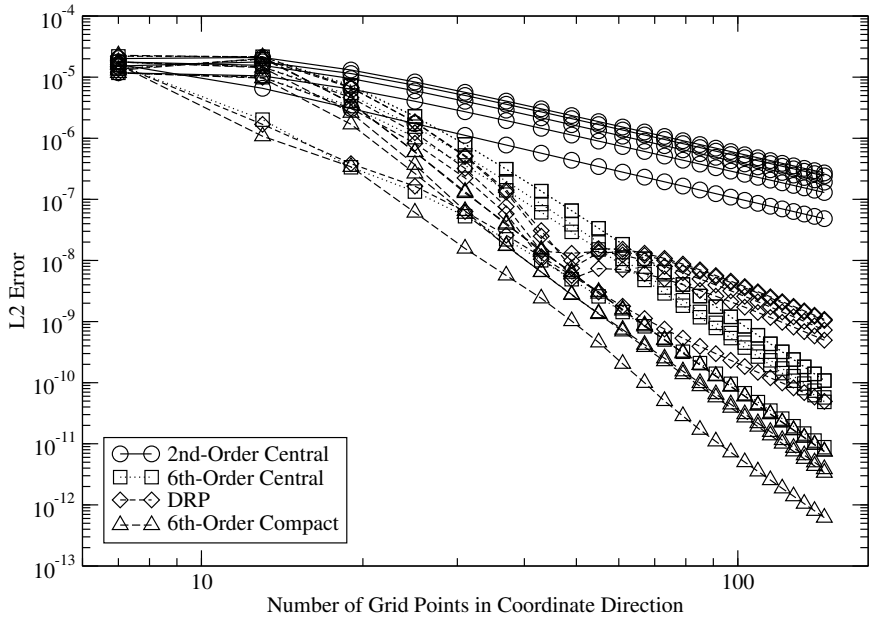


Figure 4-12: Euler 3D Spatial L_2 Error — Curvilinear Grid

shows the L_2 error for each of the spatial schemes in BASS. The results are in line with previous spatial results. The DRP scheme again performs better than the explicit 6th-order central scheme for the coarser grids and worse for the finer grids, and the

compact 6th-order scheme is again the most accurate scheme. Figure 4-13 shows

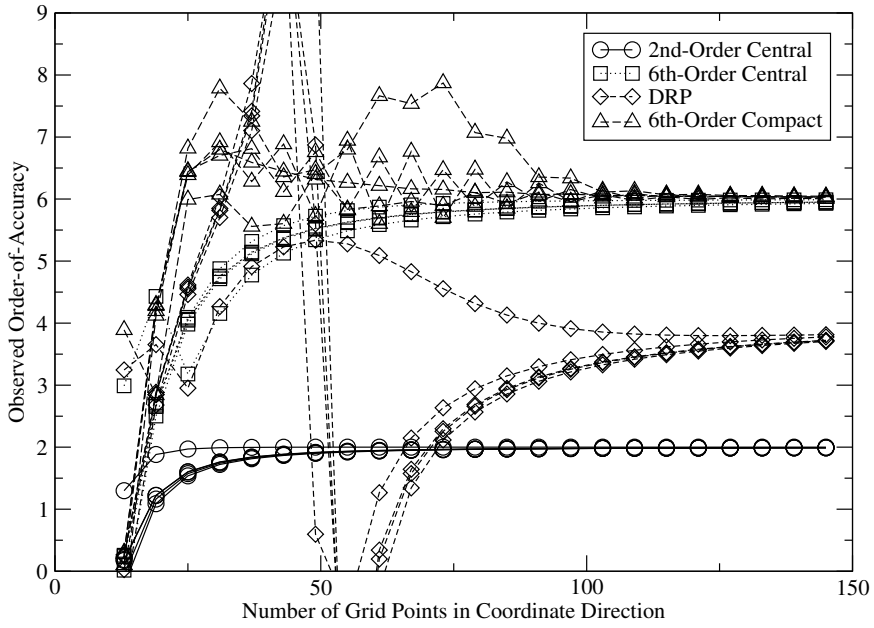


Figure 4-13: Euler 3D Spatial Observed Order-of-Accuracy — Curvilinear Grid

the observed order-of-accuracy for the BASS runs. Again, the results are similar to previous spatial results. The explicit 2nd- and 6th-order central schemes attain their formal order-of-accuracy for the majority of the grid spacings. The compact 6th-order scheme approaches 6th-order "from above," as in previous test cases. The DRP returns at first a higher and then lower order-of-accuracy before approaching its formal order-of-accuracy (4th). The DRP scheme required the most grid to observe its formal order-of-accuracy, making it the most difficult to Verify.

4.4 Temporal Order-of-Accuracy Results — Single EVA Solution

To Verify the BASS code's temporal order of accuracy, the procedure described in Section 2.2 was performed first on the two-dimensional and then three-dimensional forms of the Euler equations. As discussed in the beginning of Section 4.3, Verification of the temporal schemes of a CFD/CAA code is generally more challenging and computationally intensive than Verification of the spatial schemes. Results for specific cases are presented in the proceeding sections.

4.4.1 Two-Dimensional

The initial condition used to Verify the time-marching schemes for the two-dimensional Euler equations was very similar that used in Section 4.3.1

$$\vec{Q}(x, y, t = 0) = \bar{Q} + \tilde{Q} \exp\left[-\frac{\ln(2)}{9.0}(x^2 + y^2)\right] \tag{4.23}$$

where

$$\bar{Q} = \left\{ \begin{array}{c} \bar{\rho} \\ \overline{\rho u} \\ \overline{\rho v} \\ \bar{E}_{total} \end{array} \right\} = \left\{ \begin{array}{c} 1.0 \\ 0.1 \\ 0.1 \\ 1.78571425 \end{array} \right\} \tag{4.24}$$

and

$$\tilde{Q} = \left\{ \begin{array}{c} \tilde{\rho} \\ \widetilde{\rho u} \\ \widetilde{\rho v} \\ \tilde{E}_{total} \end{array} \right\} = \left\{ \begin{array}{c} 0.1 \\ 0.01 \\ 0.01 \\ 0.1 \end{array} \right\} \tag{4.25}$$

The final time level used was $t = 0.25$. The EVA solution was found on a uniform grid using 9^2 grid points extending from -8.0 to 8.0 in both coordinate directions. For this first Verification exercise a uniform grid with 801^3 grid points was used with each of the time-marching schemes in the BASS code — however, different grid spacings were found to be necessary for the different schemes. The two Jameson schemes (the Jameson RK4 and RK5) were able to be Verified using a grid spacing of 0.125, while the HALE_RK7 required 0.0625 and the HALE_RK67 and Hu RK56 needed 0.03125. The 6th-order compact spatial differencing scheme of Lele was used for each BASS run. Figure 4-14 shows the L_2 error norm for each of the time-marching schemes in BASS plotted against the time step size. Four curves are shown for each of the schemes as the two-dimensional Euler equations are a system of four equations. The curves for each of the four solution variables are nearly straight for the Jameson RK5 and HALE_RK7, indicating that these schemes are converging at an approximately constant order-of-accuracy for the entire range of time steps used. The Jameson RK4 and HALE_RK67 schemes also appear to converge at a nearly uniform rate, at least after the largest two or three time steps. The Hu RK56 scheme shows a noticeable "bump" in error for all four solution variables in the larger-time-step range, likely due to the "optimized" nature of the Hu RK56. Figure 4-15 shows the observed order-of-accuracy for the BASS runs. As indicated by the L_2 error plot, the Jameson RK5 and HALE_RK7 maintain an approximately constant order-of-accuracy for the entire range of time steps, and both achieve their formal orders-of-accuracy (2nd for

