LA-UR-13-26506
Approved for public release; distribution is unlimited.

Title: One Dimensional Lagrangian Hydrocode Development

Author(s): Esmond, Micah J.
Thurber, Andrew J.

Intended for: 2013 Computational Physics Summer Student Workshop,
2013-06-10/2013-08-16 (Los Alamos, New Mexico, United States)

Issued: 2013-08-16

2013 Computational Physics Workshop
Los Alamos National Laboratory

One Dimensional Lagrangian Hydrocode Development

Micah Esmond
Department of Mechanical Engineering
Virginia Tech
jemicah2@vt.edu

Andrew Thurber
Department of Mechanical Engineering
Virginia Tech
ajthurbe@vt.edu

Mentor:
Dr. Nathaniel Morgan
XCP-8: Verification and Analysis
Los Alamos National Laboratory
nmorgan@lanl.gov

August 16th, 2013

Table of Contents

Introduction

Arbitrary Lagrangian Eulerian (ALE) codes combine the Lagrangian and Eulerian techniques. The Lagrangian approach is used to evolve the mesh, and the Eulerian approach is used to remap physical quantities after the mesh has been relaxed. This project focuses on the development of a 1D Lagrangian code as a testbed for ALE techniques; various hydrodynamic methods can be compared within a single framework. Remapping and ALE have not yet been implemented, and the code remains fully Lagrangian. The development and use of this code will aid in the understanding of the mathematics behind these types of algorithms.

Code Details

The code requires user inputs stored in a text file that is called by the input function in the code. Having a separate text file containing user inputs allows for the inputs to be changed easily and eliminates the need to compile the code after a slight change in the inputs. The use of a separate text file for inputs also requires the development of a parser. The parser used in this code was developed with the help of Dr. Vincent Chiravalle. The parser reads in the input text file and separates each word, variable, and value and then stores them in an array of strings. The input function then queries the array and extracts user-defined values and assigns them to the simulation parameters. These simulation parameters include mesh density, domain length, boundary conditions, thermodynamic quantities, and output options.

Once the simulation parameters defined by the user are collected, the initialize function in the code uses them to populate 1D arrays that will be used in the simulation. The code utilizes these arrays to represent the spatial distributions of thermodynamic quantities. The arrays can represent multiple materials. For the simulations of interest in this project, the property that differentiates materials is the ratio of the specific heats, i.e. gamma. Each element in an array represents an individual control volume containing a single material. Once ALE is implemented in the code, individual elements will have to represent multiple materials because of remapping procedures. This will be done using dynamic link lists.

The physics of the simulations are carried out in separate functions called “CCH”, “PCH”, “MPC”, “SGH”. These functions contain the Runge-Kutta time integration steps which include the majority of the calculations involving the conservation equations and the equation of state. At the end of these functions, the maximum speed of sound in the computational domain is computed. This value is fed back into the main function to compute the next time step. The main function serves to calculate the next time step and print out current simulation information including the current time, time step, and the minimum Δx across the domain.

At user-specified time intervals, the output function records the spatial distributions of thermodynamic quantities in data files. The data files are named using the name of the input file with the time information appended to it. Output files are generated for both point and cell arrays. These data files can then be plotted using third party plotting software such as GNUplot.

Source Code Files

The source code files are presented and described in Table 1.

Table 1. Source code file names and descriptions.

Source code file name	Description
program.c	Contains the main function as well as optimization functions used for the computations. The main function calls the subsequent functions and organizes their respective inputs and outputs. The main function also calculates the next time steps to input into the hydro packages.
header.h	Spacial arrays and other simulation parameters are declared in this file in order to be used globally. This file is included in each of the source code files.
setDefaults.c	Contains the function that establishes simulation parameters to be used if the user does not include them in the input file. It provides additional robustness to minimize user frustration.
input.c	Contains the function that parses the user input file and saves to variables for further use. If inputs are omitted, the default values are used.
initialize.c	Contains the function allocates memory and defines arrays and variables included in header.h, based on values from the user input file.
CCH.c	Contains the cell-centered hydrodynamics function. This funtion calls a separate function that performs a Riemann solution and steps the simulation parameters forward in time. Separate volume and area have also been included. See CCH Description.
PCH.c	Contains the point-centered hydrodynamics function. This function is used for the PCH and PCHA methods. It includes various functions for the Riemann solution and the time integration. It also contains a functions that compute a volumes and areas. See PCH Description and PCH Method Comparison.
SGH.c	Contains the staggered-grid hydrodynamics code. There is also a separate function that performs a Riemann solution and steps the simulation parameters forward in time. Volumes and areas are also computed in separate functions. See SGH Description.
MPC.c	Contains the modified point-centered hydrodynamics code. This function also calls a separate function that performs a Riemann solution. Volumes and areas are also computed in separate functions.
output.c	Contains a function that generates output files at specific time intervals defined by the user. This time interval is referred to as *dtdump*. Values for relevant properties such as density, pressure, etc. are output at each point or cell location, depending on the hydro method selected by the user.

Each source code file includes the header.h file. When the source code is compiled using the makefile, an executable file named *code* is created. The executable file is run with an input file using the following syntax:

```
./code ExampleInput.inp
```

The flowchart in Figure 1 details the overall procedure of the 1D hydrocode.

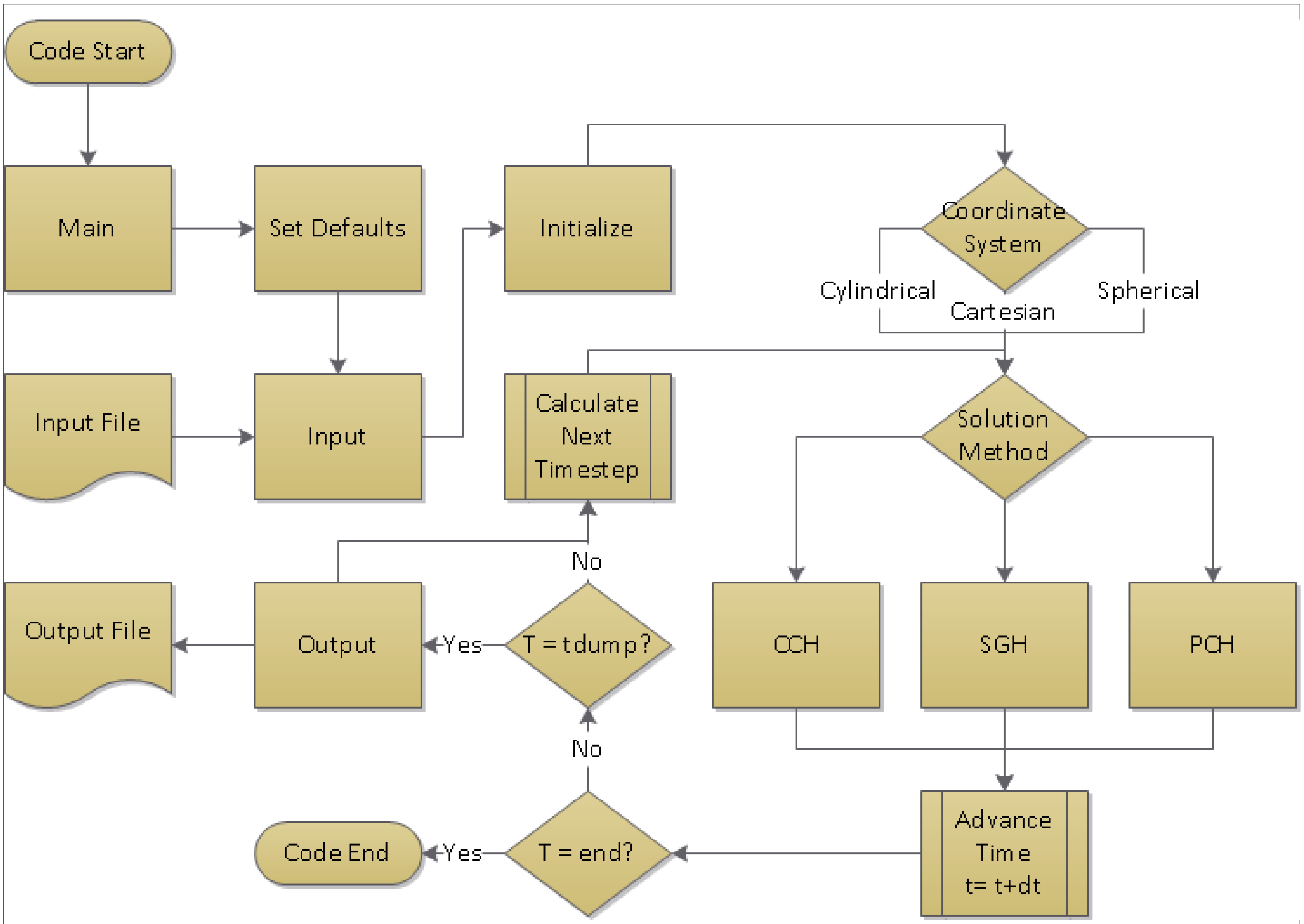


Figure 1. Flowchart of code structure.

Input

The user provides inputs to the hydrocode using a separate text file. If the user omits any inputs, default values will be used in their place. A list of the available user inputs along with the default values is shown in Table 2.

Table 2. User inputs and default values.

Input	Default Value	Comments
General Inputs		
mats	2	Number of Materials
np	20	Number of points used in the simulation
L	10	Length of domain (this is also the radius in curvilinear coordinates)
init-ie?	0	Initializes the simulation using specified internal energy (1) or pressure (0)
init_from_cell	0	Initializes point values based on the user inputs (0) or on cell values (1). Use (1) to smooth the shock front.
BC1	0	Fixed (0), Free (1)
BC2	0	Fixed (0), Free (1)
BCu1	0	Fixed wall velocity
BCu2	0	Fixed wall velocity
Method	CCH	
VelOpt	0	Computes density and total energy change from averaged point velocities (0) or from Riemann velocities (1)
AvgOpt	0	Averages the nodal velocities in space (0) or in space and time (1)
NodePosOpt	0	Updates the nodal positions based on the nodal velocities (0) or the averaged control volume boundary positions (1)
Coordinate_System	car	Planar (car), Cylindrical (cyl), Spherical (sph)
Material Inputs		
u	0	Initial velocity of each material
p	1.0 and 0.1	Initial pressures of material 1 and material 2, respectively
rho	1.0 and 0.1	Initial densities of material 1 and material 2, respectively
ie	1.0 and 0.1	Initial specific internal energies of material 1 and material 2, respectively
x1	0.0 and 5.0	Start locations of material 1 and material 2, respectively
x2	5.0 and 10.0	End locations of material 1 and material 2, respectively
gamma	1.4	Gamma of each material
I/O Parameters		
dt0	1e-9	Initial time step
tstop	30.0	Stop time
dtdump	5.0	Time interval to write output files
CFL	0.5	CFL parameter for time step control
CFLV	0.01	CFLV parameter for time step control

It is important to note that the materials must be entered in the order they appear in the domain from left to right. The 1D hydrocode assumes a consistent set of units. A typical system of units used for these problems is cm (length), μs (time), megabar (pressure), cc (volume), g (mass), megabar cc/g (specific energy).

CCH Description

Cartesian Coordinates

In CCH, all parameter values are stored at the cell's center. A Riemann-like solution is used to determine the velocity and pressure at the interface between cells, called points or nodes. See Figure 2.

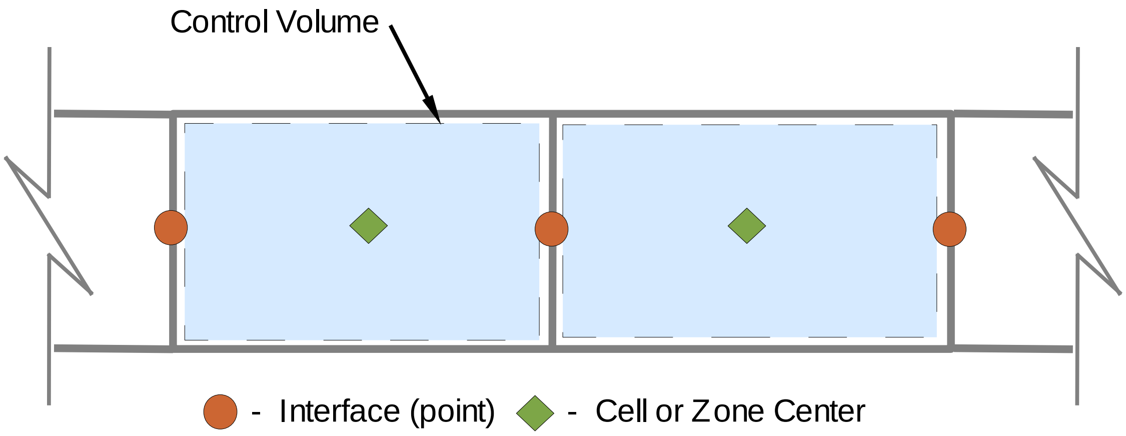


Figure 2. CCH mesh.

For 1D, the CCH method uses the following governing equations. The continuous equations are on the right, and the discretized approximations are shown on the right.

Continuous	**1D Discrete**
$\frac{d}{dt}\int_V \rho dV = 0$	$\frac{\Delta M_z}{\Delta t} = 0$
$\nabla \cdot \mathbf{u} = \lim_{V \to 0} \frac{1}{V}\frac{\delta V}{\delta t}$	$\frac{\Delta V_z}{\Delta t} = \sum_{CV} (\mathbf{n}A\ u^*)^{n+\frac{1}{2}}$
$\frac{d}{dt}\int_V \rho \mathbf{u} dV = \oint_A \mathbf{n} \cdot (-P\mathbf{I}) dA$	$M\frac{\Delta u_z}{\Delta t} = -\sum_{CV} (\mathbf{n}A\ P^*)^{n+\frac{1}{2}}$
$\frac{d}{dt}\int_V \rho \left(e + \frac{u^2}{2}\right) dV = \oint_A \mathbf{n} \cdot (-P\mathbf{I}) \cdot \mathbf{u}\, dA$	$M\frac{\Delta j_z}{\Delta t} = -\sum_{CV} (\mathbf{n}A\ P^*\ u^*)^{n+\frac{1}{2}}$

where *M* is the mass, *V* is the volume, **u** is the velocity, *P* is the pressure, *j* is the specific total energy, and **n** is the surface normal vector that points in either the positive or negative direction in 1D. The superscript * indicates the Riemann solution which is discussed in a later section. The subscript *z* indicates a zone or cell centered quantity. The superscript *n+1/2* indicates the time integration scheme.

The time integration scheme is discussed in more detail in a later section. In 1D, the change in the x-location of the points is used to compute the change in volume. The change in the x-location of a point is determined by

$$\frac{\Delta x_p}{\Delta t} = u^{*(n+\frac{1}{2})}$$

The density is updated using the volume change. The internal energy is determined using the equation

$$e_z = j_z - \frac{1}{2}u_z^2$$

where e_z is the specific internal energy in the zone. Using the updated density and the internal energy, the pressure is updated using the equation of state for a gamma-law gas. A constant gamma is assumed. *P* is given by

$$P_z = \rho_z(\gamma - 1)e_z$$

Cylindrical Coordinates

For cylindrical coordinates in 1D, the same governing equations are used as in Cartesian coordinates. The difference is in the volume and area calculations. The volume is computed as the volume per radian and is given by

$$V_z = \frac{1}{2}(r_{p+1}^2 - r_p^2)$$

where *dz*, the depth of the cylinder, is understood to be of unit length. The area that is used to compute the forces on the individual control volumes is computed in order to preserve consistency with the divergence relationship. The divergence relationship is

$$\bigtriangledown \cdot \mathbf{u} = \lim_{V \to 0} \frac{1}{V}\frac{\delta V}{\delta t}$$

The right side can be expressed as

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{V_z}\left[\frac{\delta V_z}{\delta r_{p+1}}\frac{\delta r_{p+1}}{\delta t} + \frac{\delta V_z}{\delta r_p}\frac{\delta r_p}{\delta t}\right]$$

simplifying, one obtains

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{2}(r_{p+1}^2 - r_p^2)}\left[r_{p+1}u_{p+1} - r_p u_p\right]$$

Simplifying again yields

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{2}(r_{p+1}+r_p)}\frac{[r_{p+1}u_{p+1} - r_p u_p]}{r_{p+1}-r_p}$$

and

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{2}(r_{p+1}+r_p)}\frac{\delta r u}{\delta r} \approx \bigtriangledown \cdot \mathbf{u}$$

Therefore, using the average radius at a control volume to determine the area will be consistent with the divergence relationship. The area per radian used to determine the forces acting on the cell is given by

$$A = \frac{1}{2}(r_{p+1}+r_p)$$

where *dz* is understood to be of unit length.

Spherical Coordinates

For spherical coordinates in 1D, the same governing equations are used as in Cartesian coordinates. However, the volume and area calculations are different. The volume is computed as the volume per steradian and is given by

$$V_z = \frac{1}{3}(r_{p+1}^3 - r_p^3)$$

The area used to compute the forces acting on the individual control volumes is computed in order to preserve consistency with the divergence relationship presented in the previous section. Similar to the derivation in the previous section, it can be shown that

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{3}(r_{p+1}^3 - r_p^3)}\left[r_{p+1}^2\, u_{p+1} - r_p^2\, u_p\right]$$

Simplifying yields

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{3}(r_{p+1}^2 + r_{p+1}r_p + r_p^2)}\frac{\left[r_{p+1}^2\, u_{p+1} - r_p^2\, u_p\right]}{r_{p+1}-r_p}$$

and

$$\frac{1}{V_z}\frac{\delta V_z}{\delta t} = \frac{1}{\frac{1}{3}(r_{p+1}^2 + r_{p+1}r_p + r_p^2)}\frac{\delta r^2 u}{\delta r} \approx \nabla \cdot \mathbf{u}$$

From this derivation, the area per steradian is given by

$$A = \frac{1}{3}(r_{p+1}^2 + r_{p+1}r_p + r_p^2)$$

Boundary Conditions

Two different types of boundary conditions are available in the code. The first is a fixed or reflective boundary, the second is a free boundary condition. The details of the mathematics used to simulate these conditions are presented below

CCH: Fixed (Reflective) Boundary Condition

For a fixed boundary, the velocity at the boundary is set to a specific value set by the user. Figure 3 illustrates mesh at the boundary.

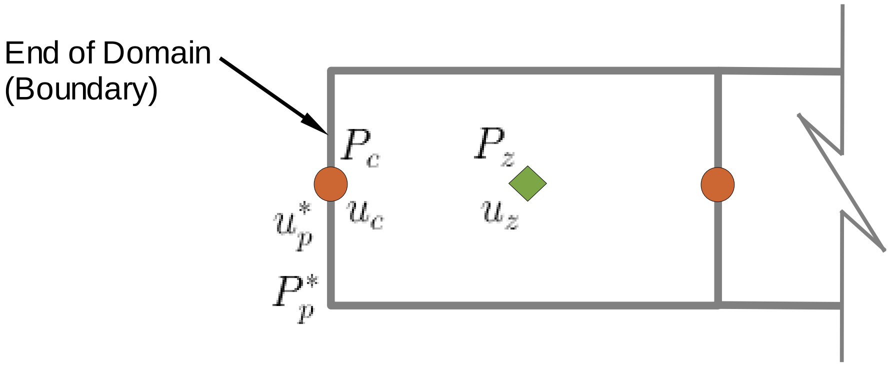


Figure 3. CCH boundary mesh.

The fixed velocity at the boundary is u^*_p. The pressure at the boundary is determined by the equation

$$P_p^* = P_c - \mu(u_p^* - u_c)\mathbf{n}$$

where **n** is the normal vector pointing in the positive or negative direction. μ at the boundary cell can be approximated by

$$\mu = \rho c$$

where the density and the speed of sound are evaluated at the boundary cell. Also, for a first order approximation,

$$u_c = u_z, \;\; P_c = P_z$$

Once the pressure and velocity at the boundary are known, the governing equations for the boundary cell can be implemented normally.

CCH: Free Boundary Condition

For a free boundary condition, the pressure at the boundary must be maintained at zero. The velocity at the boundary must be determined. By manipulating the equation for *P** in the previous section, the following equation is obtained.

$$u_p^* = \frac{P_c \mathbf{n}}{\mu} + u_c$$

Where the approximations used for μ and the projected velocity and pressure are the same as for the fixed boundary condition.

SGH Description

In the SGH approach, the pressure, internal energy, and density are stored at the cell center. The velocity is stored at the points. The momentum and energy equations are solved on two separate control volumes. See Figure 4.

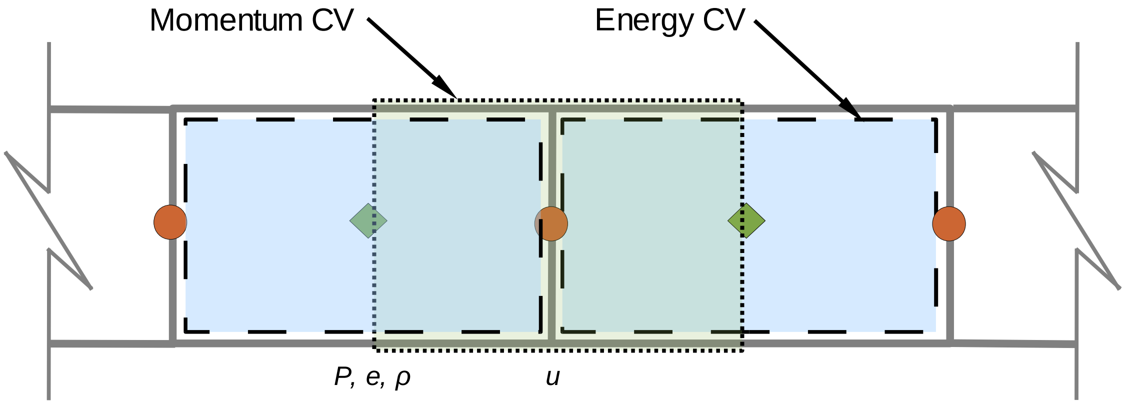


Figure 4. SGH mesh.

The momentum control volume is designated MCV, and the energy control volume is designated ECV. The discrete Lagrangian governing equations for SGH are shown below on the right and the continuous equations are shown on the left.

Continuous

$$\frac{d}{dt}\int_V \rho dV = 0$$

$$\frac{d}{dt}\int_V \rho \mathbf{u} dV = \oint_A \mathbf{n}\cdot(-P\mathbf{I})dA$$

$$\frac{d}{dt}\int_V \rho e dV = (-P\mathbf{I}) : \oint_A \mathbf{n}\,\mathbf{u}\,dA$$

1D Discrete

$$\frac{\Delta M_z}{\Delta t} = 0$$

$$M_p \frac{\Delta u_p}{\Delta t} = -\sum_{MCV} (\mathbf{n}A\ P_z^*)^{n+\frac{1}{2}}$$

$$M_z \frac{\Delta e_z}{\Delta t} = -P_z^{*(n+\frac{1}{2})} \sum_{ECV} (\mathbf{n}A\ u_p)^{n+\frac{1}{2}}$$

where M_p is the mass of the MCV centered at a point, and M_z is the mass of the ECV at a cell. e_z is the specific internal energy. The volume and density of the cell are updated based on the change in the locations of the points. The location of the points is updated using a time averaged velocity as in the equation.

$$\frac{\Delta x_p}{\Delta t} = \frac{1}{2}(u_p^n + u_p^{n+1}) \equiv u^{n+\frac{1}{2}}$$

A pressure is computed using the equation of state for a gamma-law gas where a constant gamma is assumed.

$$P_z = \rho_z(\gamma - 1)e_z$$

A Riemann-like problem is solved at the center of the each cell. In 1D, the velocities are projected from the points to the cell centers. The pressure stored at the cell center is the pressure used to determine u^* and P^*.

Curvilinear Coordinates

The equations implemented for curvilinear coordinates in SGH are derived similarly to those used for CCH. However, the area used in the momentum equation is evaluated at the center of the momentum control volume. This same area is then used in the energy equation. A predictor-corrector scheme was implemented in SGH. This procedure is explained in [1]. First, the nodal positions are estimated at the next time step using a time averaged velocity. The velocity is then updated using the areas computed from the predicted nodal positions. New nodal positions are then computed using the updated velocity and these values are compared to the predictions. The process is repeated until the difference in the predicted and computed nodal positions is negligible.

Boundary Conditions

SGH: Fixed (Reflective) Boundary Condition

The SGH approach utilizes a momentum control volume centered on the points and an energy control volume centered on the cells. To simulate a fixed boundary condition, the change in the velocity with respect to time for the boundary node is set to zero. To illustrate this concept, a customized momentum control volume is established for the boundary node as shown in Figure 5.

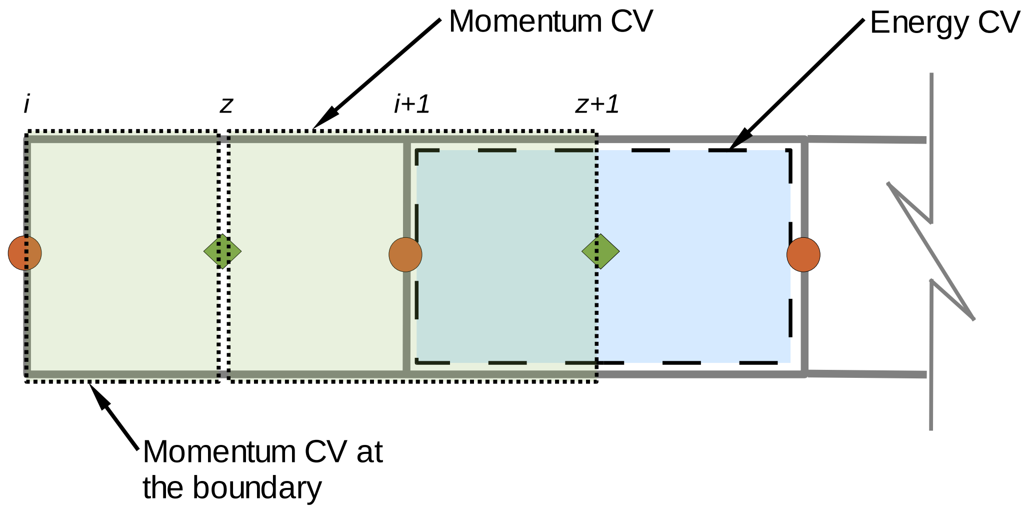


Figure 5. SGH boundary mesh.

For the momentum control volume at the boundary, the pressure on the right side must be equal to the pressure on the left side. Therefore, the change in velocity with respect to time for the boundary node must be zero to satisfy the governing equation

$$\frac{\Delta u_p}{\Delta t} = -\frac{1}{M_{MCV}} \sum_{CV} \mathbf{n}_i P_z^* A = 0$$

SGH: Free Boundary Condition

The free boundary condition for SGH is simulated by using the momentum control volume at the boundary and setting the pressure at the boundary node equal to zero. The rate of change of the velocity at the boundary node is then given by

$$\frac{\Delta u_p}{\Delta t} = -\frac{1}{M_{CV}} A\left[(P^*\mathbf{n})_i + (P^*\mathbf{n})_z\right] = -\frac{1}{M_{CV}} A(P_z^*\mathbf{n})_z$$

PCH Description

Using a point centered approach (PCH), pressure, energy, density, and velocity are stored at the points and each control volume is centered on a point. The Riemann-like solution is used to compute a *P** and *u** at the interfaces between the control volumes as shown in Figure 6.

