Method of Generated Solutions as a Numerical Verification Tool for Ice Code

Polina Milyavskaya, Christopher Sikorski, Todd Harman*

UUCS-08-07

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 is a well-known method used to verify numerical algorithms. It is used to estimate convergence and order of accuracy of the algorithms. The method involves design of analytical solutions to the set of equations solved by the algorithm and generation of the forcing function, which becomes the input to the solver. The disadvantage of this method is that the solutions it investigates may not reflect physical solutions. Method of Generated Solutions was designed to overcome this limitation. Method of Generated Solutions interpolates or approximates experimental data or data from a solver in order to design analytical solution. These solutions closely resemble physical solutions, which leads to a more accurate baseline for testing and verification of a numerical solver. The method was used to verify ICE (Implicit, Continuous fluid, Eulerian), a semi-implicit finite volume solver, that simulates fluid phenomena. This paper describes the results of numerical experiments, which demonstrate the effectiveness of the method.

1. Introduction

Numerical algorithms are widely used in various fields for different purposes including development and testing of the models resembling behavior of real life systems. Code verification and validation are methods used to assess accuracy and build confidence in the algorithms. [1]

There are many definitions of the terms verification and validation. Charles Hirsch defines them as:

> *Verification is "the process of determining that a model implementation accurately represents the underlying mathematical model and its solutions."*
>
> *Validation is "the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model."* [2]

In other words, validation and verification are two different steps in code development and assessment. Validation is used to determine how close the numerical model represents a real life phenomenon; while verification ensures that the model is working within the allowed error limits.

Roache defines code verification as:

> *"The [code] author dëfines 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. mesh increments) $\Delta \rightarrow 0$, the code produces a solution to the continuum equations"* [3]

Thus, code verification is used to verify that the code solves numerical model or set of equations consistently and follows the established theoretical order of accuracy of the discretization method. Verification is based on comparing the numerical results with analytical, exact solutions.

Various methods are used for verification of different systems and solvers including Method of Exact Solutions and Method of Manufactured Solutions. This paper focuses on the Method of Generated Solutions (MGS) [8] and its application for verification of the ICE (Implicit, Continuous fluid, Eulerian), a semi-implicit finite volume solver, that simulates fluid phenomena. In this paper, we analyze order of accuracy, consistency of the discretization errors for various 2D problems. We also analyze factors which can influence the accuracy of the solver.

2. Previous Work

Suppose one needs to solve partial differential equation of the form

$$Du = g \qquad (1)$$

on some domain Ω with boundary Γ. *D* is the differential operator, *g* is the source term and *u* is the exact solution that is sought.

2.1 Method of Exact Solutions

In the widely used verification Method of Exact Solutions one first derives exact solutions to the set of equations solved by the code. Exact solution is a mathematical expression that gives solutions at all locations in space and time. They can be derived using mathematical methods such as the separation of variables, integral transforms (Laplace transforms, Green functions, etc.), etc. Then the code is run with corresponding inputs and a numerical (discrete) solution is generated. This generated numerical solution is compared against exact solution. [3, 7]

One of the major disadvantages of the Method of Exact Solutions is that it is not always possible or often very difficult to find exact solution to the equation or set of equations (e.g. in case when D is non-linear). Also, certain exact solutions (e.g., which use Laplace transforms, infinite sums, etc.) are difficult to implement, which is required for computing their values at a number of points in space and time in order to compare exact and code-generated solutions.

2.2 Method of Manufactured Solutions

Another widely used technique for verifying numerical solvers is the Method of Manufactured solutions [4, 5, 6, 7]. In order to verify that code solves equation *(1)* correctly one has to, first, manufacture a solution *u* (e.g. using an arbitrary mathematical function), and then apply operator *D* and compute the source terms *g*, which become the input to the solver. [3, 4]

Method of Manufactured Solutions is much simpler than the Method of Exact solutions because it does not require user to solve equations (i.e., invert the differentiation operator *D*). However, this method has certain limitations. One of them is that the solutions chosen by the Method of Manufactured solutions are arbitrary and may not reflect the true nature of a real life system and the physical solution that is being simulated. For instance, in order to avoid unnatural oscillations (due to discontinuities, sharp changes in the solution, etc.), which can occur at high resolutions, solvers use special techniques such as gradient, slope limiters, etc. As a result, in this case the Method of Manufactured Solutions may not be an appropriate code verification technique.

3. Method of Generated Solutions

To overcome limitations of the Methods of Exact Solutions and Manufactured Solutions, we propose a new verification method – Method of Generated Solutions (MGS) [8]. The method is designed to verify computational performance of differential and/or integral solvers on exact (analytic) solutions which resemble physical 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 is the technique proposed in [8] to verify such numerical solvers.

In order to verify that equation (1) is solved correctly one has to:

1. Obtain an approximate solution u by a computational algorithm or by experiment.

2. Approximate or interpolate the data from u using an appropriate technique (e.g., spline interpolation or least-squares approximation). These results, u_1, in the analytical form provide the values at all locations in space and time.

3. Apply the differential operator D (exact analytical differentiation) to the functions resulting in step 2. This yields new function g_1 and new problem

$$Du_1 = g_1, \qquad (2)$$

   where exact solution u_1 and source terms g_1 are known.

4. Solve new problem (2) using the numerical solver; forcing functions g_1 generated in step 3 are used as the source terms. As a result, new solution u_2 is generated.

5. Compare solution u_2 generated in step 4 against exact solution u_1 (from step 2) and compute the errors introduced by the solver.

Diagram in figure 1 shows the verification process described above.

4. Discretization errors, consistency and order of accuracy

As a result of the experiments, we try to quantify the *discretization error*, its *consistency* and *order of accuracy* of the Advect and Advance in Time (AAT) module of the ICE algorithm developed by the C-SAFE (Center for Simulations of Accidental Fires and Explosions) research group at the University of Utah.

Discretization of the governing equations subdivides the domain of the problem into finite number of cells. The approximate solution, which satisfies these discretized equations, is not the same as the exact solution, which satisfies the mathematical continuum equations. *Discretization error* is the difference between the two. [3] We use normalized L_2 and L_∞ norms of the difference between exact solution, generated by MGS method, and approximated solution, computed using ICE, to evaluate the discretization errors.

Since in the verification we are using uniform grid NxN, we compute normalized L_2 norm using the following formula:

$$L_2 = \sqrt{\frac{1}{N^2}\sum\nolimits_{n=1..N^2}(u_n - U_n)^2} \qquad (3)$$

We compute L_∞ norm using the following formula:

$$L_\infty = \max(u_n - U_n) \qquad (4)$$

where,

u_n – Exact solution evaluated at x_n, y_n, z_n
U_n – Approximate solution of the discretized equation

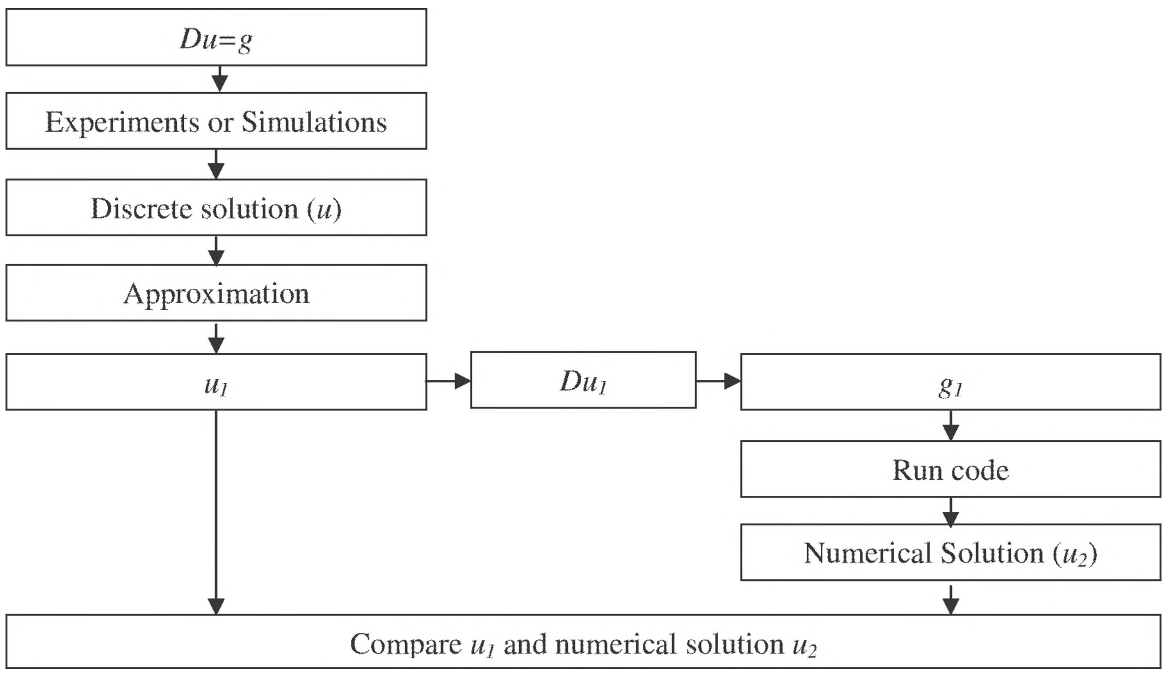

$Du=g$
Experiments or Simulations
Discrete solution (u)
Approximation
u_1
Du_1
g_1
Run code
Numerical Solution (u_2)
Compare u_1 and numerical solution u_2

Figure 1. Verification process using MGS method

Discretization methods are *consistent* if the error goes to zero as the cell size decreases to zero. [3] In order to evaluate consistency of the discretization error, we run experiments for the various cell sizes and compute ratios of L_2 and L_∞ norms.

