The Method of Generated Solutions for Numerical Verification of ICE Code

Amjidanutpan Ramanujam, Christopher Sikorski, Todd Harman*

UUCS-07-006

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

*Department of Mechanical Engineering
University of Utah
Salt Lake City, UT 84112 USA

Abstract

Method of Manufactured Solutions (MMS) is a widely used technique to verify convergence and possible coding errors in numerical algorithms. This method involves designing analytical solutions satisfying the governing equations that are solved by the numerical algorithm. The solutions investigated by MMS may not resemble the physical solutions. In order to overcome this limitation, we propose a new verification technique called Method of Generated Solutions (MGS). The MGS method designs its analytical solutions by approximating or interpolating discrete data from the solver or data measurements from physical experiments. This can be achieved using appropriate mathematical techniques such as spline interpolation or least-squares approximation. The analytical solutions obtained in that way will closely resemble the physical solutions. Hence, the verification results based on these analytical solutions will provide us with an accurate benchmark of the solver's capabilities. The proposed MGS method is implemented and utilized to verify ICE (Implicit, Continuous fluid, Eulerian), a semi-implicit finite volume solver, that simulates fluid phenomena.

Floating point analysis of the time-stepping summations in the ICE algorithm was also performed. The condition numbers of such summations are evaluated and possible use of a corrective summation algorithm was explored.

1 Introduction

Numerical verification is a procedure to demonstrate that the governing equations, as implemented in the code, are solved consistently [1]. Roache [2] defines code verification process as:

> *"The [code] author defines precisely what continuum partial differential equations and continuum boundary conditions are being solved, and convincingly demonstrates that they are solved correctly, i.e., usually with some order of accuracy, and always consistently, so that as some measure of discretization (e.g. the mesh increments)* $\Delta \to 0$*, the code produces a solution to the continuum equations; this is Verification."*

Verification of a numerical solver is performed to ensure that the solver satisfies the theoretical order of accuracy while solving the corresponding set of equations.

Validation is also a testing method for ensuring correctness of a solver. But verification differs from validation in the following respect, validation is a process to find how close the equations that we solve represent a real world model. Validations are typically performed by comparing the results obtained from the solver and a physical experiment. Whereas, verification ensures that the solver solves the designed model within the allowed error limits. Additionally, code verification is not concerned with asking if the best numerical method is being used to solve the problem [1]. Such optimality questions are the subject of computational complexity theory.

Verification is performed by comparing the numerical solutions from the solver against the exact solutions. Verification becomes harder for solvers dealing with the class of problems whose exact solutions are not readily available. A classic example of such a problem is given by the Navier-Stokes equations that need to be solved numerically in order to simulate fluid phenomena.

1.1 Previous Work and Limitations

Some of the widely used techniques to verify numerical solvers are explained below.

1.1.1 Method of Exact Solutions

Method of Exact Solutions(MES) [1] is a well known method for verifying numerical code. The exact solution to the set of equations is derived using mathematical methods such as integral transform techniques, separation of variables etc. The derived solution is a mathematical expression that provides us with the solution at all locations in space and time. Suppose one is solving a differential equation of the form,

$$Du = g \tag{1}$$

where D is the differential operator, u is the solution and g is a source term. In the MES method, one chooses a function g and then using methods from classical applied mathematics, inverts operator D to solve for u.

The major drawback of this MES procedure is that if the differential operator D is non-linear in nature, the exact inversion operator is unknown or extremely difficult to obtain. Additionally, if the exact solutions are obtained using laplace transforms, it is extremely difficult to implement them in a computer program. Finally, the exact solutions may contain singularities which are impossible to verify due to practical limitations on the number of grid points.

1.1.2 Method of Manufactured Solutions

Method of Manufactured Solutions (MMS) [3][4] is a widely used approach to verify numerical solvers. A manufactured solution is an exact solution to the set of PDE's that has been constructed by solving the problem backwards. For equation (1), the MMS method manufactures the u using arbitrary mathematical functions (say, trigonometric) and then the operator D is applied to find g. This is a much simpler procedure than MES since it does not involve inversion of any operators (non-linear or otherwise).

One of the major limitations of this procedure is that the solutions picked by the MMS method are arbitrary in nature. These solutions may not represent the true nature of the physical solutions that are being simulated. The effect of this drawback will be more pronounced if the solver utilizes a *gradient limiter* [1] to suppress the unnatural oscillations and

[1] Higher order methods are prone to introduce oscillations at places where the gradient of a quantity changes rapidly. In order to eliminate such unnatural oscillation, gradient limiters will try to reduce the order of the solver to keep the values bounded. This can be implemented by investigating the cell centered and the face-centered values in the surrounding cells. The gradient limiter scheme implemented in ICE is prescribed by Van Leer[5] method.

overshoots in the solution.

1.2 Research Contributions

To overcome the above limitation, we propose a new verification method called Method of Generated Solutions (MGS) [6]. The MGS method is designed to verify computational performance of differential and/or integral equation solvers on exact (analytic) solutions that resemble physical solutions. In order to evaluate our technique, the ICE algorithm is verified using our method and the order of accuracy of ICE is computed for various problems. The verification results show that the ICE solver's discretization errors are problem specific.

2 Method of Generated Solutions

Physical phenomena such as fluid flow are simulated numerically by solving a system of partial differential equations using some discrete approximation method.

Method of Generated Solutions (MGS) is a technique proposed in [6] to verify such numerical solvers.

Let us consider equation (1)

$$Du = g$$

The steps in verifying a solver using MGS method are as follows:

1. An approximate solution u_1 to the problem described in (1) is obtained either by a computational algorithm or by measurements of physical data.

2. The data from u_1 is approximated or interpolated using an appropriate technique such as least-squares approximation or spline interpolation. This results in an analytical expression that provides us with values at all spatial and temporal locations.

3. The differential operator D is applied to the function resulting from step 2. The exact analytical differentiation can be performed on u_1 to get the corresponding forcing functions g_1.

4. This process yields a new problem

$$Du_1 = g_1 \tag{2}$$

whose exact solution u_1 and source terms g_1 are available.

5. The new problem (2) is now solved by the numerical solver using the forcing functions generated in step 3.

6. The resulting solution u_2 is then compared with the exact solution u_1 to quantify the errors introduced by the solver.

This verification process is explained as a flow diagram shown in Fig. 1.

3 Problem Description

MGS method has the ability to verify numerical solvers that have problem dependent convergence rate. The ICE (Implicit, Continuous fluid, Eulerian) algorithm [7] [8] developed at University of Utah by the C-SAFE (Center for Simulation of Accidental Fires and Explosion) [9] research group is one such solver and hence, chosen to verify using MGS. ICE algorithm developed by C-SAFE is a cell-centered, finite volume, multi-material version of a numerical algorithm developed by Kashiwa *et. al.* [10].

The ICE algorithm is utilized by C-SAFE to simulate explosions, fires and other fluid phenomena. One of the features of ICE is the use of gradient limiter to suppress unnatural oscillations introduced by the higher-order numerical methods. The effects of gradient limiter depend upon the profile of the problem that is being solved, which in turn affects the order of accuracy of the solving process. This property makes ICE an appropriate candidate for verification using MGS.

The ICE algorithm has several modules to solve a multitude of problems. Since an exhaustive verification of all possible problems is impractical, we decided to perform a hierarchical way of verifying individual modules. As a first step towards this approach, the **Advect and Advance in Time(AAT)** module in ICE was chosen. This module computes the material derivative of a physical quantity such as energy, pressure, mass, momentum and passive scalar. This operator is chosen for our verification purpose for the following reasons:

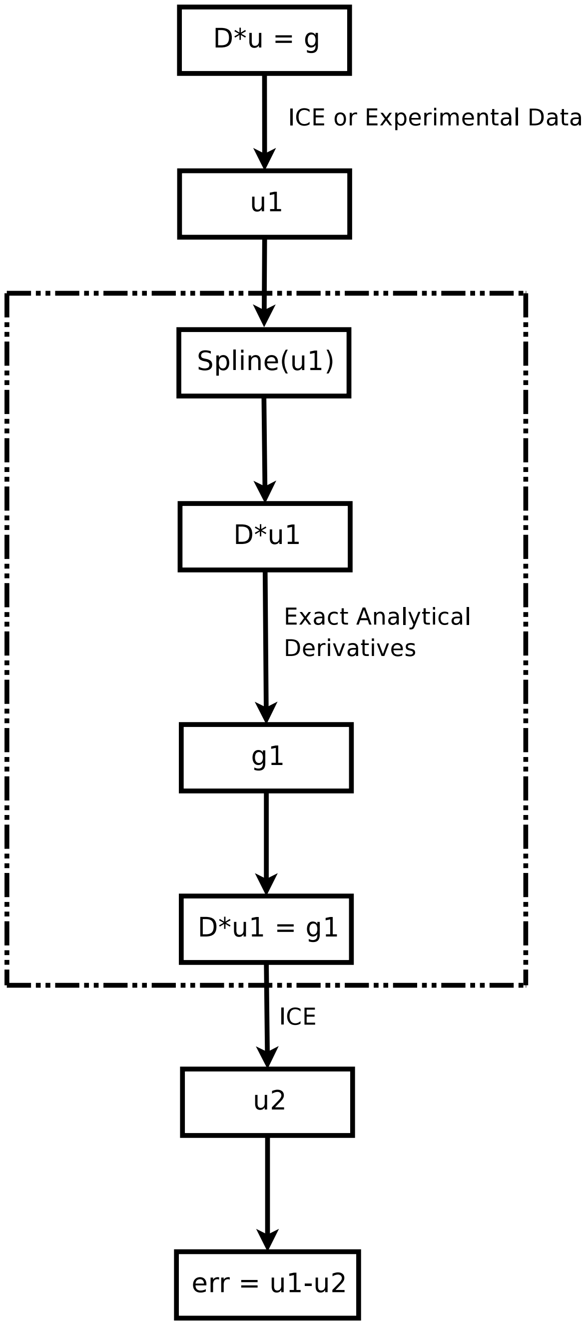

Figure 1: Verification Process Using MGS Method

1. The advection operator is an integral part of ICE as it is invoked five times per timestep during full-scale fluid simulations. Hence, the accuracy of this operator is critical.
2. The advection operator is relatively simple to verify, since it solves only one governing equation.