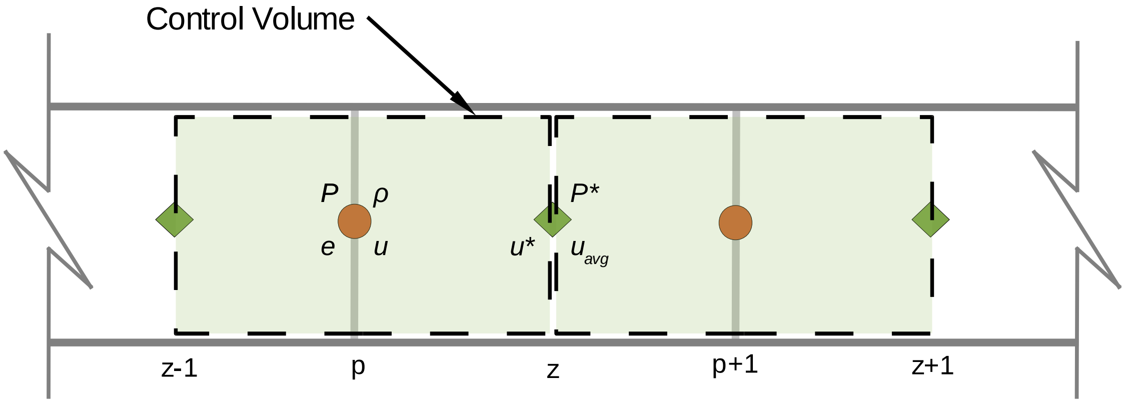


Figure 6. PCH mesh.

In 1D, the governing conservation equations reduce to following discretized forms:

$$\frac{\Delta M_p}{\Delta t} = 0$$

$$\frac{\Delta x_p}{\Delta t} = u_p^{n+\frac{1}{2}}$$

$$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p}\sum_{CV} F^*$$

$$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p}\sum_{CV} F^* u_{avg}$$

where M_p is the point centered mass and

$$F^* = (-\mathbf{n}A\ P^*)_z^{n+\frac{1}{2}}$$

$$u_{avg} = \frac{u_p^{n+\frac{1}{2}} + u_{p+1}^{n+\frac{1}{2}}}{2}$$

The internal energy is computed by subtracting the kinetic energy from the total energy, and the pressure is computed using the equation of state for a gamma-law gas assuming a constant gamma. For PCH, three different methods were used to update the density of each control volume. The governing equations presented in this section represent the first method used. Details regarding the other methods are presented in a later section.

Curvilinear Coordinates

The equations used to compute volumes and areas in PCH are similar to those used in CCH. The differences are that the volume is computed for the point centered control volumes rather than the cell centered control volumes, and that the areas are computed at the nodes rather than at the cell centers.

Boundary Conditions

At the boundaries, a control volume is not centered on a point, as it cannot extent past the domain, as seen below. The governing equations are then applied to a control volume that is typically half the size of an ordinary control volume in the domain. Figure 7 illustrates the PCH mesh at the boundary.

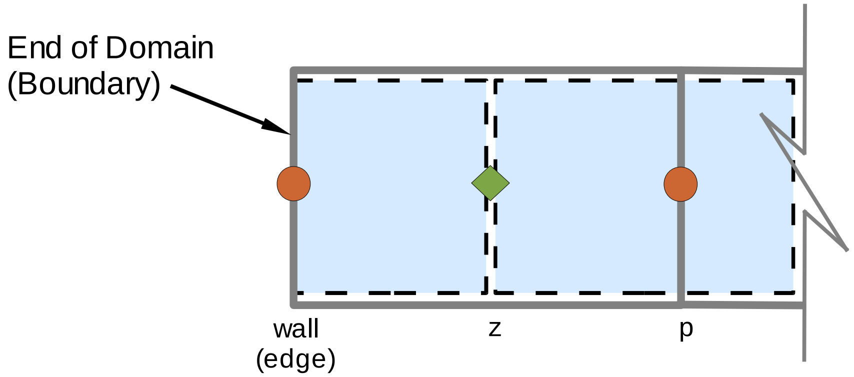


Figure 7. PCH boundary mesh.

PCH: Fixed BC

The fixed boundary is subject to the following conditions:

$$\left.\frac{\Delta x}{\Delta t}\right|_{wall} = u_{wall}$$

$$\left.\frac{\Delta u}{\Delta t}\right|_{wall} = 0$$

$$\left.\frac{\Delta j}{\Delta t}\right|_{wall} = \frac{1}{M_p}\sum_{CV} F^* u = \frac{1}{M_p}\left[(F^* u_{wall})|_{wall} + (F^* u_{avg})|_{z}\right]$$

where M_p is the mass of the control volume at the boundary and

$$u_{avg} = \frac{u_{wall} + u_p}{2}$$

$$F^*|_{wall} = -F^*|_z$$

The wall velocity, *u*$_{wall}$, is fixed. The magnitude of the forces on either side of the boundary control volume are assumed to be equal as the gradient of the pressure is set to zero for this condition.

PCH: Free BC

For a free boundary condition, the edge velocity is not fixed. The free boundary is subject to the following conditions:

$$\left.\frac{\Delta x}{\Delta t}\right|_{edge} = u|_{edge}$$

$$\left.\frac{\Delta u}{\Delta t}\right|_{edge} = \frac{1}{M_p}\sum_{CV} F^* = \frac{1}{M_p}F_z^*$$

$$\left.\frac{\Delta(te)}{\Delta t}\right|_{edge} = \frac{1}{M_p}\sum_{CV} F^* u = \frac{1}{M_p}\ F_z^* u_{avg}$$

where

$$u_{avg} = \frac{u_{edge} + u_p}{2}$$

The three different point centered methods discussed in this report vary slightly in the implementation of boundary conditions. For example, *u*$_{avg}$ in the energy equation above is replaced by the zone-centered Riemann velocity, *u**.

Solver Details

Riemann Solution

As presented in [2], a linear relationship between the shock velocity and the material velocity (the *U*-*u* curve) can be a good approximation for many materials. This simplifies the Riemann solution. As pointed out in [3], the Riemann-like solution can be used to determine a viscous force acting on the material experiencing a shock. The Riemann-like solution solves the shock problem across the interface between two cells. This solution simulates the dissipation effects that characterize a shock. In 1D, the Riemann-like solution yields a velocity and pressure at the cell interface that are given by

$$u^* = \frac{(P_c^- \mathbf{n}^- + \mu^- u_c^-) + (P_c^+ \mathbf{n}^+ + \mu^+ u_c^+)}{(\mu^+ + \mu^-)}$$

$$-P^* \mathbf{n}^+ = \mu^+ (u^* - u_c^+) - P_c^+ \mathbf{n}^+$$

$$-P^* \mathbf{n}^- = \mu^- (u^* - u_c^-) - P_c^- \mathbf{n}^-$$

Where P_c and u_c are the projected values, and μ is the shock impedance. The + and – superscripts denote the positions of the quantities relative to the interface of interest. For simplicity, a first order projection is used for the velocity and pressure. In CCH, these values are projected from the zone center to the points. In SGH, the velocities are projected from the points to the zone center. The shock impedance is determined based on the linear approximation of *U-u* curve. For a gamma law gas, the slope of of the shock impedance relation can be approximated by $\left(\frac{\gamma+1}{2}\right)$ [4]. Therefore, the shock impedance is computed by

$$\mu^+ = \rho c + \rho \left(\frac{\gamma+1}{2}\right) \left\| \left(\frac{u_c^+ + u_c^-}{2} - u_c^+\right) \right\|$$

$$\mu^- = \rho c + \rho \left(\frac{\gamma+1}{2}\right) \left\| \left(\frac{u_c^+ + u_c^-}{2} - u_c^-\right) \right\|$$

where *c* is the acoustic wave speed. For a gamma-law gas, the acoustic wave speed is approximated by

$$c = \sqrt{\gamma \frac{P}{\rho}}$$

Time Integration

The code uses a fourth order, explicit Runge-Kutta time integration scheme. The Taylor series expansion of any function is

$$f^{n+1} = f^n + \Delta t \frac{\delta f}{\delta t} + \frac{\Delta t^2}{2!} \frac{\delta^2 f}{\delta t^2} + \frac{\Delta t^3}{3!} \frac{\delta^3 f}{\delta t^3} + \frac{\Delta t^4}{4!} \frac{\delta^4 f}{\delta t^4} + \dots$$

If the higher order terms are neglected, the above equation can be represented by

$$f^{n+1} \approx f^n + \Delta t \frac{\delta}{\delta t} \left(f^n + \frac{\Delta t}{2} \frac{\delta}{\delta t} \left(f^n + \frac{\Delta t}{3} \frac{\delta}{\delta t} \left(f^n + \frac{\Delta t}{4} \frac{\delta f}{\delta t} \right) \right) \right)$$

The code uses the quantities at “n” to compute a slope and step the quantities forward to “n+1/4”. The code then uses the quantities at “n+1/4” to compute a new slope and step the quantities forward from ”n” to “n+1/3”. This process is repeated until the quantities are stepped forward from “n” to “n+1”. At each of these stages, a Riemann-like problem is solved at the control volume interfaces, and all the simulation parameters are stepped forward in time.

Time Step Control

Two methods are used in order to control the time step during the simulation. The first method is the Courant Stability Condition. The Courant stability condition requires that

$$\frac{u\Delta t}{\Delta x} \leq CFL$$

where *u* is the maximum speed of the fluid in the domain. The *CFL* parameter is chosen by the user. For explicit schemes, *CFL* must be less than or equal to 1. The maximum speed of sound, c_{max}, can be used in the place of the maximum speed of the fluid in the domain. c_{max} is used with the minimum *Δx* of a cell in the domain to calculate a new time step. The code calculates a new time step before each iteration by applying the formula

$$(\Delta t)_{new} = CFL\frac{(\Delta x)_{min}}{c_{max}}$$

The user can set an initial time step in the input file. The code allows the time step to grow by a maximum of 10% at each step.

The second method restricts the volume change of a cell in a single time step. The requirement is expressed by

$$CFLV \geq \frac{V^{n+1} - V^n}{V^n}$$

where *V* is the volume, and *CFLV* is a parameter set by the user. By restricting the volume change, other parameters, such as the density and pressure, are allowed to develop in the cell before the cell collapses and yields unphysical results. This adds stability to the computations. The fundamental formula is the divergence relationship.

$$\nabla \cdot \mathbf{u} = \lim_{V \to 0} \frac{1}{V}\frac{\delta V}{\delta t}$$

In 1D, the finite difference form of this equation can be expressed as

$$\frac{\Delta V}{\Delta t} = \frac{V^{n+1} - V^n}{\Delta t} = \frac{\Delta u}{\Delta x}V^n$$

By rearranging and substituting, the following formula gives a suitable time step.

$$\Delta t = \frac{CFLV}{\left(\left|\frac{\Delta u}{\Delta x}\right|^n\right)_{max}}$$

The hydrocode uses a CFLV parameter set by the user and finds the maximum velocity gradient in the domain. A new time step is then calculated using these values.

The time step used in the next calculation is the minimum of the time steps computed using these two methods.

PCH Method Comparison

The PCH method that was implemented in the code had difficulties on test problems with strong shocks, such as the Sod problem. As a results, a variety of PCH methods were explored. The first method is the canonical PCH method (PCH). In 1D, it was found that this method allowed the collapse of zones at shock discontinuities. The collapse of a cell causes the time step to approach zero and prevents the simulation from reaching the required time. It was found that by “smoothing” the shock interface initially eliminates the cell collapse issues. The smoothing was accomplished by averaging the thermodynamic quantities at the discontinuity. A second method seeks to solve the issue by updating the volume of each control volume using the averaged velocities at the control volume boundaries. This method is denoted by PCHA. This method allows the simulation to complete but does not update the nodal positions directly from the nodal velocities. Rather, the method updates the control volume boundaries and moves the nodal positions based on the new control volume boundaries. A third method uses the Riemann velocities at the control volume boundaries to calculate the density and total energy change. This method is denoted by MPC. This method is effectively identical to the CCH method, only shifted by half of a cell. The equations for each method are presented below.

PCH

$$\frac{\Delta M_p}{\Delta t} = 0$$

$$\frac{\Delta x_p}{\Delta t} = u_p^k$$

$$V_p^{k+1} = \frac{x_{p+1}^{k+1} - x_{p-1}^{k+1}}{2}$$

$$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$$

$$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} \left(F^* u_{avg}\right)\Big|_z^k$$

PCHA

$$\frac{\Delta M_p}{\Delta t} = 0$$

$$\frac{\Delta x_z}{\Delta t} = u_{avg}\Big|_z^k$$

$$\frac{\Delta V_p}{\Delta t} = \sum_{CV} \mathbf{n} \cdot \left(u_{avg}\right)\Big|_z^k$$

$$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$$

$$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} \left(F^* u_{avg}\right)\Big|_z^k$$

MPC

$$\frac{\Delta M_p}{\Delta t} = 0$$

$$\frac{\Delta x_z}{\Delta t} = u^*\Big|_z^k$$

$$\frac{\Delta V_p}{\Delta t} = \sum_{CV} \mathbf{n} \cdot u^*\Big|_z^k$$

$$\frac{\Delta u_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F^*$$

$$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} \left(F^* u^*\right)\Big|_z^k$$

where the area used to compute the volume is understood to be equal to 1 in planar coordinates. u_{avg} is the average of the neighboring nodal velocities. The procedures used to compute the volume change in PCH and PCHA are algebraically equivalent. The difference is that, in PCHA, the control volume boundary locations are updated using the averaged nodal velocities. These values are then used to update the nodal positions after the time integration is complete. In PCH, the nodal positions are updated throughout the time integration using the nodal velocities.

Smoothing

In the Sod problem, the PCH method moves the individual points too quickly and causes the cell ahead of the shock discontinuity to collapse. This causes the time step to approach zero because the maximum allowable time step is based on the distance between individual points. One strategy to prevent this from happening is to smooth the shock discontinuity. This technique smooths out the shock interface by placing a point directly between the high and low density regions of Sod problem and assigns to the point the average of the two densities. The same technique is applied to the initial pressure and energy distributions. Figure 8 illustrates this technique.

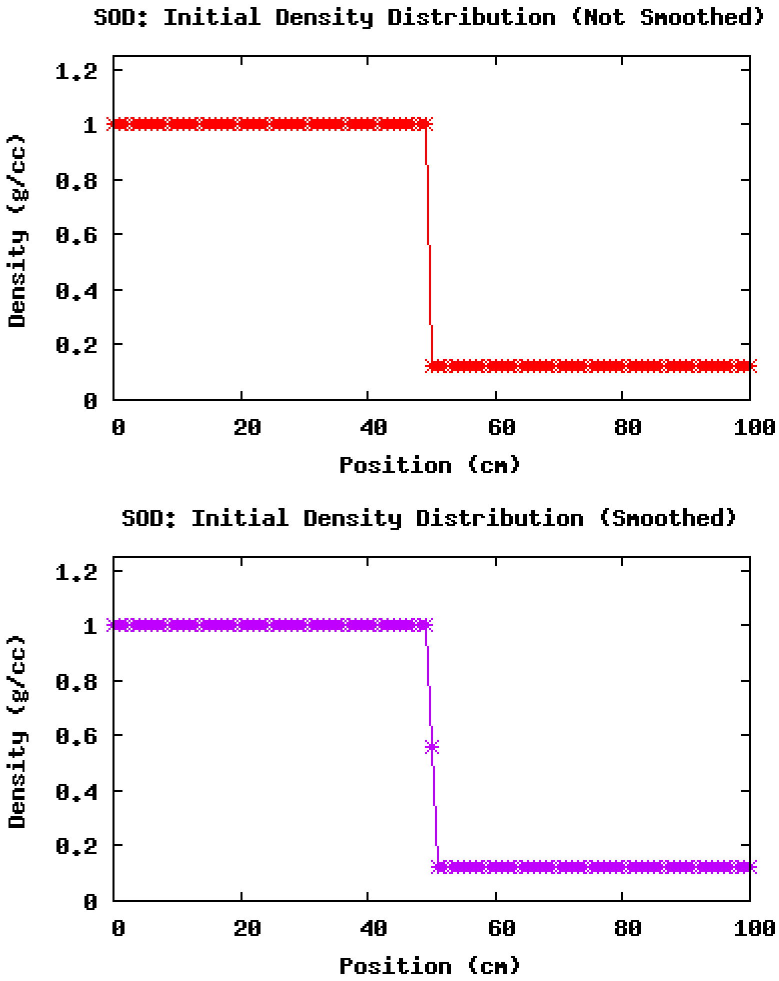


Figure 8. PCH: Smoothing technique concept.

This technique allows the simulation to run past 20 µs. The simulation results at 20 µs are compared to the analytical solution in Figure 9.

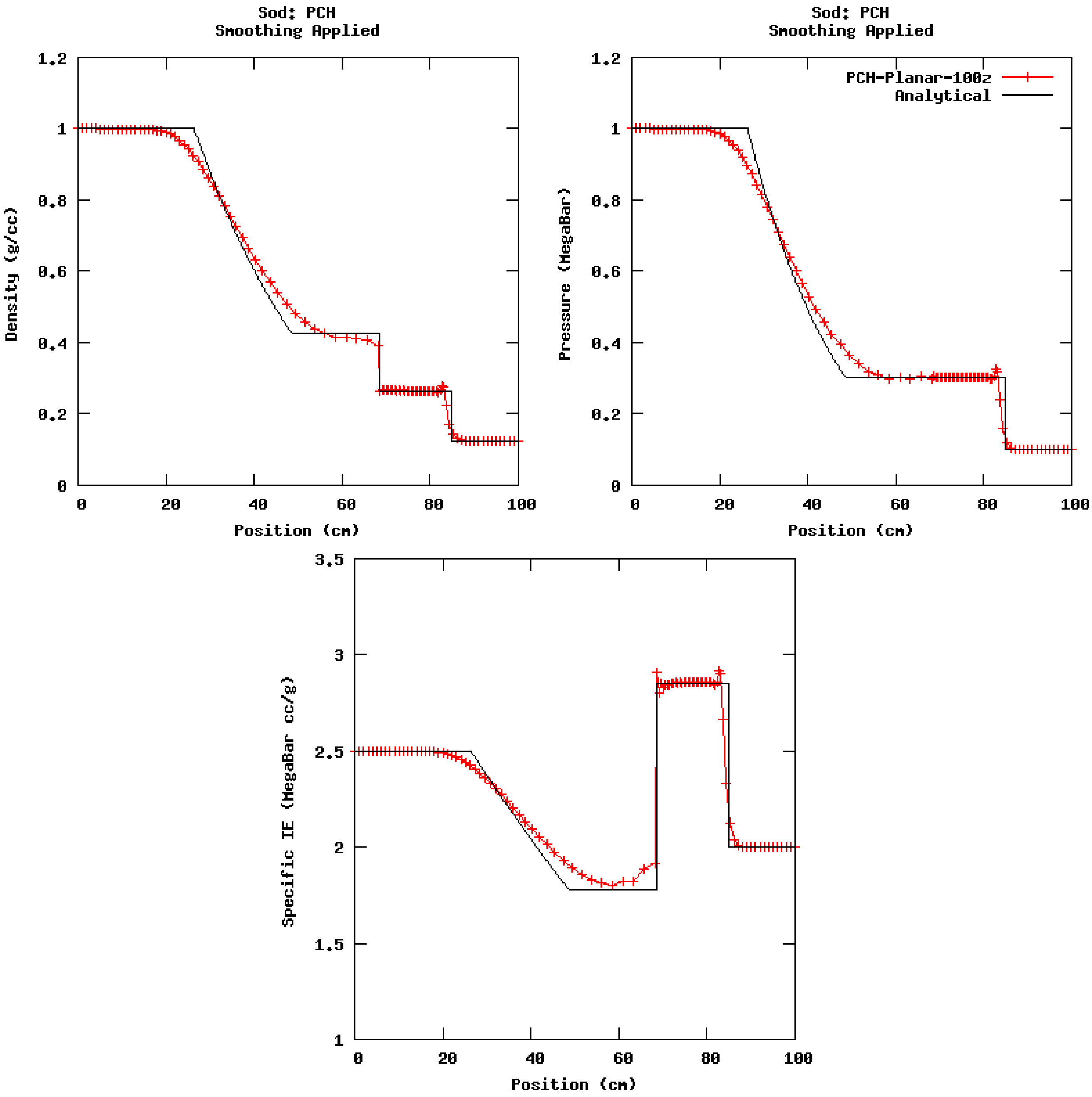


Figure 9. PCH Results for the Sod problem at 20 μs with smoothing applied.

The smoothing technique allows the simulation to run up to 41.72 μs. The results for pressure at this time are shown in Figure 10.

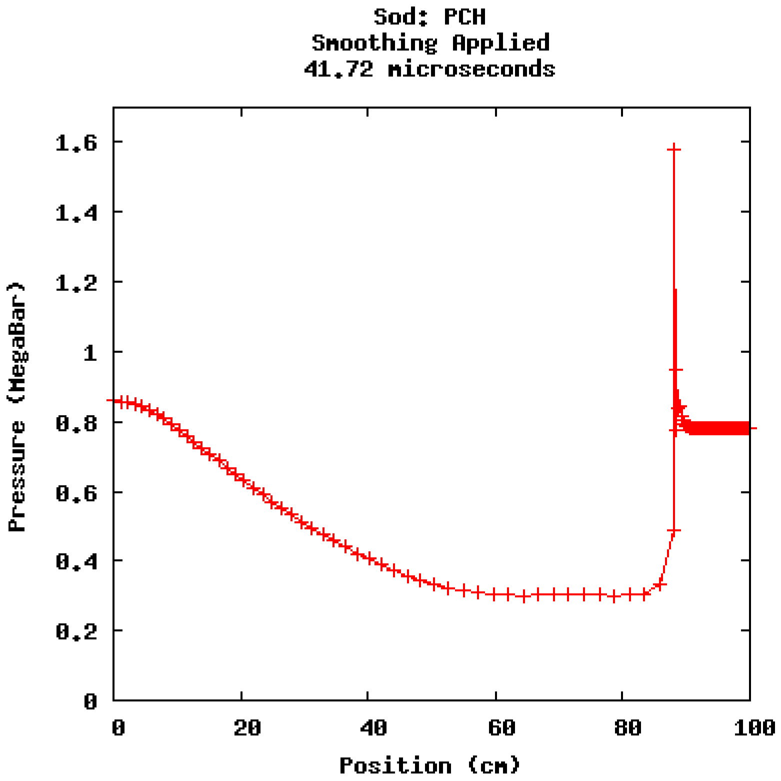


Figure 10. Maximum extent of PCH simulation with smoothing applied for the Sod problem.

At 41.72 μs, the shock has been reflected off of the boundary at 100 cm and has begun to move to the left. A new shock interface has formed, observed in the figure above at approximately 90 cm. This new interface has not been smoothed since no algorithm to automatically smooth shocks has been implemented. Therefore, the points move too close together and the time step approaches zero which stops the simulation.

Space and Time Averaged

For the PCHA method, the control volume boundaries are moved using the averaged nodal velocities. Two different approaches were investigated for this method. The first approach averages the nodal velocities in space, and the second approach averages the nodal velocity in space and time. For the second approach the governing equations for the control volume boundaries, volume, and total energy become

$$\frac{\Delta x_z}{\Delta t} = u_{avg}\Big|_z^{k+\frac{1}{2}}$$

$$\frac{\Delta V_p}{\Delta t} = \sum_{CV} \mathbf{n} \cdot u_{avg}\Big|_z^{k+\frac{1}{2}}$$

$$\frac{\Delta j_p}{\Delta t} = \frac{1}{M_p} \sum_{CV} F_z^* u_{avg}\Big|_z^{k+\frac{1}{2}}$$

where

$$u_{avg}\Big|_z^{k+\frac{1}{2}} = \frac{1}{2}\left(\frac{u_p^n + u_p^{k+1}}{2} + \frac{u_{p+1}^n + u_{p+1}^{k+1}}{2}\right)$$

The different averaging approaches have little to no effect on the results. The results from a simulation using 100 zones and both averaging methods are compared to the analytical solution for the Sod problem at 20 μs in Figure 11.

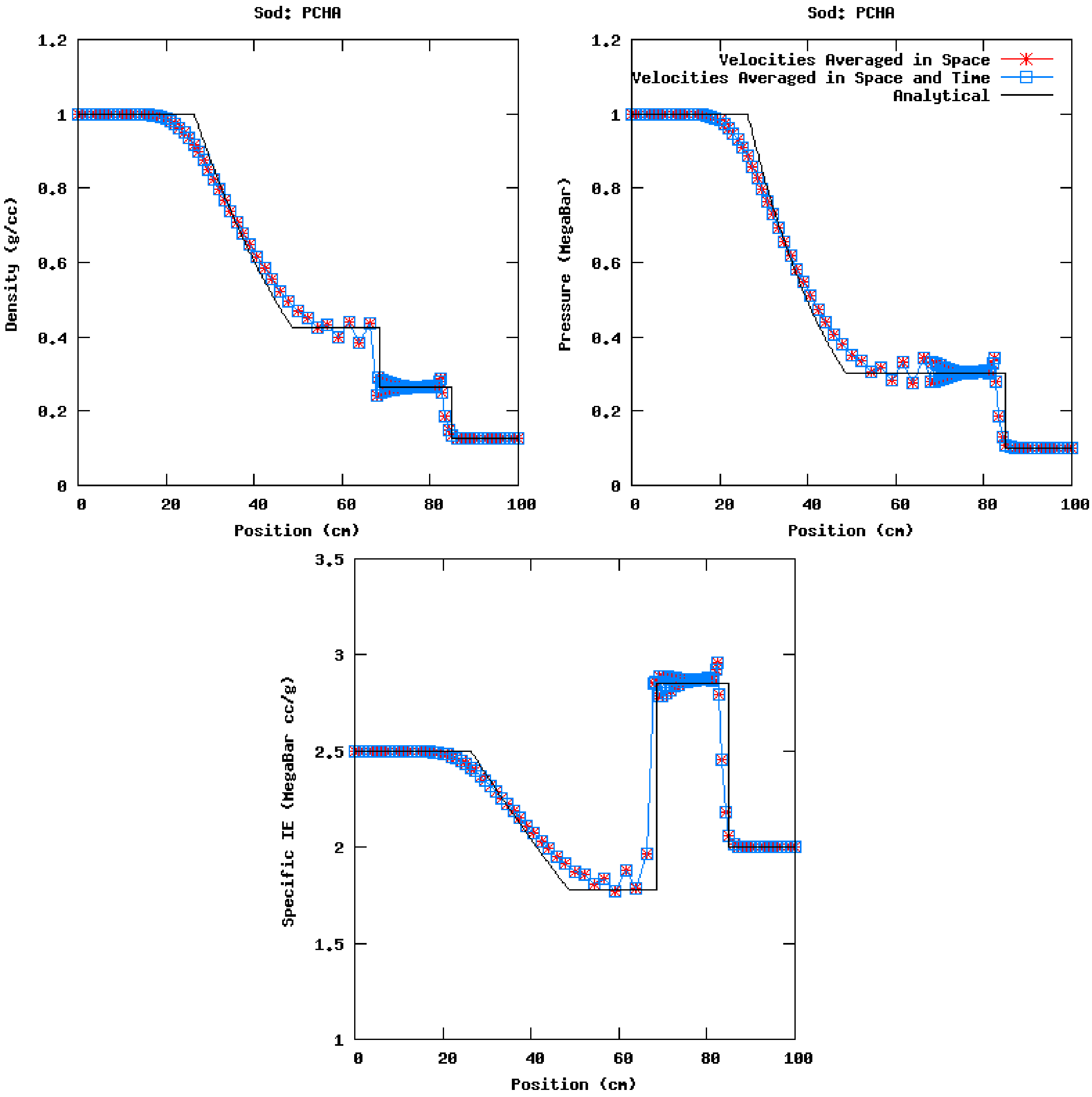


Figure 11. Comparison of results for the Sod problem. The PCHA simulation was run using the time averaged velocities as well as the time and spacial averaged velocities.

The approach used to calculate the average velocity, whether a space or a both space and time averaged velocity, does not change the results of the 1D simulation significantly. Even as the mesh is refined, there is no noticeable difference in the plotted results. For both approaches with 100 zones, a numerical ringing is produced at the contact discontinuity, located between 50 and 75 cm in Figure 11. The PCHA method used in the remainder of this report utilizes spatially averaged velocities. The effect of mesh refinement on the ringing is shown in Figure 12.