Order of accuracy is the rate at which the error decreases to zero. We use the following formula to compute the order of accuracy, ρ:

$$\rho = \frac{\log\left(\frac{E_{\text{grid}_1}}{E_{\text{grid}_2}}\right)}{\log(r)} \qquad (5)$$

where,

E_{grid_1} and E_{grid_2} are global errors for grid_1 and grid_2

r is the refinement ratio

5. Experiments

As already mentioned before, we decided to verify the performance of the ICE algorithm developed by the C-SAFE (http://www.csafe.utah.edu) research group at the University of Utah. ICE algorithm is utilized to simulate explosions, fires and other fluid phenomena. ICE is a cell-centered, finite volume version of an algorithm developed and described by Kashiwa, et. al. [9]. ICE uses gradient limiter to suppress unnatural oscillations introduced by the higher-order numerical methods. The effect of the gradient limiter on the order of accuracy will be discussed in this paper as well.

ICE algorithm has several modules which solve various problems. As part of the research, we decided to focus on the Advect and Advance in Time (AAT) module. This module was chosen for verification for the following reasons:

1. The advection operator is an important part of ICE algorithms – it's invoked multiple times during full-scale fluid simulations. Therefore, its accuracy is crucial.
2. The advection operator involves only one governing equation; therefore, it is relatively simple to verify. [10]

In order to isolate the Advect and Advence in Time module, the advection of a passive scalar is employed as a verification experiment. The profiles of the passive scalar are defined using bell-shaped exponential and squared-exponential functions. The 2D governing equation for the experiment is as follows:

$$\frac{d}{dt}(PS(x, y, t)) + U_x\frac{d}{dx}(PS(x, y, t)) + U_y\frac{d}{dy}(PS(x, y, t)) = g(x, y, t)$$

where (6)

$PS(x, y, t)$ – Passive scalar
U_x, U_y – Constant velocity in x and y directions correspondingly
$g(x, y, t)$ – Source terms

As already mentioned above, MGS method can use an approximate solution from a numerical solver or measurements from a physical experiment. Since physical experimental data is not available, we decided to use approximate solution generated by ICE to build our analytical solution. This completes step 1 of the verification process.

We decided to use natural cubic splines to approximate the data extracted from ICE to build up the exact solution. Since the differential operator D from equation (1) corresponding to equation (6), defined as

$$D = \frac{d}{dt} + U_x \frac{d}{dx} + U_y \frac{d}{dy}, \tag{7}$$

is a degree one function, the analytical solution resulting from approximation or interpolation of the approximate solution u has to be at least C^1 continuous. Since natural cubic spline interpolation produces C^2 continuous functions, which fits the requirement, it was chosen as an interpolation technique for computing exact solution and gradients at the specified points of time and space. These gradients are then used to compute source terms $g_1(x,y,t)$ on the right-hand side of the equation *(6)*.

Using these new source terms, we use ICE to solve the new problem given by equation *(2)*, and generate the solution u_2.

Finally, we compare the generated solution u_2 with the original solution u_1 and compute L_2 and L_∞ error, error consistency and order of accuracy.

Summary

1. For our verification experiments we decided to choose profile of the passive scalar to be represented by some of the well-known forms of analytical functions: a 2D bell-shaped exponential and a squared exponential;

2. We decided to choose a $1\text{x}1\text{m}^2$ 2D domain in X and Y dimensions (from -0.5 to 0.5 meters in each direction). We also chose time step, $\Delta t = 10^{-6}$, and the number of time steps in experiments equal to 20.

3. The tests are executed on the grid resolutions 100x100, 200x200, 400x400 and 800x800, which correspond to Δx and Δy equal to $1\text{x}10^{-2}$, $0.5\text{x}10^{-2}$, $0.25\text{x}10^{-2}$ and $0.125\text{x}10^{-2}$ respectively.

4. The source terms $g_l(x,y,t)$ are computed using the MGS method from two different input sources:

   a. Exact analytical solution;
   b. Ice generated solution.

5. ICE allows specifying the desired order of accuracy of the solution. We decided to verify its performance when we select both first and second orders.

6. As already mentioned before, ICE utilizes gradient limiter to suppress unnatural oscillations at places where the gradient of a quantity changes rapidly. The limiter implemented in ICE is computed using Van Leer method [9, 11]:

$$\alpha_j = \min\left(1, \alpha_{j_{\min}}, \alpha_{j_{\max}}\right)$$

where,

$$\alpha_{j_{\min}} = \max\left(0, \frac{u_{\min} - u_j}{\min[u_v] - u_j}\right), \quad \alpha_{j_{\max}} = \max\left(0, \frac{u_{\max} - u_j}{\max[u_v] - u_j}\right)$$

u_j – value of the solution at the cell center;
u_v – value of the solution at the cell vertices;
u_{min}, u_{max} – min and max values of the solution at the surrounding cell centers.

The values of the gradient limiters are used to bound the values of gradients:

$$(\nabla u)_j = \alpha_j (\nabla u)_j$$

To evaluate the effect of the gradient limiter on the solution, discretization error consistency and order of accuracy, we decided to verify the code with gradient limiter enabled and disabled (this applies only to the second order of accuracy tests).

6. Results of the experiments

6.1 Exponential profile

We chose the following function for describing the bell-shaped exponential profile:

$$f(x, y, t) = 4 \cdot \exp\left(\frac{-1}{d_x(1-d_x)}\right) \cdot \exp\left(\frac{-1}{d_y(1-d_y)}\right) \qquad (8)$$

where,

$$d_x = \frac{x+0.3}{0.6}, \quad d_y = \frac{y+0.3}{0.6}$$

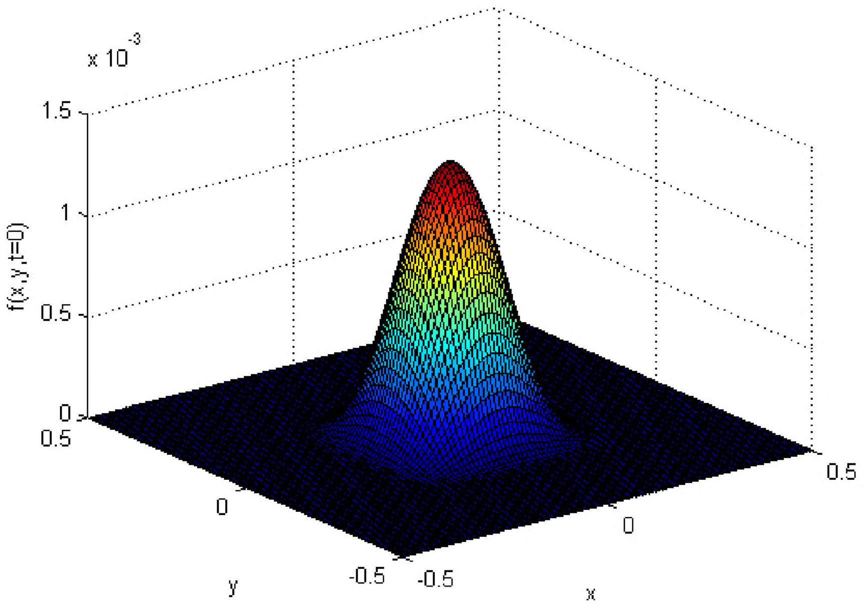


Figure 2. 2D exponential function profile at t=0

The profile of the exponential function at time t=0 is shown in Figure 2. Figure 3 shows the cross-section of the 2D exponential profile at y=0 and corresponding gradient limiter values.

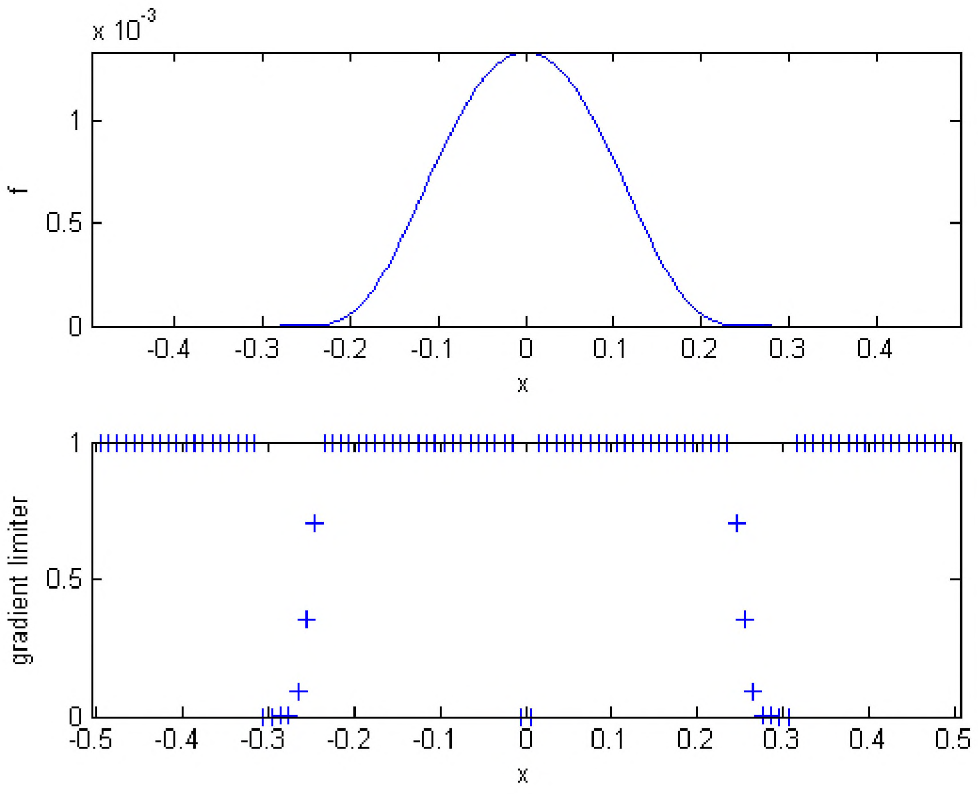


Figure 3.

- Cross-section of the 2D exponential profile at time t = 0 and y = 0;
- Cross-section of the gradient limiter profile for the 2D exponential profile at t = 0 and y = 0.

We ran three sets of tests on the exponential function profile:

1. Gradient limiter is enabled in the ICE code, and second order is the desired order of accuracy of the solution.
2. First order is the desired order of accuracy of the solution.
3. Gradient limiter disabled (or, in other words, gradient limiter values are set to 1) and second order is the desired order of accuracy of the solution.

6.1.1 Second order accuracy, gradient limiter enabled

First, we ran mesh refinement experiments (with the grid refinement ratio equal to two) on the exponential profile, gradient limiter enabled in the solver (ICE) and desired order of accuracy equal to two.