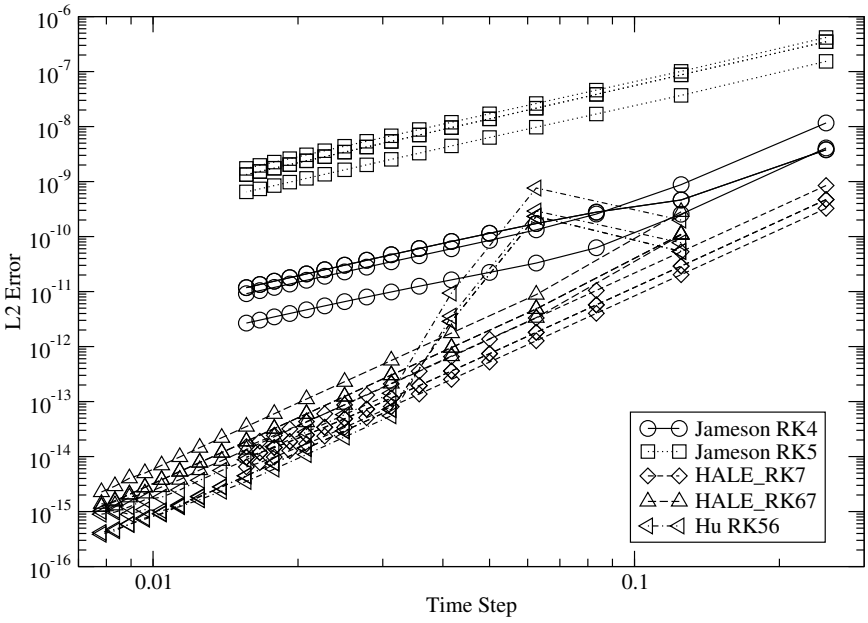


Figure 4-14: Euler 2D Temporal L_2 Error — Uniform Grid

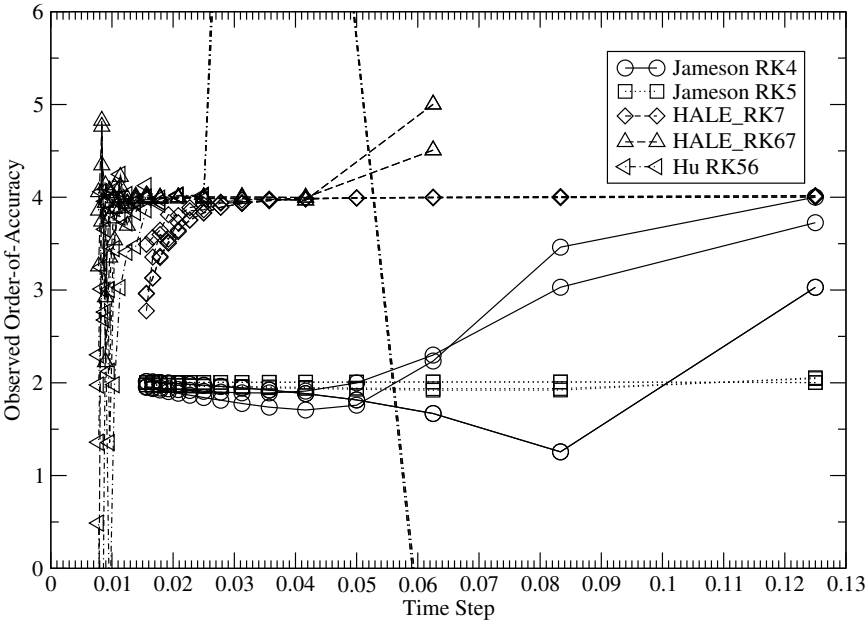


Figure 4-15: Euler 2D Temporal Order-of-Accuracy — Uniform Grid

the Jameson RK5 and 4th for the HALE_RK7). The Jameson RK4 and HALE_RK56 also return their formal order-of-accuracy (again 2nd and 4th, respectively) fairly quickly after the first few large time steps. After a large fall and spike in order-of-accuracy that was indicated by Figure 4-14 the Hu RK56 goes to 4th-order, which is its formal order-of-accuracy. The sharp drop in order-of-accuracy for the three 4th-order schemes for the smallest time steps is a result of either the accuracy of the EVA solution being exhausted, the spatial discretization error overwhelming the error from the temporal scheme, or the error of the schemes reaching the precision of the computer. This behavior is not observed in the 2nd-order schemes because the magnitude of the error is much larger than the 4th-order schemes (judging from the L_2 plot, at least an order-of-magnitude and at most about five).

Each two-dimensional time-marching scheme in BASS was also run on a rotated curvilinear grid in order to more strongly Verify the schemes. The initial condition used was identical to the uniform case, i.e., Equations (4.23)–(4.25), and the final time level was also identical at $t = 0.25$. The EVA solution was found on a uniform Cartesian grid rotated 45° counter-clockwise — the grid extended from -2.0 to 2.0 in both directions and included 3^2 grid points. The equations used to create the base grid for the BASS runs are:

$$x(i,j) = x_{\min} + (i-1)\Delta x + 0.02 \sin\left[\frac{\pi}{2}(y_{\min} + (j-1)\Delta y)\right] \quad (4.26)$$

$$y(i,j) = y_{\min} + (j-1)\Delta y + 0.01 \sin\left[\frac{\pi}{2}(x_{\min} + (i-1)\Delta x)\right] \quad (4.27)$$

Like the EVA grid, the BASS grids were rotated 45° degrees. The BASS grids extended from -4.0 to 4.0 and contained 129^2 grid points for the 2nd-order schemes and 257^2 grid points for the 4th-order schemes. The explicit 6th-order central spatial differencing scheme was used for all BASS runs. Figure 4-16 shows the $L_{\max}$ error norm of each of the time-marching schemes. Judging by the slope of the curves,