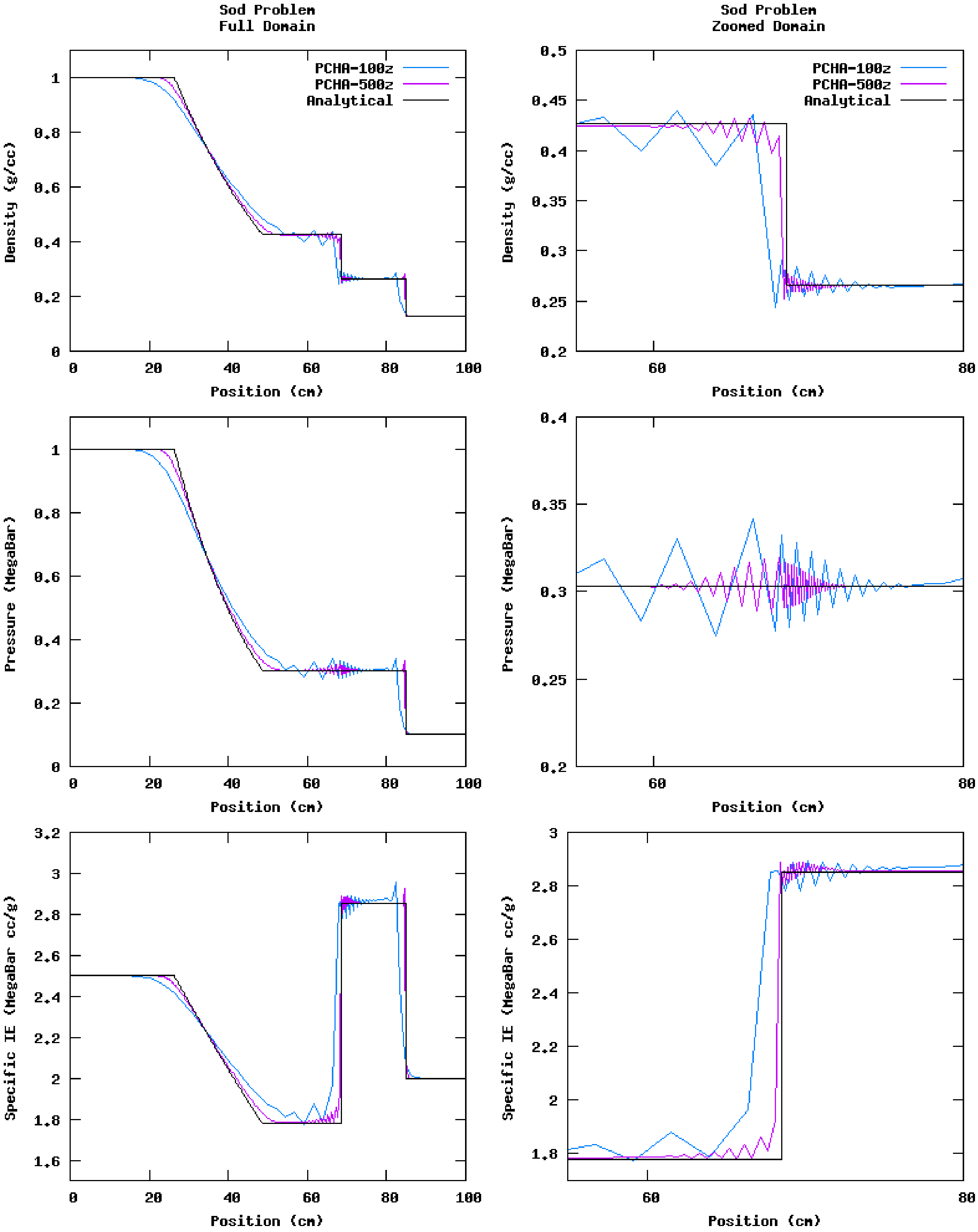


Figure 12. Effect of mesh refinement on the numerical ringing observed in the PCHA method.

PCH Comparison to PCHA

The PCH method experiences cell collapse for the Sod and Sedov test problems. However, for the Piston and Noh test problems, the PCH and PCHA methods produce similar results. These results are shown in Figure 13.

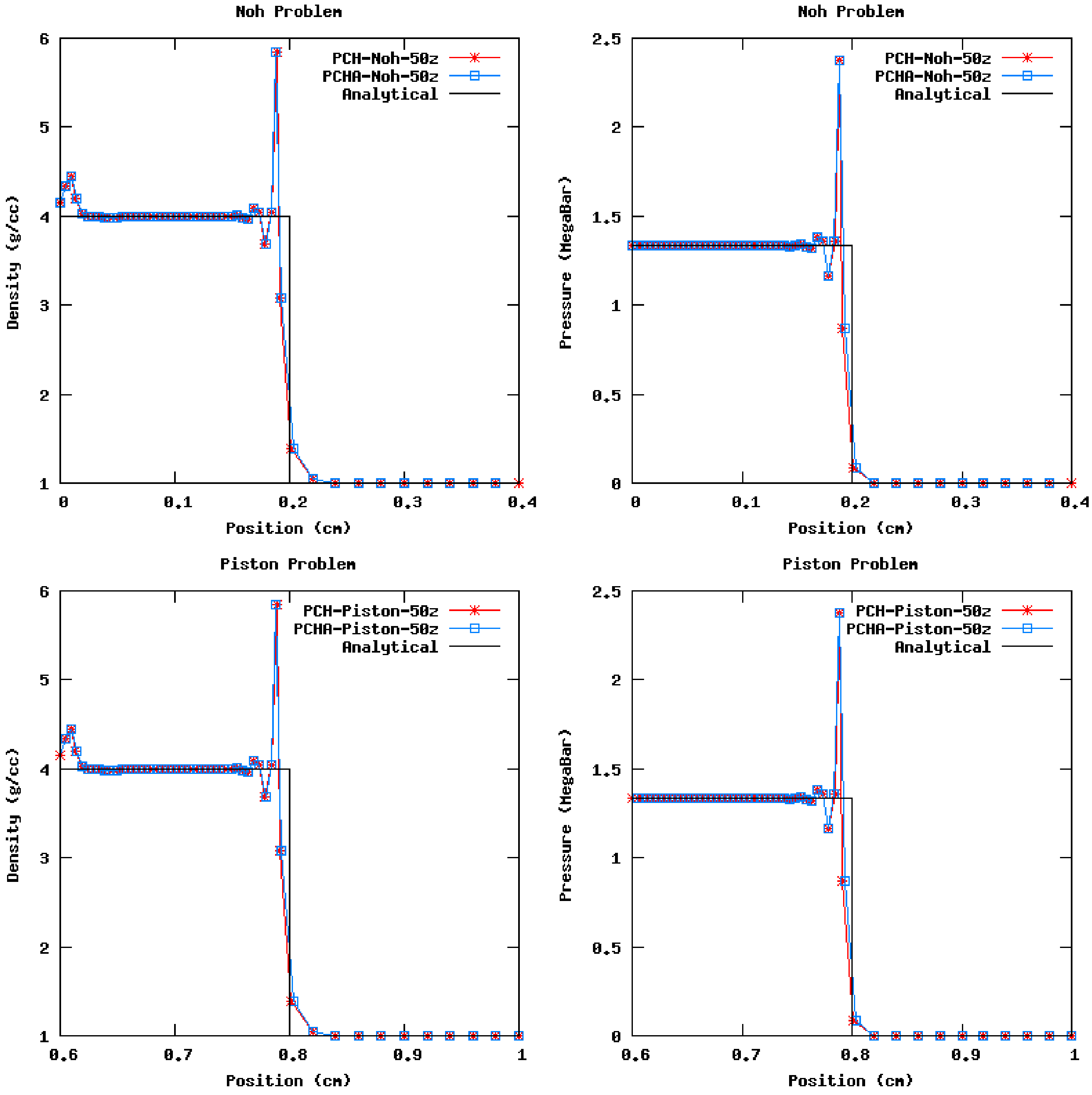


Figure 13. Comparison of the PCH and PCHA methods on the Noh and Piston test problems.

The two methods, PCH and PCHA, yield essentially identical results in the case of the the Noh and Piston test problems.

MPC Comparison to CCH

The MPC method uses an approach very similar to CCH. The results from these two different methods are compared in Figure 14. Both the Sod and the Piston problems were used to compare the two methods.

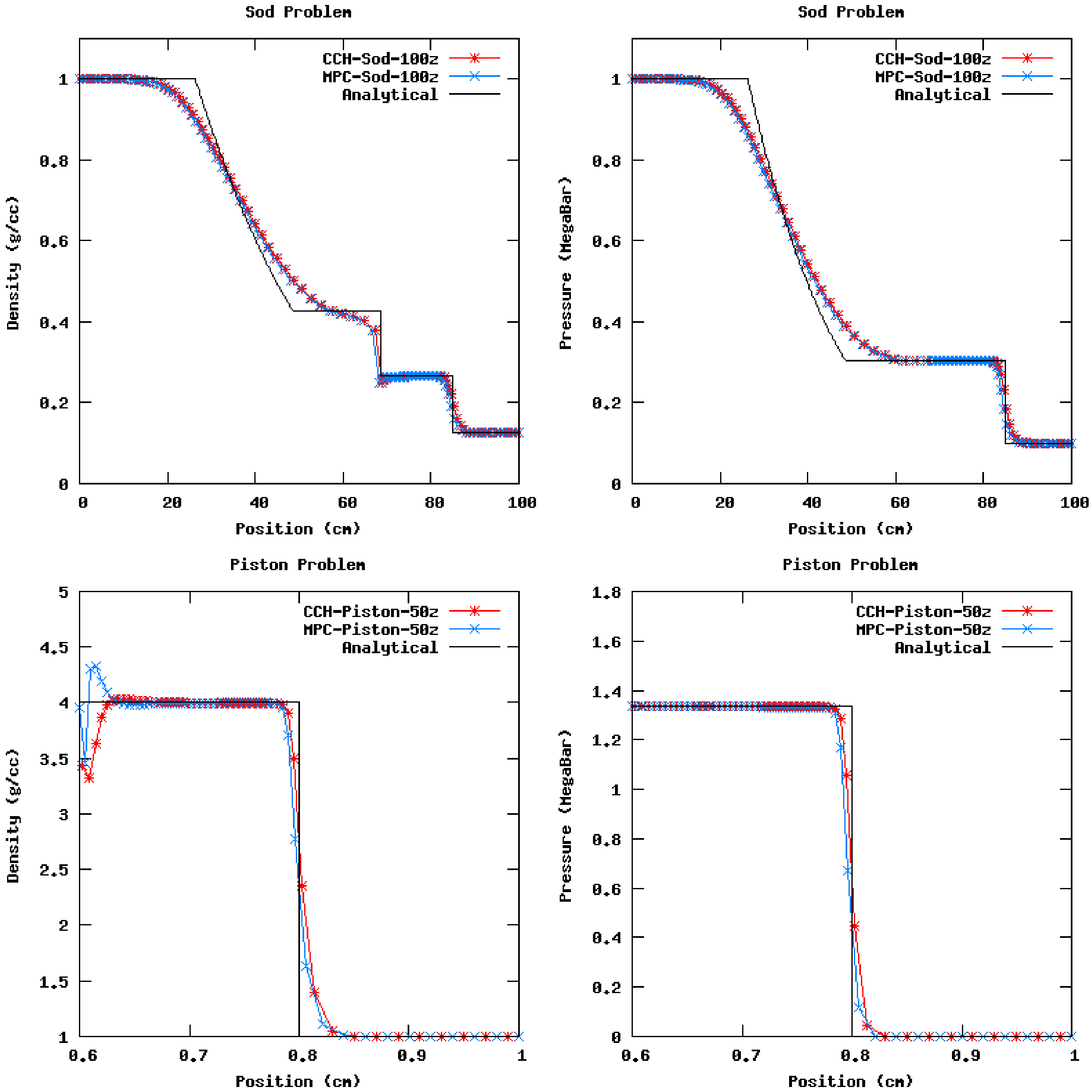


Figure 14. Comparison of the CCH and MPC methods.

The MPC and CCH methods give similar results. The most notable difference is at the fixed boundary for the Piston problem. The differing boundary conditions in MPC and CCH cause considerable differences in the density at this boundary.

Convergence Analysis

The code results from the four methods for the Sod, Piston, and Noh test problems were analyzed at various mesh resolutions to ensure that the calculated results converged to the analytical solutions. Since only first-order methods were used in the 1D hydrocode, near first order convergence is to be expected. For the Piston and Noh test problems, the analytical solutions for pressure, density, and internal energy are piecewise constant functions. For the Sod test problem, portions of the analytical solution are not piecewise constant or linear. For comparison to the hydrocode results, these portions of the analytical solution were resolved using a fixed mesh analytic code [6]. The analytical code produces data points that represent the analytical solution. The results of the hydrocode at various resolutions were mapped to the fixed mesh from the analytical code for comparison. The procedure for this mapping algorithm is as follows.

Since the exact code uses a fixed mesh, the position of any point is given by

$$x_{ex} = i * h$$

where x_{ex} is x-position for the analytical data points, *i* is the index (0, 1, 2, 3,...) and *h* is the constant mesh spacing. The appropriate position for a point mapped from the hydrocode mesh, is then

$$i = \frac{x_{calc}}{h}$$

Since the C language always rounds down in float to integer conversion (i.e. 2.24 and 2.99 both round to 2), this equation always ensures that *i* is the index of the fixed mesh point just below the calculated value, and *i+1* is the index of the analytical data point just above the calculated value. The expected analytical value and the error are then determined from linear interpolation of the analytical data points. A sample of this algorithm for pressure is as follows.

$$m = \frac{P_{exact(i+1)} - P_{exact(i)}}{x_{exact(i+1)} - x_{exact(i)}}$$

$$P_{interp} = P_{exact(i)} + \Delta x * m$$

where

$$\Delta x = x_{calc} - x_{exact(i)}$$

$$\Delta P = |P_{interp} - P_{calc}|$$

Figure 15 illustrates this procedure. This same method can be used for analytical solutions without a functional representation, such as the horizontal segments with discontinuities, but this results in a slope of infinite magnitude, making linear interpolation unnecessary.

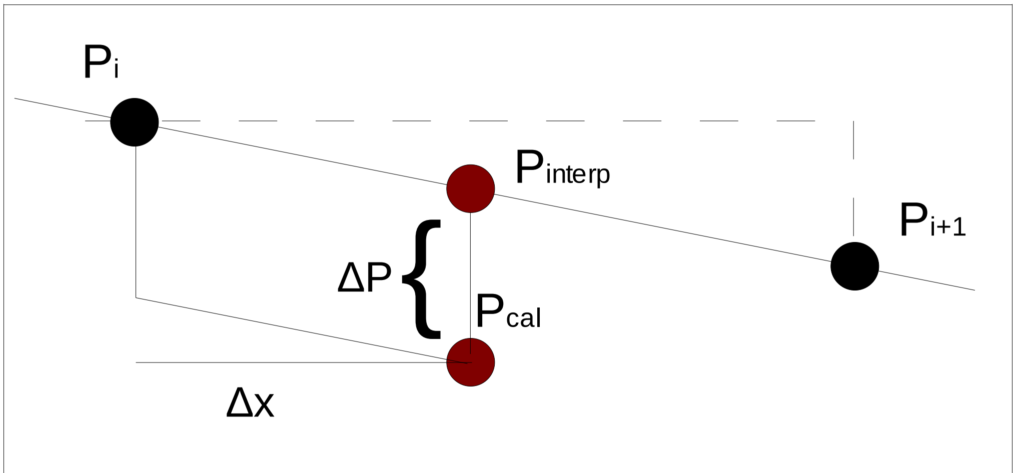


Figure 15. The calculation of error from analytical data points.

Convergence is measured by comparing the volume-weighted combination of all *ΔP* over the domain with the mesh resolution. Two combinations are common, the L^1 and L^2 norms. The volume weighted L^1 norm is given as:

$$||L||^1 = \frac{\sum_i V_i * \Delta P_i}{\sum_i V_i}$$

where V_i is the volume of the Lagrangian element at index *i*. The volume weighted L^2 norm is

$$||L||^2 = \sqrt{\frac{\sum_i (V_i * \Delta P_i)^2}{(\sum_i V_i)^2}}$$

To measure convergence, the rate of decrease in the error with respect to the increase in mesh density is measured. The error is quantified using the L^1 and L^2 norms. The rate decrease of the error should be comparable to the order of the approximations used in the hydrocode. That is, a second order scheme should show a quadratic decrease in error with increasing mesh density. The first order scheme that has been implemented in the 1D hydrocode should show a first order decrease in error with increasing mesh density. Each method is expected to converge according to the following general equation.

$$\epsilon = A\, n^k$$

where ε is the error, A is the convergence coefficient, n is the number of cells, and k is the convergence rate. A power law fit was used to determine the values of A and k that correspond to each method for each test problem given a variety of mesh densities. These results are presented with the corresponding test problem results in the following section.

Test Problems

Sod Problem

The SOD problem simulates the interactions of two materials, one in the first half of the domain, and the other in the second half of the domain. The first material has an initial density of 1 g/cc and internal energy of 2.5 megabar cc/g. The second material has an initial density of 0.125 g/cc and internal energy of 2.25 megabar cc/g. The boundary conditions on both sides of the domain are fixed and the domain is 100 cm in length. The gamma of both materials is set to 1.4. Both materials have an initial velocity of zero and the imaginary barrier between them is removed at t=0. 100 zones were used across the domain to simulate the problem. The results at 20 µs for the four different hydro methods are compared to the analytical solution in Figure 16. SGH and CCH are shown on the left, and MPC and PCHA are shown on the right. The SGH results are slightly more accurate compared the CCH results. The SGH method follows the density discontinuity more accurately than the CCH method at approximately 70 cm. The PCHA method shows numerical ringing about the density discontinuity, while MPC tracks the analytical solution without numerical ringing.

The analytical solution is known [5] and a computer code was used to calculate analytical data points [6]. Convergence studies were carried out for each method. Table 3 shows the results of these convergence studies.

Table 3. Convergence data for the Sod test problem.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^1 Convergence Coefficient
CCH	Pressure	-0.756	0.7821	-1.1207	0.8325
	Density	-0.715	0.5559	-1.0763	0.5727
	Specific Internal Energy	-0.7652	1.8149	1.3086	-1.0401
MPC	Pressure	-0.7589	0.7968	-1.1206	0.8285
	Density	-0.7223	0.5937	-1.0704	0.5581
	Specific Internal Energy	-0.7829	2.1553	-1.0441	1.5312
PCHA	Pressure	-0.844	1.1623	-1.1174	0.7582
	Density	-0.8112	0.8217	-1.0903	0.5526
	Specific Internal Energy	-0.8755	3.17	-1.0642	1.7875
SGH	Pressure	-0.7926	0.6336	-1.1439	0.6956
	Density	-0.7672	0.5212	-1.1025	0.5006
	Specific Internal Energy	-0.8122	1.6637	-1.0348	1.0958

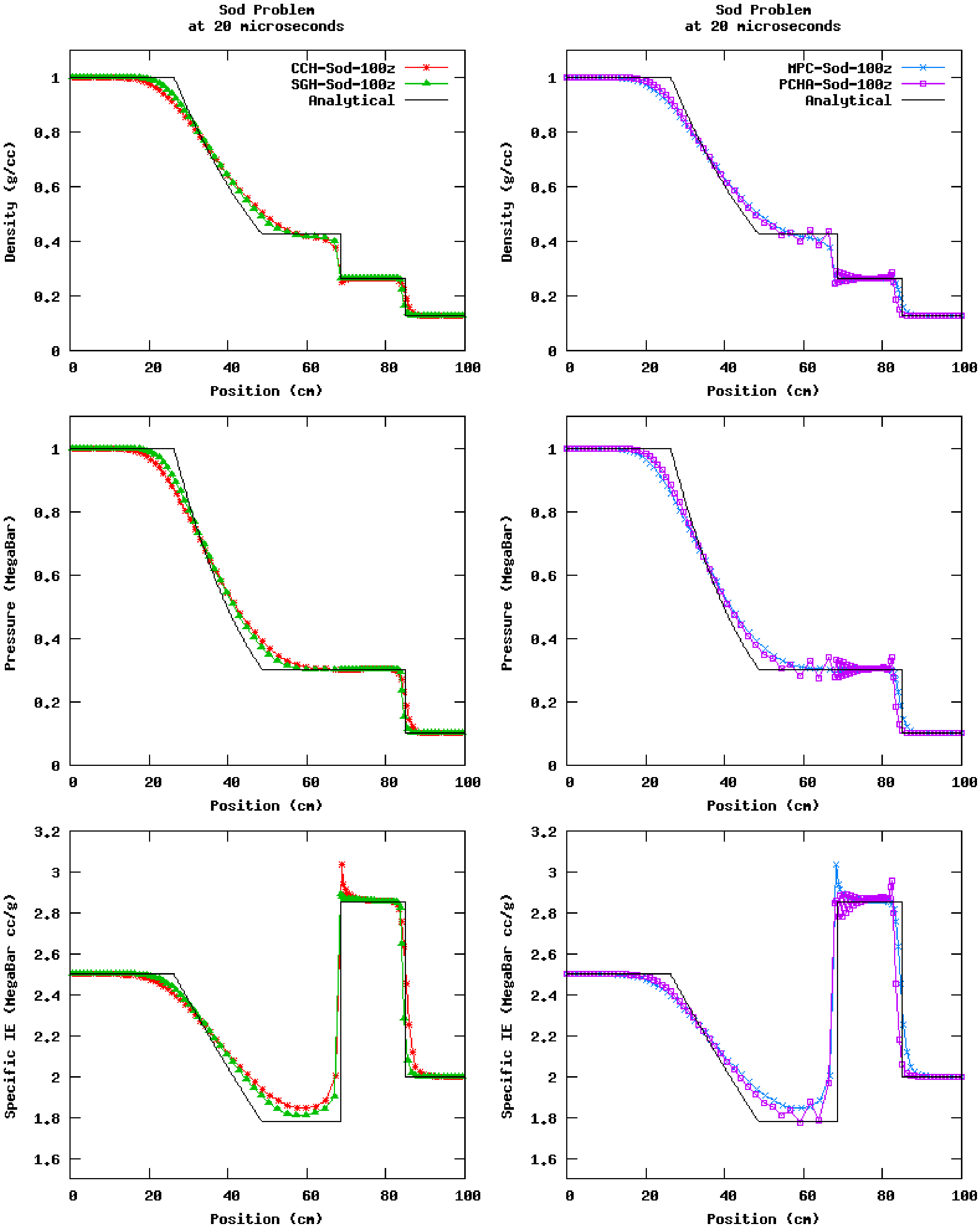


Figure 16. Comparison of the results from four methods for the Sod test problem.

Piston Problem

The piston problem consists of one material across the domain. The initial velocity, initial pressure, and initial internal energy of the material is zero. The initial density of the material is 1 g/cc. The boundary condition on the right is fixed with zero velocity. The boundary condition on the left is fixed with a velocity of 1 cm/μs. The domain is 1 cm in length and 50 zones were used to simulate the problem. The gamma of the material was set to 5/3. The results after 0.6 μs for the four hydro methods are shown in Figure 17. SGH and CCH are shown on the left, and MPC and PCH are shown in the right. The analytical solution to this problem is known. Table 4 shows the results of the convergence studies for this test problem.

Table 4. Convergence data for the Piston test problem.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-1.0145	0.4056	-1.0065	0.2534
	Density	-1.0015	1.8994	-1.0065	0.8630
	Specific Internal Energy	-1.0011	0.2905	-1.0085	0.1621
MPC	Pressure	-1.0120	0.4494	-1.0051	0.2966
	Density	-0.9995	1.6330	-1.0058	0.7468
	Specific Internal Energy	-0.9978	0.2024	-1.0054	0.1040
PCH	Pressure	-1.0101	0.5734	-1.0109	0.3007
	Density	-1.0021	1.5545	-1.0101	0.6628
	Specific Internal Energy	-0.9935	0.1702	-0.9999	0.0876
SGH	Pressure	-1.0018	0.3192	-1.0003	0.2827
	Density	-0.9962	0.9938	-1.0026	0.5665
	Specific Internal Energy	-0.9978	0.1068	-0.9934	0.0655

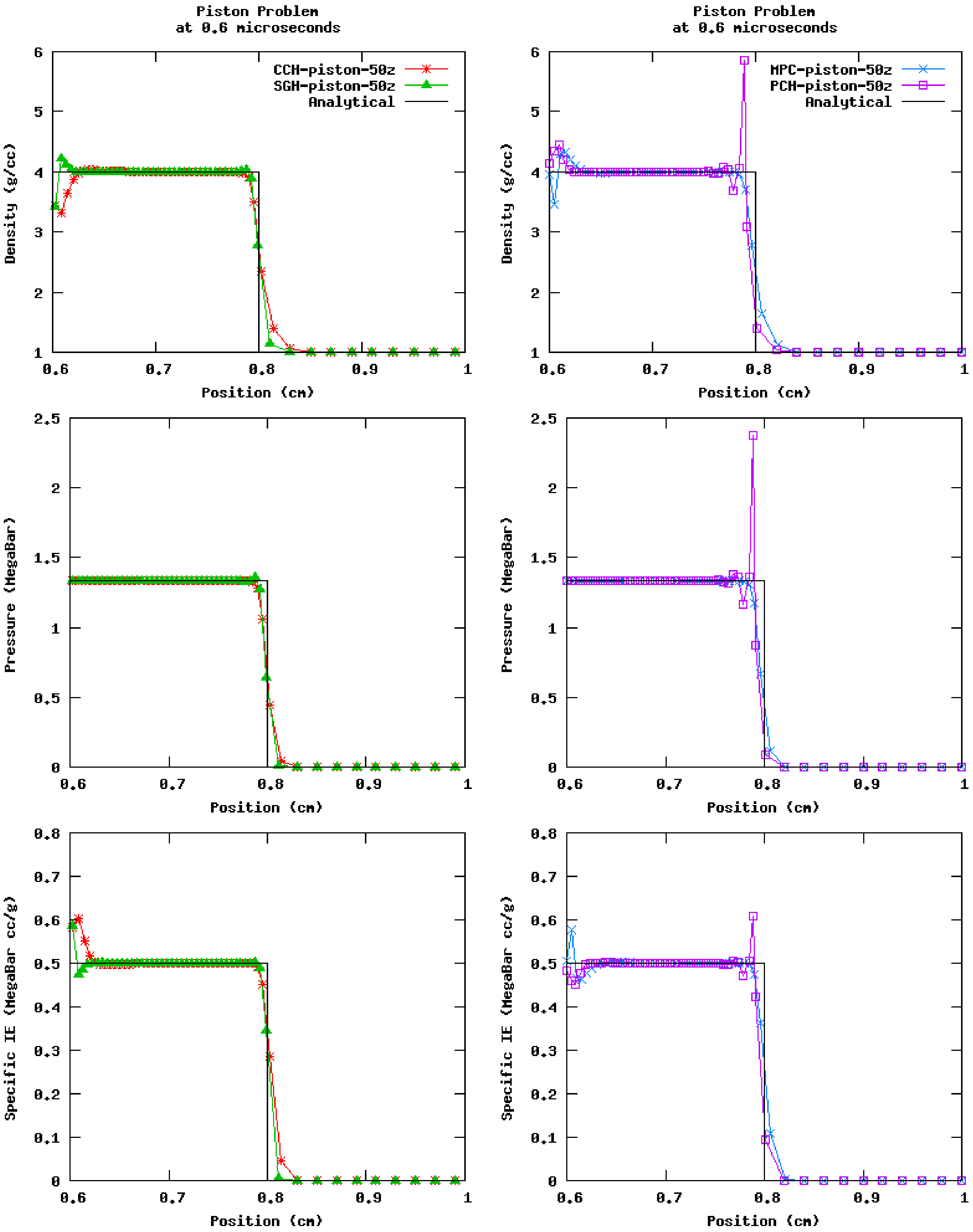


Figure 17. Comparison of results from four methods for the Piston test problem.

