A Hybrid Adaptive ENO Scheme

Robert Bruce Bauer
Department of Mathematics
Center for Applied Mathematical Sciences
University of Southern California
Los Angeles, CA 90089-1113
rbbauer@cams.usc.edu

April 29, 1997

Key Words: ENO Scheme, Adaptive Grid, Hybrid Scheme
Subject Classification: 65P05,65P45

Running title: A Hybrid Adaptive ENO Scheme

Send proofs to:

Dr. Bauer
Center for Applied Mathematical Sciences
Department of Mathematics
University of Southern California
1042 West 36th Place, DRB 155
Los Angeles, California 90089–1113

phone: (213) 740 – 9806
fax: (213) 740 – 2424

e-mail: rbbauer@cams.usc.edu

Abstract

This paper describes a new Hybrid Adaptive ENO scheme for partial differential equation in conservative form. The scheme is hybrid because it combines finite difference approximations for points away from the shock and ENO approximations near the shock. The method is adaptive in two ways. The first is that this method changes the computational grid in order to maintain a uniform approximation. Secondly, the method determines where to use the conservative ENO approximation to the derivative. The combination of these produces a quick algorithm for the solution of conservation laws.

1 Introduction

The study of compressible flows requires a special class of algorithms specifically designed to solve problems with discontinuous solutions. Algorithms in this class attempt to be high order without causing spurious oscillations. Spectral methods have been adapted by using filtering to inhibit oscillations. Van Leer used a hybrid of a high order scheme in smooth regions and a monotone scheme near shocks [13]. Shu and Osher developed a class of high-order essentially non-oscillatory conservative schemes (ENO) [11, 12]. ENO schemes are highly adaptive schemes which use an optimal stencil at each grid point. ENO schemes come in two different formulations, cell-average and point-wise. Cell-averages use a very intuitive method based on the cell averages of the function. In Bauer an adaptive grid cell-averaged ENO scheme was developed and analyzed [1]. However, the cell-average ENO formulation on regular grids is prohibitively expensive when applied in higher dimensions and even more expensive on non-uniform grids.

Point-wise ENO avoids the dimensional complications but is only applicable on a uniform grid. Unfortunately, ENO schemes are substantially more expensive to implement than finite difference methods. At every grid point, several *if-then* statements must be evaluated to decide on the proper stencil. Ami Harten [5, 4, 7, 8, 6, 9] and Lee Jameson [10] developed methods to reduce computations using wavelets and multi-resolution analysis. Harten used wavelet coefficients to locate points where a cheaper/faster method could be used to compute the derivative. His method still used a uniform grid, but only computed the costly ENO approximations at a small number of points. Jameson did not examine the problem of high-order conservative schemes, but used wavelets to determine an optimal computational grid for a finite difference computation. These ideals were joined together in Bauer [1] and tests show substantial reductions in computational costs and time. The method presented here breaks free from the confines of cell-averages at a price. The method is no longer conservative. Harabetian and Pego presented another non-conservative hybrid shock capturing scheme [3]. This method is distinctly different in a couple of ways; it uses an adaptive grid and determines which points to use ENO approximations on differently. In Harabetian and Pego, it was surmised that "if switching is prohibited too close to shocks," the method will be accurate in smooth regions and have the correct shock speed. This Hybrid Adaptive ENO uses an adaptive finite difference scheme in smooth areas and uses a conservative ENO scheme around the shocks. Sufficient care has been taken to ensure that all shocks are well within the region of ENO approximations.

This paper will first cover the basics including the general form of the Differential Equation and formulation of ENO schemes. Then I will discuss the computational grids which form the corner stone of the Hybrid Adaptive ENO method. This is followed by a discussion of when and where ENO computations are used to ensure an accurate solution. The next two sections cover how and when to change the computational grid and two ways to change the grid throughout the computation. Finally, examples are presented showing that the method is able to compute and maintain a sufficient computational grid.