Figures 4 and 5 show results of the experiments – L_2 and L_∞ errors and their ratios as functions of time – for the source terms from analytical solution and ice-generated solution correspondingly. Tables 1 and 2 show the approximate computed errors and observed order of accuracy for analytical and ice-generated solutions correspondingly at the time step 20.

From these results, we can conclude that the resulting discretization error is consistent (the L_2-norm decreases by a factor of approximately 3-3.1 and L_∞-norm decreases by a factor of 2-2.1 when we refine our grid by a factor of two). However, the order of accuracy doesn't match our expectations and theoretical predictions. Instead of order of accuracy two, we got approximately 1.6-1.7 (for L_2-norm) and 1-1.1 (for L_∞-norm).

6.1.2 First order accuracy

As a result, we decided to reduce the desired order of accuracy to one and ran the same set of tests (mesh size is 100x100, 200x200, 400x400 and 800x800) in order to see how the solver will perform. The results are shown in figures 6 and 7, and tables 3 and 4.

As we can see from the results, discretization errors are consistent and go down by a factor of two when resolution is increased by a factor of two. The resulting order of accuracy is equal to the expected value - one - for both solver generated and analytical function generated sources.

6.1.3 Second order accuracy, gradient limiter disabled

To evaluate the effect of the gradient limiter on the order of accuracy of the solution we decided to run an experiment with the desired order accuracy equal to two and gradient limiter disabled. In other words, the values of gradient limiter are set to one. We are testing the same mesh sizes as before – 100x100, 200x200, 400x400 and 800x800. Similar to the first two experiments, the results are presented in figures 8 and 9, and tables 5 and 6.

The results show that disabling the gradient limiter gives us the expected order of accuracy equal to two in case of both solver-generated sources and sources generated from the analytical function.

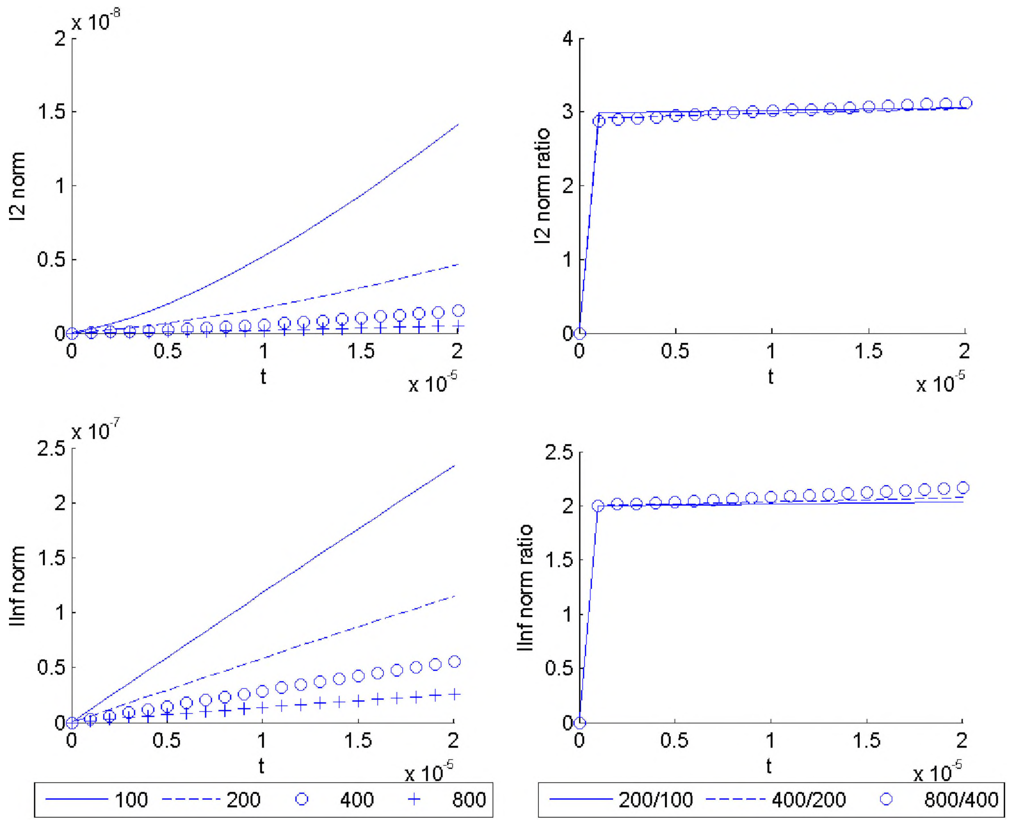


Figure 4. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter enabled* test for exponential 2D profile using source terms from analytical function for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 1. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D exponential profile (source terms are from analytical function)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	1.41659e-08			2.33979e-07		
200x200	4.63891e-09	3.05	1.61	1.15025e-07	2.03	1.02
400x400	1.52472e-09	3.04	1.61	5.54151e-08	2.08	1.05
800x800	4.88037e-10	3.12	1.64	2.55816e-08	2.17	1.12

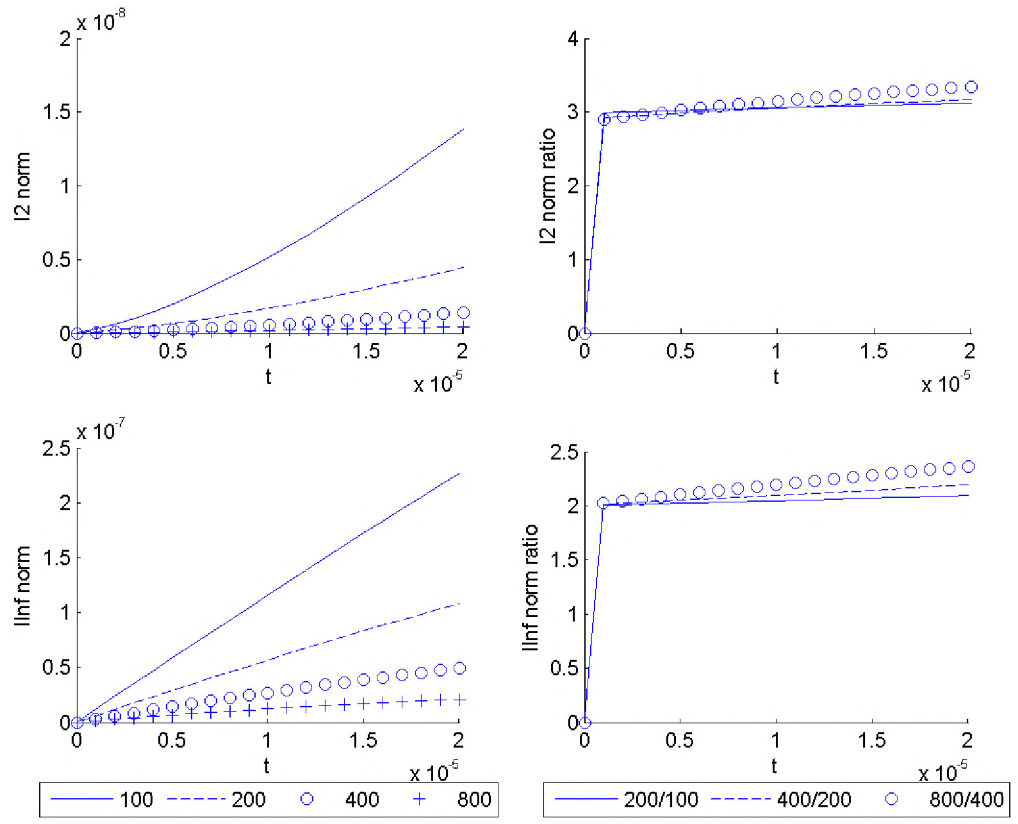


Figure 5. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter enabled* test for exponential 2D profile using source terms from solver's solution for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 2. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D exponential profile (source terms are from ICE-generated solution)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	1.38635e-08			2.26970e-07		
200x200	4.43660e-09	3.12	1.64	1.08364e-07	2.09	1.07
400x400	1.39746e-09	3.17	1.67	4.94638e-08	2.19	1.13
800x800	4.18437e-10	3.34	1.74	2.09230e-08	2.36	1.24

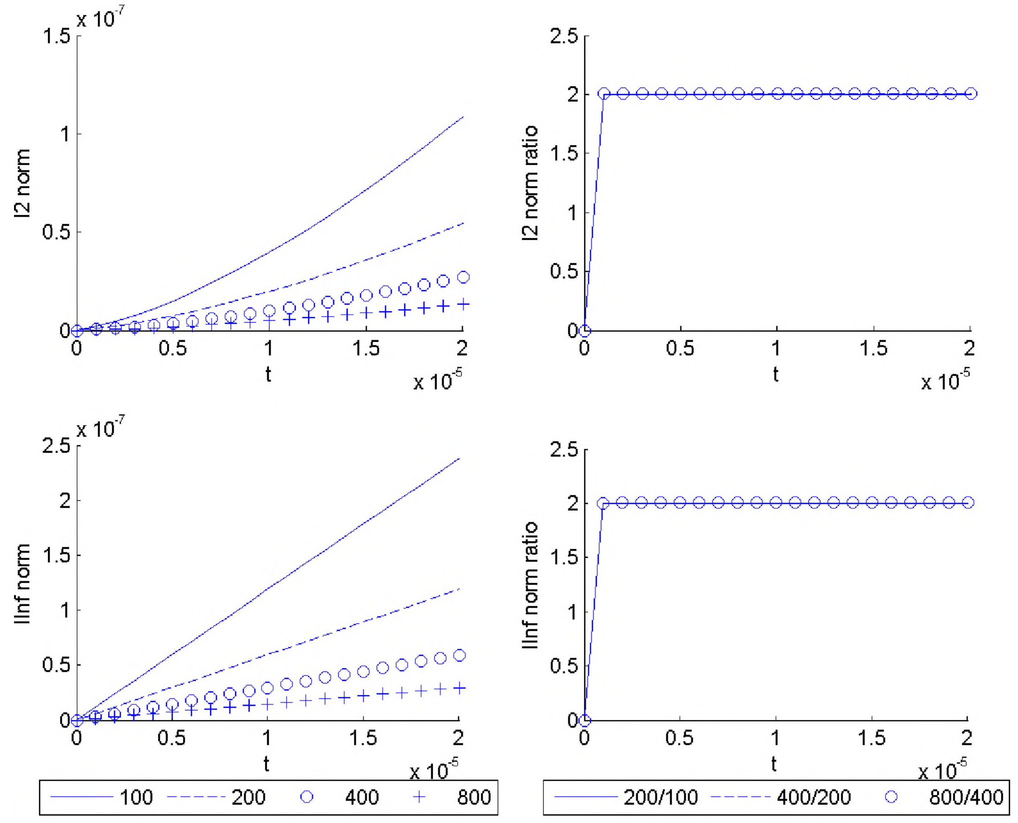


Figure 6. L_2 and L_∞ errors and their ratios as functions of time from *first second order accuracy* test for exponential 2D profile using source terms from analytical function for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 3. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D exponential profile (source terms are from analytical function)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	1.08795e-07			2.37928e-07		
200x200	5.43964e-08	2.00	1.00	1.18995e-07	2.00	1.00
400x400	2.71525e-08	2.00	1.00	5.94001e-08	2.00	1.00
800x800	1.35247e-08	2.01	1.01	2.95864e-08	2.01	1.01

