18th Australasian Fluid Mechanics Conference
Launceston, Australia
3-7 December 2012

Verification of a Compressible Flow Solver

R. J. Gollan[1] and P. A. Jacobs[2]

[1]Centre for Hypersonics
The University of Queensland, Queensland 4072, Australia

[2]Queensland Centre of Excellence for Geothermal Energy
The University of Queensland, Queensland 4072, Australia

Abstract

In the development of computational fluid dynamics software, the verification and validation of the implementation are important tasks. In this paper, we present work towards the verification of a compressible flow code, `Eilmer`. The verification case uses a manufactured solution of inviscid supersonic flow to test the numerical simulations, and is used to show that various flux calculators are all implemented correctly in the code. We also demonstrate how to assess the effects of grid nonuniformity on the code's order of spatial accuracy by using the manufactured solution case.

Introduction

In this paper, we present work on the verification of a compressible flow solver, `Eilmer`. `Eilmer` is a code which has been developed at the University of Queensland, and has historically been used to simulate the flow in hypersonic testing facilities. More recently, the code base had been adapted and extended for use in other application areas of compressible flow such as interior ballistics modelling [2] and turbomachinery modelling [14].

The `Eilmer` code continues to be under active development, and, as such, it is extremely useful to have a set of benchmark solutions to use for regression testing as development proceeds. Recently, an ongoing effort was begun to establish certain bechmark solutions for the solver as verification and validition cases. Here we use the terms 'verification' and 'validation' in the technical sense defined by Roache [10]: verification is defined as "solving the equations right"; and validation is defined as "solving the right equations".

To date, `Eilmer` has been verified using a range of cases that test various aspects of its use as a hypersonic flow solver. These cases include: use of a Manufactured Solution by Roy et al. [12] to verify the implementation for inviscid supersonic flows, and viscous subsonic flows; use of an analytical solution to an oblique detonation wave [9] to test the multi-species chemically-reacting gas modelling capabilities; and comparison to an analytical solution of shock tube flow of a gas in chemical equilibrium. In this paper, we focus on just one of these cases: verification of the implementation for inviscid supersonic flow. We demonstrate how the manufactured solution can be used to test the implementation of flux calculators and assess the effect of various nonuniform grids on the order of spatial accuracy. More generally, the verification work presented is also useful in the development and testing of other compressible flow solvers.

Flow Solver Description

The `Eilmer` code began its life as a single block integrator of the compressible Navier-Stokes equations, `CNS4U` [3], for two-dimensional geometries (planar and axisymmetric). Over the intervening years and a few name changes later (first to `mbcns` — multiple blocks but still 2D, and now to `Eilmer` — for combined 2D and 3D solver), a number of enhancements have been added to the code [4], including:

- more general thermochemistry, including look-up tables for a gas in chemical equilibrium;
- multiple-block capability with an MPI parallel implementation;
- a parametric and programmable front-end for specifying geometry and grids;
- three-dimensional geometry;
- programmable boundary conditions and source terms; and
- finite-rate chemistry.

The number-crunching core of the code is written in C++, while the pre- and post-processing stages are handled by a more user-friendly suite of Python programs and loadable libraries.

The finite-volume formulation of the code and the associated numerical schemes are described in the report by Jacobs [3] and more recently in the code's accompanying theory book [5]. The code integrates the compressible Navier-Stokes equations in a time-accurate manner. An explicit predictor-corrector update is used to advance the solution in time. The time advancement algorithm in `Eilmer` treats each piece of physics in a timestep-splitting (or operator-splitting) approach as advocated by Oran and Boris [8] (see Chapter 11 of their text). Thus the order of operations for a full multi-temperature reacting flow simulation applied over a small timestep, Δt, is:

1. compute gas transport due to inviscid flux
2. compute gas transport due to viscous flux
3. compute change of gas state due to chemical reactions
4. compute change of gas state due to thermal relaxation

For the inviscid transport of gas, the properties at the cell interfaces are reconstructed using a piecewise parabolic reconstructor. This is employed for high-order reconstruction. For low-order reconstruction, the adjacent cell centre properties are taken as the interface values. Based on the flow properties at cell interfaces, the inviscid flux is calculated using a flux calculator. In `Eilmer` there are four flux calculator options: 1) the advection upstream splitting method (AUSM) by Liou and Steffen Jr. [6], 2) the AUSMDV calculator by Wada and Liou [15], 3) the equilibrium flux method (EFM) by Macrossan [7], and 4) an adaptive calculator which uses EFM near shocks and AUSMDV elsewhere.