In order to isolate the AAT module, the advection of a passive scalar is employed as our verification experiment. The profile of the passive scalar can be defined using an analytical function such as a cubic polynomial, trigonometric function and a bell-shaped exponential function. The governing equation for our experiment solved by the AAT module is as follows:

$$\frac{d}{dt}(PS(x,t)) + U_x \frac{d}{dx}(PS(x,t)) = g(x,t) \tag{3}$$

where,

$PS(x,t)$	–	Passive Scalar
U_x	–	Constant Velocity in the x-direction
$g(x,t)$	–	Source terms

By comparing the governing equation (3) with equation (1), we arrive at the differential operator as follows:

$$D = \frac{d}{dt} + U_x \frac{d}{dx} \tag{4}$$

Since D is of degree one, the analytical solution resulting from the approximation/interpolation needs to be at least C^1 continuous. To satisfy this constraint, a cubic spline interpolation is chosen to generate the analytical functions. The results from the interpolation are piecewise cubic polynomials, that are C^2 continuous.

4 Implementation

To verify the AAT module, the MGS method is implemented in three modules:

1. A module to extract suitable information for the MGS method from the input data – *Input Module*

2. A module to interpolate the data fed from the input module and compute the MGS source terms – *Interpolation Module*

3. A module to replace ICE source terms with the MGS source terms in order to compute numerical solution – *ICE Module*.

4.1 Input Module

Either an approximate solution from a numerical solver or measurements from a physical experiment can be used by the MGS method. Highly resolved data from physical experiments are a practical rarity, hence this remains a theoratical possibility. For our experiments, we chose the approximate solution from ICE to build our analytical solution. This initial data from ICE is available in the form of a binary file. This binary file contains values of various parameters (pressure, velocity, etc) that are computed by ICE. The values for required parameters can be read from these files using a utility called *lineextract*. The extracted values are used as input to the interpolation module for further computation.

4.2 Interpolation Module

The data from the input module is interpolated using natural cubic splines to design the exact solutions. This module computes the cubic splines and its gradients at the specified spatial/temporal locations. These gradients are then used to compute the source terms (rhs) g(x,t) in equation (3).

4.3 ICE Module

This module replaces the source terms computed by ICE with the source terms from the MGS module. Using these new source terms, the solver proceeds to solve the problem numerically. The results from these computations are then compared with the original solution.

The experiments conducted using this framework are explained in the subsequent section.

5 Numerical Experiments

The experiments conducted for this verification procedure are designed to quantify the *discretization error* of the AAT module and subsequently verify the observed *order-of-accuracy* of the spatial derivative.

The governing equations can be discretized by dividing the domain of the problem into finite cells. The approximate solution, that satisfies the discretized equations, is not the same as the exact solution that satisfies the mathematical continuum equations. The difference between the two is called the **discretization error** [1]. The discretization error is evaluated by computing the normalized L-2 norm or the L-∞ norm of the difference between the approximate solution computed by ICE and the exact solution designed by the MGS method. The normalized global error using L2 norm for a uniform grid of size N cells is given by,

$$L_2 = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(u_n - U_n)^2} \tag{5}$$

The L-∞ norm is defined by the following equation

$$L_\infty = \max(u_n - U_n) \tag{6}$$

where,

u_n	$-$	Exact Solution evaluated at x_n, y_n, z_n
U_n	$-$	Approximate solution of the discretized equation

The discretization methods are consistent if the error goes to zero as the representative cell size h decreases to zero. The rate at which the error decreases to zero is called the **order-of-accuracy**[1]. By running the experiments for different mesh sizes h and computing the global error for each mesh we can estimate the observed order-of-accuracy.

Setup: The goal of our experiments is to verify the order of accuracy of the spatial derivative in the AAT module. The theoretical order of accuracy of the AAT module is second order in space and first order in time. In order to verify the second order accuracy of the spatial derivative, we halved the grid cell size to check if the errors reduce by a factor of 4. For this purpose, the errors introduced by the temporal derivative must be separated from the spatial errors. This can be achieved by reducing the time step-size Δt to be smaller than the square of the grid cell size Δx^2. The experiment used to verify the AAT module is the advection of a passive scalar in a computational domain with constant velocity. Since the performance of the AAT module is impacted by the gradient-limiters in ICE, the order-of-accuracy for the spatial derivative may not be of second order. Hence, the mesh refinement is performed on various passive scalar profiles to study the order-of-accuracy for different problems. For our verification experiments, the profile of the passive scalar is chosen to represent some of the common forms of analytical functions such as trigonometric, polynomial and exponential. The following parameters are common for all three types of profiles:

- Computational Domain is one dimensional in X-direction from -0.5 to 0.5 meters.
- Advection rate of passive scalar is 10m/s in X-direction
- Time step size $\Delta t = 10^{-6}$
- Number of time-steps in experiment = 100
- The input data (from experiment or Solver) to the MGS module is from the highest resolution run (800). The source terms for the lower resolutions are computed from the same exact solution since it is defined by the analytical function designed using the MGS method.
- The experiments are repeated for the following grid resolutions 100, 200, 400, and 800 corresponding to Δx values $1 \times 10^{-2},\ 0.5 \times 10^{-2},\ 0.25 \times 10^{-2},\ 0.125 \times 10^{-2},\ 0.625 \times 10^{-3}$ respectively.

The source terms (g(x,t)) are computed using the MGS method from two different input sources, namely, the exact solution and the ICE solution. The exact solution is the data generated using the analytical function as described in equations (7) (8) and (9). The ICE solution is the data from the discretized solution computed by ICE for a resolution of 800 grid cells in the computational domain.

The plot of the gradient limiter shown in conjunction with the profile are the points in the computational domain where the gradient-limiter has detected high gradients. The second-order solver will be made first-order at those cells to suppress the over shoot.

5.1 Sinusoidal

The trigonometric function chosen to initialize the sinusoidal profile between -0.1 and +0.1 is described by the following equation,

$$f(x) = \sin\left[2\pi\nu \times \left(\frac{x + 0.1}{0.2}\right)\right] \tag{7}$$

where,

$$\nu = \text{Frequency}(1.0)$$

The profile of the sinusoidal function initialized at time $t = 0$ is shown in figure 2.

The experiments are repeated for different cell sizes as explained earlier. The results from these experiments are shown in fig(3) and fig(4).

5.2 Cubic Polynomial

Two piece-wise cubic polynomials are juxtaposed to create a pseudo-bell curve shape. The cubic polynomial is defined between -0.1 and +0.1 and it is defined by the following equation.

$$f(x) = \begin{cases} -\left(\frac{4}{3}\right) \times d^3 + d^2 & d <= 0.5 \\ -\left(\frac{4}{3}\right) \times (1.0 - d)^3 + (1.0 - d)^2 & d > 0.5 \end{cases} \tag{8}$$

where,

$$d = \left(\frac{x + 0.1}{0.2}\right)$$

The profile of the cubic polynomial function initialized at time $t = 0$ is shown in figure 5.

The results from the mesh refinement experiment are shown in fig(6) and fig(7).