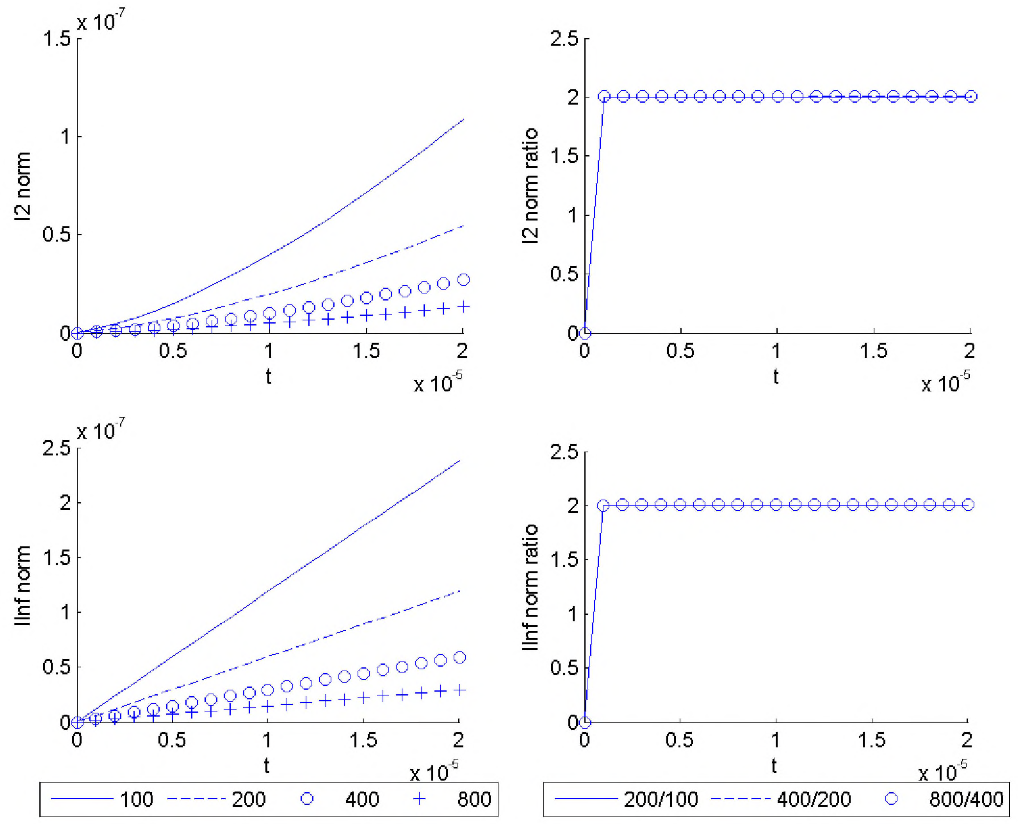


Figure 7. L_2 and L_∞ errors and their ratios as functions of time from *first order accuracy* test for exponential 2D profile using source terms from solver's solution for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01m$);
- 200x200 ($\Delta x = \Delta y = 0.005m$);
- 400x400 ($\Delta x = \Delta y = 0.0025m$);
- 800x800 ($\Delta x = \Delta y = 0.00125m$).

Table 4. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D exponential profile (source terms are from ICE-generated solution)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	1.08779e-07			2.37901e-07		
200x200	5.43921e-08	2.00	1.00	1.18989e-07	2.00	1.00
400x400	2.71514e-08	2.00	1.00	5.93984e-08	2.00	1.00
800x800	1.35244e-08	2.01	1.01	2.95860e-08	2.01	1.01

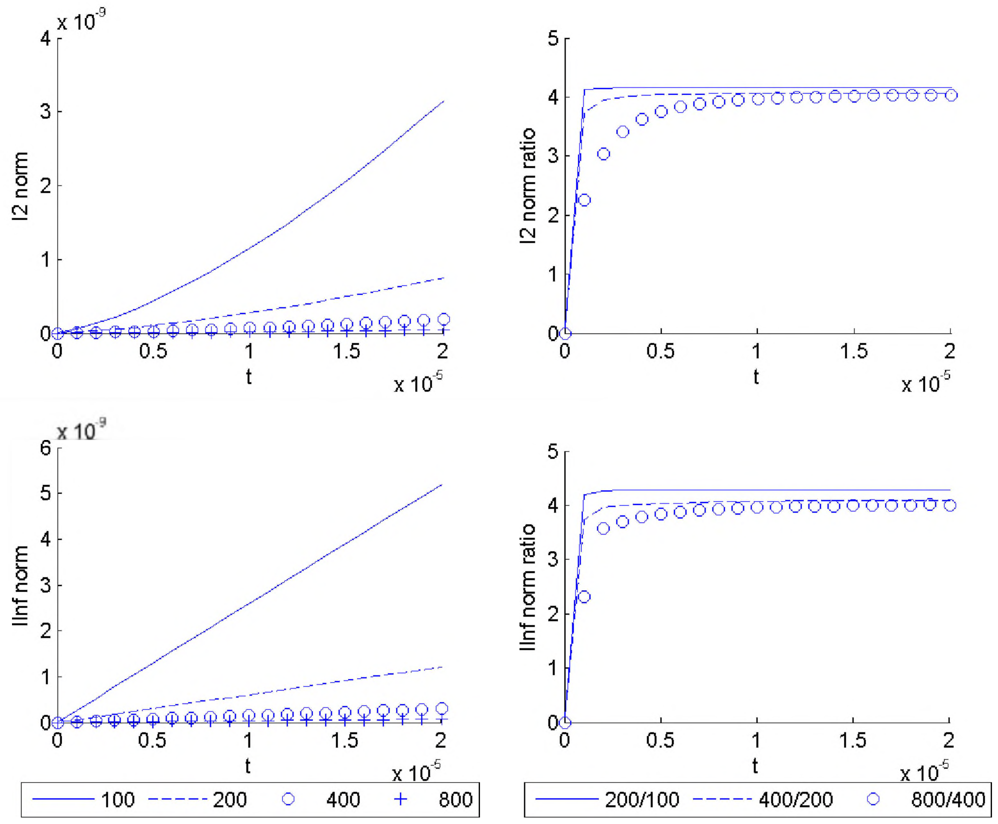


Figure 8. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter disabled* test for exponential 2D profile using source terms from analytical function for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 5. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D exponential profile (source terms are from analytical function)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	3.14777e-09			5.18447e-09		
200x200	7.55431e-10	4.17	2.06	1.21124e-09	4.28	2.10
400x400	1.85757e-10	4.07	2.02	2.96308e-10	4.09	2.03
800x800	4.60069e-11	4.04	2.01	7.41494e-11	4.00	2.00

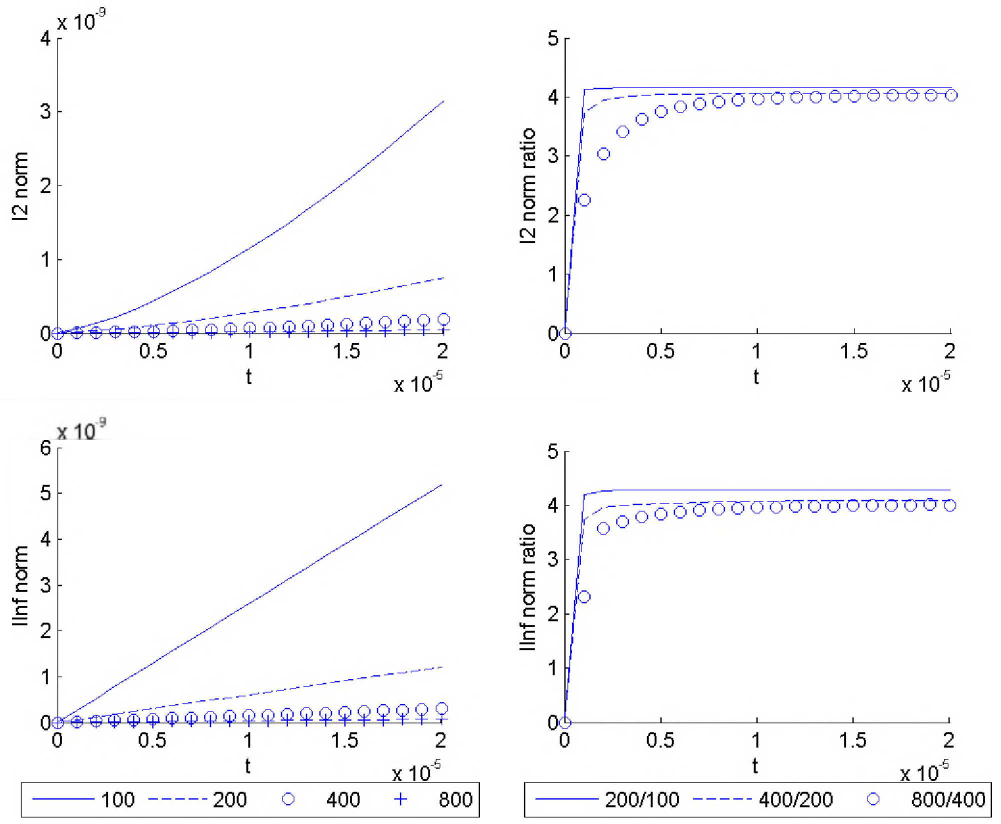


Figure 9. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter disabled* test for exponential 2D profile using source terms from solver's solution for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 6. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D exponential profile (source terms are from ICE-generated solution)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	3.14765e-09			5.18349e-09		
200x200	7.55428e-10	4.17	2.06	1.21121e-09	4.28	2.10
400x400	1.85758e-10	4.07	2.02	2.96308e-10	4.09	2.03
800x800	4.60085e-11	4.04	2.01	7.41542e-11	4.00	2.00

Results of the experiments show that the errors reduce consistently by a factor of two when the mesh size is doubled. Also, observed order of accuracy agrees with the theoretical predictions when the desired order of accuracy is set to one and when gradient limiter is disabled in the code.