Noh

The Noh problem involves a single material moving at -1 cm/µs and colliding with a fixed boundary at the origin. The domain for this simulation was set to 1 cm and 50 zones were used to simulate the problem. The boundary at the origin is fixed and the boundary at the end is moving at -1 cm/µs with the fluid. The results are for a 1D simulation in planar, cylindrical, and spherical coordinates with gamma equal to 5/3. The analytical solution is known [7].

Planar Coordinates

The results for planar coordinates are shown in Figure 18. Convergence data is recorded in Table 5.

Table 5. Convergence data for the Noh test problem in planar coordinates.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-1.0145	0.4056	-1.0065	0.2534
	Density	-1.0015	1.8994	-1.0065	0.8630
	Specific Internal Energy	-1.0011	0.2905	-1.0085	0.1621
MPC	Pressure	-1.0096	0.4392	-1.0065	0.2947
	Density	-0.9983	1.6114	-1.0054	0.7394
	Specific Internal Energy	-0.9980	0.2027	-1.0061	0.1047
PCH	Pressure	-1.0028	0.5569	-1.0049	0.2941
	Density	-0.9975	1.5221	-1.0040	0.6450
	Specific Internal Energy	-0.9938	0.1707	-0.9988	0.0863
SGH	Pressure	-1.0018	0.3192	-1.0003	0.2827
	Density	-0.9962	0.9938	-1.0026	0.5665
	Specific Internal Energy	-0.9954	0.1128	-1.0003	0.0699

Cylindrical Coordinates

The results for cylindrical coordinates are shown in Figure 19. The results of the convergence studies for the Noh problem in cylindrical coordinates are shown in Table 6.

Table 6. Convergence data for the Noh test problem in cylindrical coordinates.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-1.0068	5.7774	-1.0098	2.1002
	Specific Internal Energy	-0.8970	0.6561	-1.0037	0.3299
MPC	Pressure	-1.0169	7.0382	-0.9940	2.5974
	Specific Internal Energy	-0.88	0.4887	-1.0084	0.2416
PCHA	Pressure	-0.9901	5.8003	-0.9903	3.1239
	Specific Internal Energy	-0.8482	0.5744	-0.9986	0.2265
SGH	Pressure	-0.9888	16.9188	-0.9235	6.4584
	Specific Internal Energy	-0.7903	1.4122	-0.9314	0.7030

Spherical Coordinates

The results for spherical coordinates are shown in Figure 20. The results of the convergence studies are shown in Table 7.

Table 7. Convergence data for the Noh test problem in spherical coordinates.

Method	Parameter	L^1 Convergence Rate	L^1 Convergence Coefficient	L^2 Convergence Rate	L^2 Convergence Coefficient
CCH	Pressure	-0.9721	31.4138	-1.0191	9.3409
	Specific Internal Energy	-0.8611	0.7734	-1.0023	0.3596
MPC	Pressure	0.0477	5.3956	-0.4547	6.1347
	Specific Internal Energy	-0.8986	2.8903	-0.9886	1.4213
PCHA	Pressure	0.0516	5.2193	-0.4507	5.9481
	Specific Internal Energy	-0.8696	2.3708	-0.9904	1.0447
SGH	Pressure	-1.0074	99.1013	-0.9668	32.2163
	Specific Internal Energy	-0.7842	2.3395	-0.9812	1.3969

It is interesting to note that for spherical coordinates, the convergence rate for the point centered methods (MPC and PCHA) are positive for pressure in the L^1 norm. This is a reason to further investigate curvilinear coordinates for all the hydro methods presented. The results for a mesh resolution of 10,000 zones is shown in Figure 21. It is clear that the MPC and PCHA methods converge to a value for pressure near the wall that is less than the analytical value.

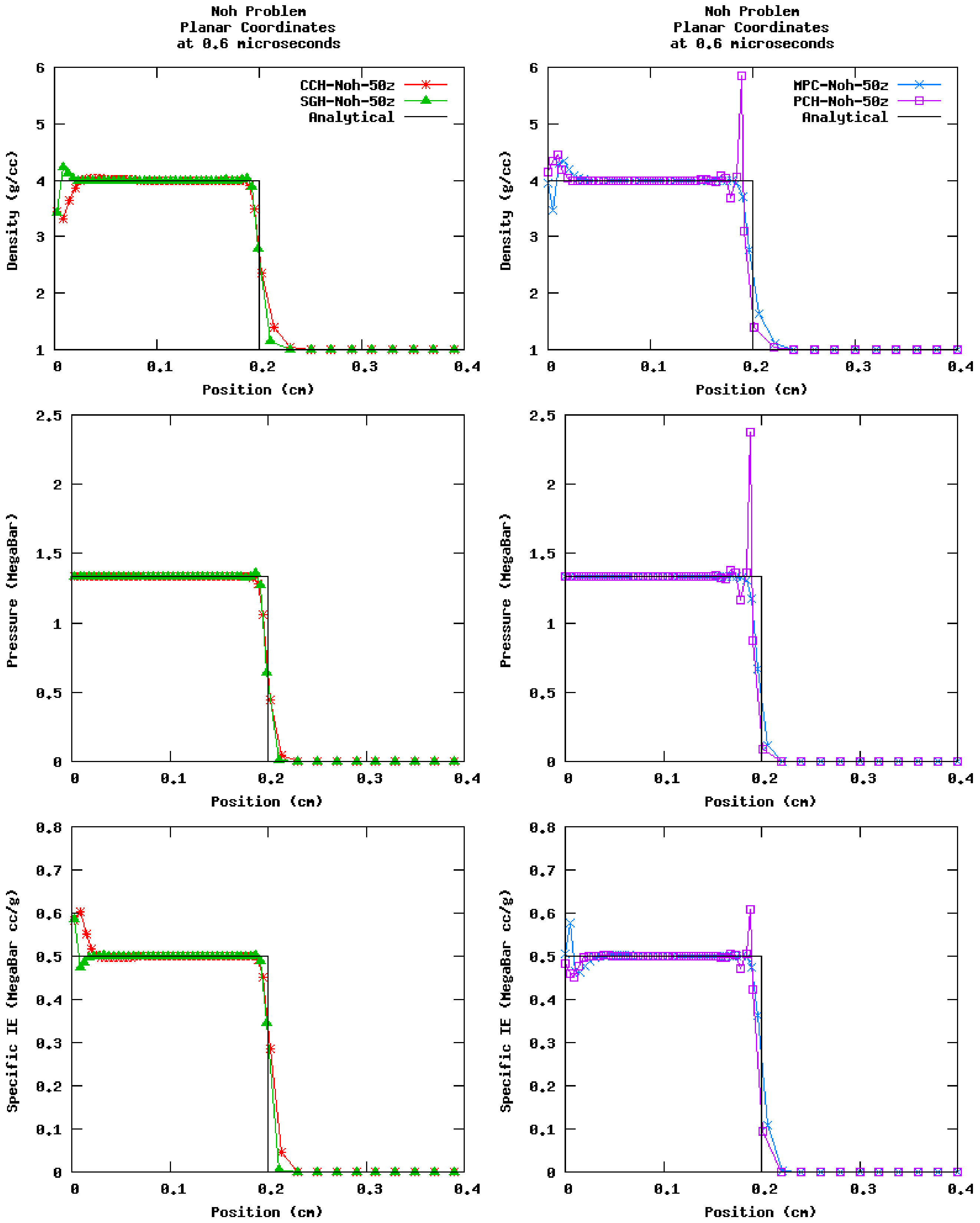


Figure 18. Comparison of results from four methods for the Noh test problem in planar coordinates.

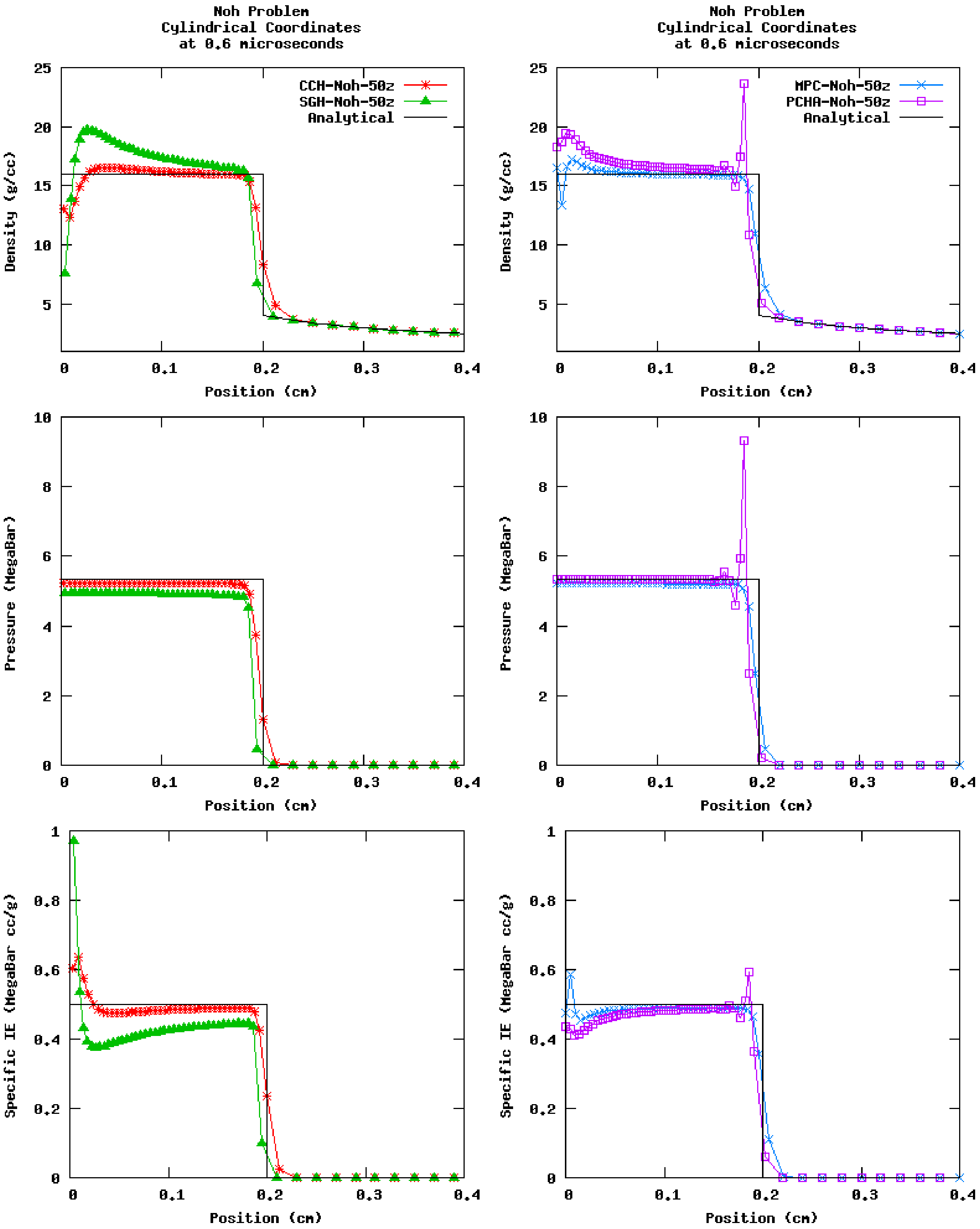


Figure 19. Comparison of results from four methods for the Noh test problem in cylindrical coordinates.

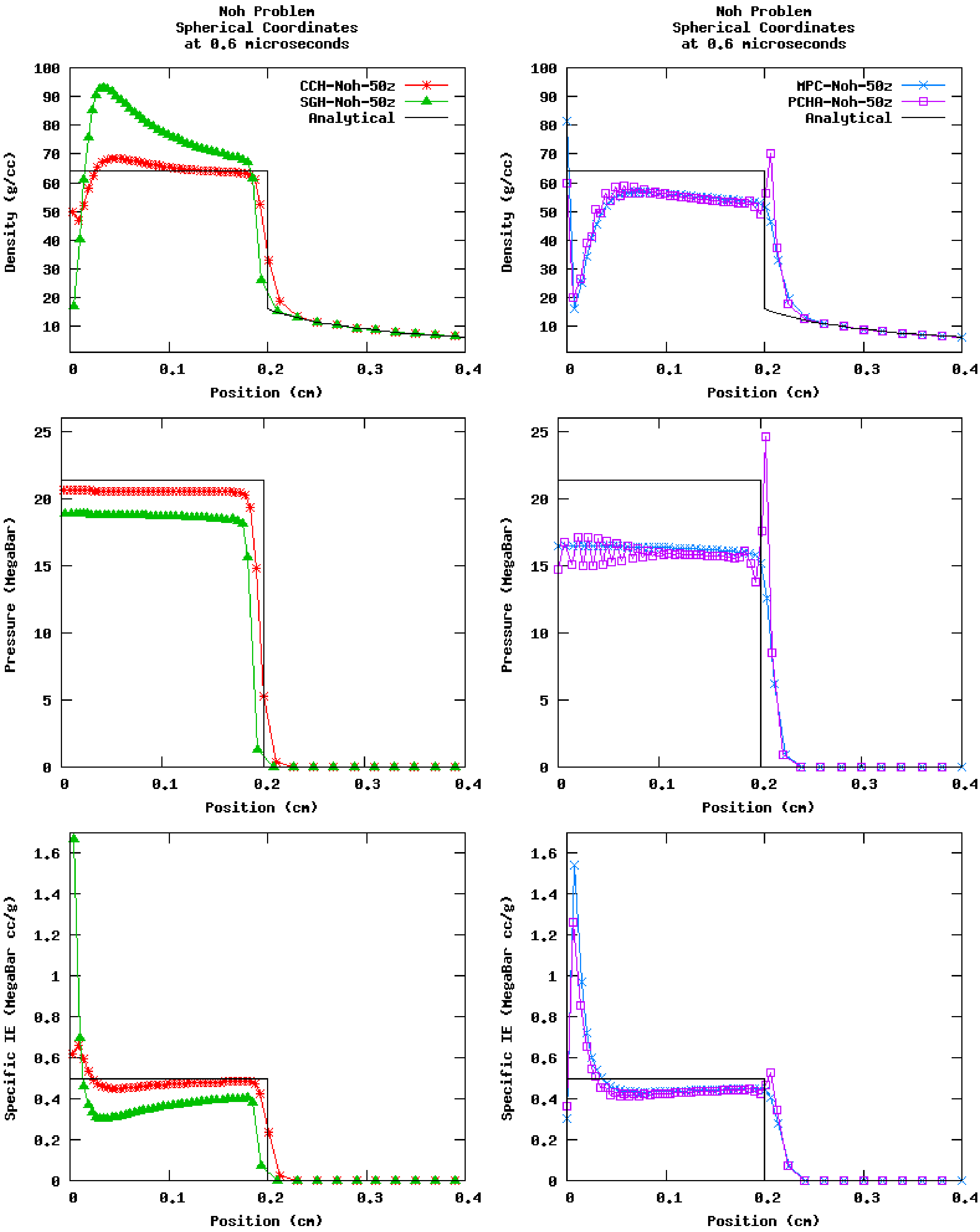


Figure 20. Comparison of results from four methods for the Noh test problem in spherical coordinates.

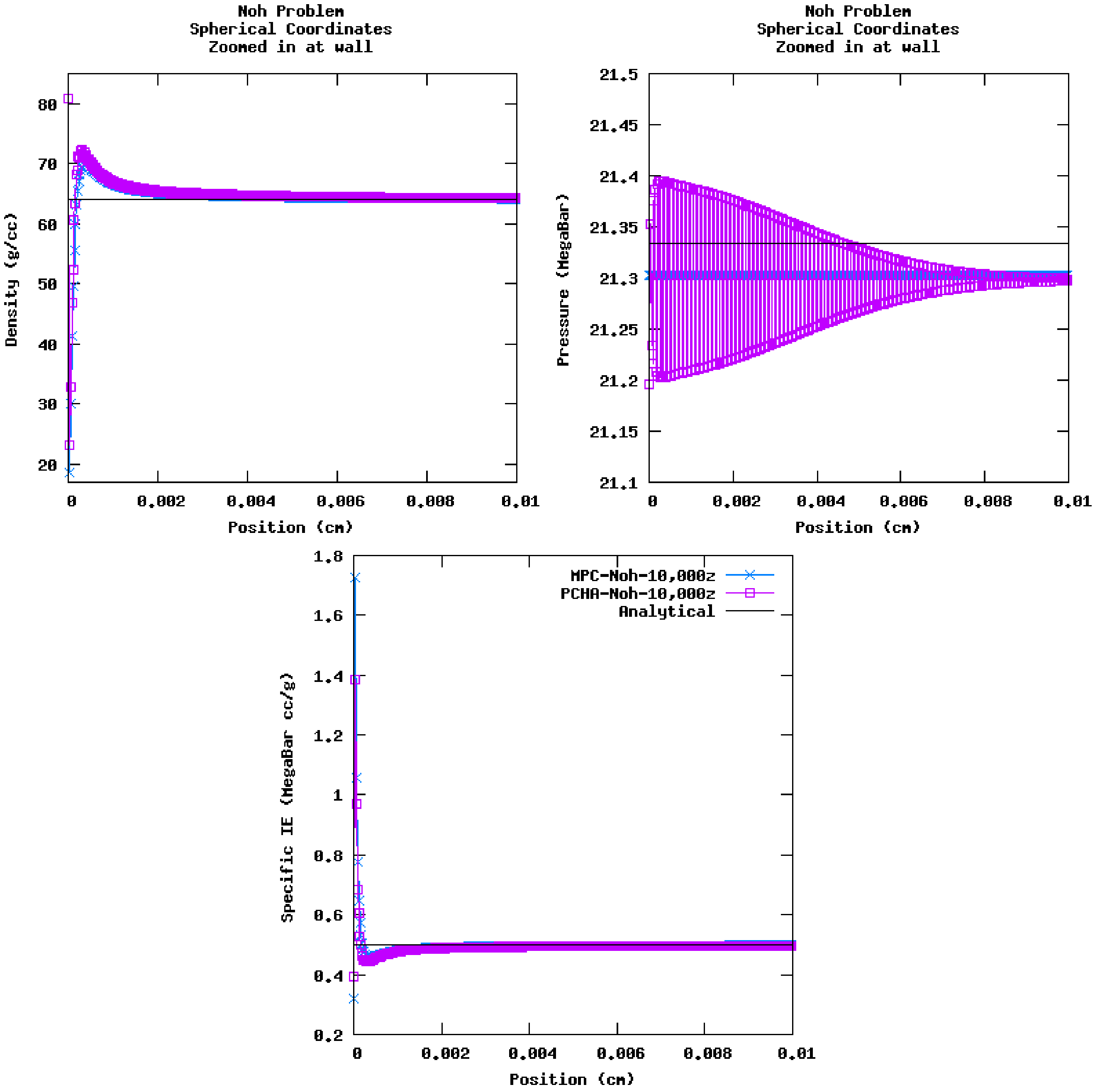


Figure 21. MPC and PCHA results for the Sedov test problem, zoomed at the wall.

The PCHA method produces numerical oscillations at the wall. Both the MPC and PCHA methods converge to a lower pressure near the wall than the analytical solution demands.

Sedov

The Sedov problem involves a blast propagating from the origin. The simulation is initialized with a specified amount of internal energy deposited in the cell at the origin. Once the simulation is started, a shock wave moves away from the origin. The domain for this simulation was set to 1.2 cm and 60 zones were used to simulate the problem. The gamma of the gas was set to 5/3 across the domain. The density is initially set to 1 g/cc across the domain, and the pressure and internal energy are set initially to zero everywhere except at the origin. The analytical solution is known [8] and the analytical data points for each simulation were computed using an analytical code [9]. The simulation was run in planar coordinates. In cylindrical and spherical coordinates, the simulation produced inconsistent results. Therefore, these results are not shown. Further investigation into the methods used in the 1D hydrocode for cylindrical and spherical coordinates is recommended.

Planar Coordinates

For planar coordinates, the amount of extensive internal energy deposited in the cell at the origin was 0.3 megabar cc. With this amount of energy, the exact form of the shock is located at approximately 1 cm after 1μs. The extensive internal energy was divided by the mass of the cell at the origin and applied to the cell at the origin as specific internal energy. The volume of the cell at the origin is initially 0.02 cc, therefore, the amount of specific internal energy deposited in the cell at the origin is 15 megabar cc/g. The results for planar coordinates are shown in Figure 22. SGH and CCH are shown on the left, and MPC and PCHA are shown together on the right. The PCH method experiences cell collapse in the Sedov test problem and the simulation stalls. Therefore no results were obtained from the PCH method for the Sedov test problem.

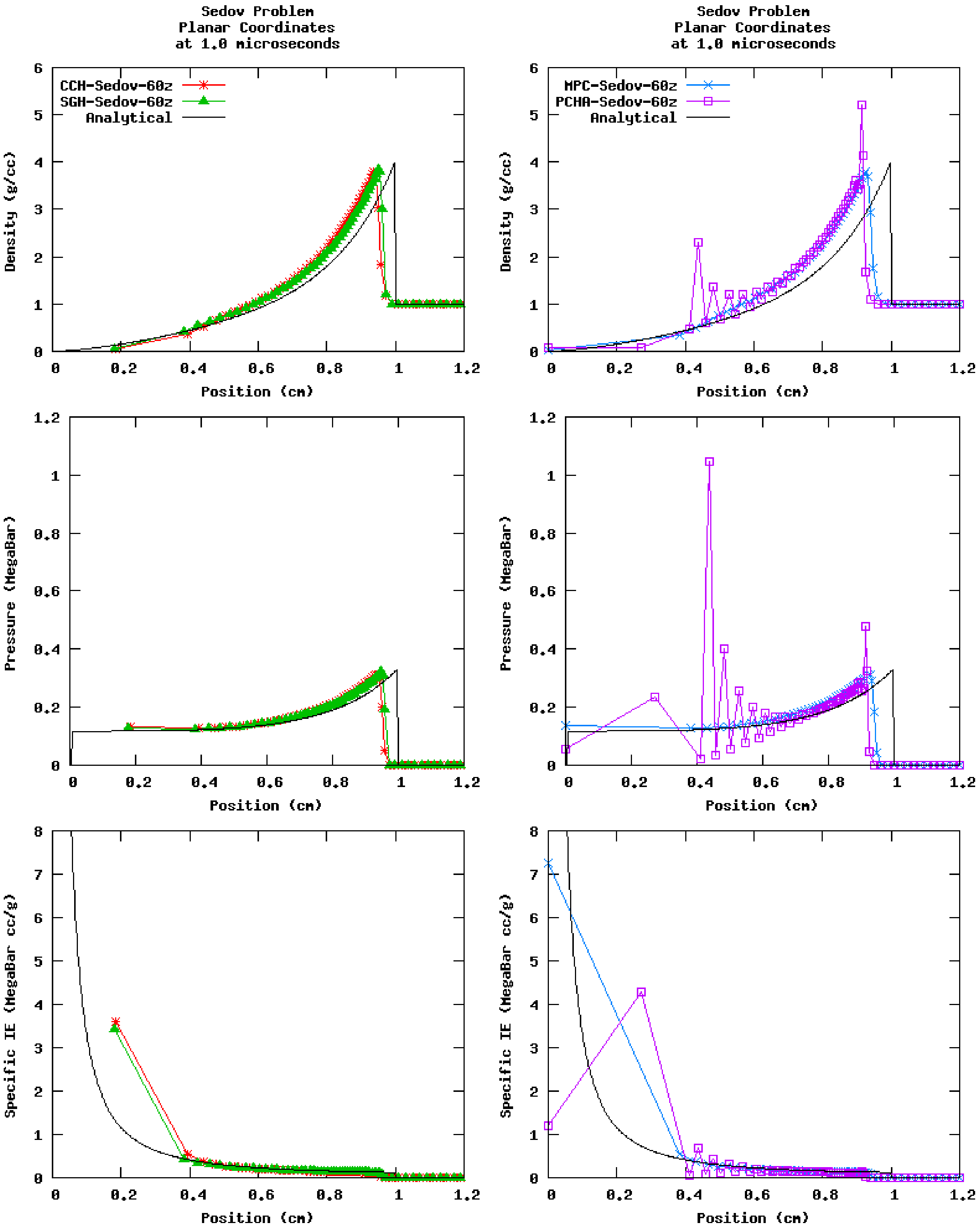


Figure 22. Comparison of results from four methods for the Sedov test problem in planar coordinates.

The results for the PCHA method for the Sedov test problem clearly demonstrate the numerical ringing associated with this method. Qualitative convergence studies were performed for the Sedov problem in planar coordinates. The results are shown in Figure 23. The CCH and SGH methods converge to the analytical solution as expected. The MPC method also converges. However, the numerical ringing in the PCHA method produces errors of greater magnitude in pressure with a mesh of 300 zones compared to the mesh of 60 zones in Figure 22. It is also interesting to note that the PCHA method does not converge to the analytical solution as the mesh is refined. The shock is still slow, as predicted with a mesh of 60 zones, even with a mesh of 300 zones.

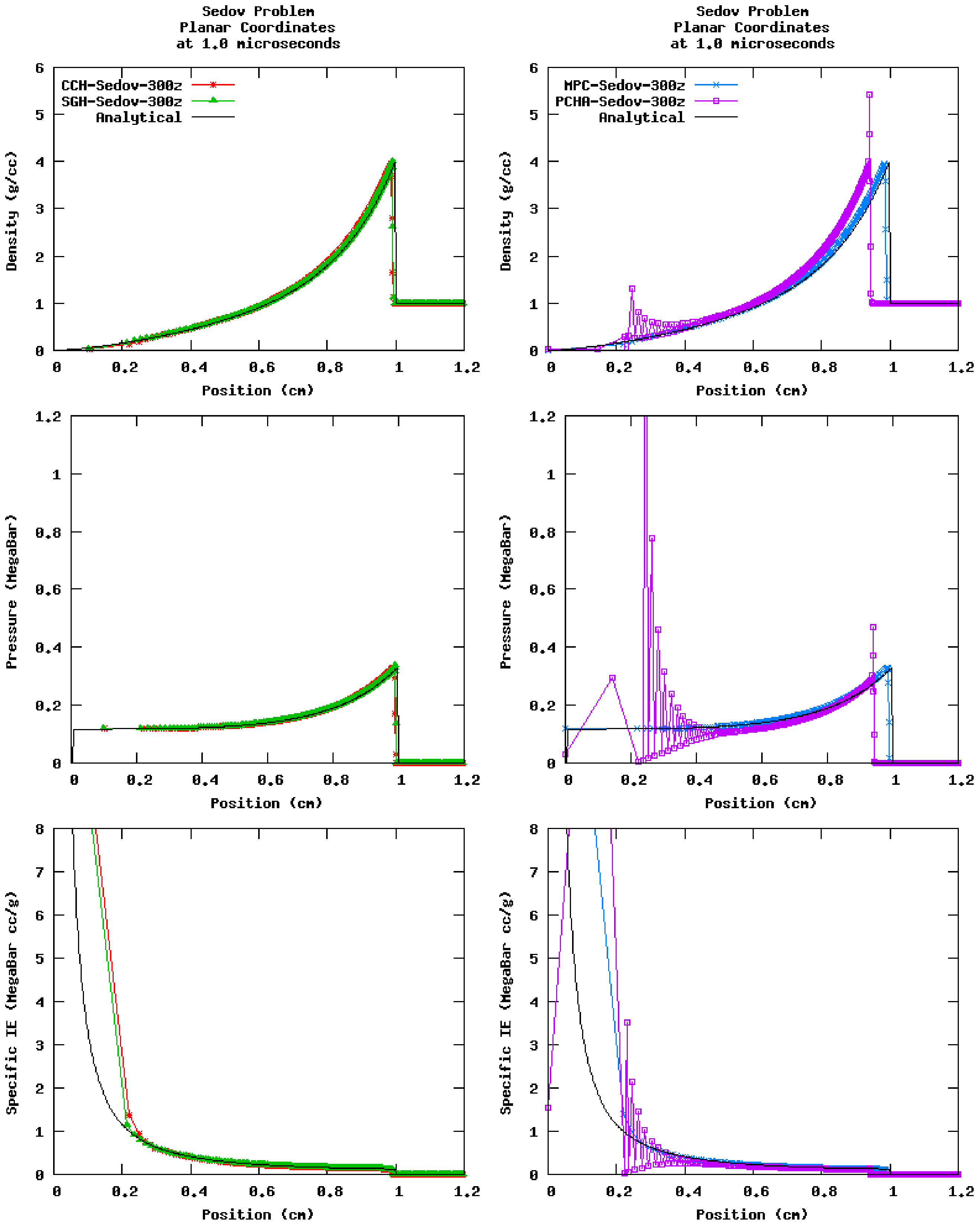


Figure 23. Effect of mesh refinement on the results for the Sedov test problem in planar coordinates.