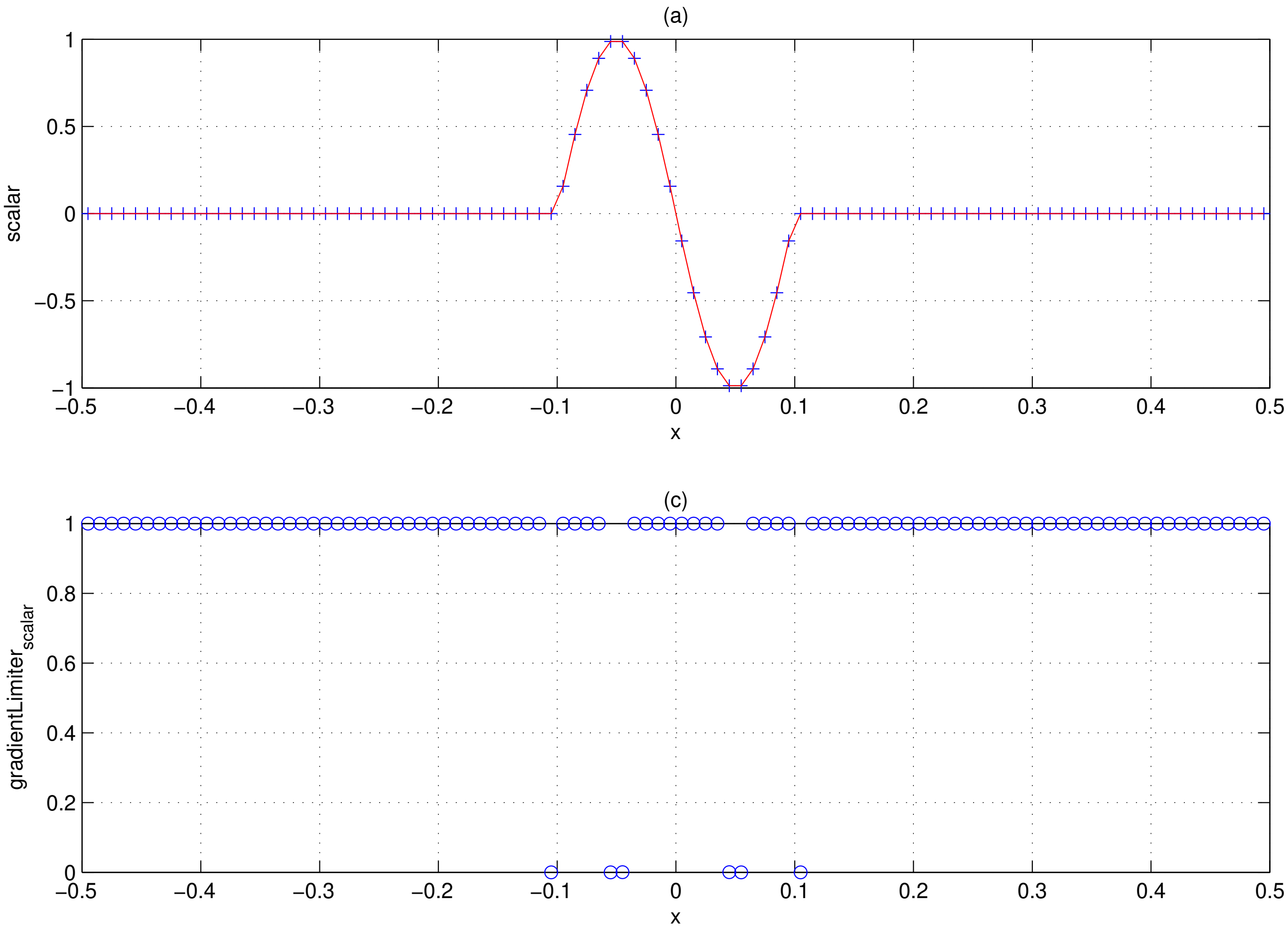

Figure 2: Sinusoidal Profile
(a) The profile of the sine function when initialized at t = 0
(c) The behavior of gradient limiter

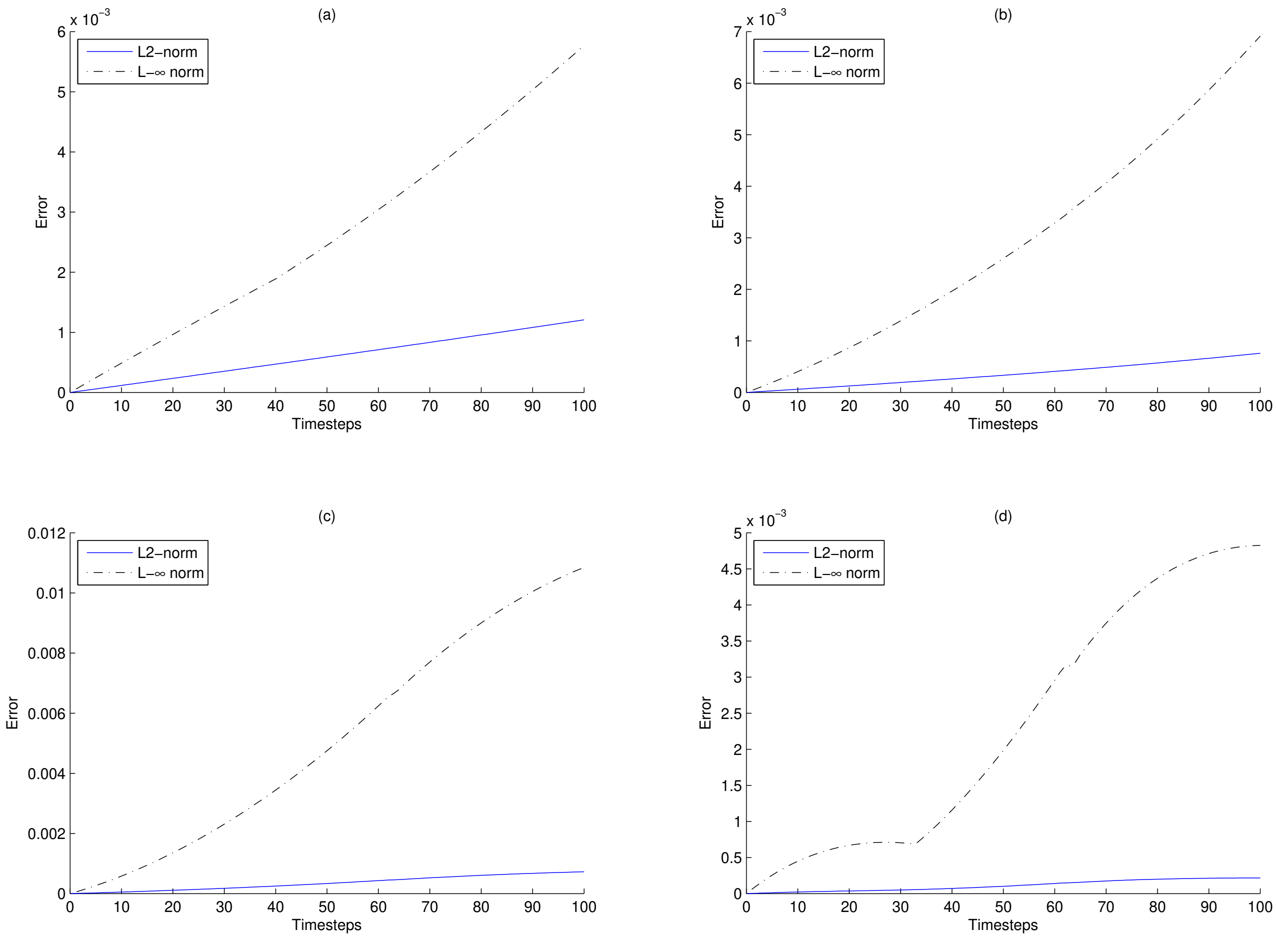

Figure 3: Discretization error as a function of time for sinusoidal profile using source terms from analytical function (a) Resolution = 100 $\Delta x = 0.01m$
(b) Resolution = 200 $\Delta x = 0.005m$
(c) Resolution = 400 $\Delta x = 0.0025m$
(d) Resolution = 800 x = 0.00125m

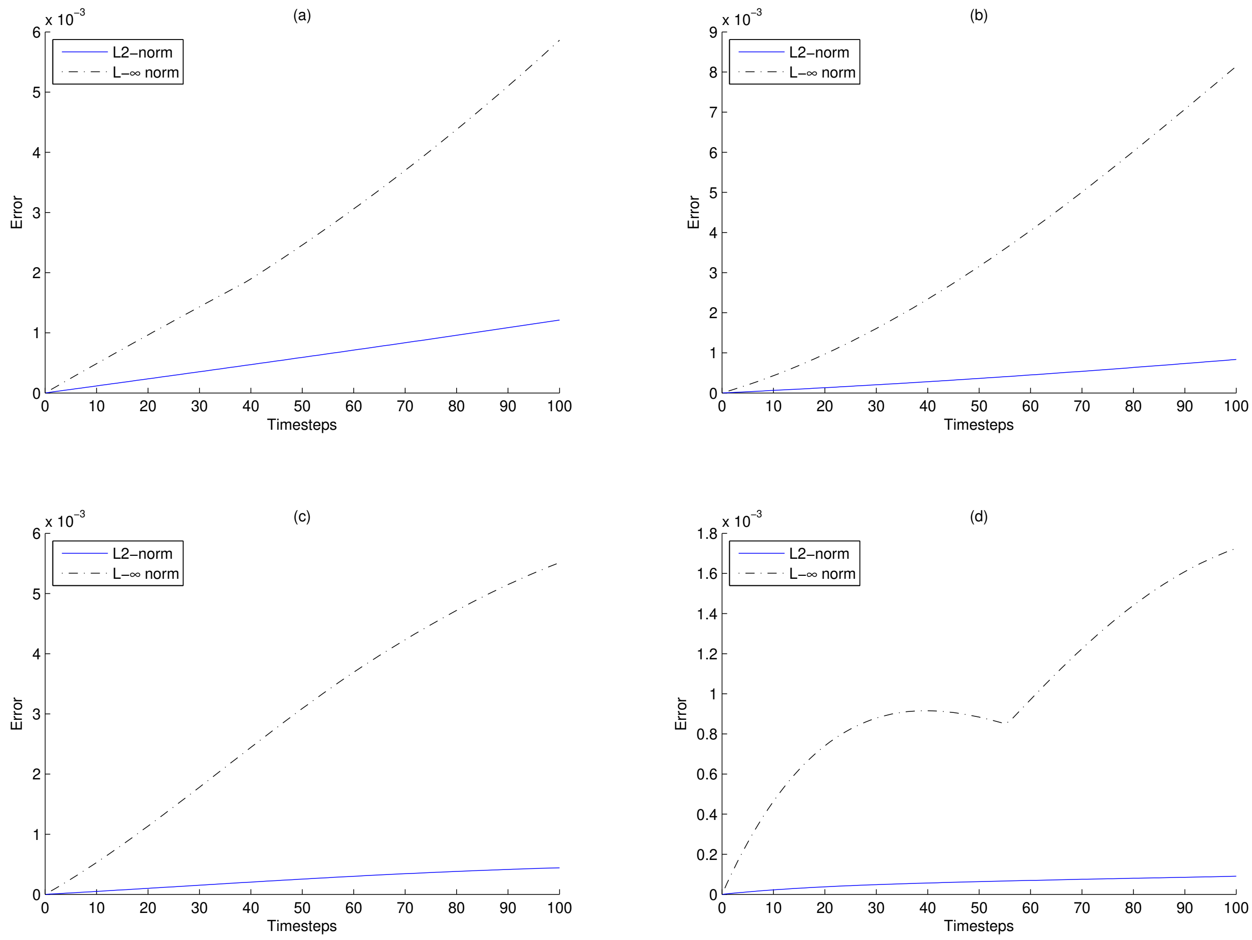

Figure 4: Discretization error as a function of time for sinusoidal profile using source terms from solver's solution
(a) Resolution = 100 $\Delta x = 0.01m$
(b) Resolution = 200 $\Delta x = 0.005m$
(c) Resolution = 400 $\Delta x = 0.0025m$
(d) Resolution = 800 $\Delta x = 0.00125m$