2 Essentially Non-Oscillatory Schemes (ENO)

Consider the solution of conservative PDEs written in the form

$$\frac{\partial}{\partial t}u(x,t) = \frac{\partial}{\partial x}f(u(x,t)). \tag{2.1}$$

Also consider the uniform grid point-wise ENO scheme written in terms of it's numerical flux, $\hat{f}_{j+1/2}$

$$u_t(x_j,t) = \frac{1}{\Delta}\left(\hat{f}_{j+1/2} - \hat{f}_{j-1/2}\right) \tag{2.2a}$$

$$\hat{f}_{j+1/2} = \mathbf{F}\left[\ldots, u(x_{j-1},t), u(x_j,t), u(x_{j+1},t), u(x_{j+2},t), \ldots\right] \tag{2.2b}$$

where $\mathbf{F}$ is consistent and Lipschitz continuous in each of it's arguments and $\Delta = x_{j+1} - x_j$. Shu and Osher [11] developed a non-oscillatory method to compute high-order approximations to $\hat{f}_{j+1/2}$. Implicitly define $h(x)$ by

$$f(u(x)) = \frac{1}{\Delta}\int_{x-\Delta/2}^{x+\Delta/2} h(\xi)d\xi. \tag{2.3}$$

Differentiating (2.3) with respect to x leads to

$$\frac{\partial f(u(x))}{\partial x} = \frac{h(x+\Delta/2) - h(x-\Delta/2)}{\Delta}. \tag{2.4}$$

Comparing (2.2a) and (2.4) results in

$$\hat{f}_{j+1/2} = h(x_{j+1/2}). \tag{2.5}$$

By using the primitive of $h(x)$ and the knowledge of $f(u(x_j,t))$, a high order approximation of $h(x)$ can be computed. However (2.3) requires that the grid be uniform with spacing Δ. If the grid is non-uniform, computation of $h(x)$ is more difficult and the scheme is no longer conservative. Since the scheme is no longer conservative, an easy to compute finite difference approximation to the derivative is used.

Computing a high order approximation is not difficult, but finding a high order ENO approximation which is not oscillatory is difficult and expensive. To compute a p^{th} order ENO approximation of $\hat{f}_{j+1/2}$ will require p *if-then* statements to be executed at each grid point and at every time step. These *if-then* statements substantially slow down the algorithm. This paper, as in [1], use an adaptive grid ENO scheme to reduce the number of ENO approximations while maintaining accuracy.

3 Grid and Mesh

The algorithm has two different sets of points. The first set is the computational grid and is the set of points $x_j \in [0,1]$ where the value of the function $u(x,t)$ is known. Throughout assume the points are ordered $x_j < x_{j+1}$. And, while computationally this is not practical, pointers are used to keep track of data points. This set of computational points is call the **grid**. There are specific requirements on which sets of points x_j will be a usable **grid**. To understand which collections of points x_j are usable examine the other set of points.

The other set of points is call the **mesh**, and will be used for maintaining the **grid**, record keeping and changing the **grid**. The **mesh** consists of evenly spaced points which can possibly be used for computations. The **mesh** is a subset of the **grid**.

The **mesh** is defined as a hierarchy of levels. Every **mesh** point on level 1 is the child of a **mesh** point on level 0 and the parent of two children on level 2. Likewise every **mesh** point on level 2 is the child of a **mesh** point on level 1 and the parent of two children on level 3. (See figure 1)

3.1 Mesh Properties

The following values are recorded for each **mesh** point in order to maintain the **mesh** and the **grid**. Figure 2 shows the values of these properties for a small grid with 4 hierarchal levels.

level(j) equals the hierarchal level on which the **mesh** point lies. The points on level 0 are the most basic points which *must* be used in each computation. Points are added on higher levels giving rise to higher accuracy.

mesh(j) refers to the j^{th} **mesh** point and x_j will refer to a **grid** point.

parent(j) equals the value of the **mesh** point which gave life to **mesh**(j). If **mesh**(j) is on the **grid**, then **parent**(j) must be on the **grid**. Points on level 0 will not have parents.

child(left,j) & child(right,j) are the values of the two **mesh** points which are the children of **mesh**(j). If the algorithm determines that the **grid** needs to be refined around the point **mesh**(j), then the points **child**(left,j) and **child**(right,j) are added. Points on the finest level will not have any children. Note that **parent**(**child**($left, j$)) $= j$.

Next define $\mathcal{L}_k$ as the set of points which make-up level k. Let N be the number of points on the coarsest level $\mathcal{L}_0$. Then equations (3.1) define the points in each of the levels $\mathcal{L}_d$ and (3.1d) shows how the levels join together to form a **mesh** which is uniformly spaced.

$$\mathcal{L}_0 = \{x_j^0 | x_j^0 = j/N, \text{ for } j = 0, N-1\} \tag{3.1a}$$
$$\mathcal{L}_1 = \{x_j^1 | x_j^1 = 1/2N + j/N, \text{ for } j = 0, N-1\} \tag{3.1b}$$
$$\mathcal{L}_d = \{x_j^d | x_j^d = 1/N2^d + j/N2^{d-1}, \text{ for } j = 0, N2^{d-1}-1\} \tag{3.1c}$$
$$\bigcup_{k=0}^{d} \mathcal{L}_k = 2^d N \qquad x_j = j/2^d N \tag{3.1d}$$

This hierarchy of levels is very similar to wavelets. However unlike wavelets, small portions of each level instead of the entire level.