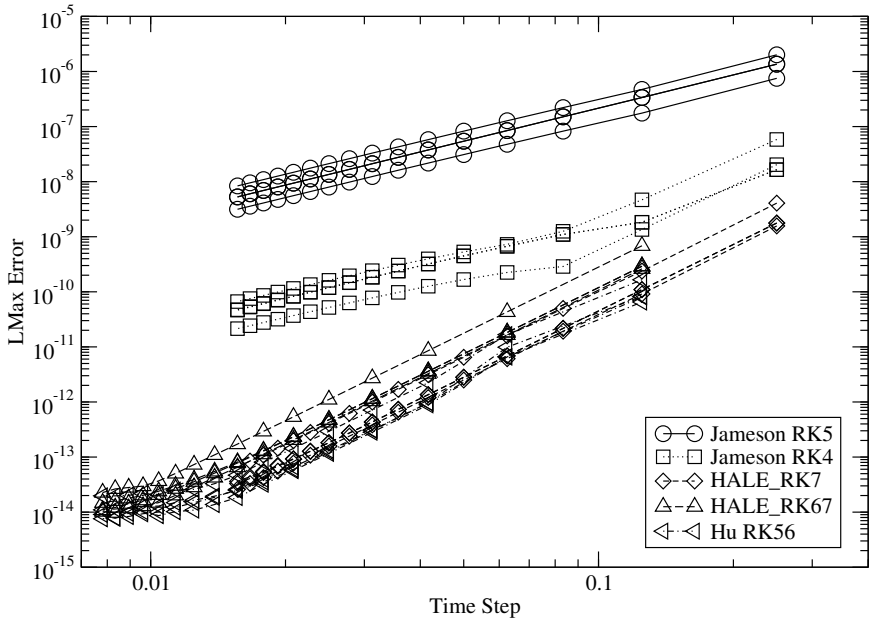


Figure 4-16: Euler 2D Temporal $L_{\max}$ Error — Curvilinear Grid

both Jameson schemes show 2nd-order results and the HALE_RK7, HALE_RK67 and Hu RK56 show 4th-order. As in previous temporal test cases, the three 4th-order schemes' error "flattens out" for the smallest time steps as the time-marching scheme's contribution to the error no longer dominates. Figure 4-17 shows the observed order-of-accuracy for each of the time marching schemes. Each scheme reaches its formal order-of-accuracy for a range of time step sizes. As indicated by the $L_{\max}$ error plot the observed order-of-accuracy drops sharply for the 4th-order schemes for the smallest time steps. Both the Jameson RK4 and Hu RK56 schemes show large variations in observed order-of-accuracy for each scheme's largest time steps. Both these schemes have comparatively lower stability limits, so it is likely that both schemes are unstable for the largest few time steps (the calculation does not "blow up" because so few time steps are taken).

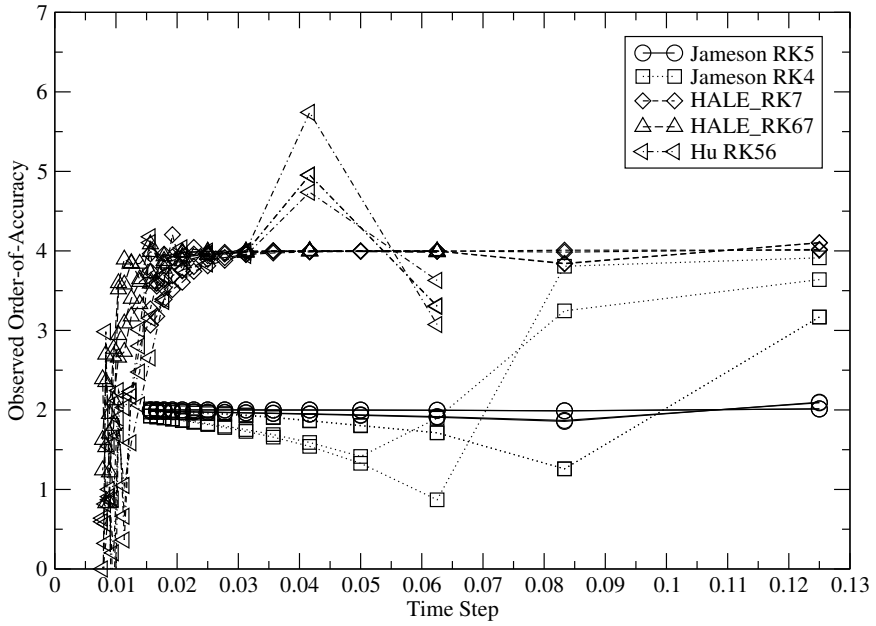


Figure 4-17: Euler 2D Temporal Observed Order-of-Accuracy — Curvilinear Grid

4.4.2 Three-Dimensional

As in Section 4.4.1, Verification of the temporal order-of-accuracy of the BASS code was first preformed on uniform Cartesian grids. The initial condition used was similar to what was used in Section 4.4.1:

$$\vec{Q}(x, y, t = 0) = \bar{Q} + \tilde{Q} \exp\left[-\frac{\ln(2)}{9.0}(x^2 + y^2 + z^2)\right] \tag{4.28}$$

where

$$\bar{Q} = \begin{Bmatrix} \bar{\rho} \\ \overline{\rho u} \\ \overline{\rho v} \\ \overline{\rho w} \\ \bar{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 1.0 \\ 0.1 \\ 0.1 \\ 0.1 \\ 1.78571425 \end{Bmatrix} \tag{4.29}$$

and

$$\tilde{Q} = \left\{ \begin{array}{c} \tilde{\rho} \\ \widetilde{\rho u} \\ \widetilde{\rho v} \\ \widetilde{\rho w} \\ \tilde{E}_{total} \end{array} \right\} = \left\{ \begin{array}{c} 0.1 \\ 0.01 \\ 0.01 \\ 0.01 \\ 0.1 \end{array} \right\} \tag{4.30}$$

The final time was again $t = 0.25$. The EVA solution was computed on a uniform grid with 9^3 points extending from -2.0 to 2.0. The BASS code runs were preformed on a uniform grid extending from -5.0 to 5.0 with 161^3 grid points. The explicit 6th-order central spatial differencing scheme was used for all BASS runs. The BASS code was run for sixteen different Δt values: $0.25, 0.125, 0.083333, 0.0625, \ldots, 0.015625$ for the single-step schemes and $0.125, 0.0625, 0.041667, 0.03125, \ldots, 0.0078125$ for the two-step schemes. Figure 4-18 shows the $L_{\max}$ error for each of the time-marching schemes.

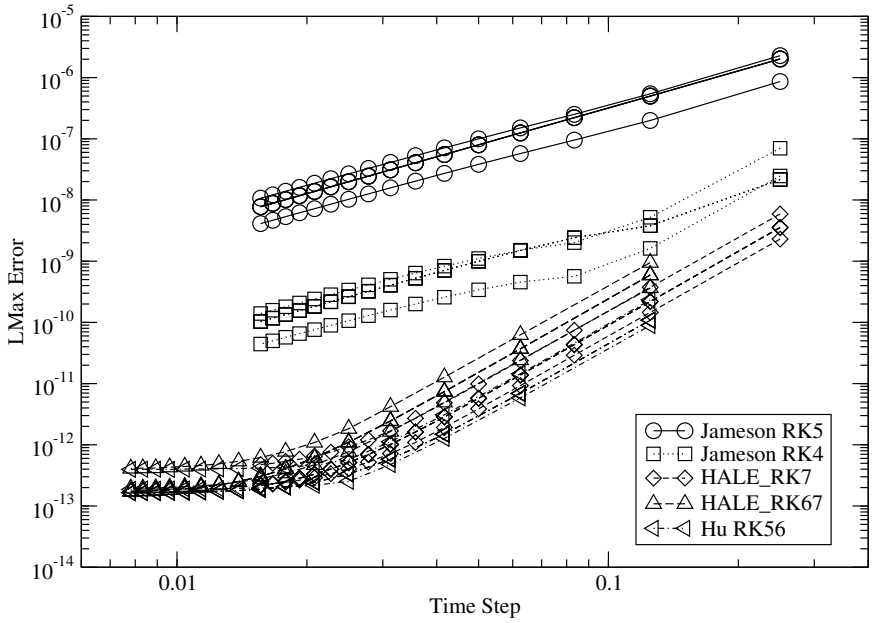


Figure 4-18: Euler 3D Temporal $L_{\max}$ Error — Uniform Grid

As in previous test cases, the Jameson RK5 scheme's error appears to decrease at a uniform rate throughout the entire range of time steps. The Jameson RK4 scheme's error shows the typical "blip" for the larger time steps before it "settles down" to a constant rate. Each of the 4th-order schemes' error flattens out for the smallest time steps as the time-marching scheme's truncation error decreases to the point where it is no longer the dominate part of the error. Figure 4-19 shows the observed order-of-