The viscous transport update is based on the calculation of viscous derivatives at the cell faces. The face-centred derivatives

are calculated as an average of the derivatives evaluated at the primary cell vertices. Those cell-vertex derivatives are calculated by application of Gauss' divergence theorem to convert a surface integral around a secondary volume to the derivative value within that volume, that is, at the vertex.

There are a number of various physical models of gases implemented in the code for various application domains. In simulation of hypersonics flows, for example, the gas models available are: mixture of ideal gases; mixture of thermally perfect gases; gas mixture in thermochemical equilibrium, and mixture of multi-temperature gases. The code is also able to compute finite-rate chemical effects, and the effects of thermal nonequilibrium in hypersonic flows.

Verification using the Method of Manufactured Solutions

The Method of Manufactured Solutions was first proposed by Steinberg and Roache [13]. The method involves choosing an analytical solution to the continuum partial differential equations that are solved by the code in question. The analytical solution is then passed through the differential operators of the governing equations to generate analytical source terms. The idea is that with these source terms included in the numerical simulation, the code should produce a numerical approximation to the original chosen analytical solution. Roache [11] advises that the solution should be non-trivial but analytical such that cross-derivative terms of the governing equations are tested. The numerical (discretised) solution may then be verified by comparison to the exact analytical solution. For a code that is behaving well, the error between the numerical and analytical solutions should diminish with increased grid resolution. Furthermore, it is possible to evaluate the order of the spatial accuracy by analysing the behaviour of successive grid refinements. The Method of Manufactured Solutions was demonstrated as a verification tool for compressible flow solvers by Roy et al. [12]. Roy et al. used the method to evaluate the order of spatial accuracy for two finite-volume codes which solve the Euler and Navier-Stokes equations. The verification case presented here is the first of two cases reported by Roy et al. This case is for the supersonic inviscid flow of an ideal gas, a flow regime that is frequently simulated in `Eilmer` calculations.

As part of the Method of Manufactured Solutions, an analytical solution is chosen for the primitive flow variables ρ, p, u and v. The particular solution chosen here is a function of sines and cosines.

$$\rho(x,y) = \rho_0 + \rho_x \sin\left(\frac{a_{\rho x}\pi x}{L}\right) + \rho_y \cos\left(\frac{a_{\rho y}\pi y}{L}\right) \quad (1)$$

$$u(x,y) = u_0 + u_x \sin\left(\frac{a_{ux}\pi x}{L}\right) + u_y \cos\left(\frac{a_{uy}\pi y}{L}\right) \quad (2)$$

$$v(x,y) = v_0 + v_x \cos\left(\frac{a_{vx}\pi x}{L}\right) + v_y \sin\left(\frac{a_{vy}\pi y}{L}\right) \quad (3)$$

$$p(x,y) = p_0 + p_x \cos\left(\frac{a_{px}\pi x}{L}\right) + p_y \sin\left(\frac{a_{py}\pi y}{L}\right) \quad (4)$$

The values for the constants in Equations 1–4 appear in table 1. The analytical solution is then substituted into the Euler equations and the analytical source terms are generated. The generation of source terms is tedious and error-prone to do by hand thus a computer algebra system (Maxima [1]) was used for this task.

The flow field for the numerical calculations was initialised to the conditions given by the ϕ_0 values given in table 1 and extends over the domain

$$0 \le x/L \le 1$$

$$0 \le y/L \le 1.$$

Table 1: Constants for the manufactured solution for a supersonic inviscid flow

Equation, ϕ	ϕ_0	ϕ_x	ϕ_y	$a_{\phi x}$	$a_{\phi y}$
ρ (kg/m^3)	1.0	0.15	-0.1	1.0	0.5
u (m/s)	800.0	50.0	-30.0	1.5	0.6
v (m/s)	800.0	-75.0	40.0	0.5	2/3
p (Pa)	1.0×10^5	0.2×10^5	0.5×10^5	2.0	1.0

The gas is modelled as a calorically perfect gas with $\gamma = 1.4$ and $R = 287.0$ J/(kg K), which are the values for ideal air. Exact Dirichlet values given by the manufactured solution are specified in the ghost cells at the south and west boundaries, while the north and east boundaries use an extrapolation boundary condition. All of the boundary conditions are first order in space, that is, a linear extrapolation is used at the boundaries. The boundary conditions are applied by using the programmable boundary condition option implemented in the code. Similarly, the specialised source terms that drive the manufactured solution are implemented using the programmable source terms option and evaluated based on the cell-centre spatial coordinates.