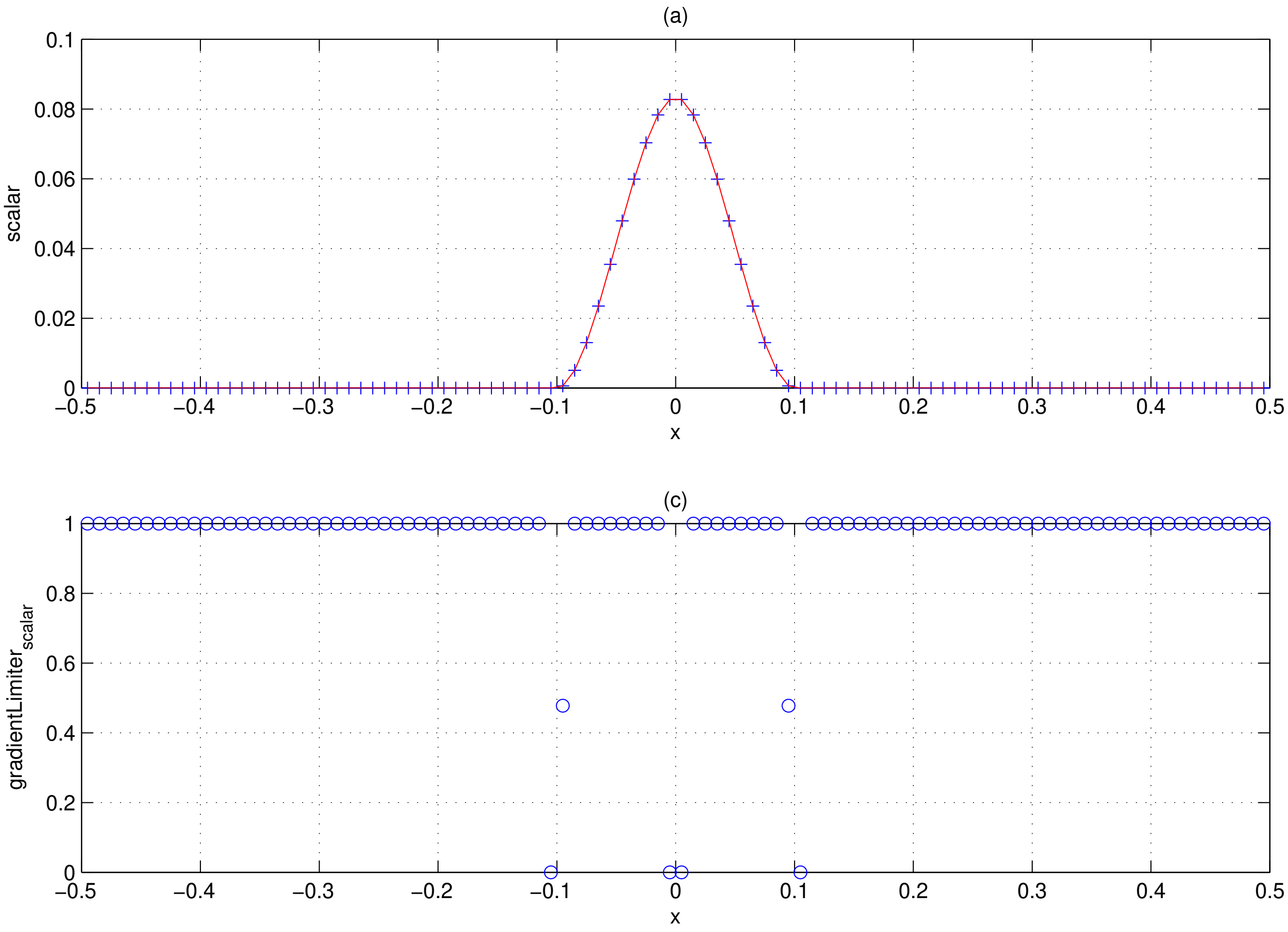

Figure 5: Cubic Profile
(a) The profile of the cubic polynomial when initialized at t = 0
(c) The behavior of gradient limiter

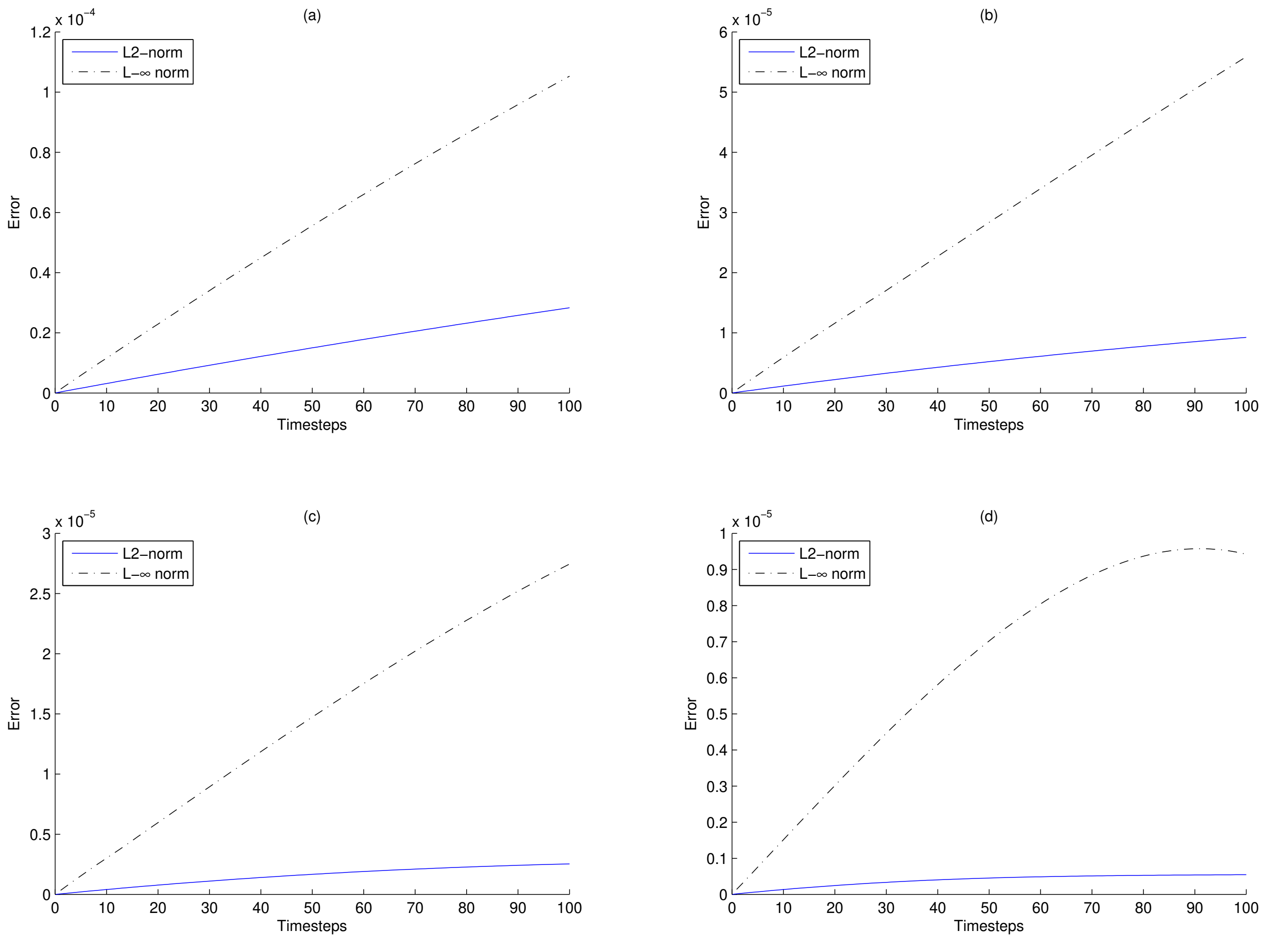

Figure 6: Discretization error as a function of time for cubic polynomial profile using source terms from analytical function
(a) Resolution = 100 $\Delta x = 0.01m$
(b) Resolution = 200 $\Delta x = 0.005m$
(c) Resolution = 400 $\Delta x = 0.0025m$
(d) Resolution = 800 $\Delta x = 0.00125m$