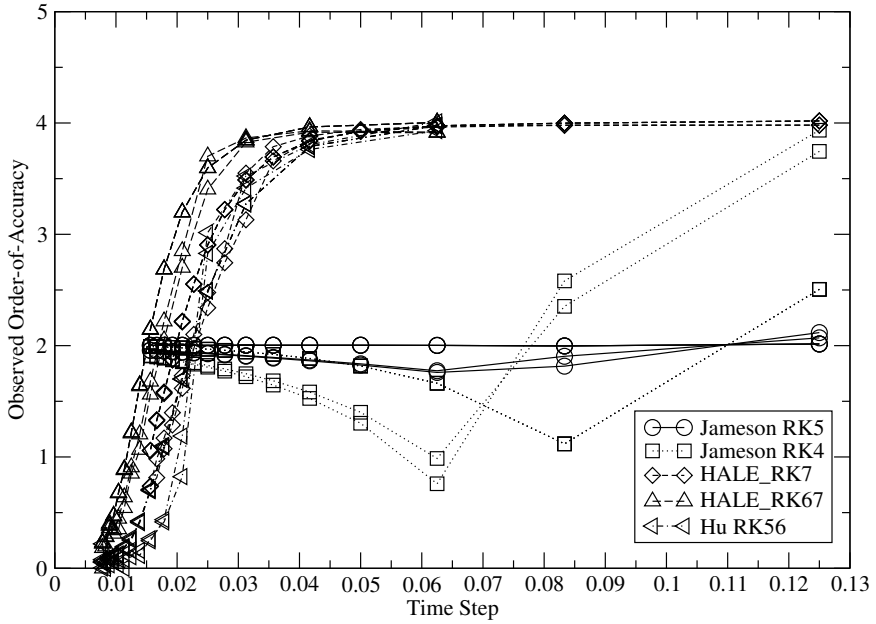


Figure 4-19: Euler 3D Temporal Observed Order-of-Accuracy — Uniform Grid

accuracy for the sixteen BASS runs. As expected from consideration of Figure 4-18, the Jameson RK5 scheme's error is decreasing at approximately a 2nd-order rate for the entire range of Δt values, and the Jameson RK4 goes to 2nd-order after overshooting and then undershooting its formal order-of-accuracy. The HALE_RK7 scheme returns 4th-order for the largest Δt values before sharply dropping off as the magnitude of the error (again, shown in Figure 4-18) becomes very small. The HALE_RK67 and Hu RK56 — both two-step schemes — show 4th-order results before

quickly dropping off. High-accuracy multi-step schemes like the HALE_RK67 and Hu RK56 are particularly challenging to Verify, as they require taking multiple time steps to the final time level, thus reducing Δt and the temporal truncation error even further and increasing the accuracy requirement of the EVA solution.

As in previous test cases, the three-dimensional time-marching schemes of the BASS code were also run on curvilinear grids in order to Verify the BASS code more strongly. The initial condition used for these test cases was identical to the uniform test case:

$$\vec{Q}(x, y, z, t = 0) = \bar{Q} + \tilde{Q} \exp\left[-\frac{\ln(2)}{9.0}(x^2 + y^2 + z^2)\right] \tag{4.31}$$

where

$$\bar{Q} = \begin{Bmatrix} \bar{\rho} \\ \overline{\rho u} \\ \overline{\rho v} \\ \overline{\rho w} \\ \bar{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 1.0 \\ 0.1 \\ 0.1 \\ 0.1 \\ 1.78571425 \end{Bmatrix} \tag{4.32}$$

and

$$\tilde{Q} = \begin{Bmatrix} \tilde{\rho} \\ \widetilde{\rho u} \\ \widetilde{\rho v} \\ \widetilde{\rho w} \\ \tilde{E}_{total} \end{Bmatrix} = \begin{Bmatrix} 0.1 \\ 0.01 \\ 0.01 \\ 0.01 \\ 0.1 \end{Bmatrix} \tag{4.33}$$

The EVA solution was found on a curvilinear grid of the form

$$\begin{aligned}
x(i,j,k) &= x_{\min} + (i-1)\Delta x + 0.06250 \sin\left[\tfrac{\pi}{2}(y_{\min} + (j-1)\Delta y)\right] \\
&+ 0.03125 \sin\left[\pi(z_{\min} + (k-1)\Delta z)\right] \\
y(i,j,k) &= y_{\min} + (j-1)\Delta y + 0.12500 \sin\left[\tfrac{\pi}{4}(x_{\min} + (i-1)\Delta x)\right] \\
&+ 0.03125 \sin\left[\pi(z_{\min} + (k-1)\Delta z)\right] \\
z(i,j,k) &= z_{\min} + (k-1)\Delta z + 0.12500 \sin\left[\tfrac{\pi}{4}(x_{\min} + (i-1)\Delta x)\right] \\
&+ 0.06250 \sin\left[\tfrac{\pi}{2}(y_{\min} + (j-1)\Delta y)\right]
\end{aligned} \tag{4.34}$$

with a final time level of $t = 0.25$. The EVA domain ranged from -1.0 to 1.0 with 3^3 grid points. After using Equation (4.34) to construct the grid, transformation matrices were used to rotate it 45.0° about the x-axis, 30.0° about the y and 27.4° about the z. Equation (4.34) was also used to construct the BASS grids with one modification: the amplitudes of each of the sine functions were divided by ten. Different numbers of grid points were found to be necessary for some of the schemes: the Jameson RK5, Jameson RK4 and HALE_RK7 used 97^3, the HALE_RK67 used 121^3 and the Hu RK67 used 145^3. The explicit 6th-order central scheme was used for the spatial differencing. Figure 4-20 shows the L_2 error for each of the time marching schemes. As in the previous test cases, the Jameson RK5 is the least accurate scheme, followed by the Jameson RK4, and both of these schemes appear to be 2nd-order accurate. Each of the 4th-order schemes "flatten out" for the very small time step values, likely indicating that the error from the spatial differencing is overwhelming the temporal error at these small time steps. Figure 4-21 shows the observed order-of-accuracy of each of the BASS time-marching schemes. As the L_2 error plot indicated, the two Jameson schemes achieve their formal order-of-accuracy for the larger time steps while the three 4th-order schemes reach their formal order-of-accuracy for the larger time steps. The sharp drop in order-of-accuracy for the 4th-order schemes is also predicted by the L_2 error plot. Because each scheme reaches its formal order-of-accuracy, one