There are a few additional quantities which need to be defined: (See figures 2 and 3)

sibling(j) points to the **mesh** point which has the same parent as **mesh**(j).

next(left,j) points to the closest point to the left *on the same level* as **mesh**(j).

next(right,j) points to the closest point to the right *on the same level* as **mesh**(j).

3.2 Defining the Grid

Now that the **mesh** is defined, it is much easier to describe the computational **grid**. All points $x_j^0 \in \mathcal{L}_0$ are **grid** points. For the points $x_j^k \in \mathcal{L}_{k>0}$ to be a **grid** point, x_j^k must satisfy the following **Grid Rules**:

1. if **mesh**(j) $\in$ **grid**, then **parent**(j) $\in$ **grid**

2. if **mesh**(j) $\in$ **grid**, then **sibling**(j) $\in$ **grid**

3. if **child**(left,j) $\in$ **grid**, then **next**(left,j) $\in$ **grid**
4. if **child**(right,j) $\in$ **grid**, then **next**(right,j) $\in$ **grid**

3.3 Grid & Mesh Combined

The **mesh** defines the **grid**. But now arrises the need to examine the **grid** and determine what properties it has. Consider a **mesh** with 5 levels and $N = 20$. Then the **mesh** will have 320 points and the **grid** will have between 20 and 320 points. Assume that there are P **grid** points, x_j for $j = 0 \ldots P - 1$.

Grid Rules 1 & 2 ensure that for all j, there exists an integer $w_j \in [0, 4]$, such that

$$\Delta_j = x_{j+1} - x_j = 2^{w_j}/320 = 2^{w_j} * \delta \tag{3.2a}$$

$$\delta = 1/320. \tag{3.2b}$$

In other words, the distance between neighboring points will be either 1/320, 2/320, 4/320, 8/320 or 16/320.

Grid Rules 3 & 4 result in another important relationship,

$$|w_{j+1} - w_j| \leq 1. \tag{3.3}$$

The importance of this is that $x_{j+1} - x_j$ will not vary too quickly throughout the domain. For example if the distance from x_{10} to x_{11} is 8/320, then the distance from x_{11} to x_{12} can only be 4/320, 8/320 or 16/320 which are multiples of 1/2, 1 or 2 times 8/320.

These two rules are the equivalent of requiring that each grid point should be overlapped by a courser level grid point that is not more than one level courser than itself. These requirements are similar to ones which Harten [4, 9] and Jameson used [10].

3.4 Stencils for Finite Difference Approximation

Unlike uniform grids, finite difference approximations need to be computed for many different grid spacings. The grid rules restrict how many different stencils need to be computed.

In this section consider equations for which $f'(u) \geq 0$ for all $u(x, t)$. This will allow me to only consider stencils for waves moving to the left. Stencils are created using the points x_{j-1}, x_j, x_{j+1}, & x_{j+2} which are used in turn to approximate $f_x(u(x_j, t))$.

When the requirement on $f'(u)$ does not hold, a flux splitting is used such that $f(u) = f^+(u) + f^-(u)$ where $df^+/du \geq 0$ and $df^-/du \leq 0$. One possible splitting is $f^+(u) = f(u) + \alpha u$ and $f^-(u) = f(u) - \alpha u$ where $\alpha = max_u|f'(u)|$. There are other choices for $f^+(u)$ & $f^-(u)$ which are more physical and the actual choice will depend on the equation to be solved. (See Shu and Osher [11, pages 36 & 37])

Once $f^+(u)$ & $f^-(u)$ are determined, then use the following stencils to compute $f_x^+(u)$ and use similar stencils using the points x_{-2}, x_{-1}, x_0 and x_1 to compute $f_x^-(u)$.