Figure 10 shows where the worst errors are occurring at y=0 and time step = 1. The figure shows the plot of the computed solution u_2 and the difference between computed and exact solutions ($u_2 - u_1$) when:

1. the desired order of accuracy is one;
2. the desired order of accuracy is two and gradient limiter values are set to 0;
3. the desired order of accuracy is two and gradient limiter values are set to 0.5;
4. the desired order of accuracy is two and gradient limiter values are set to 1 (or, in other words, gradient limiter is disabled);
5. the desired order of accuracy is two and gradient limiter values are set to 0.75;
6. the desired order of accuracy is two and gradient limiter values are enabled (in other words set to the computed values).

As we can see from the figure when the desired order of accuracy is one, the worst errors occur at the peak of the bell-shaped profile (the error at the peak is on order of 10^{-8}) and around point -0.2 and 0.2 – around the points where gradient changes rapidly. Setting the desired order of accuracy to two and gradient limiter value to zero is equivalent to the first order test. The figure indicates that, indeed, in this case we get the same errors as in case of the first order accuracy.

Setting the desired order of accuracy to two and enabling the gradient limiter (setting it to the computed values) reduces the error at the peak by a factor of two, and is equal to the error when gradient limiter is set to 0.5. The errors at the other locations where the gradient changes rapidly are approximately ten times smaller than in case of the first order of accuracy test.

Finally, we can conclude that for fixed values of the gradient limiter as their (gradient limiters') values approach one the errors go to zero.

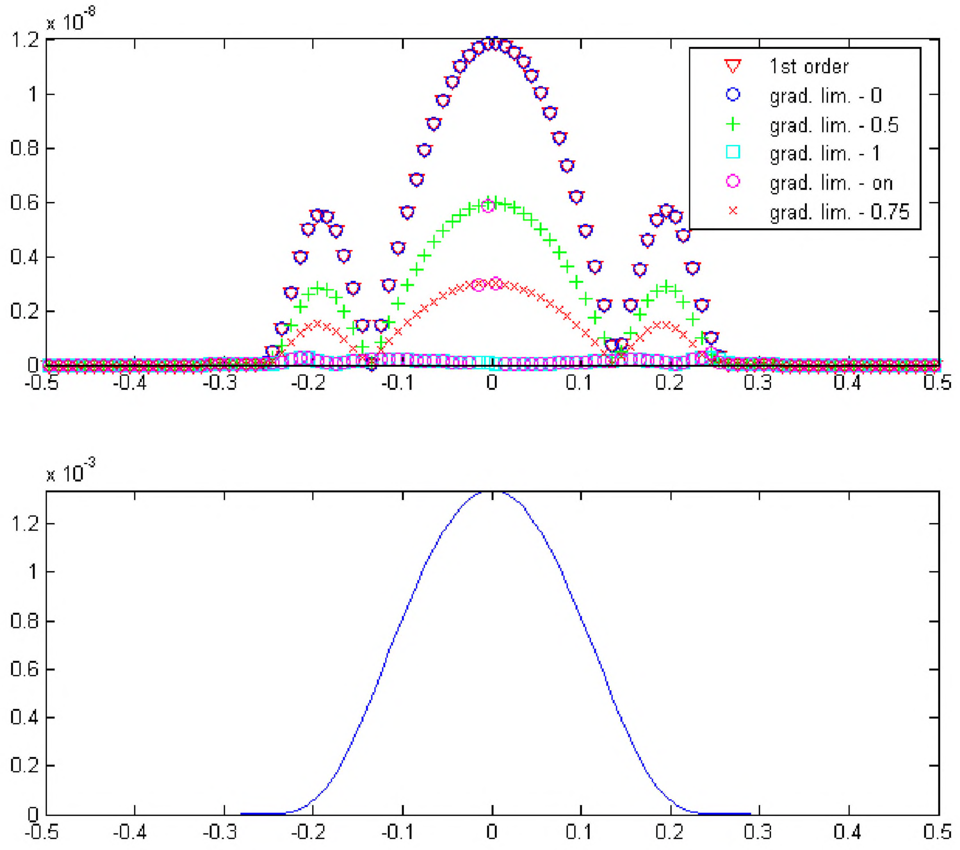


Figure 10. Errors ($u_2 - u_1$) for the exponential function for time step = 1 and y=0, when:

- the desired order of accuracy to one;
- the desired order of accuracy is two and gradient limiter is set to 0;
- the desired order of accuracy is two and gradient limiter is set to 0.5;
- the desired order of accuracy is two and gradient limiter is set to 1 (in other words disabled);
- the desired order of accuracy is two and gradient limiter is set to 0.75;
- the desired order of accuracy is two and gradient limiter is enabled (in other words set to the computed values).

6.2 Exponential squared profile

We also chose the following squared exponential function to describe the profile of the passive scalar:

$$f(x, y, t) = 4 \cdot 10^{6} \cdot \exp\left(\frac{-1}{d_x^{2}(1-d_x)^{2}}\right) \cdot \exp\left(\frac{-1}{d_y^{2}(1-d_y)^{2}}\right) \tag{8}$$

where,

$$d_x = \frac{x+0.3}{0.6}, \quad d_y = \frac{y+0.3}{0.6}$$

The profile of the exponential function at time t=0 is shown in Figure 11. Figure 12 shows the cross-section of the squared exponential profile at y=0 and corresponding gradient limiter values.

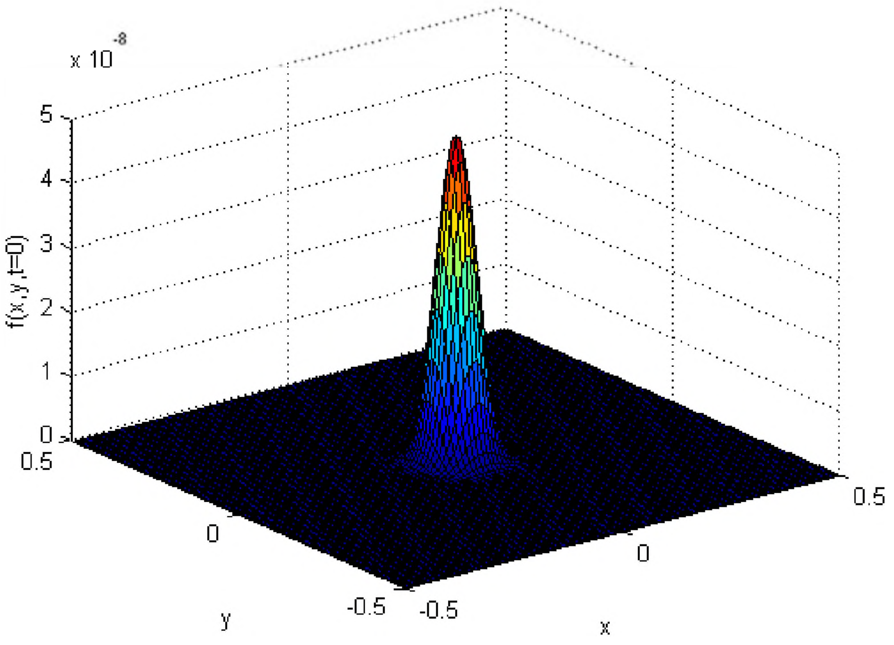


Figure 11. 2D squared exponential function profile at t=0

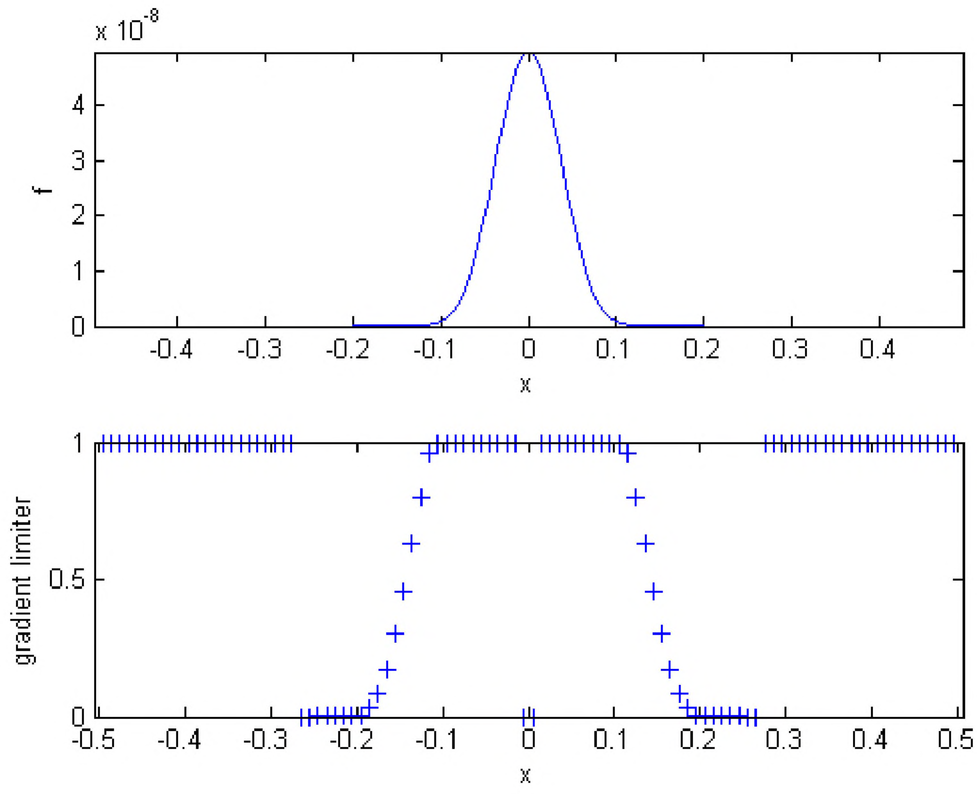


Figure 12.