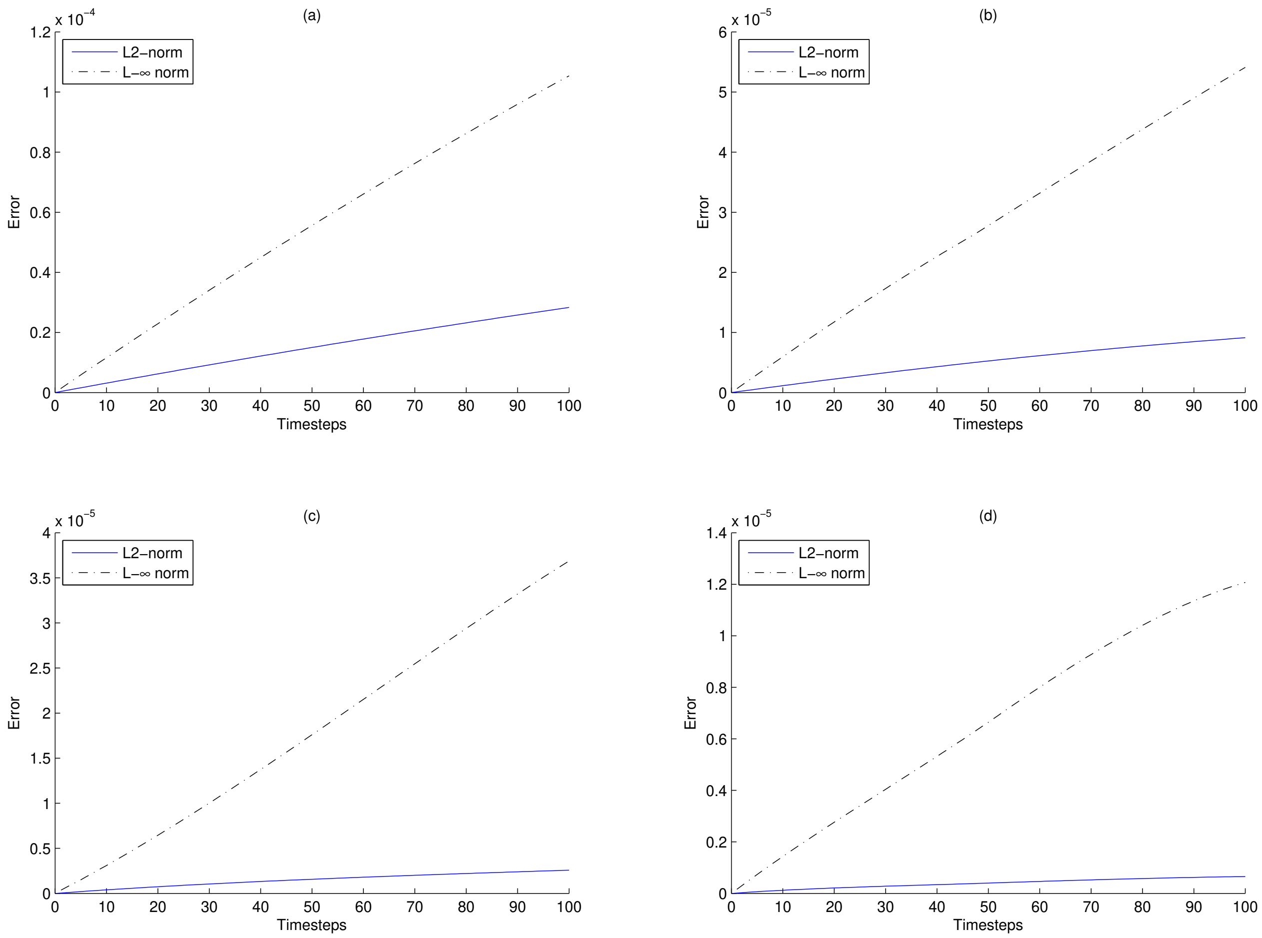

Figure 7: Discretization error as a function of time for cubic polynomial profile using source terms from solver's solution
(a) Resolution = 100 $\Delta x = 0.01m$
(b) Resolution = 200 $\Delta x = 0.005m$
(c) Resolution = 400 $\Delta x = 0.0025m$
(d) Resolution = 800 $\Delta x = 0.00125m$

5.3 Exponential

The exponential function chosen for describing the third profile is explained below.

$$f(x) = 4.5469432 \times \exp\left(\frac{-1.0}{d(1-d)}\right) \tag{9}$$

where,

$$d = \left(\frac{x+0.1}{0.2}\right)$$

The profile of the exponential function initialized at time $t = 0$ is shown in figure 8.

The results from the mesh refinement experiment are shown in fig(9) and fig(10).

5.4 Analysis

The results from the experiments clearly indicate that the order-of-accuracy of the AAT module depends upon the problem that is being solved. Using a finer mesh to solve the problem results in a better quality of the solution. This can be verified by observing the decrease in global error for decrease in Δx.

The spatial order-of-accuracy that is evaluated using the following formula:

$$p \approx \log\left(\frac{E_{grid1}}{E_{grid2}}\right) / \log(r) \tag{10}$$

where,

$$\begin{aligned} p &- \text{Observed order of accuracy} \\ E_{grid1} &- \text{The global discretized error from resolution 1} \\ E_{grid2} &- \text{The global discretized error from resolution 2} \\ r &- \text{refinement ration between grid size 1 and grid size 2} \end{aligned} \tag{11}$$

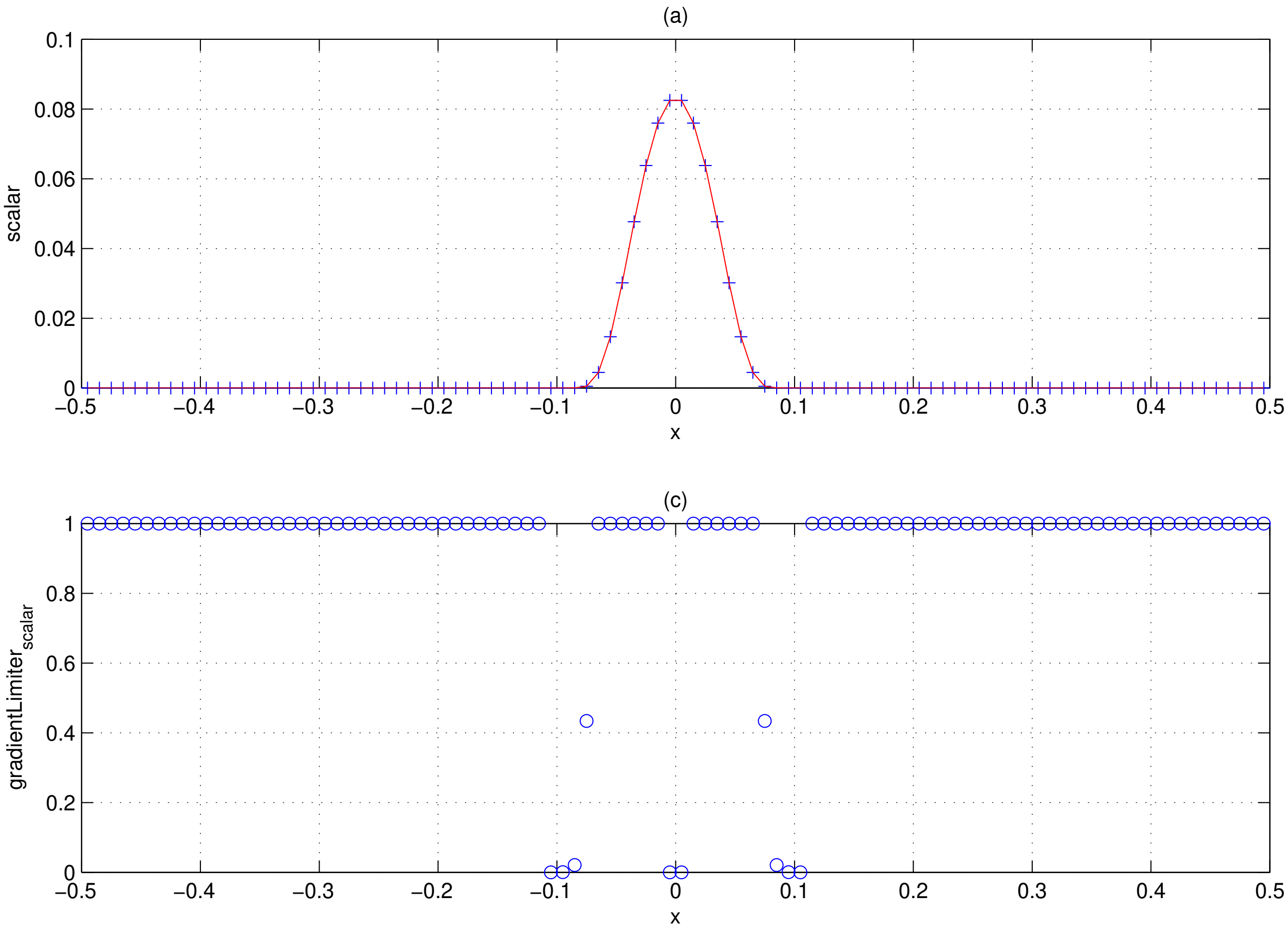

Figure 8: Exponential Profile
(a) The profile of the exponential function when initialized at t = 0
(c) The behavior of gradient limiter