Grid Rules 3 & 4 also ensure the relation

$$|w_{j+1} - w_j| + |w_j - w_{j-1}| \leq 1. \tag{3.4}$$

To understand the impact of this statement, consider for example a third order, four point stencil. Knowing w_j and using (3.4), leads to only 5 possible combinations of w_{j-1}, w_j, and w_{j+1}. These 5 possibilities are labeled stencils -2 through $+2$ and shown in table 1. Furthermore consider the four points x_{j-1}, x_j, x_{j+1}, and x_{j+2}

Stencil #	w_{j-1}	w_j	w_{j+1}
-2	$w_j + 1$	w_j	w_j
-1	w_j	w_j	$w_j - 1$
0	w_j	w_j	w_j
+1	$w_j - 1$	w_j	w_j
+2	w_j	w_j	$w_j + 1$

Table 1: Stencils for third order scheme

related to w_{j-1}, w_j, and w_{j+1}. The spacing for each of these stencils between grid points are given in table 2. Now the algorithm needs only 5 sets of coefficients to

Stencil #	x_{j-1}	x_j	x_{j+1}	x_{j+2}
-2	$x_j - 2\Delta_j$	x_j	$x_j + \Delta_j$	$x_j + 2\Delta_j$
-1	$x_j - \Delta_j$	x_j	$x_j + \Delta_j$	$x_j + 3\Delta_j/2$
0	$x_j - \Delta_j$	x_j	$x_j + \Delta_j$	$x_j + 2\Delta_j$
+1	$x_j - \Delta_j/2$	x_j	$x_j + \Delta_j$	$x_j + 2\Delta_j$
+2	$x_j - \Delta_j$	x_j	$x_j + \Delta_j$	$x_j + 3\Delta_j$

Table 2: Grid spacing for each stencil

compute the derivative. To compute $f_x(u(x_j, t))$, use the formula

$$f_x(u(x_j,t)) = \frac{1}{\Delta_j} \sum_{k=-1}^{2} c_k\, f(u(x_{j+k},t)) + \mathcal{O}(\Delta_j^3) \tag{3.5}$$

Stencil #	c_{-1}	c_0	c_1	c_2
-2	$-1/12$	-1	$4/3$	$-1/4$
-1	$-3/10$	$-2/3$	$3/2$	$-8/15$
0	$-1/3$	$-1/2$	1	$-1/6$
+1	$-16/15$	$1/2$	$2/3$	$-1/10$
+2	$-3/8$	$-1/3$	$3/4$	$-1/24$

Table 3: Coefficients for third order approximations

and the coefficients for the appropriate stencil found in table 3. This small set of coefficients can now be stored and used as needed.

3.4.1 Higher Order

The above stencils are derived for a third order, four point stencil. There are two ways to modify this method for higher order. Consider a fourth order, five point stencil. The first involves allowing 6 more stencils which are shown in table 4.

Stencil #	w_{j-2}	w_{j-1}	w_j	w_{j+1}
-5	$w_j - 1$	w_j	w_j	$w_j - 1$
-4	$w_j - 1$	w_j	w_j	$w_j + 1$
-3	$w_j + 1$	$w_j + 1$	w_j	w_j
-2	$w_j + 1$	w_j	w_j	w_j
-1	w_j	w_j	w_j	$w_j - 1$
0	w_j	w_j	w_j	w_j
+1	w_j	w_j	w_j	$w_j + 1$
+2	$w_j - 1$	w_j	w_j	w_j
+3	$w_j - 1$	$w_j - 1$	w_j	w_j
+4	$w_j + 1$	w_j	w_j	$w_j - 1$
+5	$w_j + 1$	w_j	w_j	$w_j + 1$

Table 4: Stencils for fourth order scheme

This requires 6 more stencils than before and two stencils have three different spacings within them (stencils #-4 and #4). Previously there were only two different spacings within each stencil. This difference is reflected in the equation

$$|w_{j+1} - w_j| + |w_j - w_{j-1}| + |w_{j-1} - w_{j-2}| \leq 2 \tag{3.6}$$

where the right hand side determines the total number of changes in w_j within the stencil.

Alternatively, additional **Grid Rules** can be defined. This will reduce the possible stencils from 11 to 7. Moreover, each stencil permits only two different grid spacings. Consider the five point stencil again with these additional two **Grid Rules**:

5. if **child**(left,x_j) $\in$ **grid**, then **next**(left,**next**(left,x_j)) $\in$ **grid**

6. if **child**(right,x_j) $\in$ **grid**, then **next**(right,**next**(right,x_j)) $\in$ **grid**

This enforces

$$|w_{j+1} - w_j| + |w_j - w_{j-1}| + |w_{j-1} - w_{j-2}| \leq 1 \tag{3.7}$$

and eliminates stencils #-4, #-5, #4 and #5.

4 ENO Implementation

The scheme will use an ENO approximation to the derivative on the finest grid. Section 5 will discuss the **grid** modifying algorithms which ensure that all discontinuities lie within a region of high resolution. But for now, consider only the implementation of ENO on a uniformly fine mesh.