- Cross-section of the 2D squared exponential profile at time t = 0 and y = 0;
- Cross-section of the gradient limiter profile for 2D squared exponential profile at t = 0 and y = 0.

Similarly to the bell-shaped exponential profile (section 6.1), we ran three sets of tests on the bell-shaped squared exponential function profile:

1. Gradient limiter is enabled in the ICE code, and second order is the desired order of accuracy of the solution;
2. First order is the desired order of accuracy of the solution;
3. Gradient limiter is disabled (or, in other words, gradient limiter values are set to one), and second order is the desired order of accuracy of the solution.

6.2.1 Second order of accuracy, gradient limiter enabled

Similarly to the exponential function experiments, first we ran mesh refinement tests (with the grid refinement ratio equal to two) on the squared exponential profile with gradient limiter enabled in the solver (ICE) and the desired order of accuracy equal to two.

Figures 13 and 14 show the results – L_2 and L_∞ errors and their ratios as functions of time – for the source terms from analytical and ice-generated solutions correspondingly. Tables 7 and 8 show the approximate computed errors and observed order of accuracy for analytical and ice-generated solutions correspondingly.

We can conclude that the resulting discretization errors are consistent (the L_2-norm decreases by a factor of approximately 3.2-3.3 and L_∞-norm decreases by a factor of 2-2.3 when we refine our grid by a factor of two). However, the order of accuracy does not match our expectations and theoretical predictions. Instead of order of accuracy equal to two, we got approximately 1.7 (for L_2-norm) and 1-1.2 (for L_∞-norm).

6.2.2 First order accuracy

As a result, similarly to the case of exponential function profile, we decided to run experiments with the desired order of accuracy equal to one. The results of the experiments are shown in the figures 15 and 16, and tables 9 and 10.

As we can see from the results, when one is the desired order of accuracy discretization errors are also consistent. In addition, they go down by a factor of two; therefore, the resulting order of accuracy is equal to the expected value - one - for both solver-generated and analytical function generated sources.

6.2.3 Second order accuracy, gradient limiter disabled.

Results for the second order accuracy test with the gradient limiter disabled (in other words, gradient limiter values are set to one) for the squared exponential function are presented in figures 17, 18 and tables 11 and 12.

The results show that disabling the gradient limiter gives us the expected order of accuracy equal to two in case of both solver-generated sources and sources generated from the analytical function.

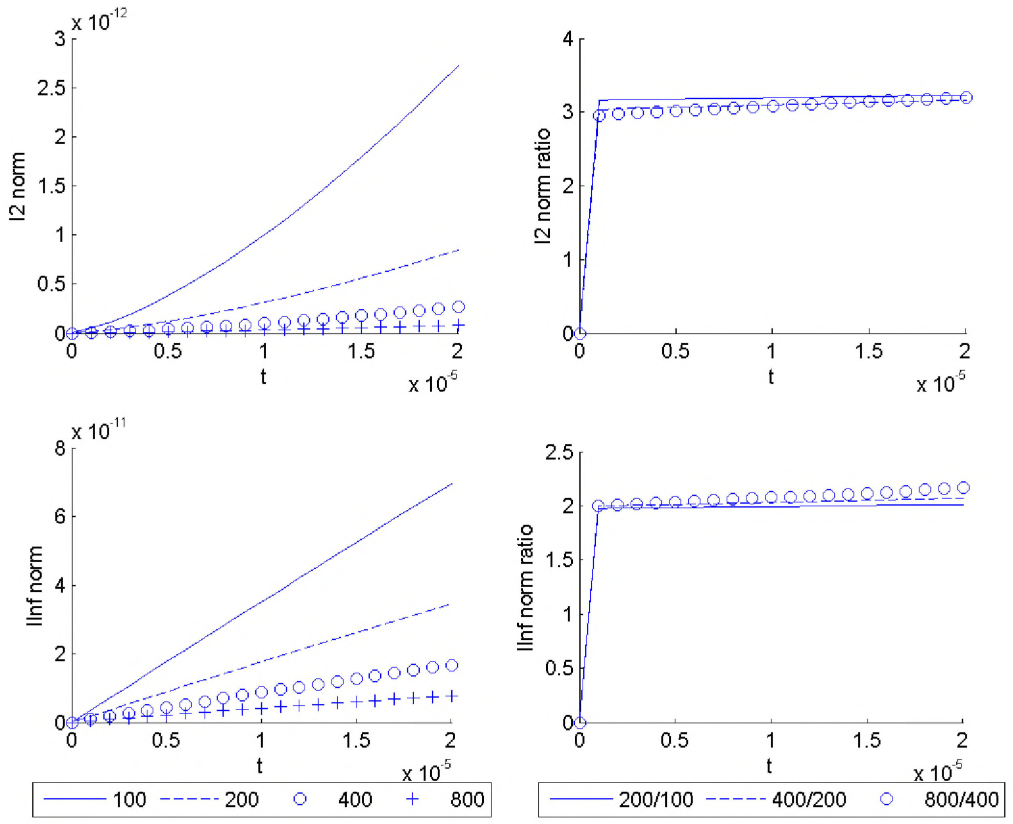


Figure 13. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter enabled* test for squared exponential 2D profile using source terms from analytical function for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 7. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D squared exponential profile (source terms are from analytical function)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	2.71748e-12			6.96352e-11		
200x200	8.41135e-13	3.23	1.69	3.46223e-11	2.01	1.01
400x400	2.65838e-13	3.16	1.66	1.67238e-11	2.07	1.05
800x800	8.31542e-14	3.20	1.68	7.72489e-12	2.16	1.11

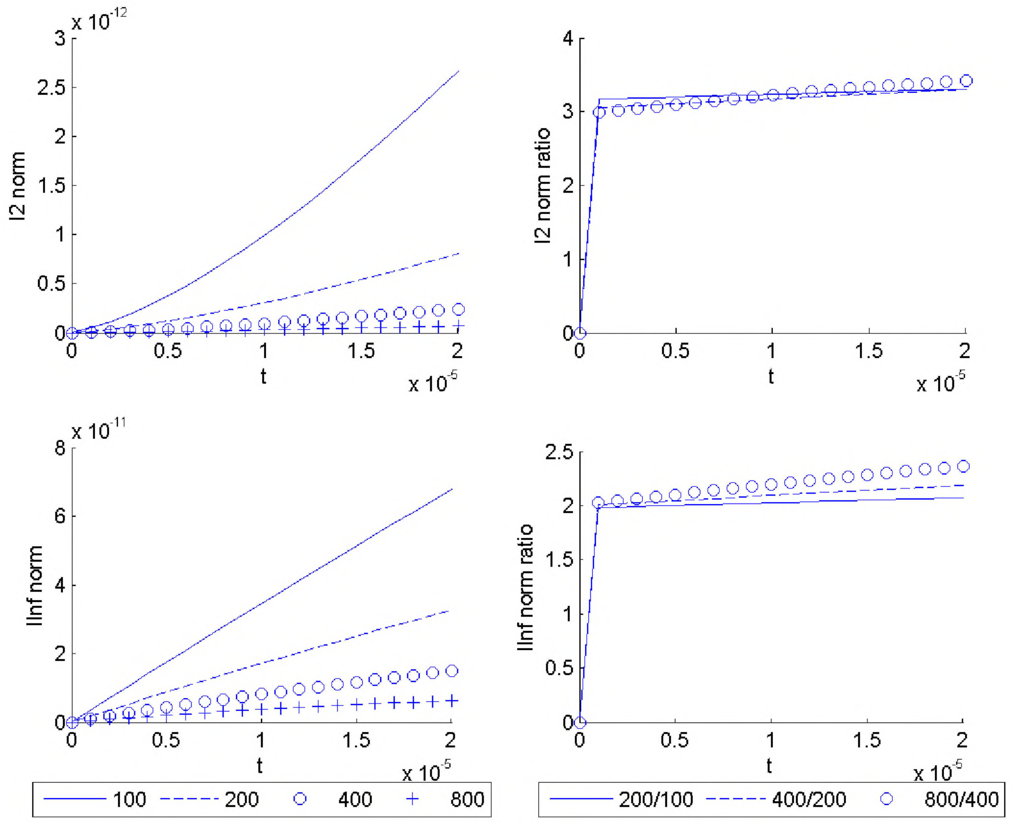


Figure 14. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter enabled* test for squared exponential 2D profile using source terms from solver's solution for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 8. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D squared exponential profile (source terms are from ICE-generated solution)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	2.66661e-12			6.77022e-11		
200x200	8.06013e-13	3.31	1.73	3.26456e-11	2.07	1.05
400x400	2.44279e-13	3.30	1.72	1.49334e-11	2.19	1.13
800x800	7.14662e-14	3.42	1.77	6.31949e-12	2.36	1.24