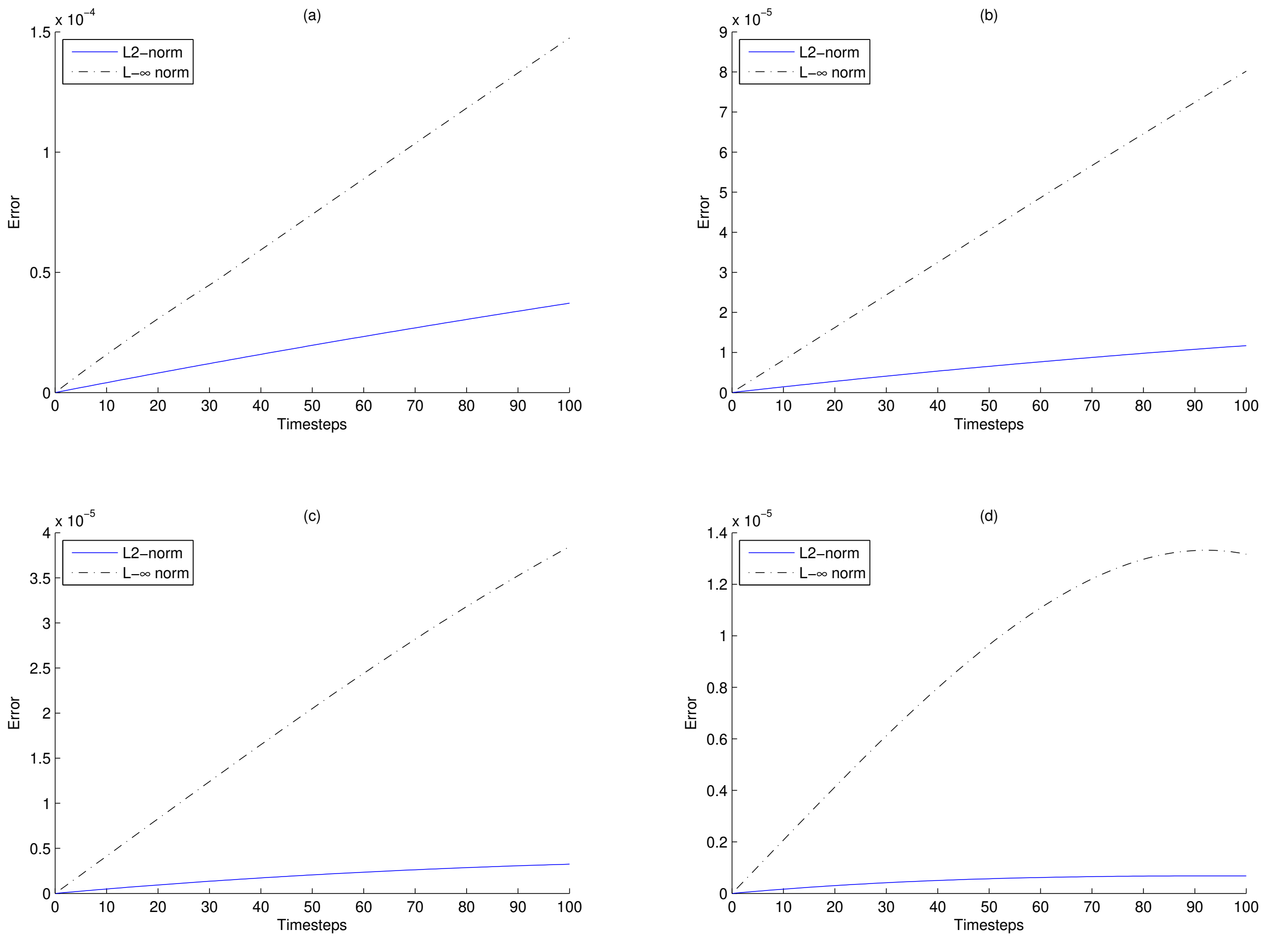

Figure 9: Discretization error as a function of time for exponential profile using source terms from analytical function
(a) Resolution = 100 $\Delta x = 0.01m$
(b) Resolution = 200 $\Delta x = 0.005m$
(c) Resolution = 400 $\Delta x = 0.0025m$
(d) Resolution = 800 $\Delta x = 0.00125m$

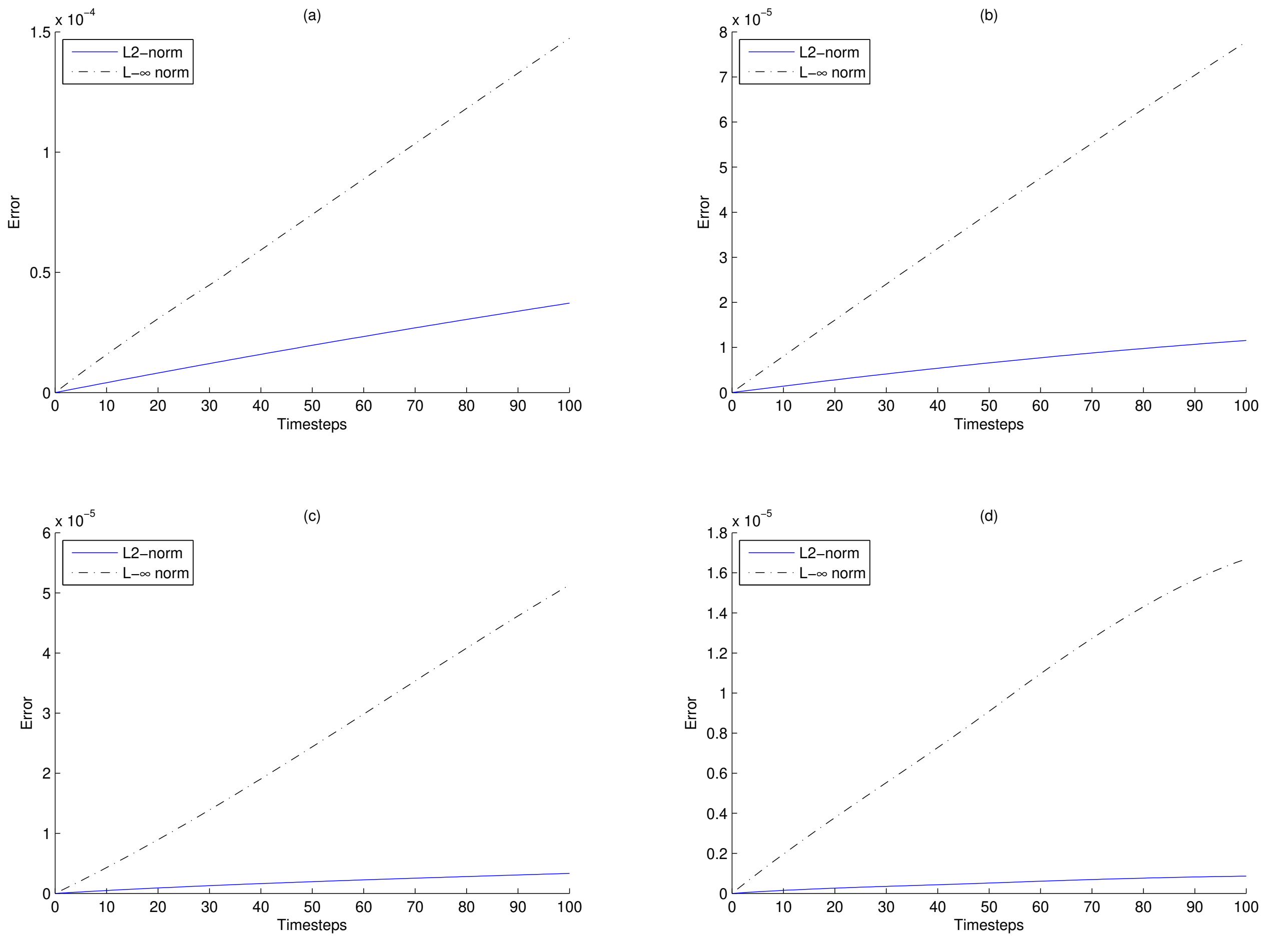

Figure 10: Discretization error as a function of time for exponential profile using source terms from solver's solution
(a) Resolution = 100 $\Delta x = 0.01m$
(b) Resolution = 200 $\Delta x = 0.005m$
(c) Resolution = 400 $\Delta x = 0.0025m$
(d) Resolution = 800 $\Delta x = 0.00125m$

The discretized errors (E_{grid1} and E_{grid2}) are computed using the formula given in equation 5. The order-of-accuracy obtained from the consequtive resolution pairs are averaged to obtain the following.

Table 1: Order-of-Accuracy of the spatial derivative in the AAT module

Profile	Source Type	Order-of-Accuracy
Sinusoidal	Analytical Sources	1.3038
	ICE Sources	1.245
Cubic Polynomial	Analytical Sources	1.895
	ICE Sources	1.808
Exponential	Analytical Sources	1.919
	ICE Sources	1.807

The order of the AAT module is close to second-order accurate for smooth profiles with continuous derivatives. The poor performance of the module while solving sinusoidal profile is because the derivative of the profile is not C^1 continuous. Also the sinusoidal profile invokes the gradient limiter more frequently than the cubic polynomial and the exponential profile.