In order to rigorously test the implementation, the verification case has been computed a number of times exercising a range of code input options. As such, the test case was computed using various flux calculators, various reconstruction schemes, on different grid types, as a single block and as a multiple-block arrangement, and with and without reconstruction limiters. We will limit our focus here to verifying the flux calculator implementations and assessing the effect of different grid types.

To demonstrate verification, we need to show that the discretisation error drops as the grid is refined. Furthermore, the discretisation error should drop as $1/r^p$ where r is the grid refinement factor and p is the formal order of spatial accuracy. By computing a discretisation error at various levels of mesh refinement, an observed order of accuracy can then be compared to the formal order of accuracy of the numerical scheme. `Eilmer` is nominally 2nd order accurate in space when using the high-order reconstructor; we seek to show that the observed order of accuracy approaches this formal order of accuracy of $p = 2$. Following Roy et al. [12], the L_2 and L_∞ norms of the discretisation error are assessed in order to establish an observed order of accuracy. The norms on mesh level k are computed as

$$L_2\text{norm}_k = \left(\frac{\sum_{n=1}^{N} |\phi_n - \phi_{\text{exact}}|^2}{N}\right)^{1/2} \quad (5)$$

$$L_\infty\text{norm}_k = \max|\phi_n - \phi_{\text{exact}}|. \quad (6)$$

An observed order of accuracy between two successive grid refinement levels is computed as

$$p_k = \log\left(\frac{L_{k+1}}{L_k}\right) / \log(r) \quad (7)$$

where L_k is one of the error norms at the kth grid level, L_{k+1} is at the coarser level, and r is the grid refinement factor. In this case, the grid refinement factor is 2 between all grid levels. In total, the solutions were computed on five grids. The dimensions and average cell width for each of the grids is listed in table 2.

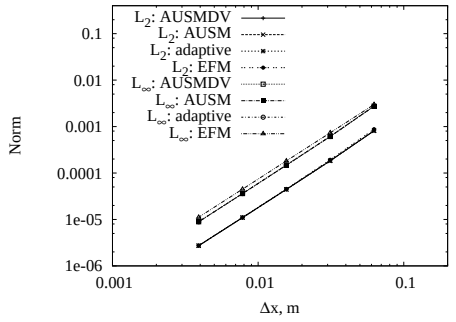
Comparison of flux calculators

As mentioned earlier in the description of the flow solver, there are a number of flux calculator options implemented in the code. The verification case was computed using each of these flux calculators in turn. The global discretisation errors for the L_2 and

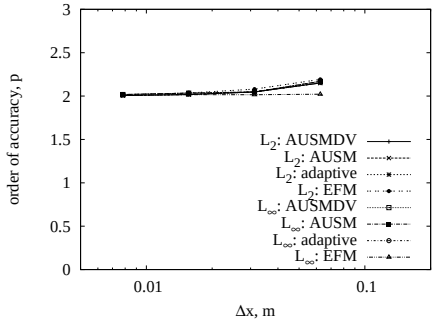
Table 2: Levels of grid refinement used for verification.

Grid	Dimensions	Cell width, Δx (m)
1	16×16	0.0625
2	32×32	0.03125
3	64×64	0.015625
4	128×128	7.8125×10^{-3}
5	256×256	3.90625×10^{-3}

L_∞ norms for each of the flux calcultors tested are shown as a function of cell width in figure 1(a). Figure 1(a) shows that the discretisation error is reducing in a consistent manner as the grid is refined for each of the flux calculators. The observed order of spatial accuracy is computed by comparing the error at two successive grid refinement levels. The results of computing equation 7 for each case are plotted in figure 1(b). The pleasing result is that the observed order of spatial accuracy has a value of 2 which matches the formal order of spatial accuracy. This is demonstrated in both the L_2 and L_∞ norms. This gives confidence that there is no implementation bugs in the code for these flux calculators. For this supersonic flow without shocks, using the adaptive flux calculator is equivalent to selecting the AUSMDV flux calculator because there are no shocks in the flow. As such, the discretisation errors and order of accuracy are identical. The adaptive flux calculator case was computed as a confirmation that the shock detector is not erroneously signalling for the use of the EFM flux calculator.