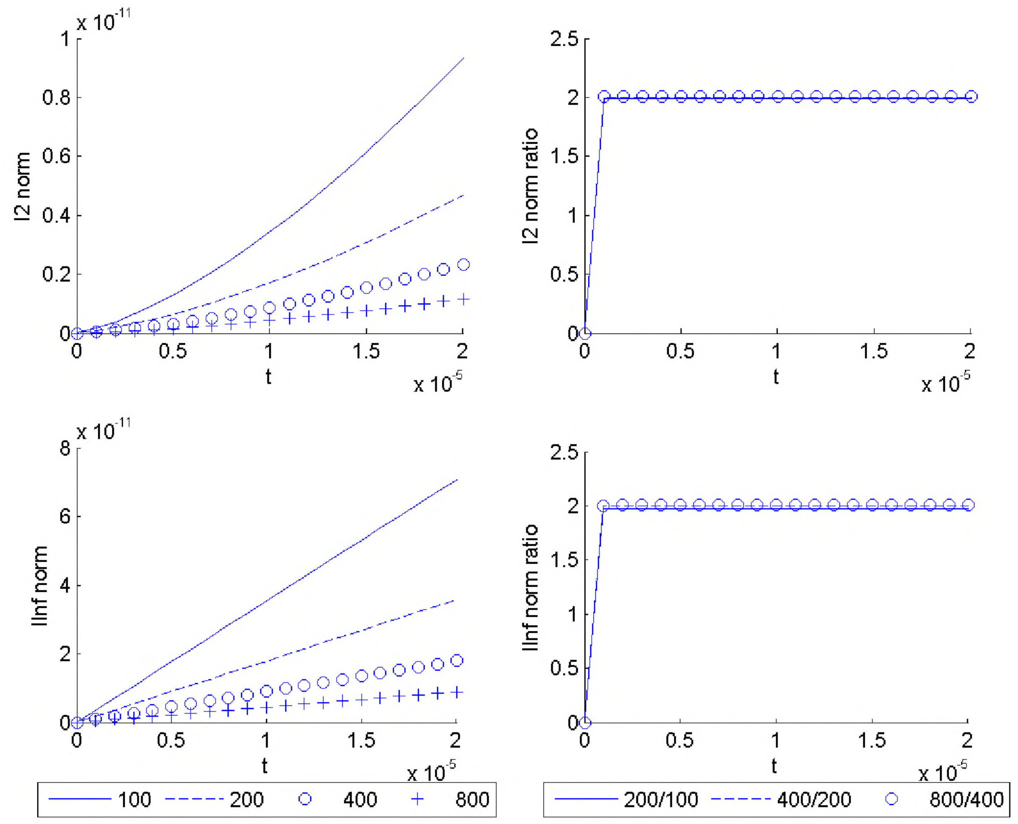


Figure 15. L_2 and L_∞ errors and their ratios as functions of time from *first order accuracy* test for squared exponential 2D profile using source terms from analytical function for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 9. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D squared exponential profile (source terms are from analytical function)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	9.33918e-12			7.07463e-11		
200x200	4.68822e-12	1.99	0.99	3.58000e-11	1.98	0.98
400x400	2.34264e-12	2.00	1.00	1.79217e-11	2.00	0.99
800x800	1.16722e-12	2.01	1.01	8.93293e-12	2.01	1.00

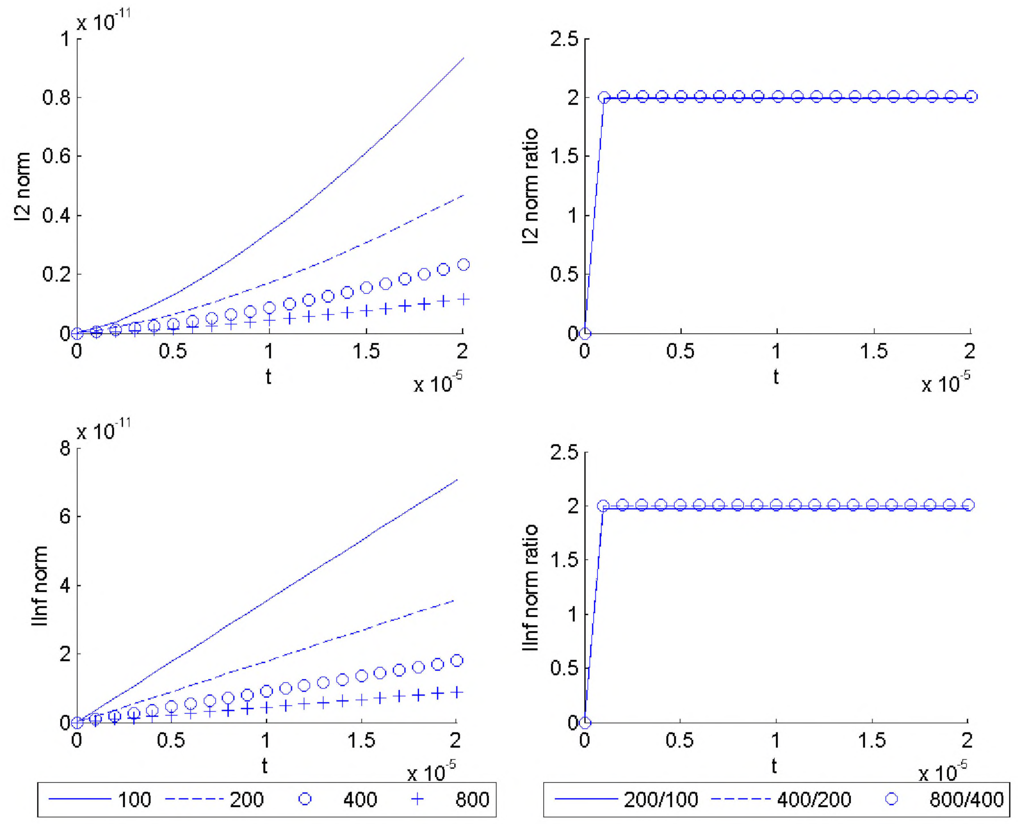


Figure 16. L_2 and L_∞ errors and their ratios as functions of time from *first order accuracy* test for squared exponential 2D profile using source terms from solver's solution for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 10. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D squared exponential profile (source terms are from ICE-generated solution)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	9.33147e-12			7.06531e-11		
200x200	4.68628e-12	1.99	0.99	3.57758e-11	1.97	0.98
400x400	2.34218e-12	2.00	1.00	1.79158e-11	2.00	1.0
800x800	1.16712e-12	2.01	1.00	8.93147e-12	2.01	1.0

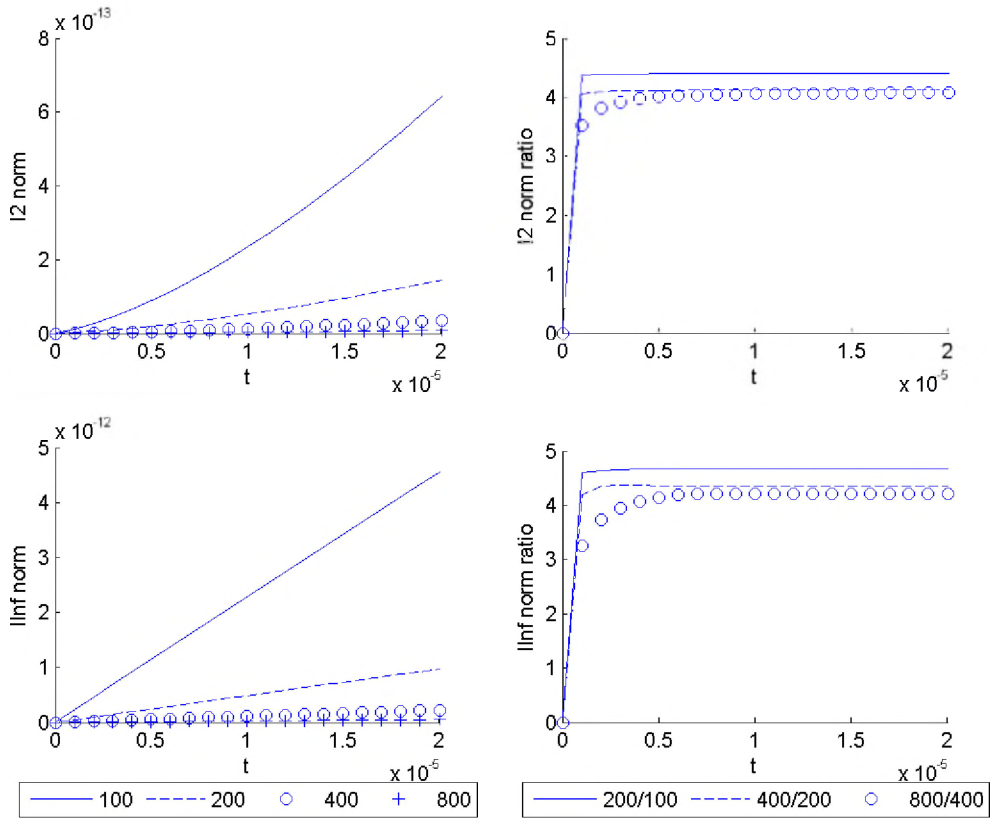


Figure 17. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter disabled* test for squared exponential 2D profile using source terms from analytical function for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 11. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D squared exponential profile (source terms are from analytical function)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	6.41230e-13			4.56135e-12		
200x200	1.45620e-13	4.4	2.14	9.75007e-13	4.68	2.23
400x400	3.52794e-14	4.13	2.05	2.24302e-13	4.35	2.12
800x800	8.65682e-15	4.08	2.03	5.33483e-14	4.20	2.07