6 Conclusion

The Method of Generated Solutions was developed to evaluate the discretization errors in ICE. The MGS method designs analytical solutions by interpolating numerical solutions from the solver or data measurements from physical experiments. Since the solutions produced by the MGS method originate from an actual problem the verification results are more representative than those obtained by the MMS method. The MGS method was implemented to verify the ICE algorithm and AAT module in the ICE solver was verified using this method. The order of accuracy of the module was evaluated for different problems using our method.

This method can be used to verify individual modules in ICE. A future research requires implementation for 2-d and 3-d domains. The current implementation uses natural cubic-spline interpolation for generating exact solutions. This can be replaced by Hermite interpolation, Least squares approximation or Sinc function approximations to represent the nature of the problem.

Appendix

A Verification of round-off errors in ICE

The numerical sensitivity of the time-stepping summations in the ICE algorithm are verified in this section. A corrective-summation algorithm was implemented inside ICE to study the usefulness of such a technique.

The experiment chosen for the verification is a 2-d Helium Plume simulation. The problem can be explained as the Helium gas escaping from a nozzle at the center of the bottom wall of a container. The computational grid chosen for this experiment was a 6m $\times$ 6m container which is sub-divided into a 100×100 grid. The computational domain is further sub-divided into 16 patches to solve them in 8 parallel nodes (2 processors per node). Hence, each patch contains 625 grid points.

The three primary variables mass, specific volume and internal-energy computed by ICE using the time-stepping summation were chosen for the condition-number analysis. The condition number of the summation shown in equation (12) were computed using the formula (13).

$$\text{Sum (S)} = \sum_{i=0}^{N} x_i \tag{12}$$

$$\text{Condition Number} = \frac{\sum_{i=1}^{N} \mid x_i \mid}{\left| \sum_{i=0}^{N} x_i \right|} \tag{13}$$

The value N in computing the condition number denotes the total number of timesteps for which the experiment is executed. For this verification experiment the tests were run for $\approx 45,000$ timesteps.

The upper bound of the Relative Error(RE) in the summation of the series can be expressed using the condition number as follows,

$$\text{Relative Error (RE)} \leqslant \epsilon \times N \times C \tag{14}$$

where,

$$\begin{aligned}
\varepsilon &- \text{Machine epsilon } (\approx 10^{-19}) \text{ for double precision computation}\\
C &- \text{Condition Number from equation (13)}\\
N &- \text{Number of timesteps } (\approx 45,000)
\end{aligned}$$

A.1 Corrective Summation Algorithm

In order to get rid of the time-step dependence in the relative error computation we can use a corrective summation algorithm.

Comparison of the pseudocode for the regular summation and the corrective summation algorithm:

<table>
<tr><th>Summation</th><th>Corrective Summation</th></tr>
<tr><td><pre>for i = 1 to n
   S = S + x(i)
end</pre></td><td><pre>U = E = 0
for i = 1 to n
   V = x(i)
   S = U + V
   E = U - S + V + E
   U = S
end
S = S+E</pre></td></tr>
</table>

This algorithm is implemented in the ICE code for the summation of mass, internal energy and specific volume computations. The relative error for the corrective summation is given by,

$$\text{Corrected Relative Error(CRE)} \leq \varepsilon \times C$$

This improvement in accuracy is achieved at the cost of additional computation. The distribution of the local min and max of condition numbers in the computational domain, is shown in figures (12)(13) and (14). The min/max of the condition numbers are computed

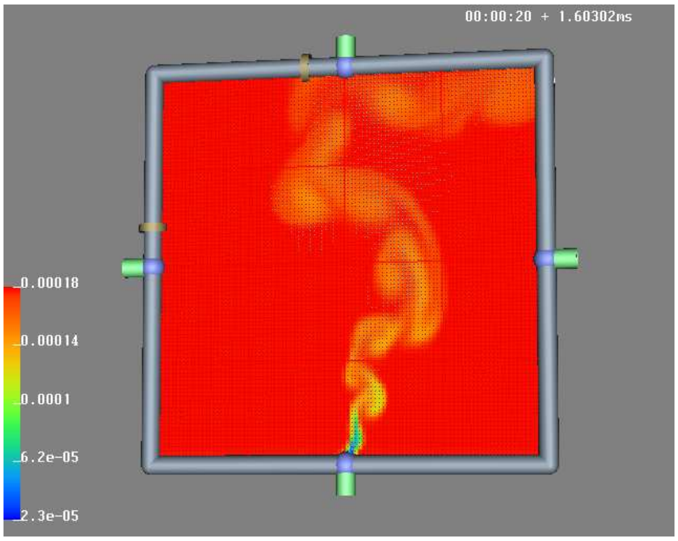

Figure 11: Simulation Snapshot: Computational Grid of 100x100 cells on a 6m x 6m computational domain

locally for each patch in the domain. It is observed that the condition numbers are maximum where there is more physical activity (near the nozzle).

The maximal condition number calculated using the equation (13) and the maximal global correction introduced by the corrective summation algorithm for the three variables are given below.

	O(Global Correction)	O(Condition Number)
Mass	10^{-18}	10^{3}
Sp. Vol.	10^{-18}	10^{3}
Int. Energy	10^{-16}	10^{3}

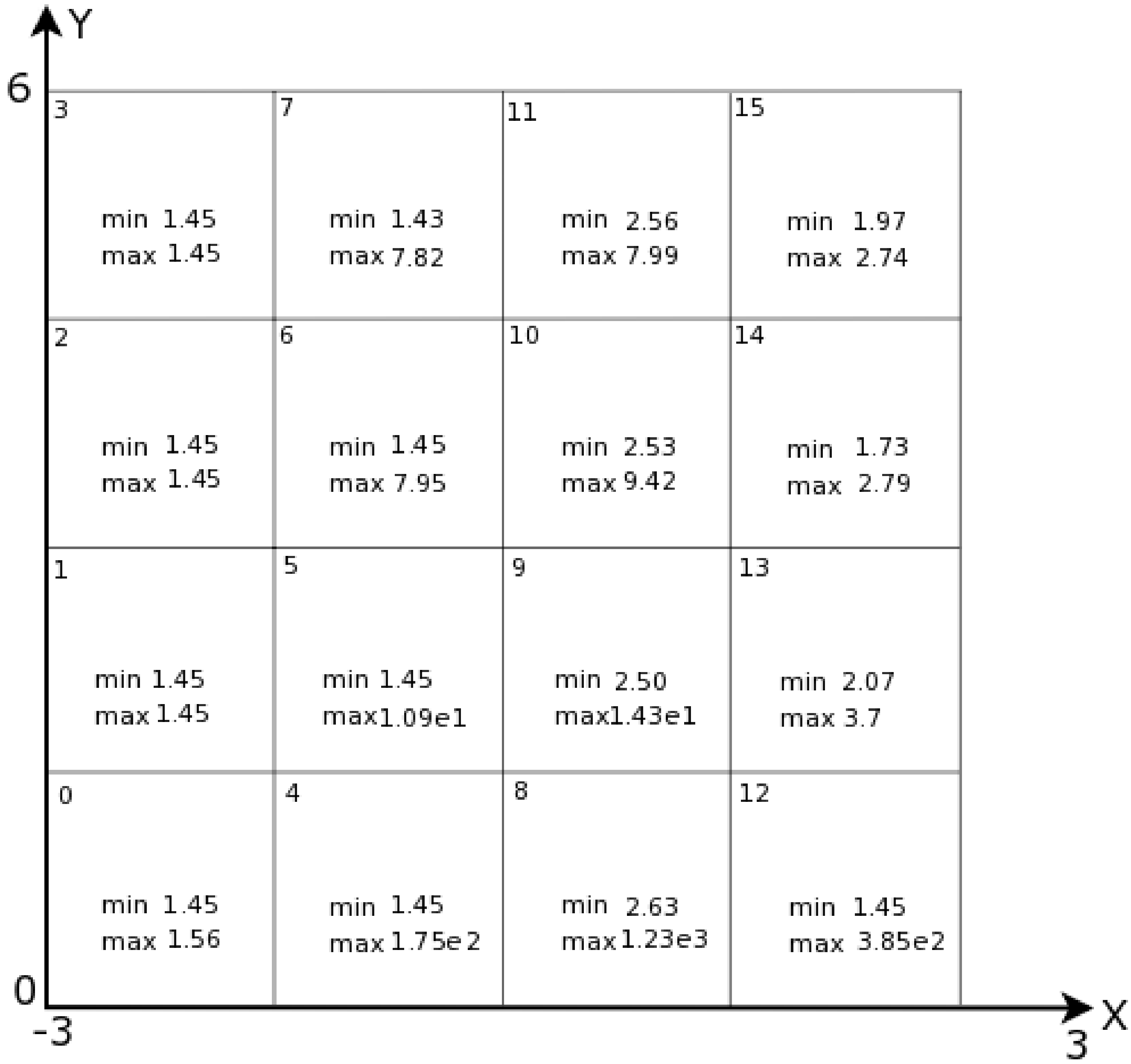

Figure 12: Mass: Computational Grid of 100x100 cells on a 6m x 6m computational domain