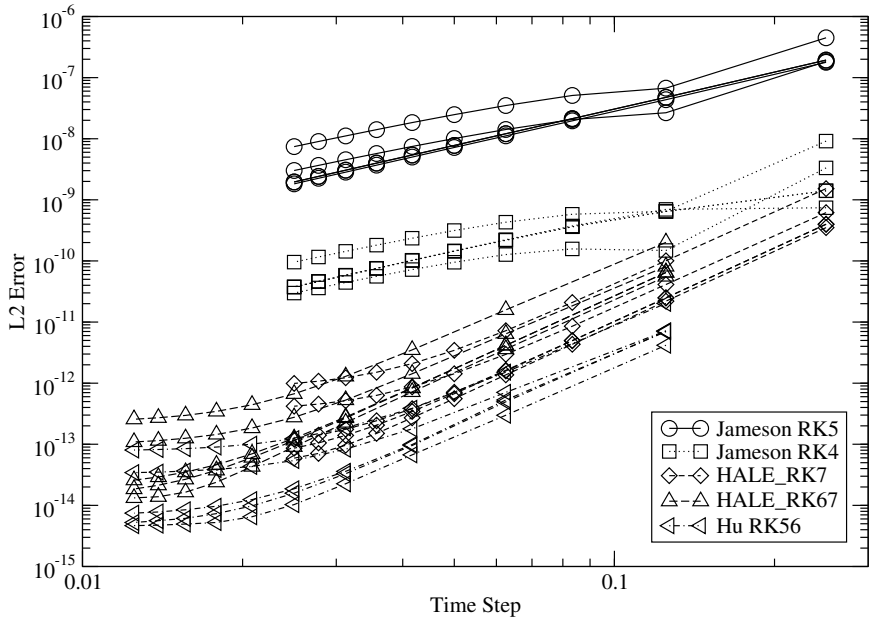


Figure 4-20: Euler 3D Temporal L_2 Error — Curvilinear Rotated Grid

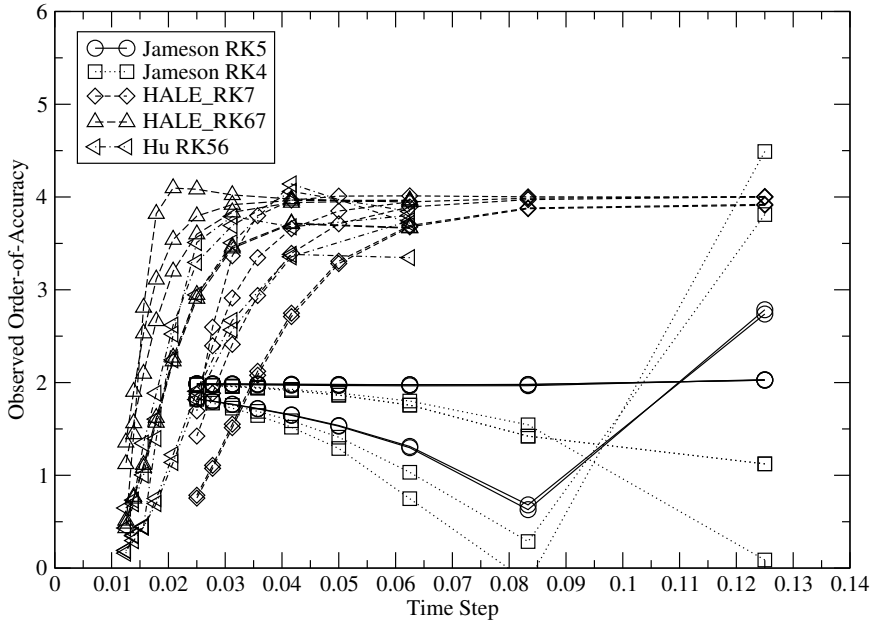


Figure 4-21: Euler 3D Temporal Observed Order-of-Accuracy — Curvilinear Rotated Grid

may conclude that each is correctly implemented in BASS for the three-dimensional case on non-uniform grids.

4.5 Temporal Order-of-Accuracy Results — Multiple EVA Solutions

As discussed in Section 2.2.5, an alternative method of Verifying the temporal order-of-accuracy consists of using multiple EVA solutions and marching the code to be Verified one time step for each run. This method has the advantage of lowering the accuracy requirement and computation cost of the EVA solution. Results are shown for the two- and three-dimensional Euler Equations on rotated curvilinear grids.

4.5.1 Two-Dimensional

The initial condition used to Verify the two-dimensional temporal schemes in the BASS code was:

$$\begin{aligned} \rho(x,y) &= 1.0 + 0.1 \exp\left[\tfrac{-\ln(2)}{9.0}\left((x-0.5)^2 + y^2\right)\right] \\ \rho u(x,y) &= 0.5 + 0.01 \exp\left[\tfrac{-\ln(2)}{9.0}\left(x^2 + (y-1.0)^2\right)\right] \\ \rho v(x,y) &= 0.5 + 0.01 \exp\left[\tfrac{-\ln(2)}{9.0}\left((x-1.0)^2 + y^2\right)\right] \\ E_{total}(x,y) &= 1.78571425 + 0.1 \exp\left[\tfrac{-\ln(2)}{9.0}\left(x^2 + (y+0.5)^2\right)\right] \end{aligned} \tag{4.35}$$

The EVA solutions were found on a uniform Cartesian grid rotated 45° with 5^2 grid points extending from -2.0 to 2.0 in both coordinate directions. Thirty-two EVA solutions were found with final time levels consisting of integer multiples of 0.001953125 (i.e., $0.001953125, 0.00390625, 0.005859375, 0.0078125, \ldots, 0.0625$). The BASS code was run on a curvilinear grid rotated 45°; the grid before the rotation was constructed

using the equations:

$$x(i,j) = x_{\min} + (i-1)\Delta x + 0.0001 \sin\left[2\pi(y_{\min} + (j-1)\Delta y)\right] \quad (4.36)$$

$$y(i,j) = y_{\min} + (j-1)\Delta y + 0.0010 \sin\left[\pi(x_{\min} + (i-1)\Delta x)\right] \quad (4.37)$$

For the single-step time-marching schemes, the BASS grid extended from -4.0 to 4.0 and contained 129^2 grid points, giving an average grid spacing $\Delta x = \Delta y = 0.0625$. The grid for the two-step time-marching schemes was identical in form but used 257^2 grid points, giving an average grid spacing $\Delta x = \Delta y = 0.03125$. The denser grid was used for the two-step schemes in order to keep the CFL numbers consistent from scheme to scheme, as the two-step schemes require two time steps to reach the final time levels discussed above, thus halving Δt. Since each Runge-Kutta scheme is at most a 4th-order scheme, a 5th-order EVA Taylor series solution was initially used; however, it was found that a 6th-order solution was necessary to Verify the two-step schemes, as they use half the time step of the EVA solution. The explicit 6th-order central spatial differencing scheme was used for all BASS runs. Figure 4-22 shows the L_2 error norm for each of the time marching schemes. More separation between in the error of each of the solution variables for a given scheme can be seen, likely due to the asymmetric initial condition used. The Jameson RK5 scheme, as in previous cases, is considerably less-accurate than the other schemes and appears to converge at a fairly uniform rate. The results for the Jameson RK4 scheme are more interesting: three of the four solution variables appear to converge at the same rate as the Jameson RK5 and one (density) converges much faster. This is not expected, as the Jameson RK4 is formally a 2nd-order scheme for nonlinear equations. Numerous test cases were run using the Jameson RK4 (using different initial conditions, grids, order of the EVA solution, etc.) but this behavior was always observed. The explanation may lie in the fact that the continuity equation is a linear PDE, and the Jameson