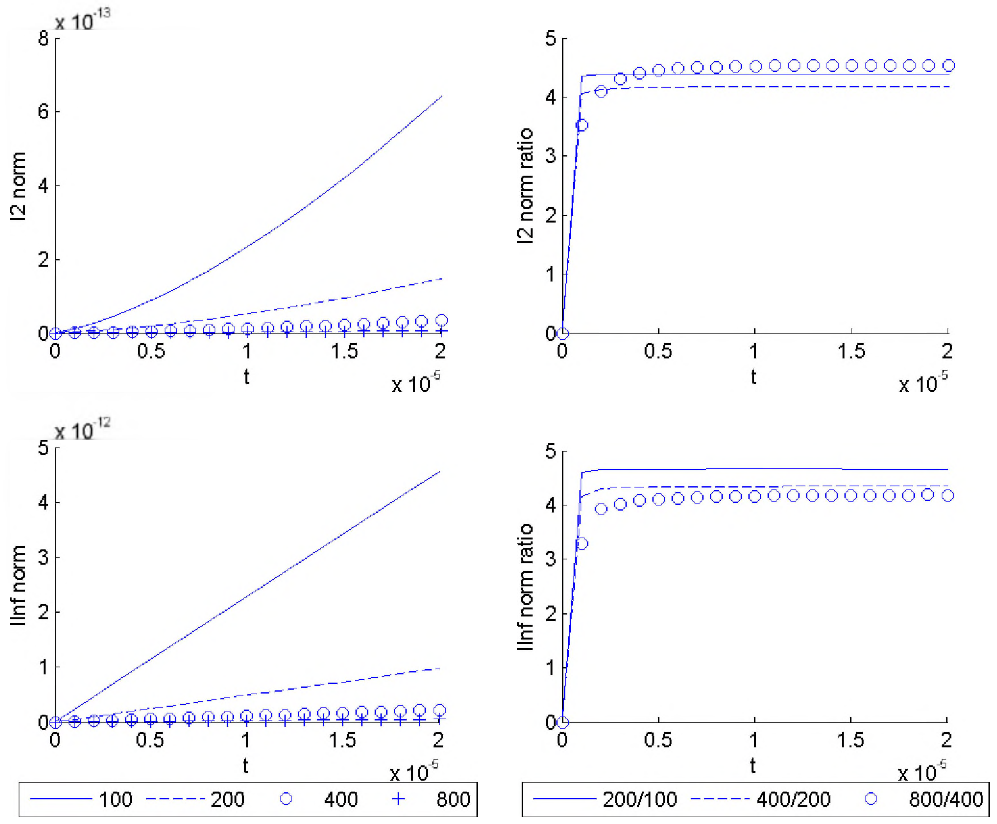
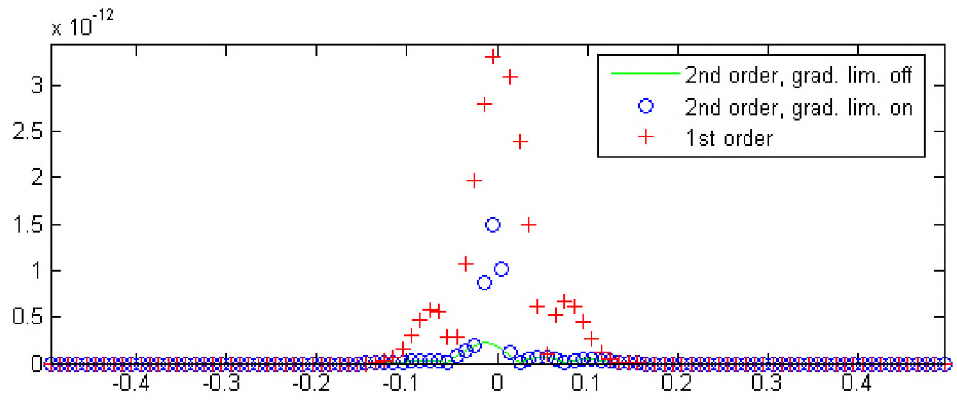


Figure 18. L_2 and L_∞ errors and their ratios as functions of time from *second order accuracy with gradient limiter disabled* test for squared exponential 2D profile using source terms from solver's solution for resolutions:

- 100x100 ($\Delta x = \Delta y = 0.01$m);
- 200x200 ($\Delta x = \Delta y = 0.005$m);
- 400x400 ($\Delta x = \Delta y = 0.0025$m);
- 800x800 ($\Delta x = \Delta y = 0.00125$m).

Table 12. L_2 and L_∞ - norms, error ratios and order of accuracy for 2D squared exponential profile (source terms are from ICE-generated solution)

Grid	L2-norm	Ratio	Observed order of accuracy	Max Error	Ratio	Observed order of accuracy
100x100	6.42001e-13			4.56070e-12		
200x200	1.46065e-13	4.40	2.14	9.78557e-13	4.66	2.22
400x400	3.49763e-14	4.18	2.06	2.25096e-13	4.35	2.12
800x800	7.70293e-15	4.54	2.18	5.38521e-14	4.18	2.06



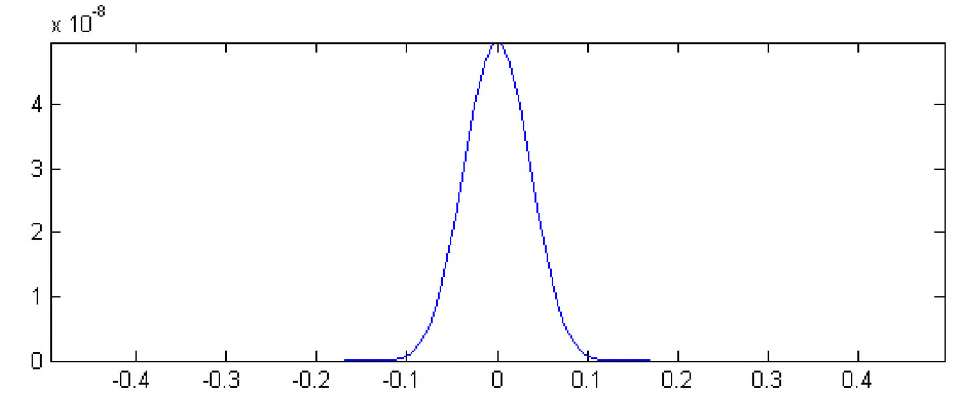


Figure 19. Summary of the errors ($u_2 - u_1$) for the squared exponential function for time step = 1 and y = 0, when:

- desired order of accuracy is 1;
- desired order of accuracy is 2 and gradient limiter is enabled;
- desired order of accuracy is 2 and gradient limiter is disabled.

We can conclude that in all three scenarios the L_2 and L_∞ errors decrease consistently when the resolution is increased by a factor of two. Also, observed order of accuracy agrees with the theoretical predictions when the desired order of accuracy is set to one and when gradient limiter is disabled in the code. However, when the gradient limiter is enabled in the second order of accuracy tests, the order of accuracy decreases by approximately 0.4 in case of L_2 norm and almost by one in case of L_∞ norm.

Figure 19 shows where the worst errors are occurring at y = 0 and time step = 1. The figure shows the plot of the computed solution u_2 and the difference between computed and exact solutions ($u_2 - u_1$) when:

1. the desired order of accuracy is one;
2. the desired order of accuracy is two and gradient limiter values are set to one (or, in other words, the gradient limiter is disabled);
3. the desired order of accuracy is two and gradient limiter values are enabled (in other words, set to the computed values).

As we can see from the figure when the desired order of accuracy is one, the worst errors occur at the peak of the profile (the error at the peak is on order of 10^{-12}) and around -0.07 and 0.07 – the points where gradient changes rapidly. Increasing the desired order of accuracy from one to two (in case when gradient limiter is enabled) reduces the error at the peak by a factor of two. In addition, when gradient limiter is enabled in the algorithm, the errors increase in the areas where the limiter values are not equal to one.

7. Analysis

The results of the experiments indicate that the order of accuracy of the Advect and Advance Module of the solver depends on the problem. Squared exponential function is a c^{∞} function and smoother than regular bell-shaped exponential. As a result, the errors are smaller and the order of accuracy is slightly better for the squared exponential function compared to regular exponential.

Also the experiments demonstrated that the L_2 and L_{∞} errors are decreasing when we are using higher resolution. Moreover they are decreasing consistently (by the same factor).

Also, we can see that the gradient limiter reduces order of accuracy in case of both functions. Although, the limiter limits the value of the gradients only in a few places (places where the gradient changes rapidly), overall order of accuracy decreases significantly. If the reduction in the order of accuracy is not as bad in case of L_2 norm (it reduces from order two to approximately 1.7), in case of L_{∞} norm it reduces almost by one order (from 2 to 1.1 approximately). Disabling the gradient limiter reduces error at the peak to a smaller number and increases order of accuracy.

8. Conclussion

The Method of Generated Solutions was developed to evaluate the discretization errors, their consistency and order of accuracy of ICE algorithm. The method designs analytical solution by interpolating numerical solution from physical experiments or from a solver's solution. Since MGS solutions originate from the actual problems, the results are more representative than the ones obtained by using Method of Manufactured Solutions.

We have used MGS to verify Advect and Advance in Time module of ICE solver on a 2D domain. In the future it is necessary to verify the solver in a 3D domain and on practical test problems. All other modules of the software also need to be verified. Finally, more research should be done in the area of the gradient limiter. Gradient limiter that is being used now does not preserve the order of accuracy well for the functions we used in the experiments (especially in case of L_∞-norm). So, there is a need for a different gradient limiter that would better preserve second order of accuracy.

References

1. Dr. W. L. Oberkampf, *Verification and Validation in Computational Simulations*, Sandia National Laboratories, 2004 Transport Task Force Meeting, 2004

2. C. Hirsch, *Numerical computation of internal and external flows: Fundementals of computational fluid dynamics*, 2007, pp. 541, 542.

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

4. C. J. Roy, C. C. Nelson, T. M. Smith, C. C. Ober, *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.

5. L. Shunn, F. Ham, *Method of Manufactured Solutions Applied to variable density flow solvers"*, Center for Turbulence Research, 2007

6. W. L. Oberkampf and T. G. Trucano, *Verification and validation in computational fluid dynamics*, Sandia National Laboratories, 2002 (SAND2002-0529)

7. K. Salari, P. Knupp, *Code Verification by the Method of Manufactured Solutions*, Sandia National Laboratories, 2000 (SAND2000-1444)

8. A. Ramanujam, P. Milyavskaya, K. Sikorski, T. Harman, *Method of Generated Solutions as a Verification Tool for Numerical Code*, Work in Progress.

9. B. A. Kashiwa, N. T. Padial, R. M. Rauenzahn, W. B. VanderHeyden, *A Cell-Centered Ice Method For Multiphase Flow Simulation*, 1994.

10. A. Ramanujam, C. Sikorski, T. Harman, *The Method Of Generated Solutions for Numerical Verification of the ICE Code*, Technical Report, School of Computing, University of Utah, 2007 (UUCS-07-006)

11. W. B. VanderHeyden, B. A. Kashiwa, *Compatible Fluxes for van Leer Advection*, Journal of Computational Physics, vol. 146, pp. 1-28, 1998