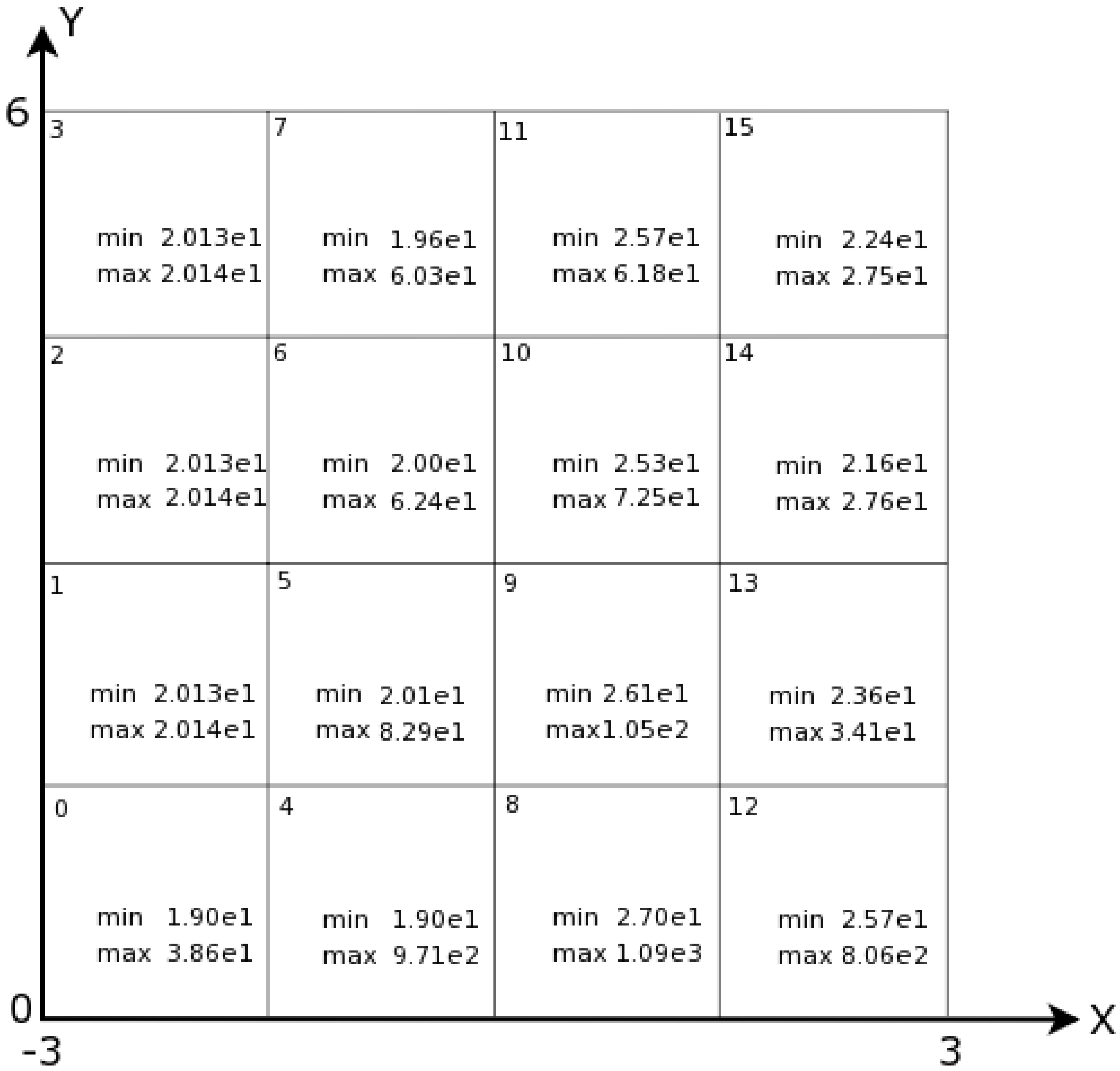

Figure 13: Internal Energy: Computational Grid of 100x100 cells on a 6m x 6m computational domain

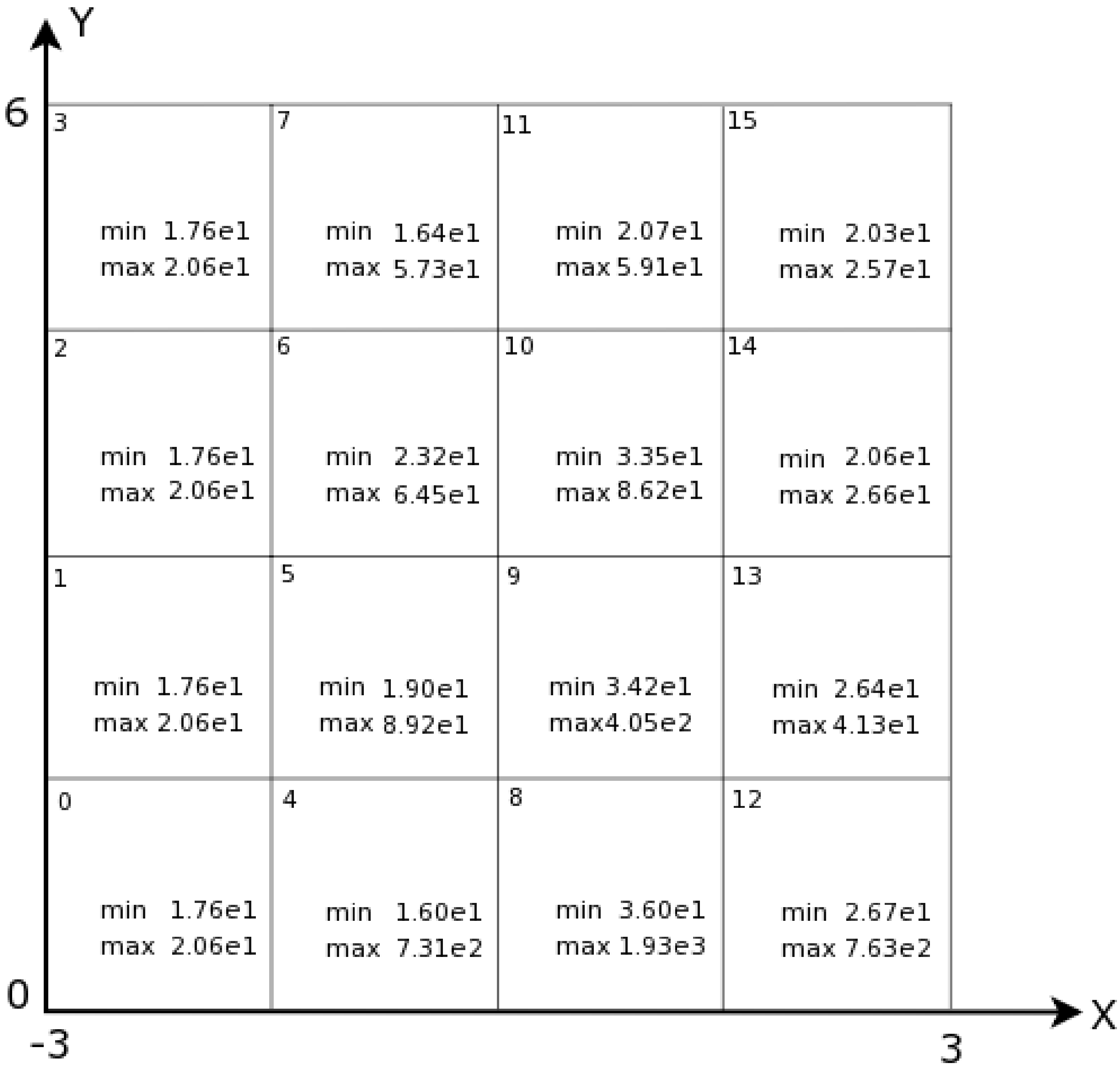

Figure 14: Specific Volume: Computational Grid of 100x100 cells on a 6m x 6m computational domain

A.2 Conclusion

The corrective summation algorithm was not adopted as our choice for calculations in the algorithm for two reasons:

- The summations analyzed in these experiments are not ill-conditioned
- The global error corrections included by the corrective summation algorithm are insignificant

References

[1] K. Salari and P. Knupp, “Code Verification by the Method of Manufactured Solutions”, Technical Report SAND2000 - 1444, Sandia National Laboratory, 2000.

[2] P. J. Roache, *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, 1998.

[3] T. M. Shih, “A Procedure to Debug Computer Programs”, *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 1027–1037, 1985.

[4] W. L. Oberkampf and F. G. Blottner, “Issues in Computational Fluid Dynamics: Code Verification and Validation”, *AIAA Journal*, vol. 36, pp. 687–695, 1998.

[5] W. B. VanderHeyden and B. A. Kashiwa, “Compatible fluxes for van Leer advection”, *J. Comput. Phys.*, vol. 146, pp. 1–28, 1998.

[6] R. Amjidanutpan, K. Sikorski, and T. Harman, “Method of Generated Solutions as a Verification Tool for Numerical Code”, Work in Progress.

[7] J. E. Guilkey, Harman T., Xia A., Kashiwa B., and McMurtry P., “An Eulerian-Lagrangian approach for large deformation fluid structure interaction problems, Part 1: algorithm development”, *ADVANCES IN FLUID MECHANICS*, vol. 36, pp. 143–156, 2003.

[8] S. G. Parker, J. Guilkey, and T. Harman, “A Component-based Parallel Infrastructure for the Simulation of Fluid-tructure Interaction”, *Engineering with Computers*, vol. 22, pp. 277–292, 2006.

[9] “C-SAFE Research Group”, http://www.csafe.utah.edu.

[10] B. A. Kashiwa and Rauenzahn, “A Cell-centered ICE Method of Multiphase Flow Simulations”, Technical Report LA-UR-93-3922, Los Alamos National Laboratory, 1994.