Conclusions

The project focused on developing a 1D hydrocode to simulate gas dynamics using three different discretization techniques. The techniques used were cell-centered (CCH), staggered grid (SGH), and point-centered hydrodynamics (PCH). The PCH method proved to fail under strong shock conditions. The failure was induced by zone collapse and a diminishing time step. Two additional PCH methods were developed to mitigate this failure. These methods are called modified PCH (MPC) and averaged PCH (PCHA). These two methods produced results for all test problems considered. MPC is similar to CCH, simply shifted by half a cell. PCHA produces numerical oscillations at shock discontinuities.

Four test problems were used to verify the hydrocode. These problems were the Sod, Piston, Noh, and Sedov problems. For the Sod, Piston, and Noh problems, the convergence rates were computed for each method. In planar coordinates, all convergence rates were consistent with the first order numerical scheme used in the code. However, in spherical coordinates for the Noh problem, the pressure did not converge to the analytical solution. Further investigation into curvilinear coordinates is recommended for future work.

The CCH, SGH, and MPC methods showed qualitative convergence in the Sedov test problem in planar coordinates. In curvilinear coordinates, the 1D hydrocode did not produce consistent results for the Sedov test problem. The numerical oscillations produced by the PCHA method are more apparent in the Sedov test problem than in other test problems.

In general, the SGH method performs better for the Sod and Sedov test problems. The CCH method performs better for the Piston and Noh test problems. It is recommended that the different PCH methods be investigated further in order to evaluate the merits of each method and perhaps continue their development.

Future Work

Curvilinear Coordinates

For the Sedov test problem in cylindrical and spherical coordinates, the simulation did not converge to a solution. It was found that as the cell size decreased and the amount of extensive internal energy deposited in the cell at the origin remained constant, the simulation produced varying results. Also, the simulation did not conserve total energy over the course of the simulation. In the future, this issue needs to be studied, and solution needs to be determined. Some hypotheses have been formed. The 1D hydrocode conserves total energy in the Sedov test problem in planar coordinates, but not in cylindrical or spherical coordinates. Therefore, it seems most likely that the error in energy conservation is due to the calculation of the areas and volumes in cylindrical and spherical coordinates. Alternatively, the way the boundary conditions are formulated could be resulting in error for cylindrical and spherical coordinates.

Second Order Scheme

The addition of a second order scheme would involve adjusting the Riemann solver in each of the hydro method programs. Instead of assigning a value of pressure or velocity to an interface from the nearest control volume, the pressure and velocity are projected via a central gradient. The projection of the velocity is illustrated in Figure 24.

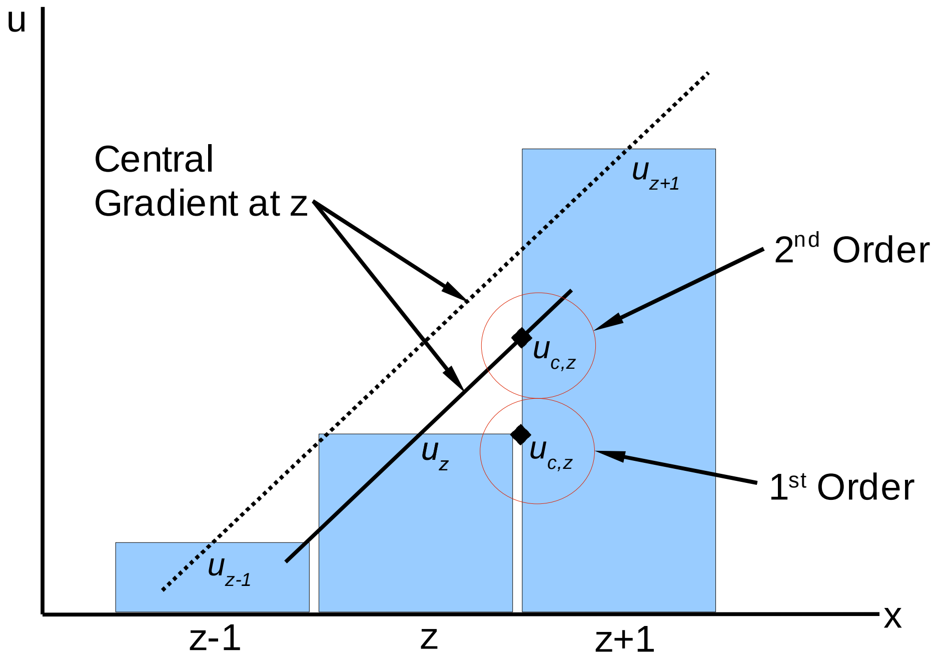


Figure 24. Second order velocity projection.

$u_{c,z}$ is evaluated at the cell interfaces using a gradient in a second order scheme. Assuming a constant value across the control volume is a first order approximation. The central gradient is given by

$$\nabla u = \frac{u_{z+1} - u_{z-1}}{\frac{1}{2}(\Delta x)_{z-1} + (\Delta x)_z + \frac{1}{2}(\Delta x)_{z+1}}$$

and $u_{c,z}$ is given by

$$u_{c,z} = u_z + \frac{1}{2}(\Delta x)_z(\nabla u)$$

Limiters will also need to be used in a second order scheme. Limiters are used to limit the gradient if the gradient projects a value that is greater than the highest value or less than the lowest value. Figure 25 illustrates this concept, where $\theta_{z,p}$ is a limiter.

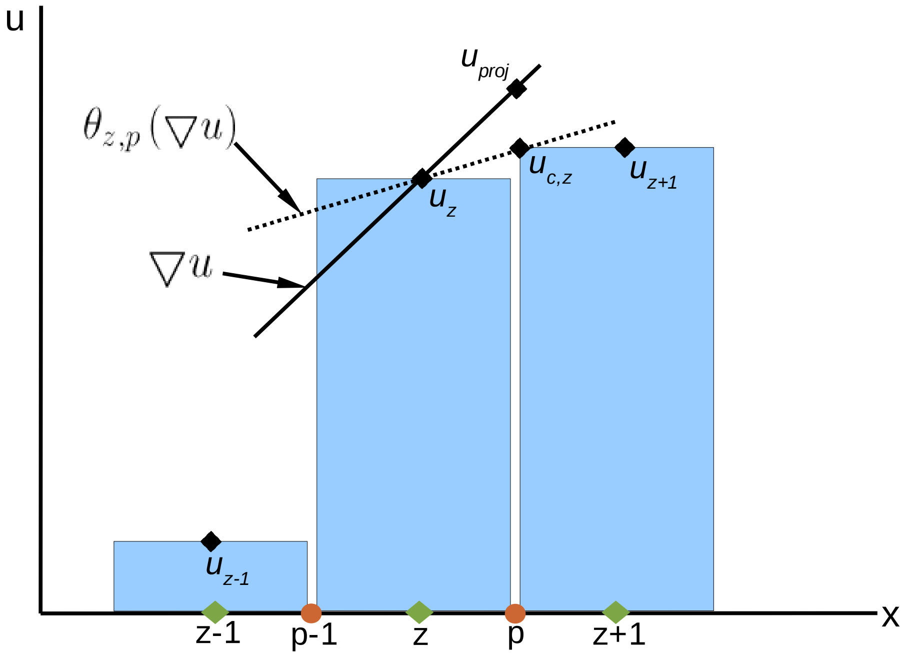


Figure 25. Second order velocity projection with a gradient limiter.

The limiter is computed using the formula

$$\theta_{z,p} = \min\left[\frac{u_z - u_{z+1}}{u_z - u_{proj}}, 1\right]$$

Similarly, at the interface at *p-1*, the limiter would be evaluated by

$$\theta_{z,p-1} = \min\left[\frac{u_z - u_{z-1}}{u_z - u_{proj}}, 1\right]$$

The projected value, or corner value, at *p* can then be computed using

$$u_{c,z} = u_z + \frac{1}{2}(\Delta x)_z\ \theta_{z,p}(\nabla u)$$

The corner values computed with the limited gradients are then used in the Riemann solver.

References

[1] R. Loubere, M. Shashkov, B. Wendroff, Volume consistency in a staggered grid Lagrangian hydrodynamics scheme. Journal of Computational Physics 227 (2008); 3731-3737.

[2] J. Dukowicz, A general, non-iterative Riemann solver for Godunovs method. Journal of Computational Physics 1985; 61:119-137.

[3] N. Morgan, A Lagrangian Staggered Grid Godunov-like Approach. Not yet published.

[4] N. Morgan, A dissipation model for staggered grid Lagrangian hydrodynamics. Computers and Fluids 2013; 83:48-57.

[5] G.A. Sod, Survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. Journal of Computational Physics 26 (1978).

[6] sod.py (Python Code for Sod Solution). Written by Scott Doebling, Los Alamos National Lab, February 2013.

[7] W.F. Noh, Errors for calculations of strong shocks using artificial viscosity and an artificial heat flux. Journal of Computational Physics 72 (1987); 78–120.

[8] L. I. Sedov, Propagation of strong shock waves, Journal of Applied Mathematics and Mechanics 10 (1946), 241–250.

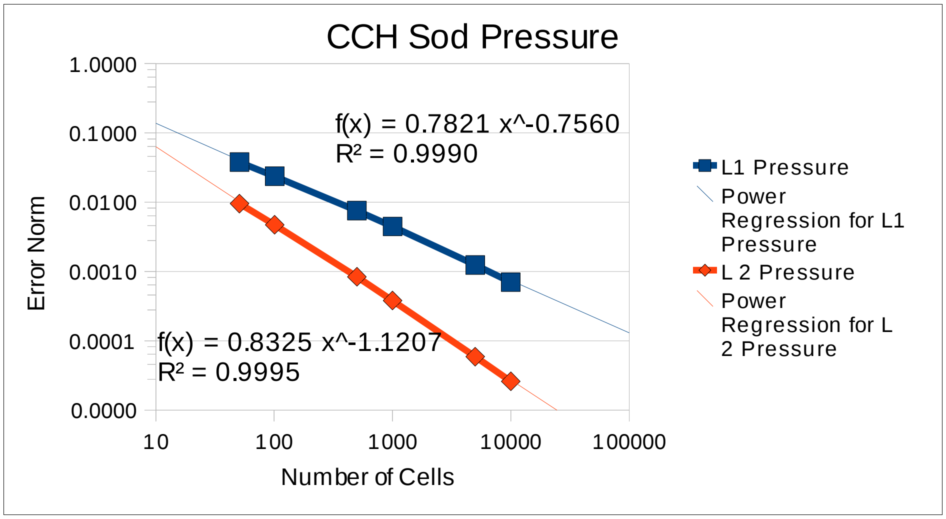
[9] sedov_exact.py (Python Code for Sedov Solution). Written by Scott Doebling, Los Alamos National Lab, August 2012.

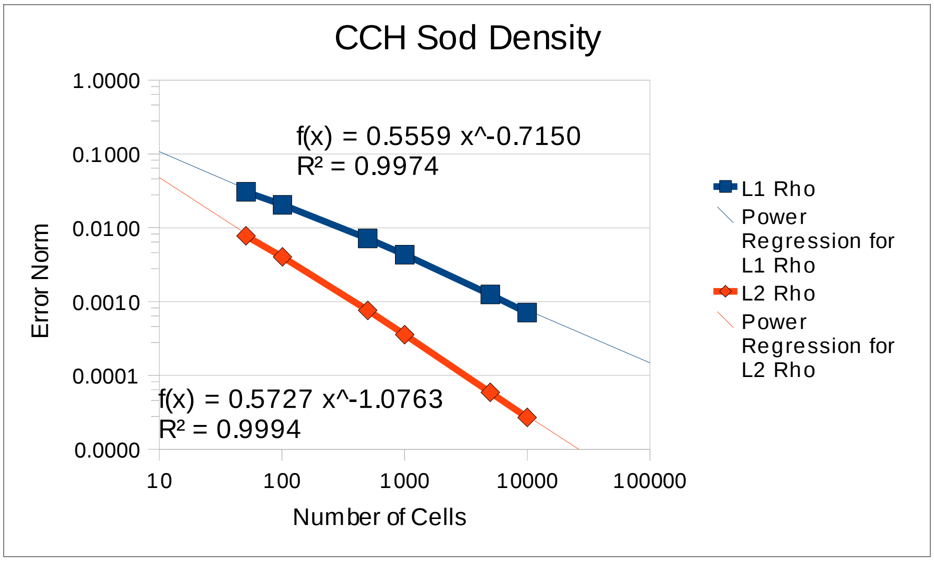
Appendix

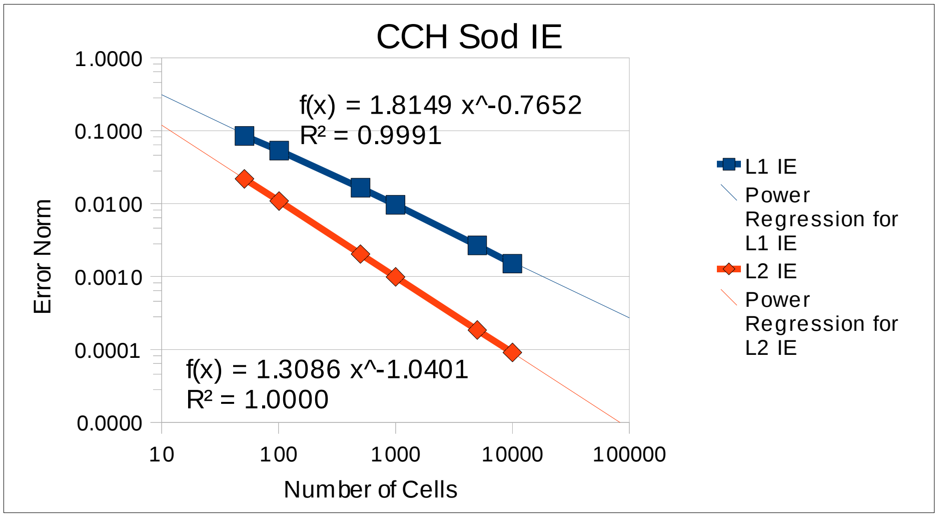
Convergence Plots

SOD

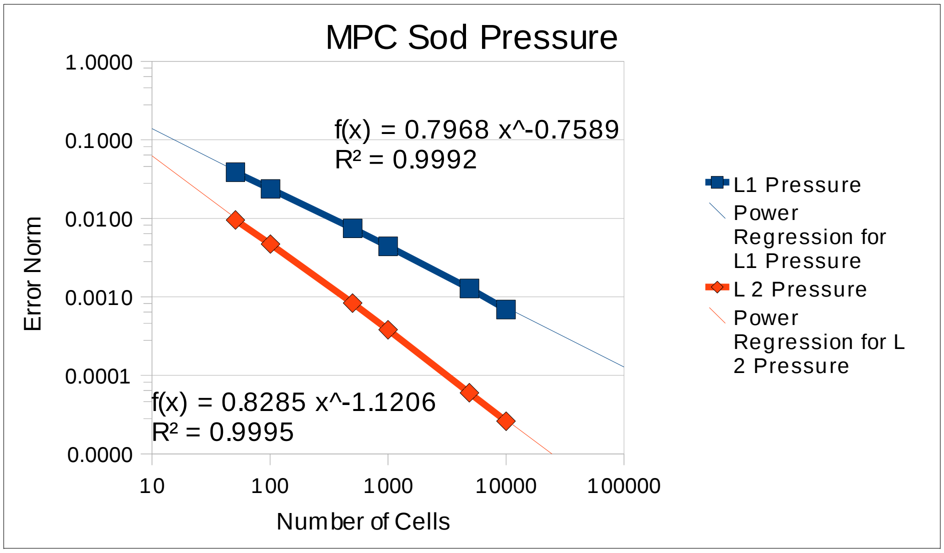
CCH

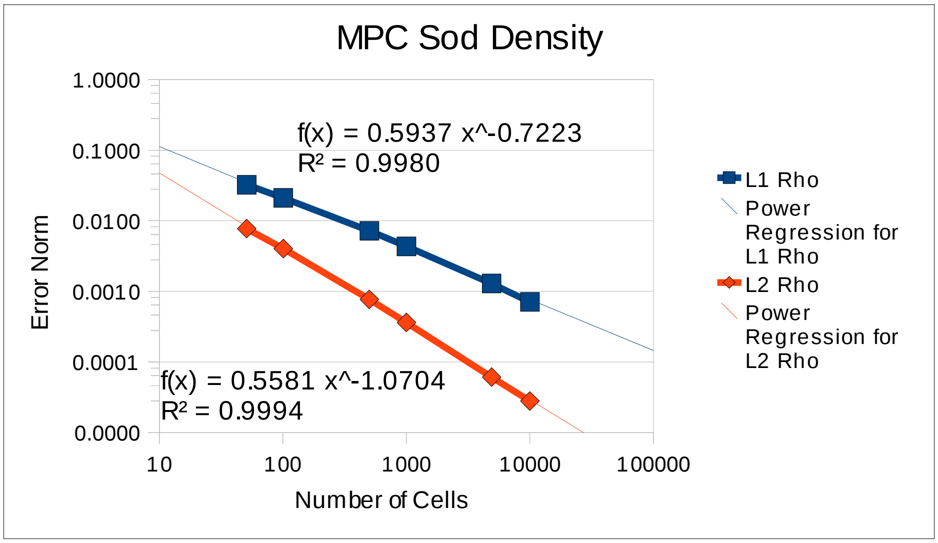


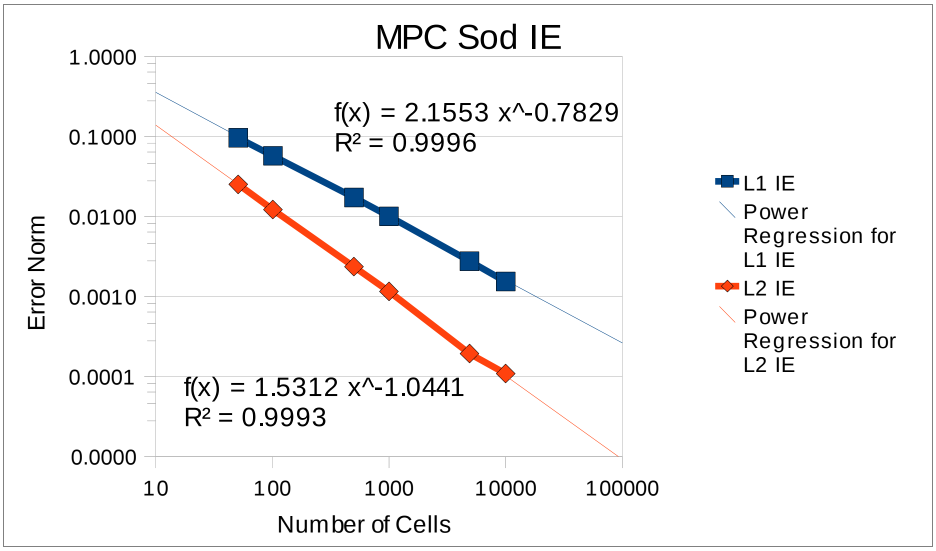



CCH Sod IE
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
f(x) = 1.8149 x^-0.7652
R² = 0.9991
f(x) = 1.3086 x^-1.0401
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

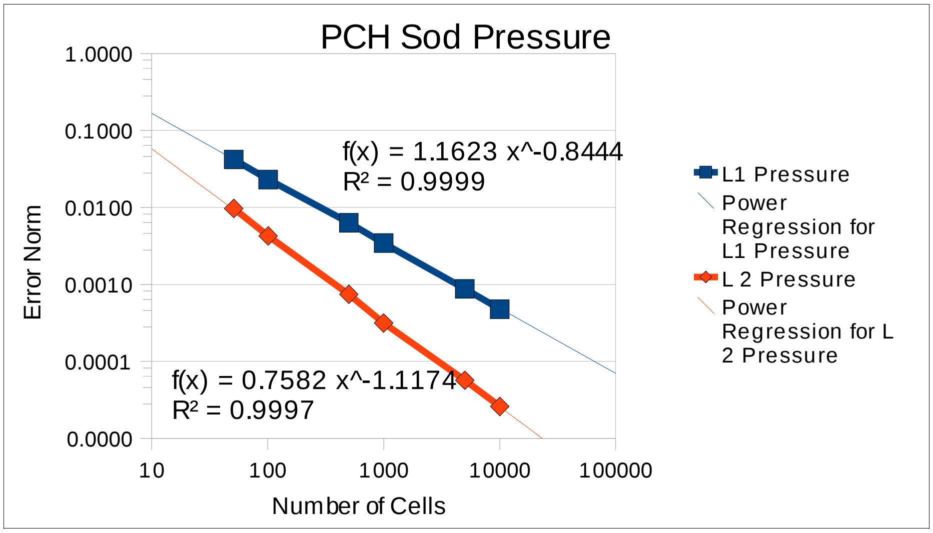
MPC

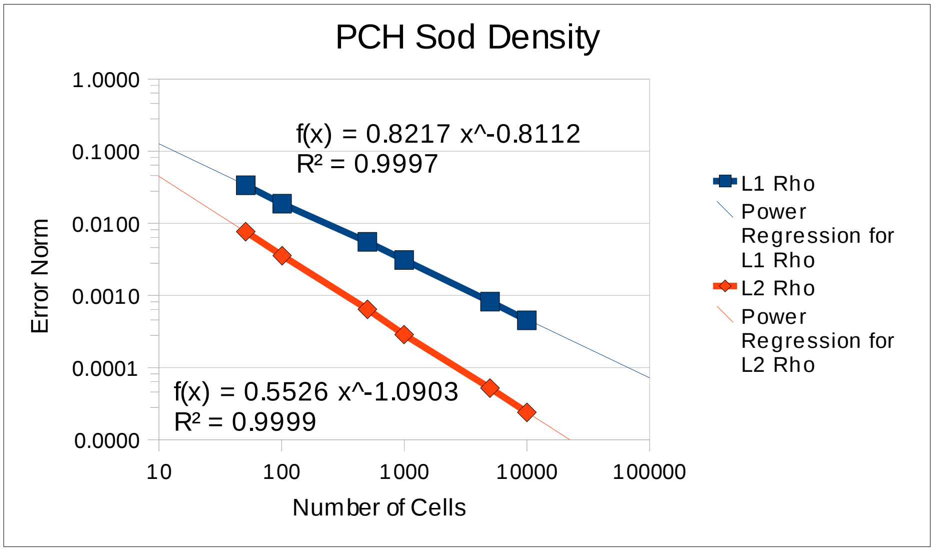

MPC Sod Pressure
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
f(x) = 0.7968 x^-0.7589
R² = 0.9992
f(x) = 0.8285 x^-1.1206
R² = 0.9995
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

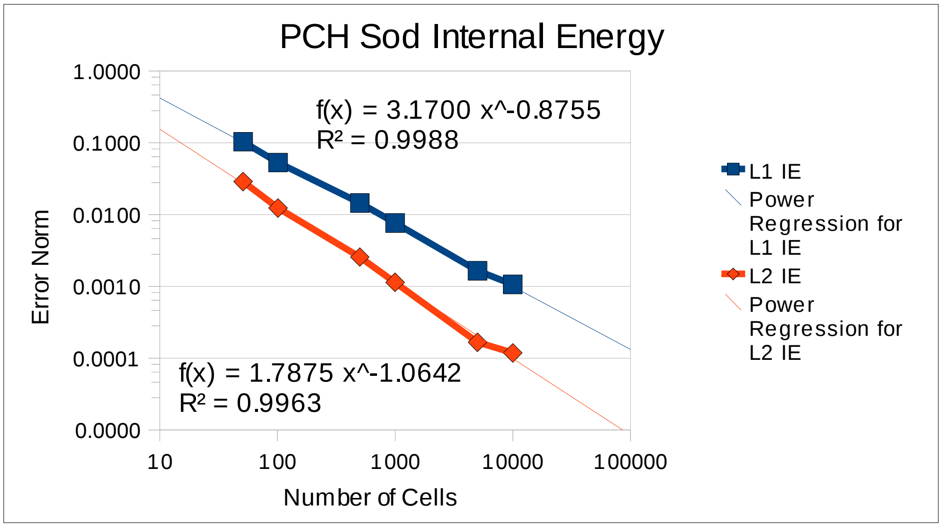

MPC Sod Density
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
f(x) = 0.5937 x^-0.7223
R² = 0.9980
f(x) = 0.5581 x^-1.0704
R² = 0.9994
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


MPC Sod IE
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 2.1553 x^-0.7829
R² = 0.9996
f(x) = 1.5312 x^-1.0441
R² = 0.9993
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

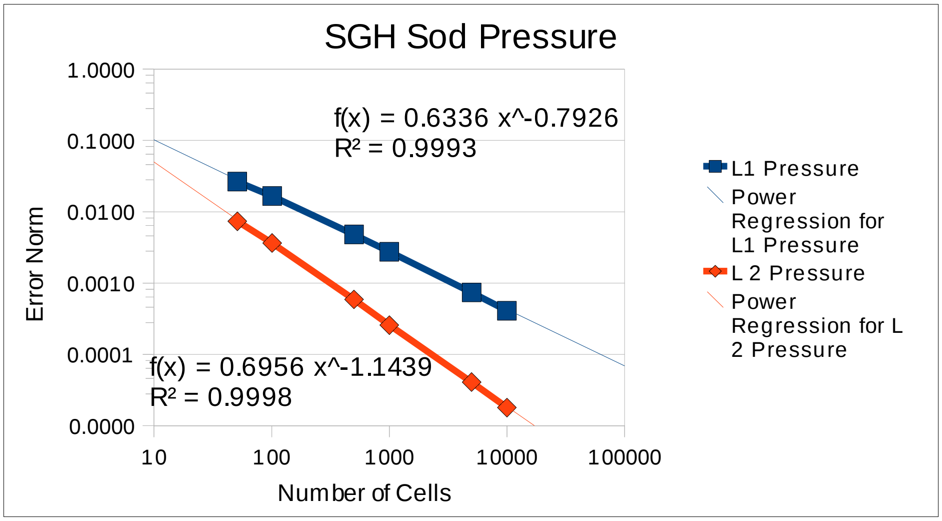
PCHA

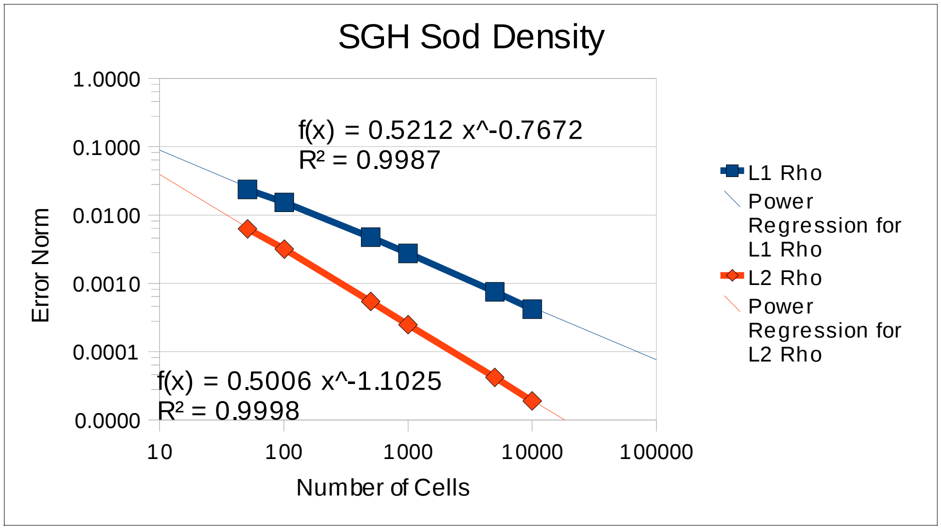

PCH Sod Pressure
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 1.1623 x^-0.8444
R² = 0.9999
f(x) = 0.7582 x^-1.1174
R² = 0.9997
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