(a) norms

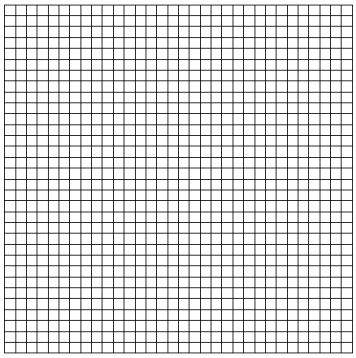


(b) observed order of accuracy

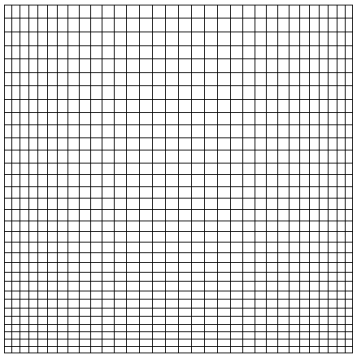
Figure 1: A comparison of various flux calculators used to compute the Euler version of the Method of Manufactured Solutions case.

Effect of grid types

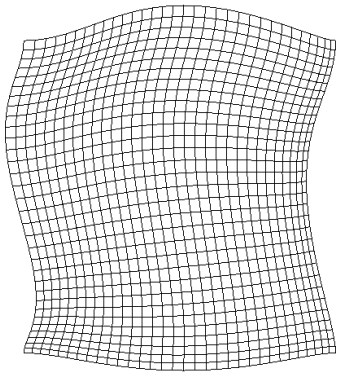
The next comparison examined the effect of nonuniform grids on the spatial order of accuracy. One of the main applications of `Eilmer` is in the simulation of hypersonic impulse facilities. The grids required to efficiently simulate these facilities typically have moderate changes in cell size: large aspect ratio cells in the long tube sections of the domain, and cells of shorter aspect ratio near features of interest such as sudden expansion regions into the test section. Therefore, the piecewise parabolic reconstructor was constructed to use information about the change in cell size so that moderate changes in cell size in the grid can be accommodated. The observed order of spatial accuracy was assessed on three grids using this Euler verification case. The three grids have been labelled 'regular', 'stretched' and 'distorted' in figure 2. The stretched grid has clustering applied such that the grid is stretched and compressed in both the x and y directions, but the cells are all orthogonal. For the distorted grid, the straight line boundaries have been replaced by 3rd order Bèzier curves. Note these curves do not conform to the original domain, however the manufactured solution is still appropriate to use because it is continuous beyond the extents of the domain, being a function of sines and cosines. For each grid, the AUSMDV flux calculator was used.



(a) regular grid



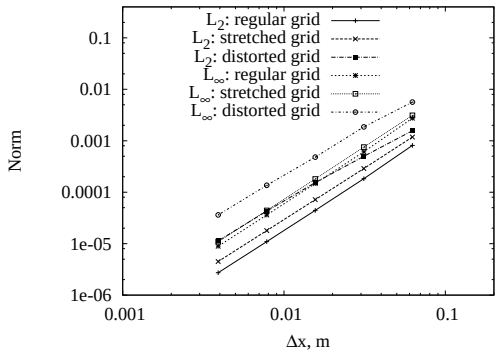
(b) stretched grid



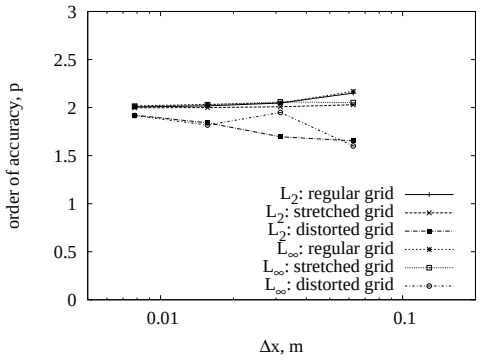
(c) distorted grid

Figure 2: Various grids used when computing the Method of Manufactured Solutions case