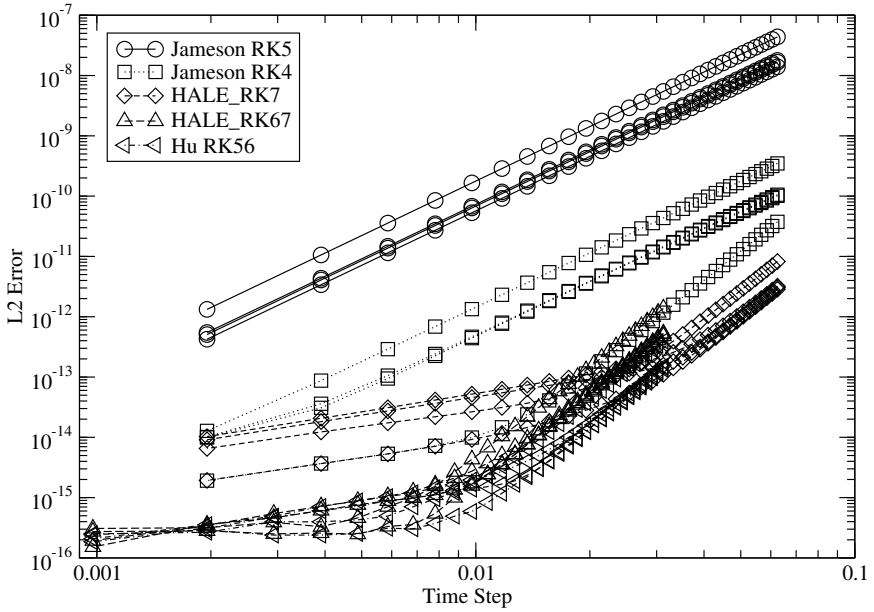


Figure 4-22: Euler 2D Temporal L_2 Error — Curvilinear Rotated Grid — Multiple EVA Solutions

RK4 is 4th-order accurate for linear problems. It may be the case that the Jameson RK4 scheme requires multiple time steps to "build up" nonlinearity in the density variable and return the correct order-of-accuracy. The HALE_RK7, HALE_RK67 and Hu RK56 seem to converge at a uniform rate for the larger time steps and then begin to "flatten out" as the time step decreases. This is likely due to the error from the spatial differencing scheme — as the time step is reduced, the spatial differencing truncation error gradually becomes more significant relative to the temporal scheme's truncation error, thus reducing the convergence rate. The fact that the level of error at which this happens seems to be more or less consistent from scheme to scheme (recall that a finer grid spacing was used with the two-step schemes) supports this belief. Figure 4-22 shows the observed order-of-accuracy for each of the BASS runs. The Jameson RK5 uniformly shows it is a 2nd-order scheme. The Jameson RK4 does as well, expect for the density solution variable, which behaves as a 4th-order scheme. The HALE_RK7 shows 4th-order convergence for the larger time steps for

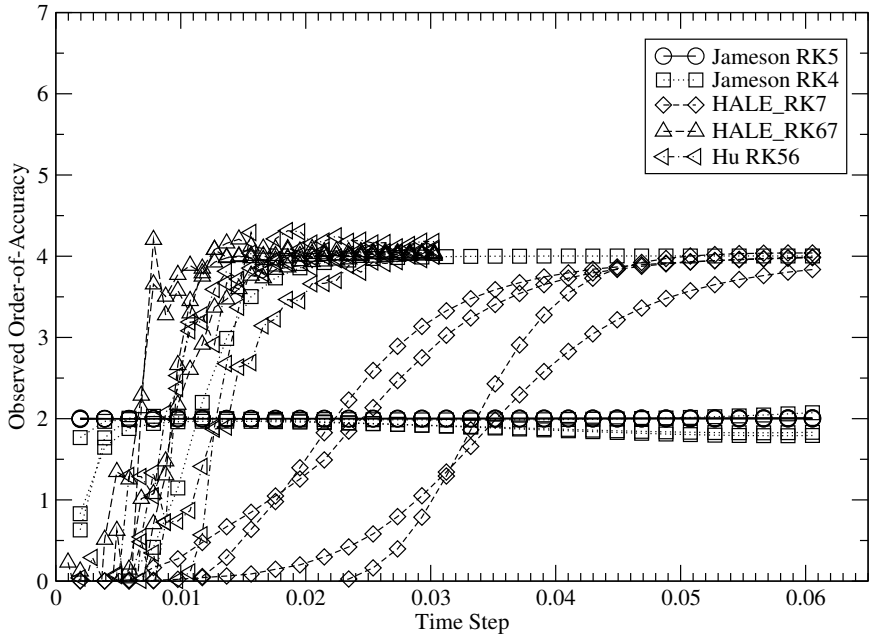


Figure 4-23: Euler 2D Temporal Observed Order-of-Accuracy — Curvilinear Rotated Grid — Multiple EVA Solutions

each of its solution variables. The HALE_RK67 and Hu RK56 both converge at a slightly higher rate than expected — this may be a result of their "optimized" nature, especially with the Hu RK56 scheme, as it is designed to converge very quickly for larger time steps.

4.5.2 Three-Dimensional

The initial condition used to Verify the three-dimensional temporal schemes in BASS was:

$$\begin{aligned}
\rho(x,y,z) &= 1.0 + 0.1\exp\left[\tfrac{-\ln(2)}{9.0}\left((x+1.0)^2+(y+1.0)^2+z^2\right)\right] \\
\rho u(x,y,z) &= 0.5 + 0.4\exp\left[\tfrac{-\ln(2)}{9.0}\left(x^2+(y-0.5)^2+z^2\right)\right] \\
\rho v(x,y,z) &= 0.4 + 0.3\exp\left[\tfrac{-\ln(2)}{9.0}\left(x^2+y^2+(z-0.5)^2\right)\right] \\
\rho w(x,y,z) &= 0.3 + 0.2\exp\left[\tfrac{-\ln(2)}{9.0}\left((x-0.5)^2+y^2+z^2\right)\right] \\
E_{total}(x,y,z) &= 1.78571425 + 0.1\exp\left[\tfrac{-\ln(2)}{9.0}\left((x-0.5)^2+(y-0.5)^2+z^2\right)\right]
\end{aligned} \tag{4.38}$$

The EVA solutions were found on a curvilinear grid rotated 45.0° about the x-axis, 30.0° about the y-axis and 27.4° about the z-axis with 3^3 grid points extending from -1.0 to 1.0. The BASS grids were identical to the EVA grids but contained 129^3 grid points and extended from -2.0 to 2.0. The equations used to create the grid were

$$\begin{aligned}
x(i,j,k) &= x_{\min} + (i-1)\Delta x + 0.031250\sin\left[\tfrac{\pi}{6}(y_{\min}+(j-1)\Delta y)\right] \\
&+ \qquad\qquad 0.015625\sin\left[\tfrac{\pi}{6}(z_{\min}+(k-1)\Delta z)\right] \\
y(i,j,k) &= y_{\min} + (j-1)\Delta y + 0.062500\sin\left[\tfrac{\pi}{6}(x_{\min}+(i-1)\Delta x)\right] \\
&+ \qquad\qquad 0.015625\sin\left[\tfrac{\pi}{6}(z_{\min}+(k-1)\Delta z)\right] \\
z(i,j,k) &= z_{\min} + (k-1)\Delta z + 0.062500\sin\left[\tfrac{\pi}{6}(x_{\min}+(i-1)\Delta x)\right] \\
&+ \qquad\qquad 0.031250\sin\left[\tfrac{\pi}{6}(y_{\min}+(j-1)\Delta y)\right]
\end{aligned} \tag{4.39}$$

where Δx, Δy and Δz of the EVA grid will differ from that of the BASS grid. Twenty-four BASS/EVA solutions were found with final time levels consisting of integer multiples of 0.001953125 (i.e., $0.001953125, 0.00390625, 0.005859375, 0.0078125, \ldots, 0.046875$). A fifth- and sixth-order EVA Taylor series solution was used to calculate the error for the single- and two-step schemes, respectively. The explicit 6th-order central scheme was used for each BASS run. Figures 4-24 and 4-25 show the error norm and order-