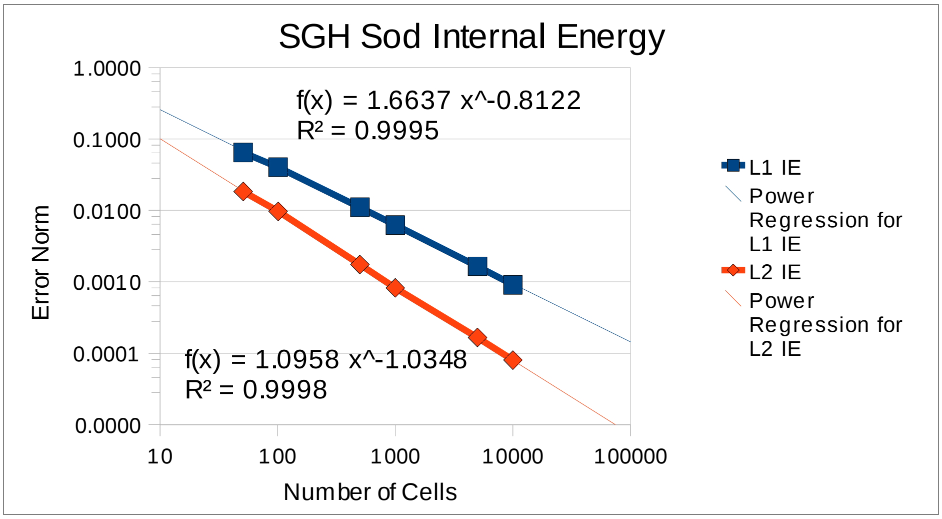

PCH Sod Density
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 0.8217 x^-0.8112
R² = 0.9997
f(x) = 0.5526 x^-1.0903
R² = 0.9999
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


PCH Sod Internal Energy
f(x) = 3.1700 x^-0.8755
R² = 0.9988
f(x) = 1.7875 x^-1.0642
R² = 0.9963
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

SGH

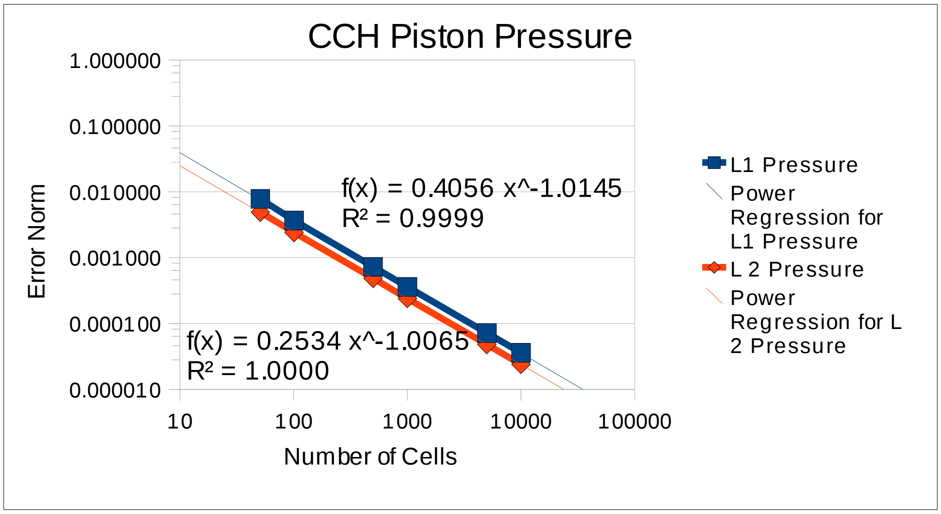

SGH Sod Pressure
f(x) = 0.6336 x^-0.7926
R² = 0.9993
f(x) = 0.6956 x^-1.1439
R² = 0.9998
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

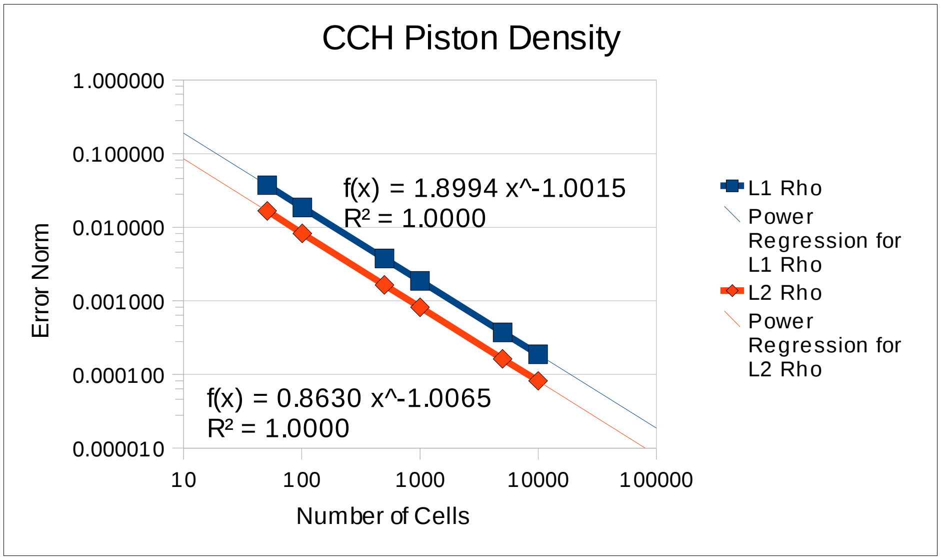

SGH Sod Density
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
f(x) = 0.5212 x^-0.7672
R² = 0.9987
f(x) = 0.5006 x^-1.1025
R² = 0.9998
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho

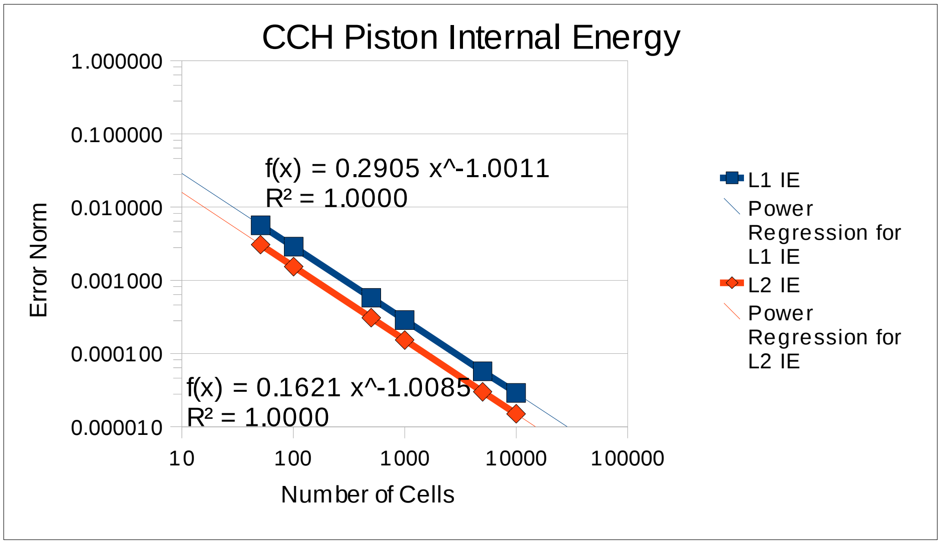

SGH Sod Internal Energy
Error Norm
1.0000
0.1000
0.0100
0.0010
0.0001
0.0000
10
100
1000
10000
100000
Number of Cells
f(x) = 1.6637 x^-0.8122
R² = 0.9995
f(x) = 1.0958 x^-1.0348
R² = 0.9998
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

PISTON

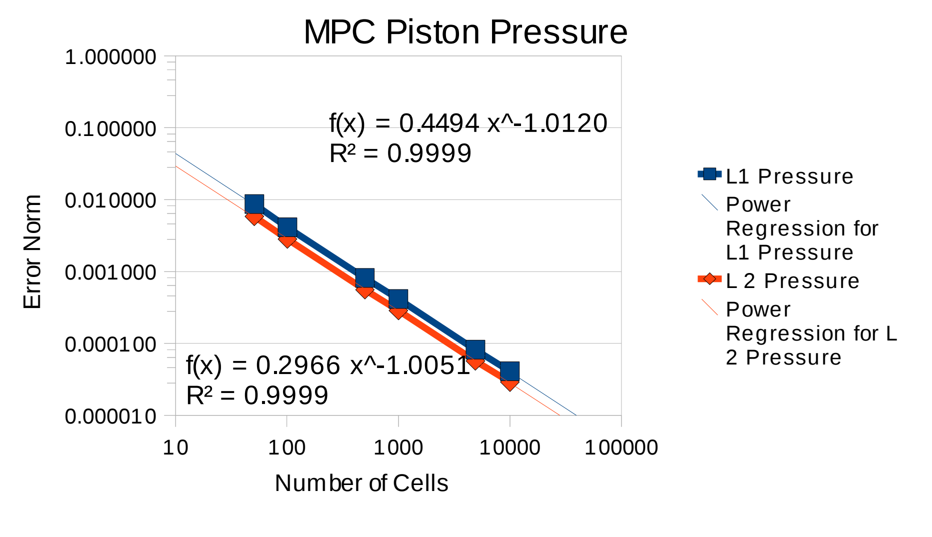
CCH

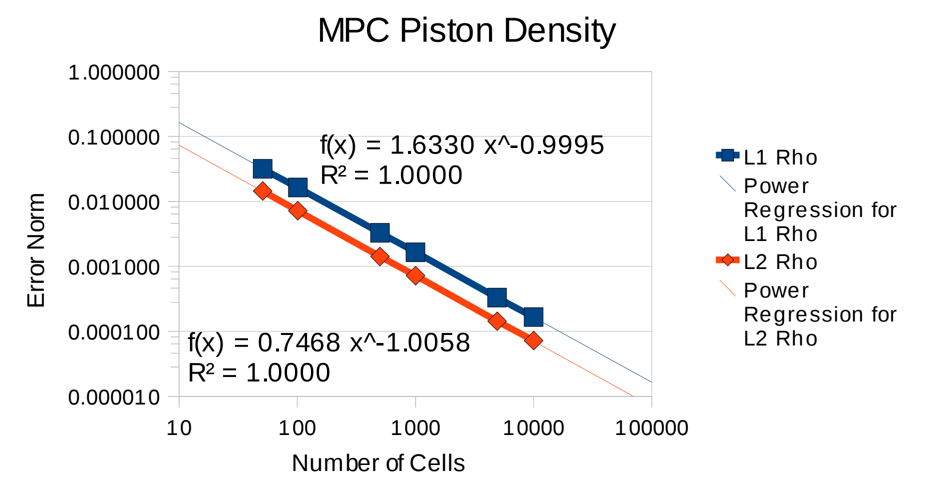

CCH Piston Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.4056 x^-1.0145
R² = 0.9999
f(x) = 0.2534 x^-1.0065
R² = 1.0000
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

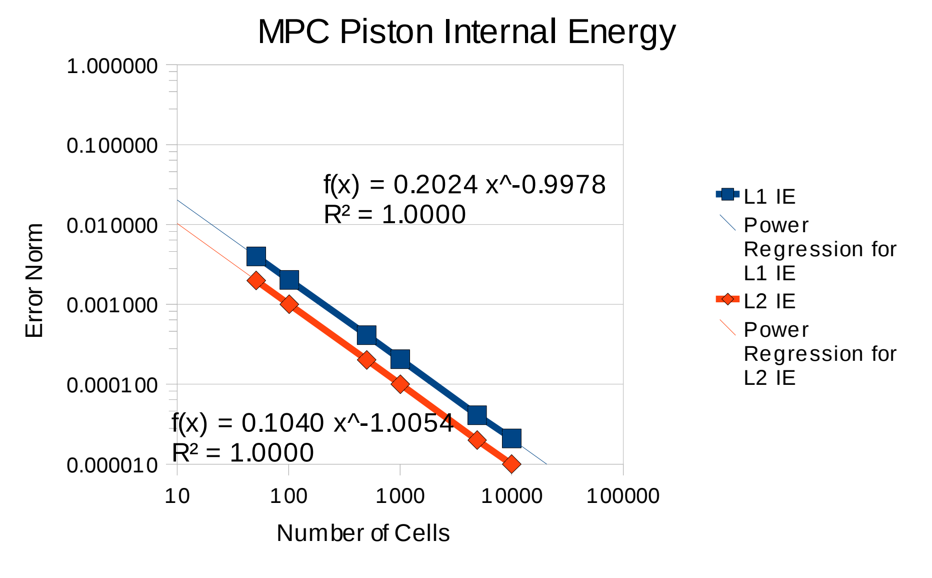

CCH Piston Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 1.8994 x^-1.0015
R² = 1.0000
f(x) = 0.8630 x^-1.0065
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


CCH Piston Internal Energy
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 0.2905 x^-1.0011
R² = 1.0000
f(x) = 0.1621 x^-1.0085
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

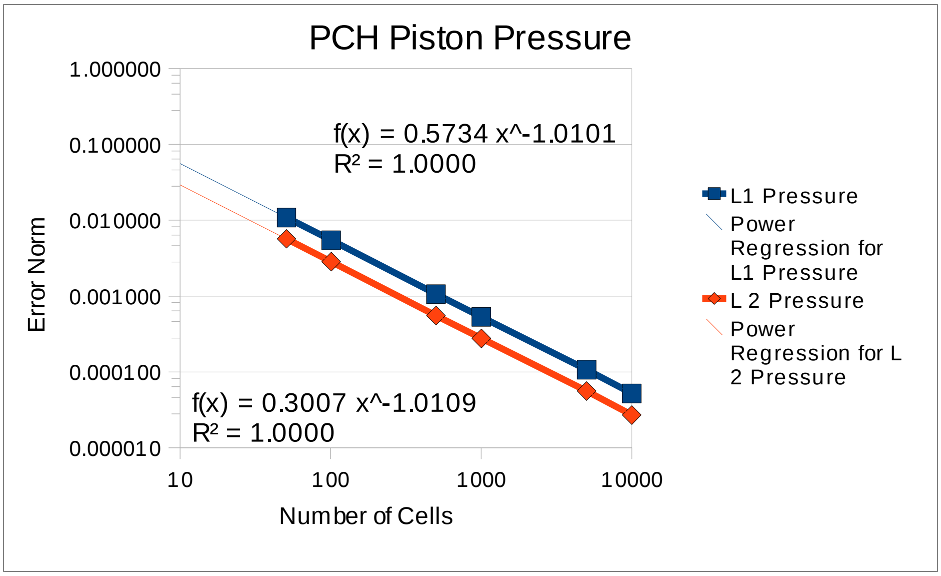
MPC

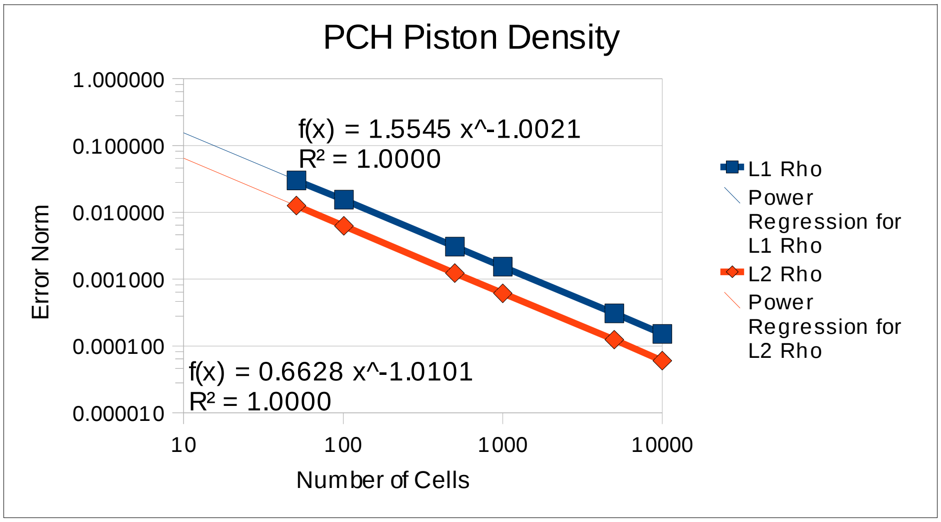

MPC Piston Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.4494 x^-1.0120
R² = 0.9999
f(x) = 0.2966 x^-1.0051
R² = 0.9999
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

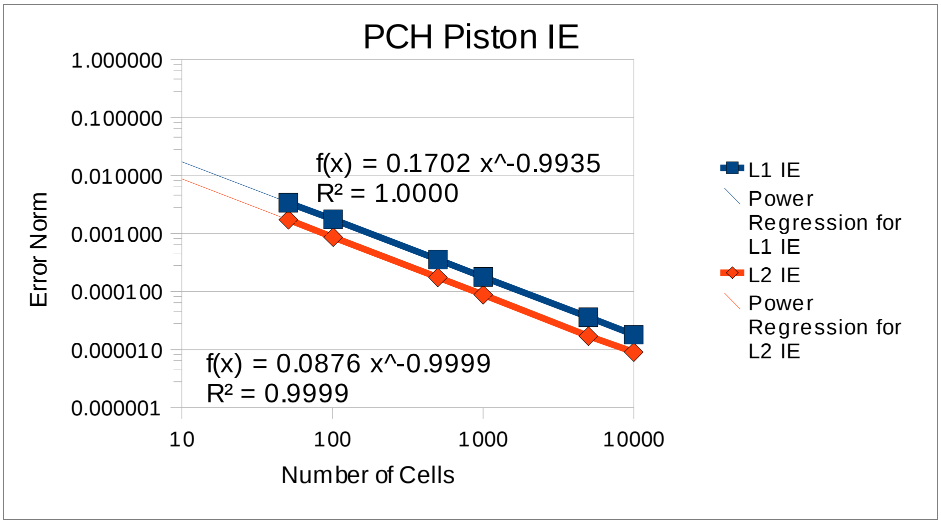

MPC Piston Density
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 1.6330 x^-0.9995
R² = 1.0000
f(x) = 0.7468 x^-1.0058
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


MPC Piston Internal Energy
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.2024 x^-0.9978
R² = 1.0000
f(x) = 0.1040 x^-1.0054
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

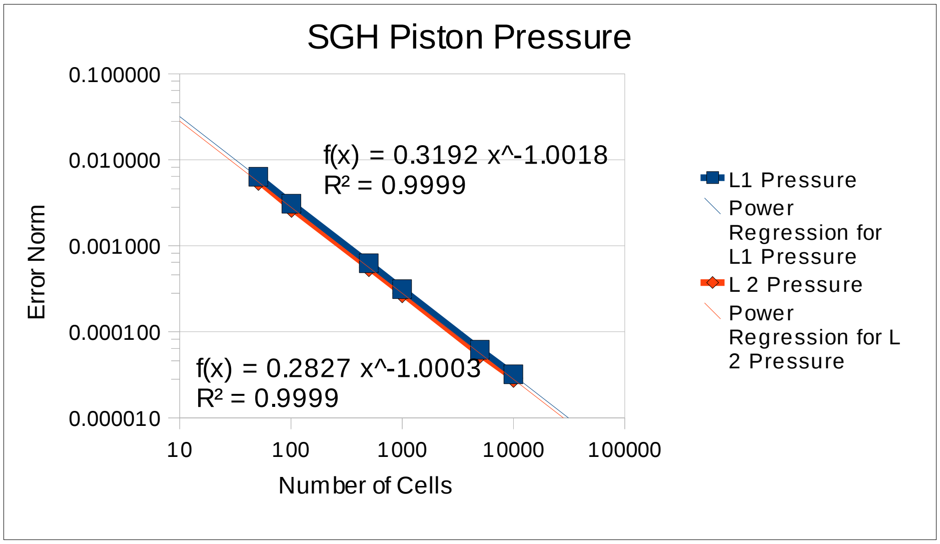
PCH


PCH Piston Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
Number of Cells
f(x) = 0.5734 x^-1.0101
R² = 1.0000
f(x) = 0.3007 x^-1.0109
R² = 1.0000
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

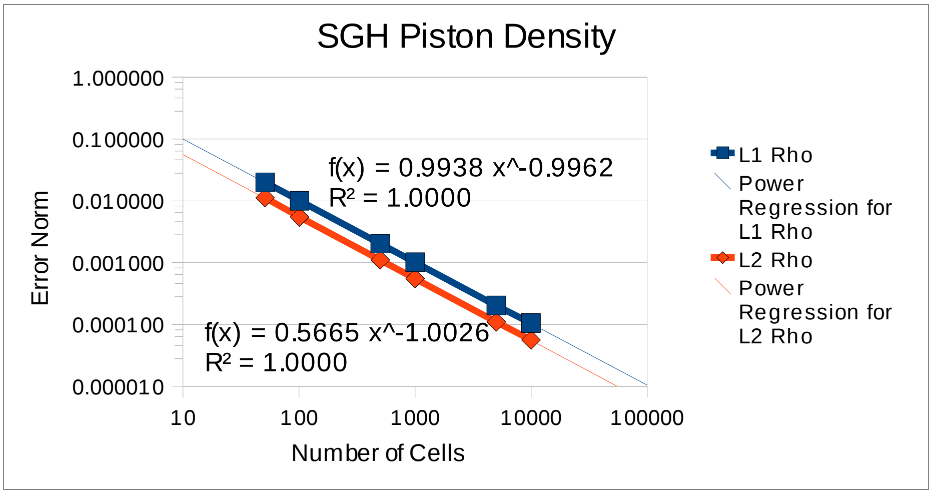

PCH Piston Density
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
Number of Cells
f(x) = 1.5545 x^-1.0021
R² = 1.0000
f(x) = 0.6628 x^-1.0101
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


PCH Piston IE
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
10
100
1000
10000
Number of Cells
f(x) = 0.1702 x^-0.9935
R² = 1.0000
f(x) = 0.0876 x^-0.9999
R² = 0.9999
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

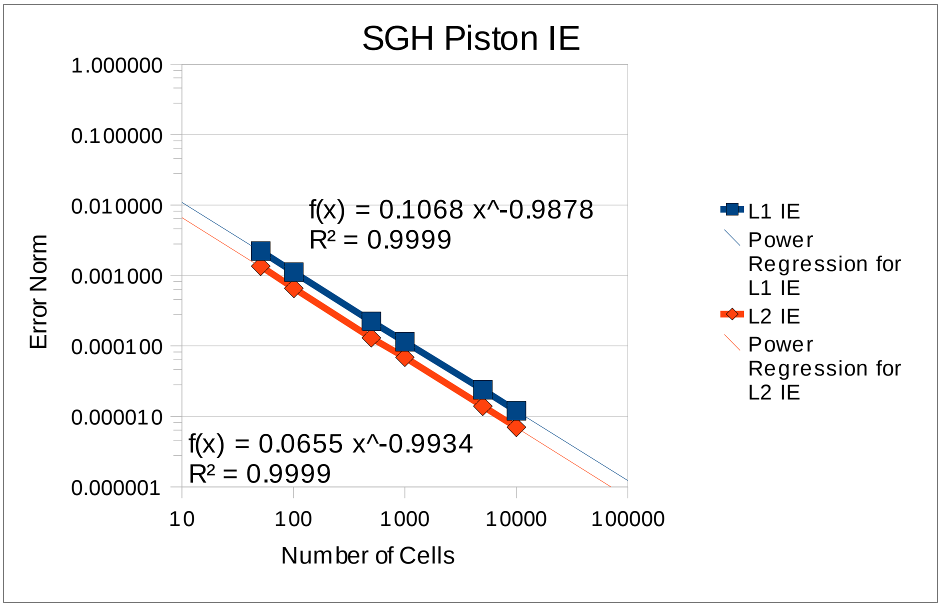
SGH



SGH Piston Pressure
Error Norm
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.3192 x^-1.0018
R² = 0.9999
f(x) = 0.2827 x^-1.0003
R² = 0.9999
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure



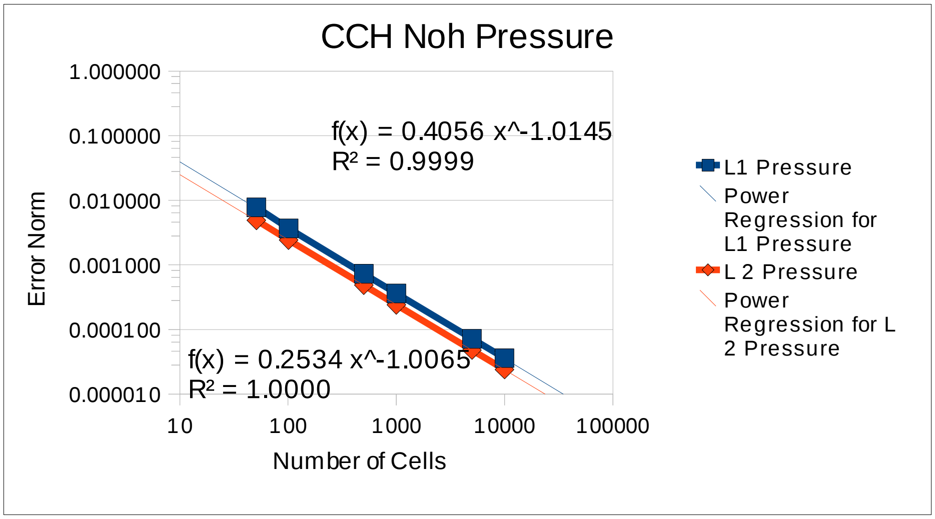
SGH Piston Density
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.9938 x^-0.9962
R² = 1.0000
f(x) = 0.5665 x^-1.0026
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho

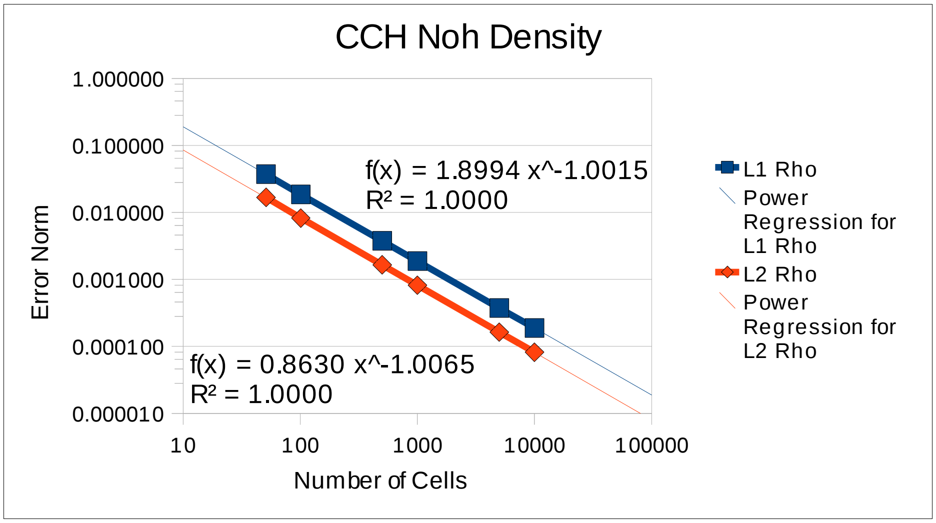


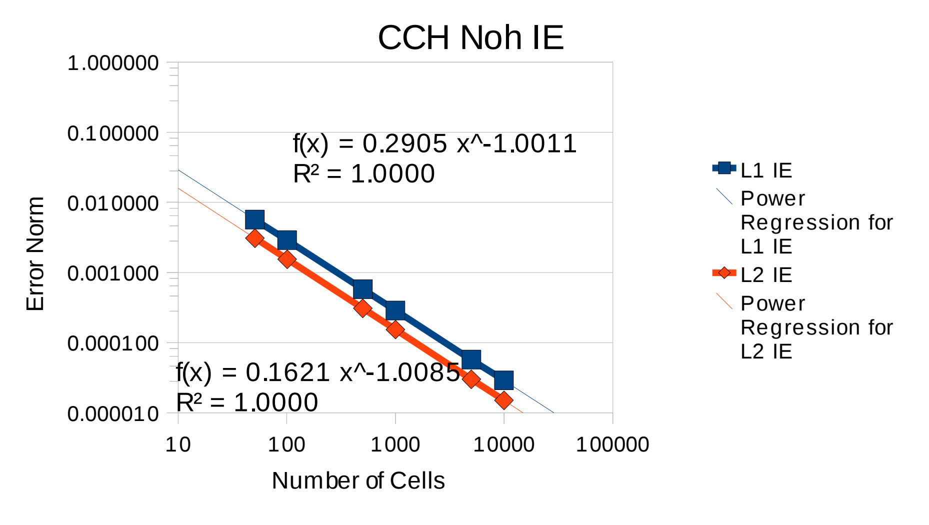
SGH Piston IE
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
10
100
1000
10000
100000
Number of Cells
f(x) = 0.1068 x^-0.9878
R² = 0.9999
f(x) = 0.0655 x^-0.9934
R² = 0.9999
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