The behaviour of the discretisation error and the observed order of accuaracy for the various grid types is shown in figure 3. Note that the magnitude of discretisation error for the distorted grid is not directly comparable to the regular or stretched grids. This is because the distorted grid is solved on a different domain. The observed order of spatial accuracy on the stretched grid is 2. This is a pleasing result because the stretched grid is representative of the class of grid we use to simulate hypersonic impulse facilities. The order of accuracy for the distorted grid approaches 2 at higher grid resolutions. The increased order of spatial accuracy at higher levels of grid refinement is most likely due to the better alignment of the collection of cells used in the piecewise parabolic reconstructor.



(a) norms



(b) observed order of accuracy

Figure 3: A comparison of the various grid types and their effect on order of spatial accruacy.

Conclusion

In recent and continuing work, we have been performing verification and validation of our in-house flow code, `Eilmer`. In this paper, we presented the results from just one of these recent verification efforts: a manufactured solution for an inviscid supsersonic flow. The global error norms were assessed for five levels of grid refinement and were shown to converge with an observed order of spatial accuracy that matched the formal order of accuracy, which is 2nd order. This was demonstrated for each of the flux calculators implemented in the code. Furthermore, this verification case was also used to assess what effect the use of nonuniform grids had on the observed order of accuracy. It was shown that on stretched grids (but everywhere orthogonal) the observed order of spatial accuracy is 2, while the spatial accuracy on distorted grids approached 2 at higher resolutions. The results of this verification exercise are a useful addition towards demonstrating verification of `Eilmer` as a compressible flow solver.

References

[1] Maxima, a computer algebra system, version 5.24.0, http://maxima.sourceforge.net/, 2011.

[2] Gollan, R. J., Johnston, I. A., O'Flaherty, B. F. and Jacobs, P. A., Development of Casbar: a two-phase flow code for the interior ballistics problem, in *16th Australasian Fluid Mechanics Conference*, Gold Coast, Queensland, 2007.

[3] Jacobs, P. A., Single-block Navier-Stokes integrator., ICASE Interim Report 18, 1991.

[4] Jacobs, P. A. and Gollan, R. J., The Eilmer3 code: User guide and example book., Mechanical Engineering Report 2008/07, The University of Queensland, Brisbane, Australia, 2010.

[5] Jacobs, P. A., Gollan, R. J., Denman, A. J., O'Flaherty, B. T., Potter, D. F., Petrie-Repar, P. J. and Johnston, I. A., Eilmer's theory book: Basic models for gas dynamics and thermochemistry., Mechanical Engineering Report 2010/09, The University of Queensland, Brisbane, Australia, 2010.

[6] Liou, M. S. and Steffen, C. J., A new flux splitting scheme., *Journal of Computational Physics*, **107**, 1993, 23–39.

[7] Macrossan, M. N., The equilibrium flux method for the calculation of flows with non-equilibrium chemical reactions., *Journal of Computational Physics*, **80**, 1989, 204–231.

[8] Oran, E. and Boris, J., *Numerical Simulation of Reactive Flow*, Cambridge University Press, New York, USA, 2001, 2nd edition.

[9] Powers, J. M. and Aslam, T. D., Exact solution for multidimensional compressible reactive flow for verifying numerical algorithms, *AIAA Journal*, **44**, 2006, 337–344.

[10] Roache, P. J., Verification of codes and calculations, *AIAA Journal*, **36**, 1998, 696–702.

[11] Roache, P. J., Code verification by the Method of Manufactured Solutions, *Journal of Fluid Engineering*, **124**, 2002, 4–10.

[12] 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*, **44**, 2004, 599–620.

[13] Steinberg, S. and Roache, P. J., Symbolic manipulation and computational fluid dynamics, *Journal of Computational Physics*, **57**, 1985, 251–284.

[14] Ventura, C., Sauret, E., Jacobs, P. A., Petrie-Repar, P., Gollan, R. J. and van der Laan, P., Adaption and use of a compressible flow code for turbomachinery design, in *European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010*, Lisbon, Portugal, 2010.

[15] Wada, Y. and Liou, M.-S., An accurate and robust flux splitting scheme for shock and contact discontinuities, *SIAM Journal on Scientific Computing*, **18**, 1997, 633–657.