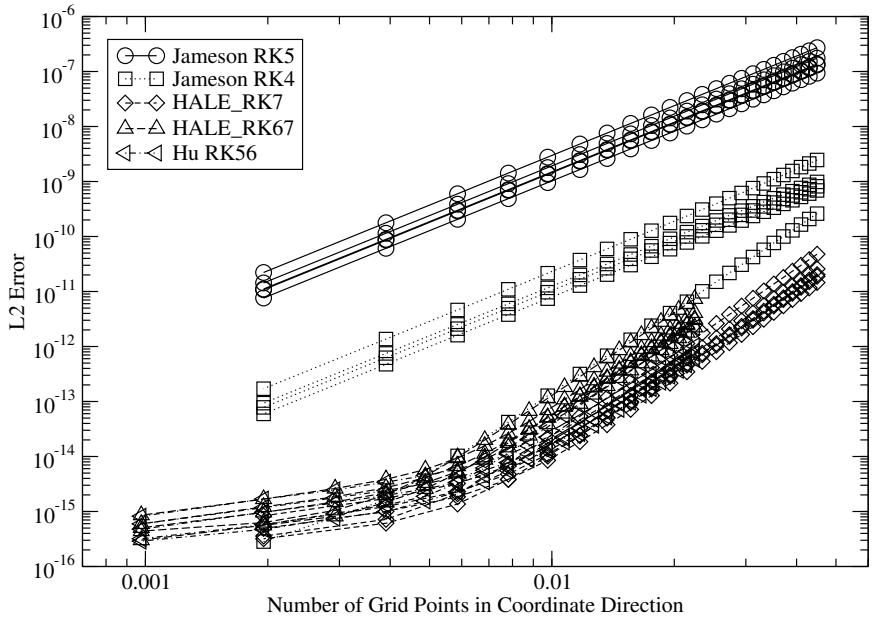


Figure 4-24: Euler 3D Temporal L_2 Error — Curvilinear Rotated Grid — Multiple EVA Solutions

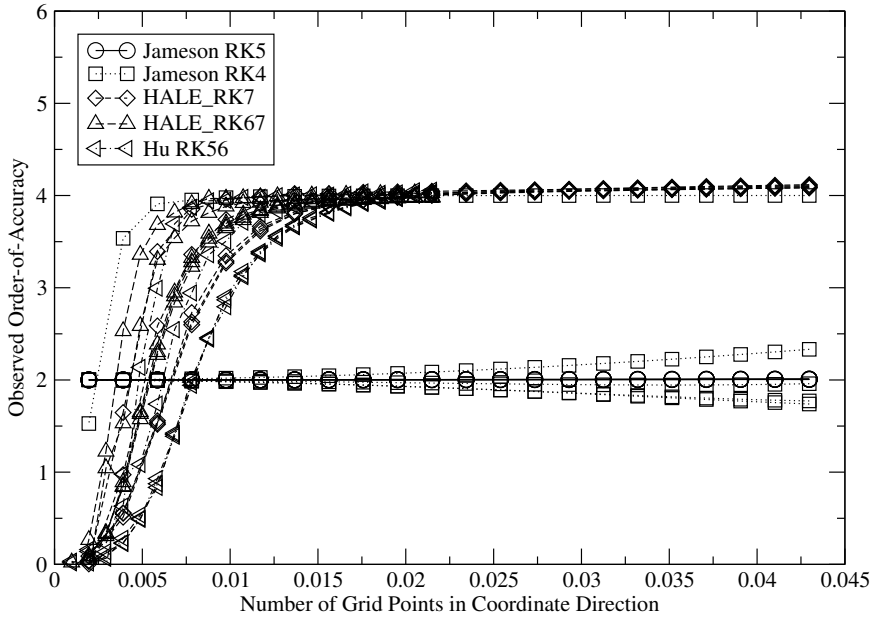


Figure 4-25: Euler 3D Temporal Observed Order-of-Accuracy — Curvilinear Rotated Grid — Multiple EVA Solutions

of-accuracy, respectively, for each of the BASS runs. Results are very similar to the corresponding two-dimensional results. The Jameson RK5 is again the least accurate scheme and converges at a 2nd-order rate for all of the time steps. The Jameson RK4 scheme shows the correct 2nd-order accuracy for all of the flow variables except density, which shows 4th-order convergence. It was thought that the larger fluctuating component of the momentum variables in Equation (4.38) would cause the continuity equation to behave in a more nonlinear manner, but this was not the case. Again, it is thought that using multiple time steps with the scheme would fix this issue. The results for the 4th-order schemes are similar to previous findings. The Hu RK56 scheme shows 4th-order accuracy for the fewest number of time steps — its error approaches machine precision the fastest.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this work, the method of External Verification Analysis (EVA) was discussed and used to Verify two codes, including FD1D, a simple linear/nonlinear inviscid Burgers' equation solver using the finite difference method and BASS, the NASA Glenn Research Center's high-order Computational Aeroacoustic code. For all test cases the observed order-of-accuracy of each spatial and temporal discretization scheme matched the formal order-of-accuracy of the scheme, indicating each is correctly implemented and thus Verified. It was found through experience with using the EVA3 tool that the spatial discretization schemes are more easily Verified than the temporal. Two methods of Verifying the temporal order-of-accuracy of a time-marching scheme were presented; the first used one very high-order EVA solution and marched the code to be Verified various numbers of time steps to the EVA solution time level, and the second used multiple lower-order EVA solutions at a range of time levels and marched the code to be Verified one time step to each EVA solution. It was found that the multiple-EVA-solution technique was computationally more efficient than using a single EVA solution, though one must be careful of being mislead by

superconvergent behavior caused by unstable CFLs. No modifications to either code or any governing equations were necessary to use EVA, which is its main advantage over the Method of Manufactured Solutions.

5.2 Potential Applications of the EVA Method

In this work the EVA method was used to Verify two CFD/CAA codes: the FD1D code and the BASS code. Both of these codes use the Finite Difference method and structured grids in their solvers. It would be interesting to use EVA to Verify codes using other numerical strategies, such as the Finite Element, Finite Volume or Spectral methods, and unstructured grids/meshes. As noted in Section 2.2, the EVA method is not influenced by the numerical method used by the code to be Verified or the style of grid used (structured vs. unstructured, uniform vs. nonuniform).