Noh Planar Coordinates

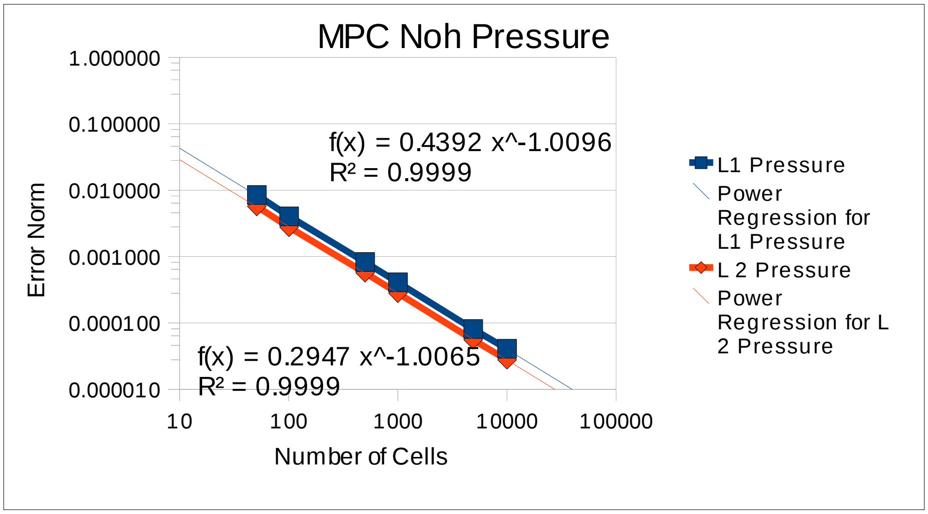
CCH

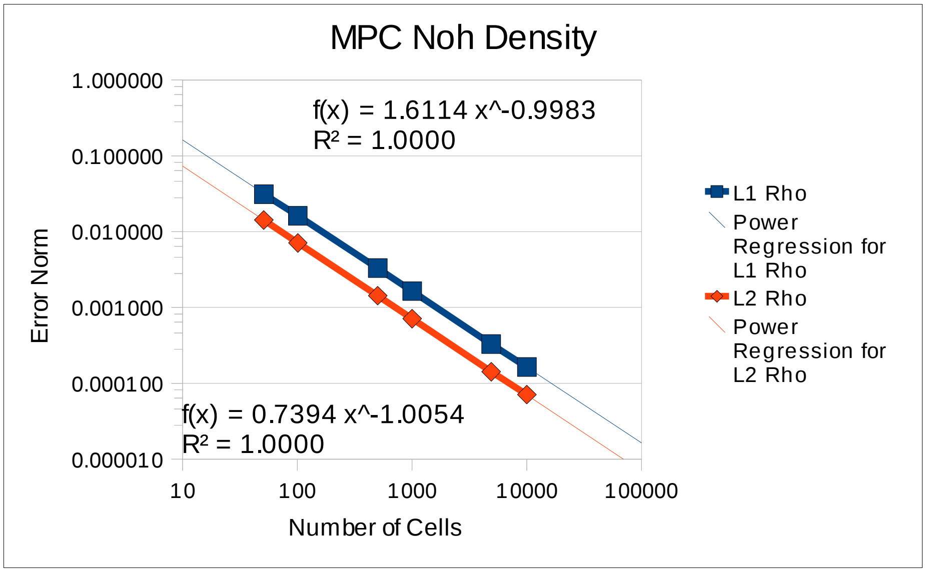


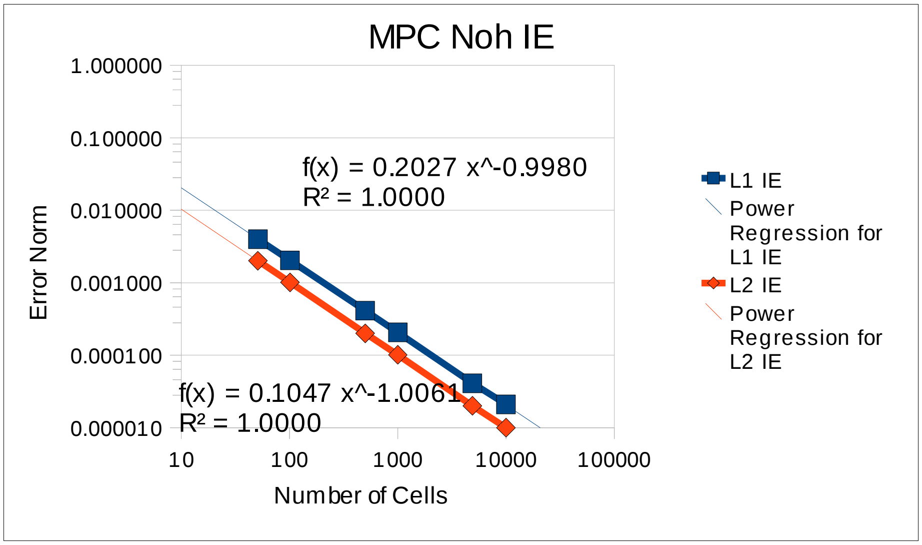




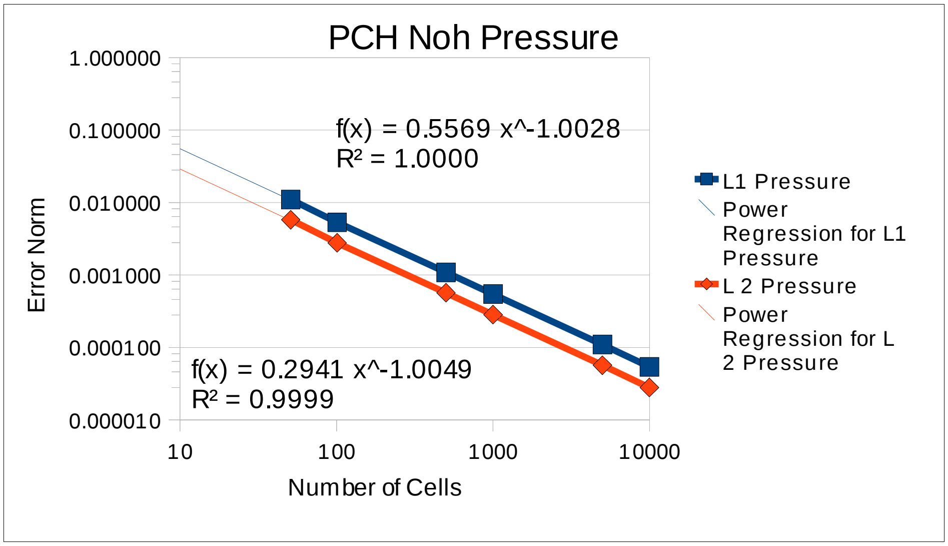
MPC

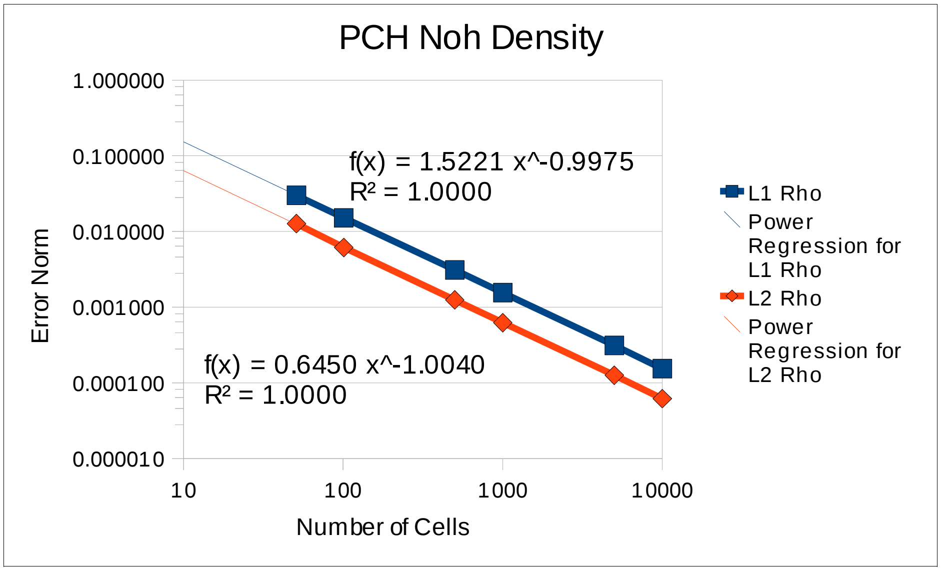

MPC Noh Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.4392 x^-1.0096
R² = 0.9999
f(x) = 0.2947 x^-1.0065
R² = 0.9999
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

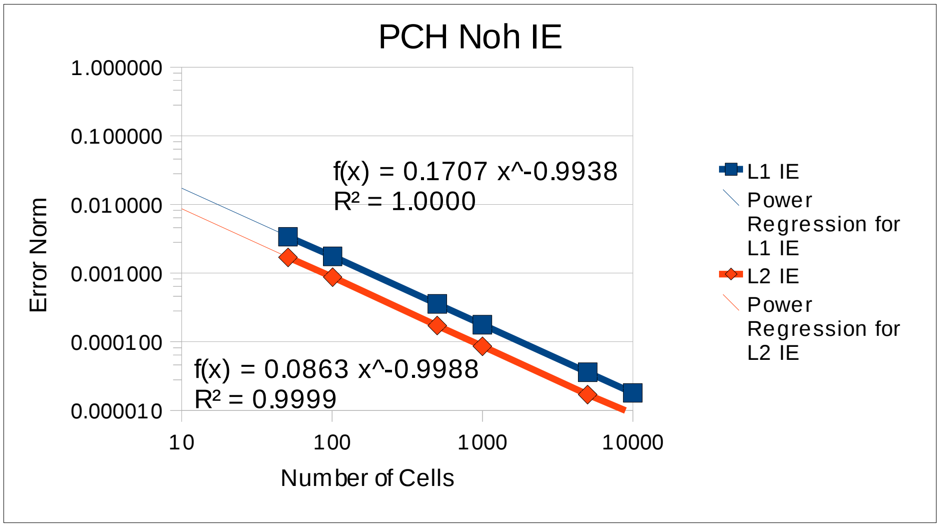

MPC Noh Density
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 1.6114 x^-0.9983
R² = 1.0000
f(x) = 0.7394 x^-1.0054
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


MPC Noh IE
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.2027 x^-0.9980
R² = 1.0000
f(x) = 0.1047 x^-1.0061
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

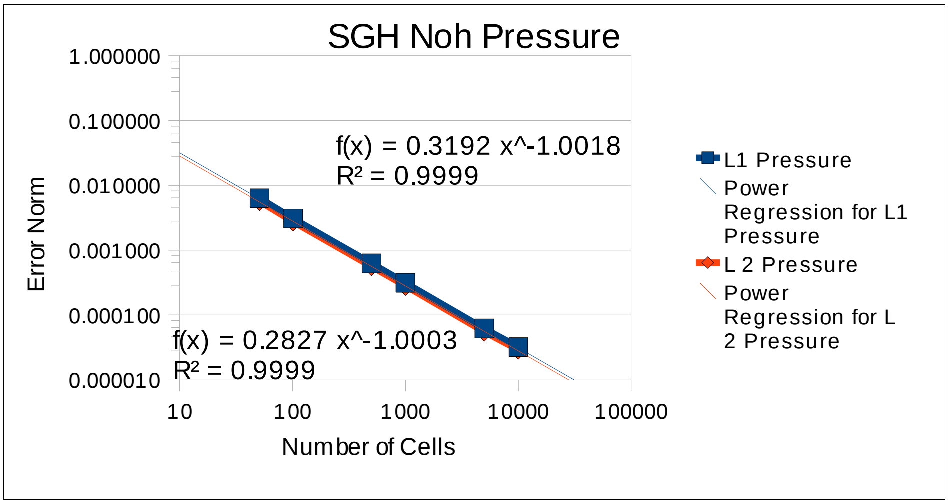
PCH

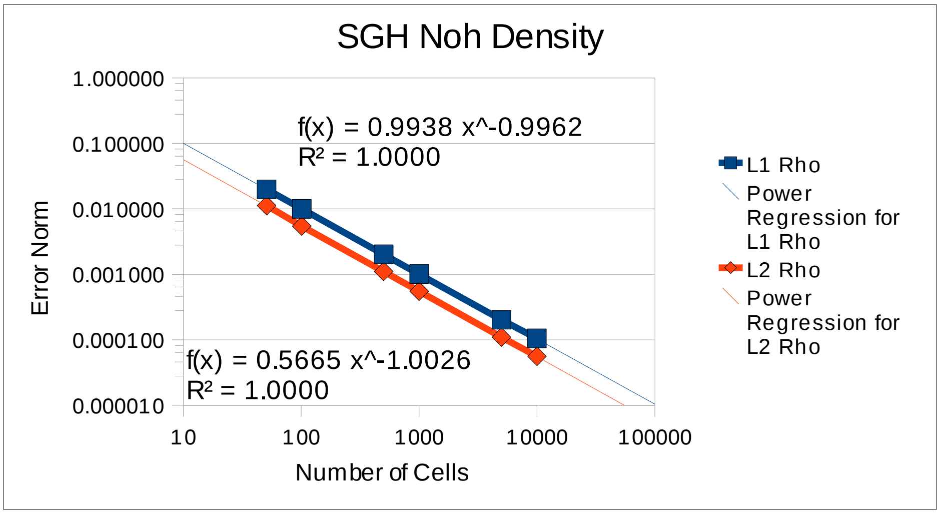

PCH Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
Number of Cells
f(x) = 0.5569 x^-1.0028
R² = 1.0000
f(x) = 0.2941 x^-1.0049
R² = 0.9999
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

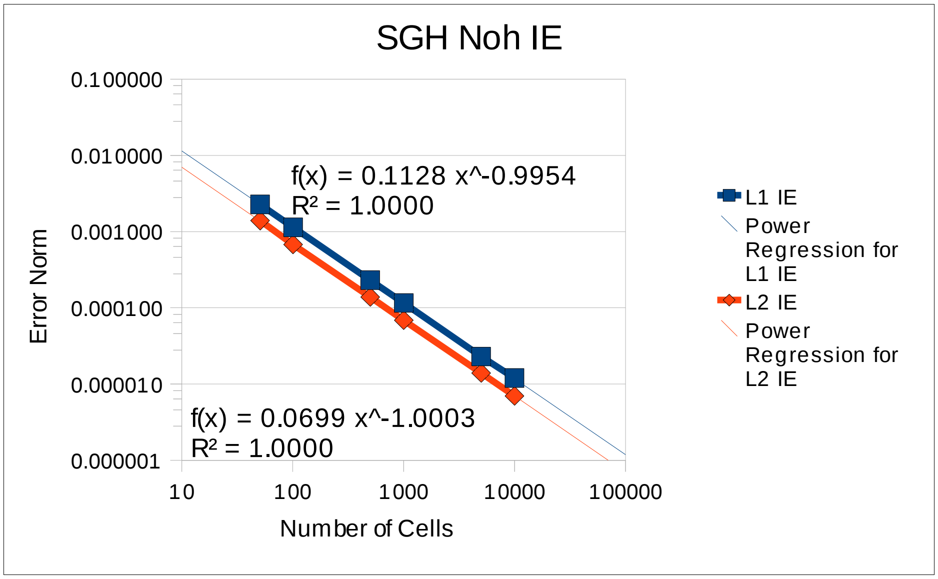

PCH Noh Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
Number of Cells
f(x) = 1.5221 x^-0.9975
R² = 1.0000
f(x) = 0.6450 x^-1.0040
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


PCH Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
Number of Cells
f(x) = 0.1707 x^-0.9938
R² = 1.0000
f(x) = 0.0863 x^-0.9988
R² = 0.9999
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

SGH

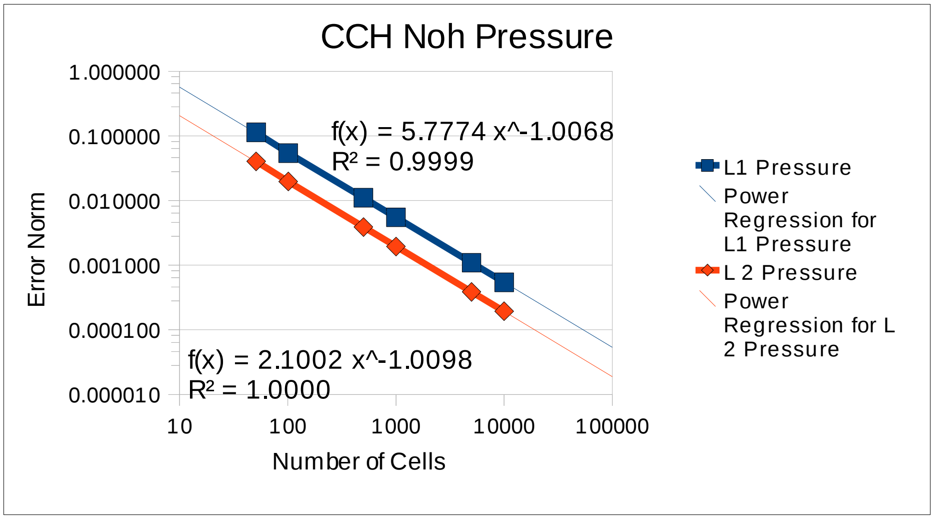

SGH Noh Pressure
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 0.3192 x^-1.0018
R² = 0.9999
f(x) = 0.2827 x^-1.0003
R² = 0.9999
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

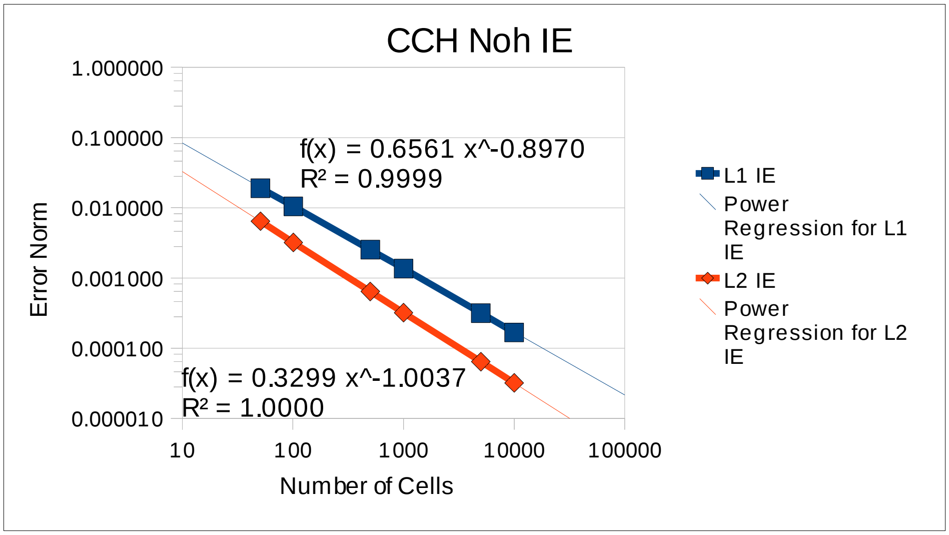

SGH Noh Density
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 0.9938 x^-0.9962
R² = 1.0000
f(x) = 0.5665 x^-1.0026
R² = 1.0000
L1 Rho
Power Regression for L1 Rho
L2 Rho
Power Regression for L2 Rho


SGH Noh IE
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 0.1128 x^-0.9954
R² = 1.0000
f(x) = 0.0699 x^-1.0003
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

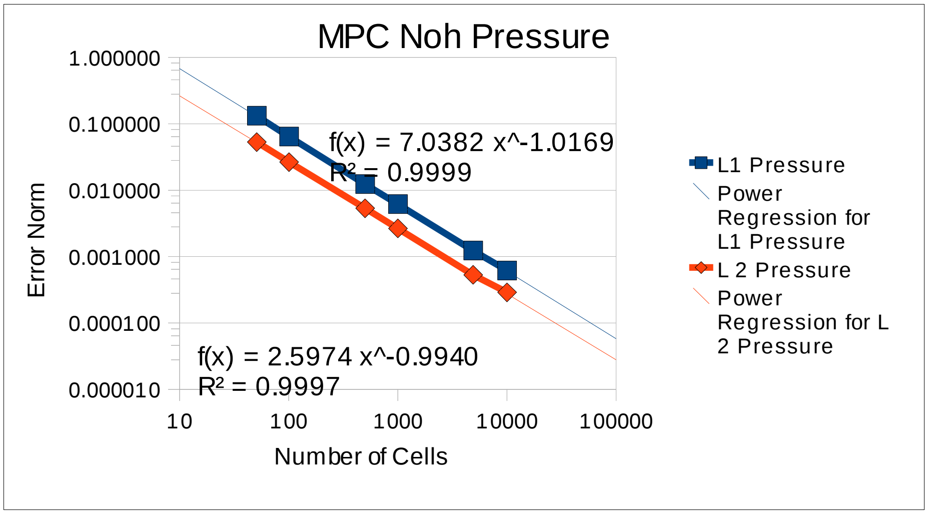
Noh Cylindrical

CCH

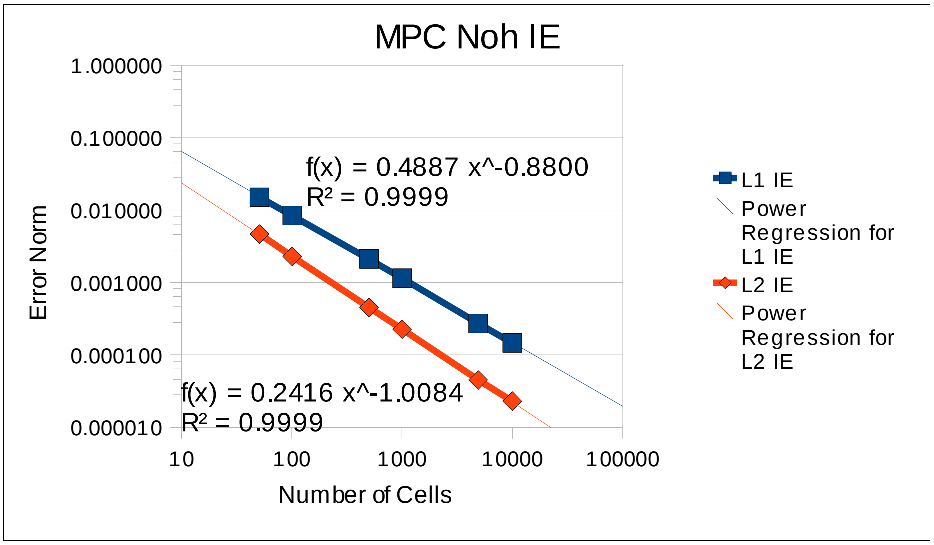




MPC

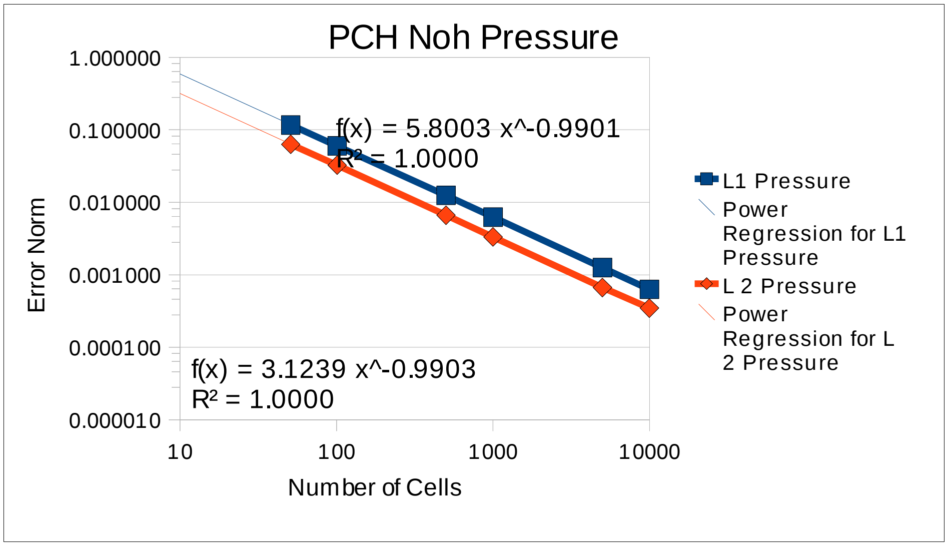


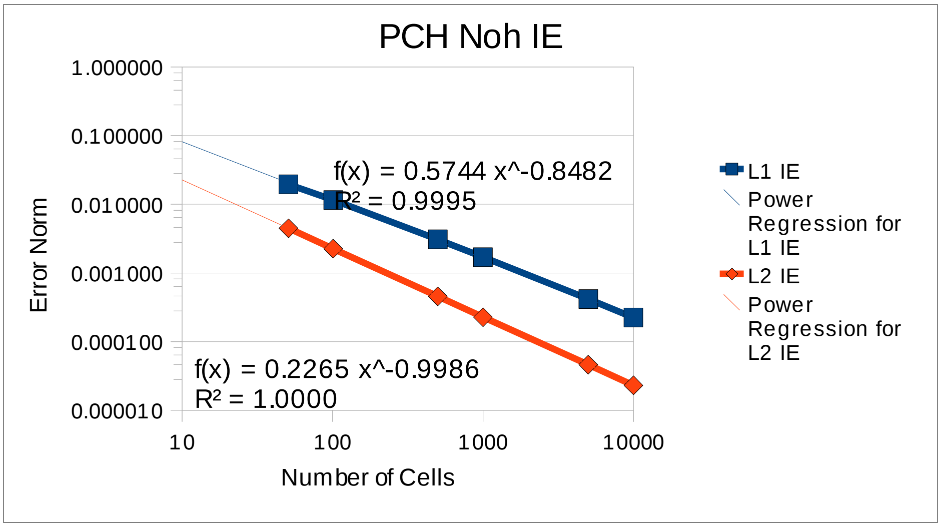
MPC Noh Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 7.0382 x^-1.0169
R² = 0.9999
f(x) = 2.5974 x^-0.9940
R² = 0.9997
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure



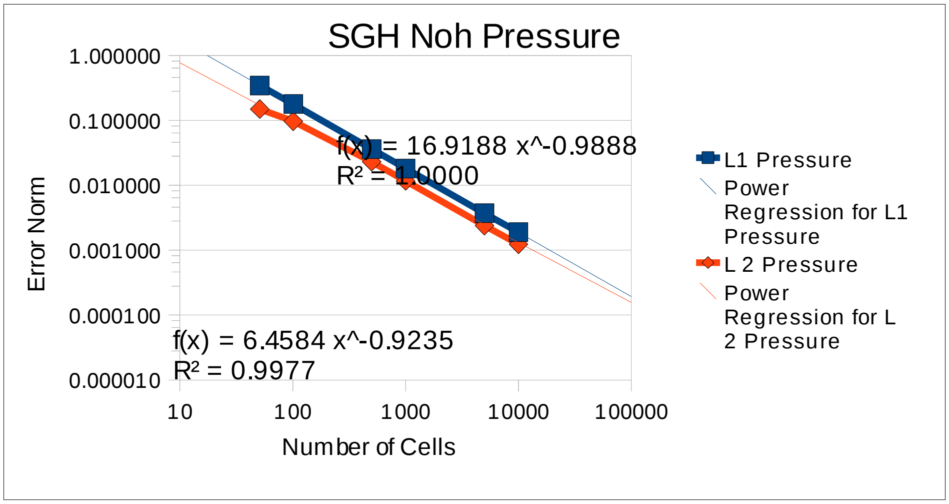
MPC Noh IE
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 0.4887 x^-0.8800
R² = 0.9999
f(x) = 0.2416 x^-1.0084
R² = 0.9999
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

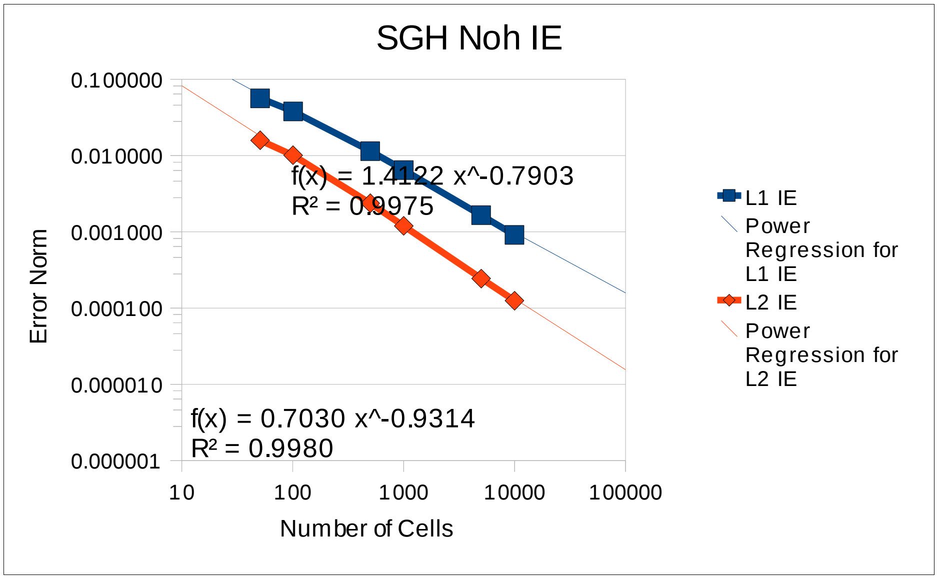
PCHA


PCH Noh Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
Number of Cells
f(x) = 5.8003 x^-0.9901
R² = 1.0000
f(x) = 3.1239 x^-0.9903
R² = 1.0000
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


PCH Noh IE
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
Number of Cells
f(x) = 0.5744 x^-0.8482
R² = 0.9995
f(x) = 0.2265 x^-0.9986
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

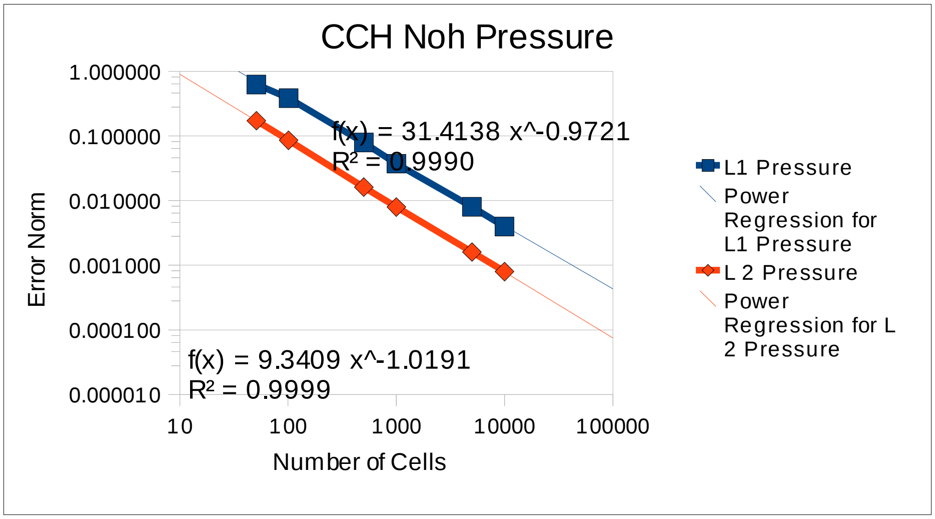
SGH

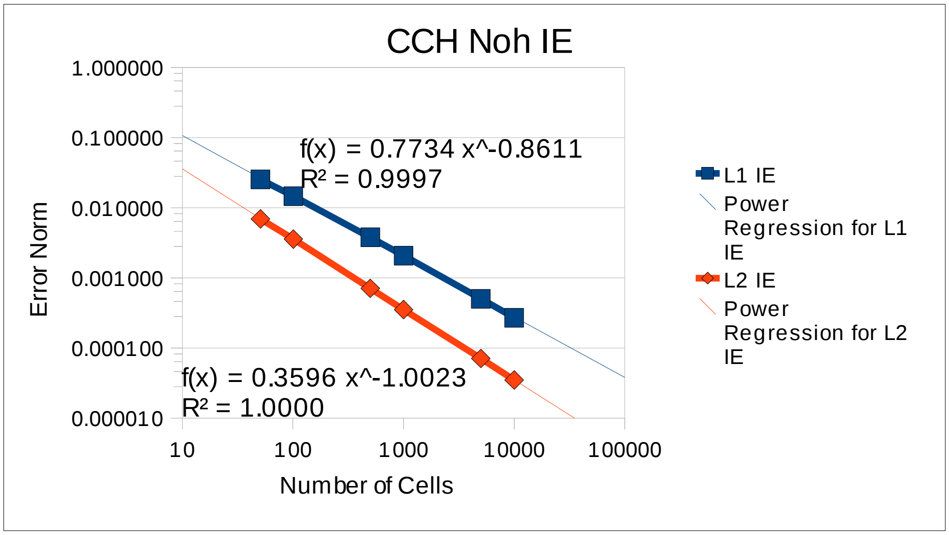

SGH Noh Pressure
Error Norm
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
10
100
1000
10000
100000
Number of Cells
f(x) = 16.9188 x^-0.9888
R² = 1.0000
f(x) = 6.4584 x^-0.9235
R² = 0.9977
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


SGH Noh IE
Error Norm
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
10
100
1000
10000
100000
Number of Cells
f(x) = 1.4122 x^-0.7903
R² = 0.9975
f(x) = 0.7030 x^-0.9314
R² = 0.9980
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

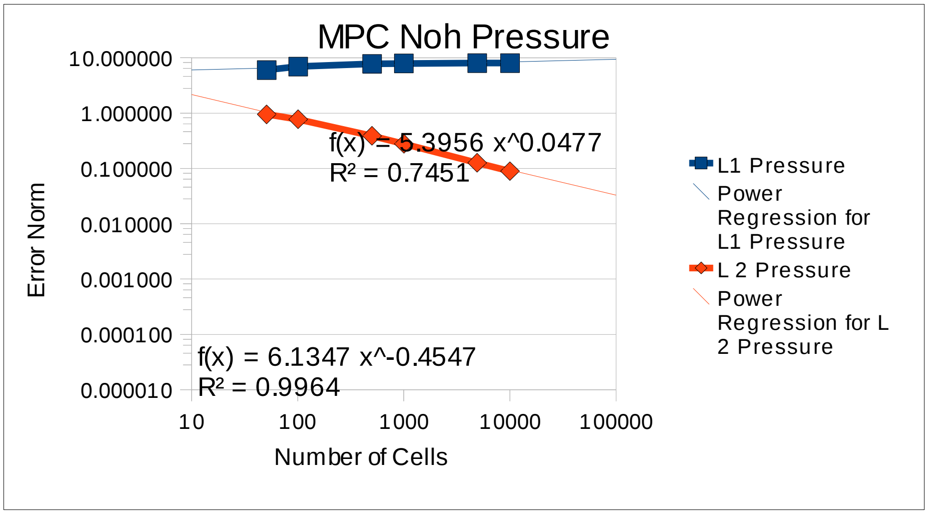
Noh Spherical

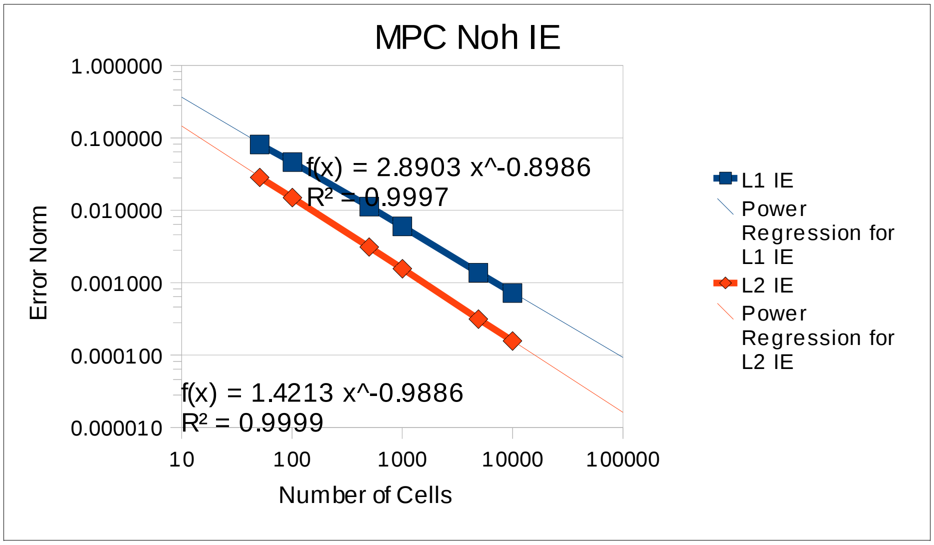
CCH



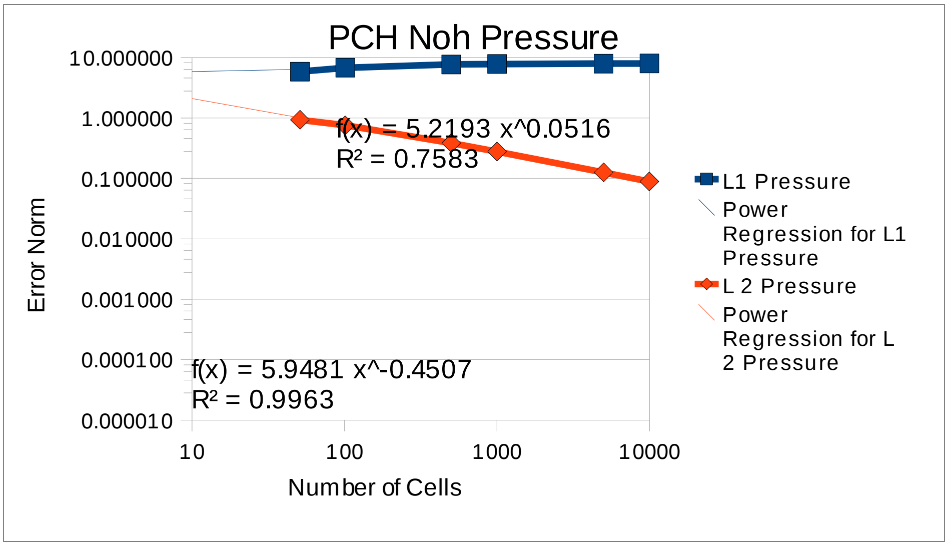


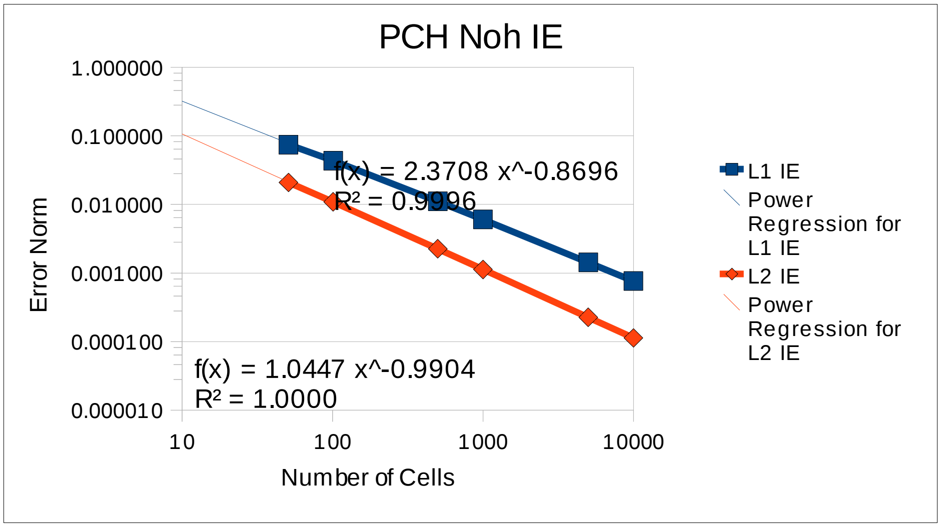
MPC


MPC Noh Pressure
10.000000
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 5.3956 x^0.0477
R² = 0.7451
f(x) = 6.1347 x^-0.4547
R² = 0.9964
10
100
1000
10000
100000
Number of Cells
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure

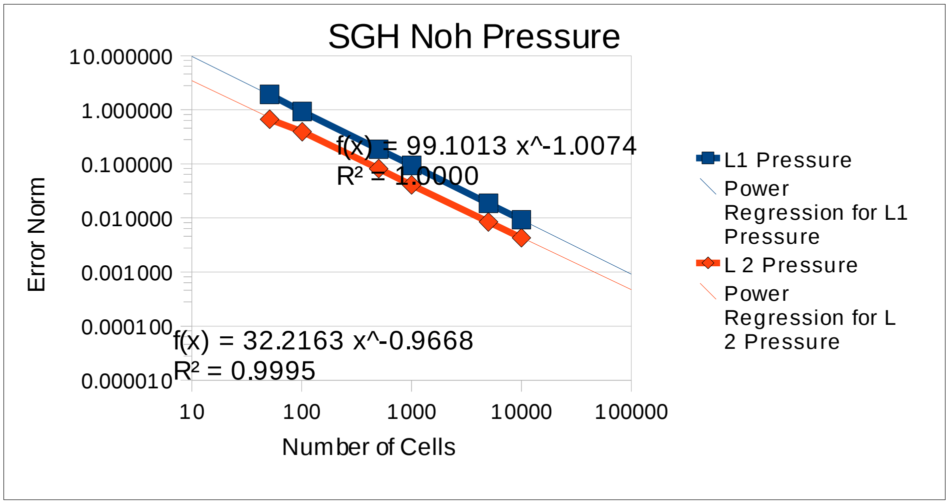

MPC Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
f(x) = 2.8903 x^-0.8986
R² = 0.9997
f(x) = 1.4213 x^-0.9886
R² = 0.9999
10
100
1000
10000
100000
Number of Cells
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

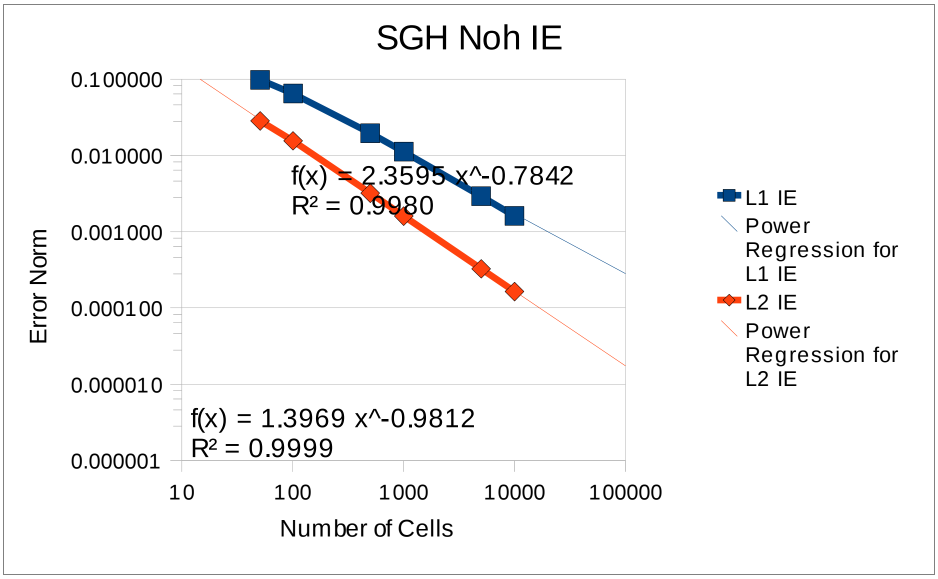
PCHA


PCH Noh Pressure
10.000000
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
Number of Cells
f(x) = 5.2193 x^0.0516
R² = 0.7583
f(x) = 5.9481 x^-0.4507
R² = 0.9963
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


PCH Noh IE
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
Number of Cells
f(x) = 2.3708 x^-0.8696
R² = 0.9996
f(x) = 1.0447 x^-0.9904
R² = 1.0000
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

SGH


SGH Noh Pressure
10.000000
1.000000
0.100000
0.010000
0.001000
0.000100
0.000010
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 99.1013 x^-1.0074
R² = 1.0000
f(x) = 32.2163 x^-0.9668
R² = 0.9995
L1 Pressure
Power Regression for L1 Pressure
L 2 Pressure
Power Regression for L 2 Pressure


SGH Noh IE
0.100000
0.010000
0.001000
0.000100
0.000010
0.000001
Error Norm
10
100
1000
10000
100000
Number of Cells
f(x) = 2.3595 x^-0.7842
R² = 0.9980
f(x) = 1.3969 x^-0.9812
R² = 0.9999
L1 IE
Power Regression for L1 IE
L2 IE
Power Regression for L2 IE

Input Files

Sod Problem

```
$general
mats=2
np=101
L=100  (Radius in Cylindrical and Spherical)
init-ie?=1 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    (0: No, the point values will be initialized using the spacial variation
                          from user input.)
                    (1: Yes, the point values will be initialized using the average of the
                          cell values.)
BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=0.0     (Boundary Velocity at start, used when fixed at left)
BCu2=0.0     (Boundary Velocity at end, used when fixed at right)
Method=CCH   (CCH, SGH, PCH, or MPC [Modified PCH = Shifted CCH] )

(Options for PCH)
VelOpt=0     (0: Computes the density and total energy change with the averaged point
                    velocities)
             (1: Computes the density and total energy change with the Riemann velocities)
(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spatially averaged velocities)
NodePosOpt=0 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                    at the end of the time integration loop)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=0.0
p=1.0
rho=1.0
ie=2.5
x1=0
x2=50
gamma=1.4
$end

$mat2
u=0.0
p=1.0
rho=0.125
ie=2.0
x1=50
x2=100
gamma=1.4
$end

$IO
dt0=0.0000000000001
tstop=20.1
dtdump=20
CFL=0.03
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Piston

```
(Note: Use parentheses to enclose comments)
$general
mats=1
np=51
L=1
init-ie?=0 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    1: Yes, the point values will be initialized using the average of the
cell values.
                    0: No, the point values will be initialized using the spacial variation
from user input.
BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=1.0     (Boundary Velocity at start, used when fixed at left)
BCu2=0.0     (Boundary Velocity at end, used when fixed at right)
Method=CCH

(3 Options for PCH)
VelOpt=0     (0: Computes the density and total energy change with the averaged point
velocities)
             (1: Computes the density and total energy change with the Riemann velocities)

(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spacially averaged velocities)
NodePosOpt=1 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                    at the end of the time integration loop)
             (2: computes new point_dx values based on 0.5*[point_u[i+1]-point_u[i-1]].
                    The point_x values are updated the same as in NodePosOpt=0.
                    This should behave the same as NodePosOpt=0.)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=0.0
p=0.0
rho=1.0
ie=0
x1=0
x2=1
gamma=1.66666666667
$end

$IO
dt0=0.0000000001
tstop=0.61
dtdump=0.6
CFL=0.03
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Noh

```
(Note: Use parentheses to enclose comments)
$general
mats=1
np=51
L=1
init-ie?=0 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    1: Yes, the point values will be initialized using the average of the
cell values.
```

```
                   0: No, the point values will be initialized using the spacial variation
from user input.
BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=0.0     (Boundary Velocity at start, used when fixed at left)
BCu2=-1.0    (Boundary Velocity at end, used when fixed at right)
Method=CCH

(3 Options for PCH)
VelOpt=0     (0: Computes the density and total energy change with the averaged point
velocities)
             (1: Computes the density and total energy change with the Riemann velocities)

(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spacially averaged velocities)
NodePosOpt=1 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                   at the end of the time integration loop)
             (2: computes new point_dx values based on 0.5*[point_u[i+1]-point_u[i-1]].
                   The point_x values are updated the same as in NodePosOpt=0.
                   This should behave the same as NodePosOpt=0.)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=-1.0
p=0.0
rho=1.0
ie=0
x1=0
x2=1
gamma=1.66666666667
$end

$IO
dt0=0.0000000001
tstop=0.61
dtdump=0.6
CFL=0.03
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Sedov

```
(Note: Use parentheses to enclose comments)
$general
mats=2
np=61
L=1.2  (Radius in Cylindrical and Spherical)
init-ie?=1 (1 to use prescribed internal energy, 0 to use prescribed pressure)
init_from_cell=0    (0: No, the point values will be initialized using the spacial variation
from user input.)
                   (1: Yes, the point values will be initialized using the average of the
                          cell values.)

BC1=0        (0 or 1 for reflected or free boundary condition, respectively)
BC2=0
BCu1=0.0     (Boundary Velocity at start, used when fixed at left)
BCu2=0.0     (Boundary Velocity at end, used when fixed at right)
Method=CCH   (CCH, SGH, PCH, or MPC [Modified PCH = Shifted CCH] )

(3 Options for PCH)
```

```
VelOpt=1     (0: Computes the density and total energy change with the averaged point 
velocities)
             (1: Computes the density and total energy change with the Riemann velocities)

(Note: If VelOpt=1, the following options are inconsequential)
AvgOpt=0     (0: Uses the spacial average of the point velocities)
             (1: Uses the time and spacially averaged velocities)
NodePosOpt=1 (0: Updates the Nodal Positions based on the nodal velocities)
             (1: Updates the nodal positions based on the average of of the CV boundaries
                   at the end of the time integration loop)
             (2: computes new point_dx values based on 0.5*[point_u[i+1]-point_u[i-1]].
                   The point_x values are updated the same as in NodePosOpt=0.
                   This should behave the same as NodePosOpt=0.)
Coordinate_System=car  (car=Cartesian, cyl=Cylindrical, sph=Spherical)
$end

$mat1
u=0.0
p=0.0
rho=1.0
ie=15  (Extensive IE=0.3 MBar cc  Volume=0.02 cc [that's 0.3/1])
x1=0
x2=0.02
gamma=1.66666666666667
$end

$mat2
u=0.0
p=0.0
rho=1.0
ie=0.0
x1=0.02
x2=1.2
gamma=1.66666666666667
$end

$IO
dt0=0.0000000000001
tstop=1.01
dtdump=1.0
CFL=0.2
CFLV=0.01 (Recommended: between 0.01 and 0.05. Never above 0.1)
$end
```

Selected Portions of Source Code

CCH Riemann Solver:

```
avg=(cell_u[i-1]+cell_u[i])/2.0;

                   c=sqrt(cell_gam[i-1]*cell_p[i-1]/cell_rho[i-1]);    // c for cell on 
left
                   if (c<0.000000001 || c!=c) { c=0.000000001; }       //If c<1e-9 or 
c=nan,set a minimum value for c

                   /* Calculate Compression-Aware mu on the left */
                   if ((cell_u[i]-cell_u[i-1])>0)
                   {
                          /* Expansion */
                          point_mu[0]=cell_rho[i-1]*c;
```

```
                    }
                    else
                    {
                           /* Compression */
                           point_mu[0]=cell_rho[i-1]*c+cell_rho[i-1]*
                                 ((cell_gam[i-1]+1.0)/2.0)*fabs(avg-cell_u[i-1]); //mu on
                                                                                     left
                    }


                    c=sqrt(cell_gam[i]*cell_p[i]/cell_rho[i]);    // c for cell on right
                    if (c<0.000000001 || c!=c) { c=0.000000001; } //If c<1e-9 or c=nan,set a
                                                                        minimum value for c

                    /* Calculate Compression-Aware mu on the right */
                    if ((cell_u[i]-cell_u[i-1])>0)
                    {
                           /* Expansion */
                           point_mu[1]=cell_rho[i]*c;
                    }
                    else
                    {
                           /* Compression */
                           point_mu[1]=cell_rho[i]*c+cell_rho[i]*
                                 ((cell_gam[i]+1.0)/2.0)*fabs(avg-cell_u[i]); //mu on right
                    }

                    /* Compute u* at each point */
                    point_u[i]=(cell_p[i-1]*point_normal[0]+cell_p[i]*
                           point_normal[1]+point_mu[0]*cell_u[i-1]
                           +point_mu[1]*cell_u[i])/(point_mu[0]
                           +point_mu[1]);

                    /* Compute p* at each point */
                    point_p[i]=point_mu[1]*(point_u[i]-cell_u[i])-cell_p[i]*
                           point_normal[1];

                    /* Check to verify Riemann Pressures are equal and opposite */
                    point_pstar_check=point_mu[0]*(point_u[i]-cell_u[i-1])-cell_p[i-1]*
                           point_normal[0];
                    if (fabs(point_p[i]+point_pstar_check)>pstar_tol)
                    {      check++;     }
```

Runge-Kutta Time Integration Solution Loop:

```
do
       {
             alpha=1.0/(nstage+1.0-istage);  // Establish the coefficient
             CCHSolveRHS(mass,alpha,SphAvgOpt);
             istage++;    // increase the RK cycle number

       } while (istage<=nstage);
```

CCH Conservation Equations:

```
for (i=0;i<nz;i++)
       {
             if (PiOpt==1) Area=GetAreaCCH(i);
```

```
              else if (PiOpt==0) Area=GetAreaCCH_NoPi(i);

              /* Calculate Forces on either side of the control volume */
              Fstar[0]=-point_p[i]*point_normal[1]*Area;    // Force on Left side of CV
              Fstar[1]=-point_p[i+1]*point_normal[0]*Area;  // Force on Right side of CV

              v[i]=(1.0/mass[i])*(Fstar[0]+Fstar[1]);
              e[i]=(1.0/mass[i])*(Fstar[0]*point_u[i]+Fstar[1]*point_u[i+1]);
       }
/*  +++++++++++++++End Calculate RHS ++++++++++++++++++++++++++++++ */
/* +++++++++++++++Begin Time Step Forward ++++++++++++++++++++++++++ */
       for (i=0;i<np;i++)
       {
              /* Calculate new point_x values */
              point_x[i]=point_x0[i]+alpha*dt*point_u[i];
       }
       for (i=0;i<nz;i++)
       {
              /* Calculate new dx */
              cell_dx[i]=point_x[i+1]-point_x[i];

              /* Calculate new rho */
              if (PiOpt==1) Volume=GetVolumeCCH(i);
              else if (PiOpt==0) Volume=GetVolumeCCH_NoPi(i);
              cell_rho[i]=mass[i]/Volume;

              /* Calculate new u */
              cell_u[i]=cell_u0[i]+alpha*dt*v[i];

              /* Calculate new total energy */
              cell_te[i]=cell_te0[i]+alpha*dt*e[i];

              /* Calculate new internal energy */
              cell_ie[i]=cell_te[i]-0.5*pow(cell_u[i],2.0);

              /* Calculate new pressure using EOS-Gamma Law Gas */
              cell_p[i]=cell_rho[i]*(cell_gam[i]-1.0)*cell_ie[i];
       }
```