Another application that the EVA method may be applied to is determining the effect of grid singularities on the accuracy of CFD/CAA codes. Grid singularities are points on the interfaces of grid blocks with more or less than 4 or 6 adjacent grid points in a structured two- or three-dimensional grid, respectively, that arise frequently in the grid generation process. Again, since the EVA method is not influenced by the grid spacing or structure, a very accurate solution to the flow at the grid singularity could be found and then used to investigate the behavior of the spatial differencing error near a grid singularity as the grid is refined.

Yet another interesting application that the EVA method has not been applied to is Verifying the spatial and temporal order-of-accuracy for a code using grid motion. This could be accomplished by moving the grid during the calculation without any wall boundaries present in the domain — thus the grid motion should not affect the solution and the EVA tool could still be used.

5.3 Proposed Extensions to the EVA3 Code

The usefulness of the EVA3 code in Verifying high-order CFD/CAA codes has been shown in this work; however, added functionality would make the EVA3 code even more effective. Perhaps the most obvious improvement would be to add functionality that would allow the EVA3 code to solve the Navier-Stokes equations. The EVA3 code manipulates expressions symbolically, making it easy to add new governing equations. Adding any form of the Navier-Stokes equations, however, would require some modification of the EVA3 code, as it currently assumes that all fluxes are functions only of the solution vector, (for example) $\vec{F} = \vec{F}(\vec{Q})$, which affects how the Jacobians of the fluxes are evaluated. This assumption is not true for the Navier-Stokes equations, as (for example) $\vec{F} = \vec{F}(\vec{Q}, \vec{Q}_x, \vec{Q}_y, \vec{Q}_z)$.

Another area of improvement for the EVA3 tool is parallelization. Currently EVA3 can only be run on one processor. Parallelizing the EVA3 code has the potential to drastically improve the two most computationally-intensive functions of EVA3: finding expressions for the temporal and mixed-spatial-temporal derivatives of $\vec{Q}$ (i.e., the terms in Equations (2.3)–(2.7)) and evaluating the solution at each grid point in the domain. For the latter function, because the EVA solution at a given point does not need information from neighbouring points to proceed, no information would need to be exchanged between nodes during the calculation. Finding expressions for the temporal and mixed-spatial-temporal derivatives could also proceed without any communication if each expression was broken up into n "pieces", where n is the number of processes in the calculation. Each process could find the derivative of its "piece" in isolation from the other processes, and the final result could be "assembled" after all processes finish. These modifications have the potential to lower the computational effort of the entire EVA process.

References

[1] Abanto, J., Pelletier, D., Garon, A., Trépanier, J.-Y., and Reggio, M., "Verification of some Comercial CFD Codes on Atypical CFD Problems," *43rd AIAA Aerospace Sciences Meeting and Exhibit* [27].

[2] Roache, P. and Steinburg, S., "Symbolic manipulation and computational fluid dynamics," *AIAA Journal*, Vol. 22, October 1984, pp. 1390–1394.

[3] Hixon, R. and Andersen, B., "Verification of Unsteady CFD Codes using the Method of Manufactured Solutions," *RTO-MP-AVT-147*, Athens, Greece, December 2007, Paper 11.

[4] Boehm, B. W., *Software Engineering Economics*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[5] Blottner, F. G., "Accurate Navier-Stokes Results for the Hypersonic Flow over a Spherical Nosetip," *AIAA Journal of Spacecraft and Rockets*, Vol. 27, No. 2, March-April 1990, pp. 113–122.

[6] Roache, P., *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, New Mexico, 1998.

[7] Roache, P., "Quantification of Uncertainty in Computational Fluid Dynamics," *Annual Review of Fluid Mechanics*, Vol. 29, 1997, pp. 123–160.

[8] Roache, P. J., "Verification of Codes and Calculations," *AIAA Journal*, Vol. 36, No. 5, May 1998, pp. 696–702.

[9] Oberkampf, W. L. and Trucano, T. G., "Verification and validation benchmarks," *Nuclear Engineering and Design*, 2007.

[10] Knupp, P. and Salari, K., *Verification of Computer Codes in Computational Science and Engineering*, Chapman & Hall/CRC, 2003.

[11] Salari, K. and Knupp, P., "Code Verification by the Method of Manufactured Solutions," Tech. Rep. SAND2000-1444, Sandia National Laboratories, June 2000.

[12] Roy, C. J., "Review of code and solution verification procedures for computational simulation," *Journal of Computational Physics*, Vol. 205, No. 1, 2005, pp. 131 – 156.

[13] Roy, C. J., Nelson, C. C., Smith, T. M., and Ober, C. C., "Verification of Euler/Navier-Stokes Codes using the Method of Manufactured Solutions," *International Journal for Numerical Methods in Fluids*, Vol. 44, No. 6, 2004, pp. 599–620.

[14] Bond, R., Knupp, P., and Ober, C., "A Manufactured Solution for Verifying CFD Boundary Conditions," *34th AIAA Fluid Dynamics Conference and Exhibit*, 2004.

[15] Bond, R., Knupp, P., and Ober, C., "A Manufactured Solution for Verifying CFD Boundary Conditions, Part II," *43rd AIAA Aerospace Sciences Meeting and Exhibit* [27].

[16] Bond, R., Ober, C., and Knupp, P., "A Manufactured Solution for Verifying CFD Boundary Conditions, Part III," *36th AIAA Fluid Dynamics Conference and Exhibit*, 2006.

[17] Eça, L., Hoekstra, M., Hay, A., and Pelletier, D., "Verification of RANS Solvers with Manufactured Solutions," *Engineering with Computers*, 2007.

[18] Lele, S. K., "Compact finite difference schemes with spectral-like resolution," *Journal of Computational Physics*, Vol. 103, No. 1, November 1992, pp. 16–42.

[19] Tam, C. and Webb, J., "Dispersion-Relation-Preserving Finite Difference Schemes for Computational Acoustics," *Journal of Computational Physics*, Vol. 107, No. 2, 1993, pp. 262–281.

[20] Hu, F. Q., Hussaini, M. Y., and Manthey, J. L., "Low-dissipation and low-dispersion Runge-Kutta schemes for computational acoustics," *Journal of Compututational Physics*, Vol. 124, No. 1, 1996, pp. 177–191.

[21] Allampalli, V., Hixon, R., Nallasamy, M., and Sawyer, S. D., "High-accuracy large-step explicit Runge-Kutta (HALE-RK) schemes for computational aeroacoustics," *Journal of Computational Physics*, Vol. 228, No. 10, 2009, pp. 3837 – 3850.

[22] Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," *AIAA 10th Computational Fluid Dynamics Conference*, Honolulu, HI, June 1991, AIAA 91-1596.

[23] Smitch, R. T. and Minton, R. B., *Single Variable Calculus*, McGraw-Hill, 2nd ed., 2002.

[24] Anderson, D. A., Tannehill, J. C., and Pletscher, R. H., "Computational fluid mechanics and heat transfer," Series in Computational Methods in Mechanics and Thermal Sciences, Hemisphere, Philadelphia, 1984, pp. 40–41.

[25] Nadjafikhah, M., "Exact solution of generalized inviscid Burgers' equation," 2009.

[26] R. Hixon, M. Nallasamy, S. S., "Parallelization Strategy for an Explicit Computational Aeroacoustics Code," *8th AIAA/CEAS Aeroacoustics Conference and Exhibit*, Breckenridge, Colorado, June 2002, AIAA 2002-2583.

[27] *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.