Define a fine grid point as a **grid** point satisfying

$$\Delta_{j-1} = \delta = \Delta_j. \tag{4.1}$$

And define a region of uniform high resolution as a section where fine grid points are grouped together. These regions will be where an ENO approximation can be applied.

4.1 ENO at a single point

In order to compute $f_x(u(x_j))$, both $\hat{f}_{j-1/2}$ and $\hat{f}_{j+1/2}$ need to be computed. To find a ρ^{th} order solution, it is required that

$$\hat{f}_{j+1/2} = h(x_{j+1/2}) + \mathcal{O}(\delta^{\rho+1}). \tag{4.2}$$

For a $(\rho+1)^{st}$ order ENO reconstruction of $h(x)$, $\rho+1$ points on either side of $x_{j-1/2}$ and $x_{j+1/2}$ are required. Therefore, to compute the derivative at x_j, the points x_k for $k = (j-\rho-1)\ldots(j+\rho+1)$ must be fine grid points.

In other words for any point x_j such that $x_{j+\rho+1} - x_{j-\rho-1} = (2\rho+2)\delta$ use an ENO approximation to the derivative.

5 Grid Modifications

This section describes the algorithms which will modify the computational **grid**. The analysis is for the specific case of a third order scheme. A brief overview of the construction of the stencils and results from this section for a fifth order scheme are given in Appendix A.

5.1 Accuracy

This method relies on two algorithms to change the grid. The first adds **grid** points to ensure the scheme maintains accuracy. How does one "maintain accuracy?"

Consider the 1-D wave equation

$$u_t = u_x \tag{5.1}$$

with $u(x,t)$ a C^∞ function, and the semi-discrete approximation to $u_t = u_x$,

$$\frac{d}{dt}u(x_j,t) = G_j(u(x,t)) \tag{5.2}$$

where $G_j(u(x,t))$ is a third order approximation to $u_x(x,t)$,

$$G_j(u(x,t)) = c_{-1}u(x_{j-1},t) + c_0 u(x_j,t) + c_1 u(x_{j+1},t) + c_2 u(x_{j+2},t). \tag{5.3}$$

The coefficients, c_k, are the same ones computed in section 3. Using a Taylor's expansion about x_j,

$$\begin{aligned} G_j(u(x,t)) = u_x(x_j,t) + \frac{1}{24}\Big[& c_{-1}\,(x_{j-1}-x_j)^4 u^{(iv)}(\xi_1) \\ & + c_1\,(x_{j+1}-x_j)^4 u^{(iv)}(\xi_2) + c_2\,(x_{j+2}-x_j)^4 u^{(iv)}(\xi_3)\Big] \end{aligned} \tag{5.4}$$

for some $\xi_1, \xi_2, \xi_3 \in [x_{j-1}, x_{j+2}]$. Then the local truncation error is

$$\begin{aligned} e_j = \frac{1}{24}\Big| & c_{-1}\,(x_{j-1}-x_j)^4 u^{(iv)}(\xi_1) + c_1\,(x_{j+1}-x_j)^4 u^{(iv)}(\xi_2) \\ & c_2\,(x_{j+2}-x_j)^4 u^{(iv)}(\xi_3)\Big| \end{aligned} \tag{5.5}$$

Using

$$x_{j-1} - x_j = -\delta 2^{w_{j-1}} \tag{5.6a}$$

$$x_{j+1} - x_j = \delta 2^{w_j} \tag{5.6b}$$

$$x_{j+2} - x_j = \delta 2^{w_j} + \delta 2^{w_{j+1}} \tag{5.6c}$$

and defining

$$U_j = \max_{\xi\in[x_{j-1},x_{j+2}]} |u^{(iv)}(\xi)|, \tag{5.7}$$

the error can be bounded by

$$e_j \le \frac{\delta^4}{24}\left[|c_{-1}|\,16^{w_{j-1}} + |c_1|\,16^{w_j} + |c_2|\,(2^{w_{j+1}} + 2^{w_j})^4\right] U_j. \tag{5.8}$$

Using the coefficients found in section 3, a bound for each different stencil can be found

$$e_j \le \frac{\Delta_j^3}{24}\, B(w_{j-1}, w_j, w_{j+1})\, U_j \tag{5.9}$$

where the value of $B(w_{j-1}, w_j, w_{j+1})$ is given in table 5. Since all of these are less

Stencil #	$B(w_{j-1}, w_j, w_{j+1})$
-2	20/3
-1	9/2
0	4
+1	7/3
+2	9/2

Table 5: Bounds $B(w_{j-1}, w_j, w_{j+1})$

than 20/3, the error can be bounded by

$$e_j \le \frac{5\Delta_j^3}{18}\, U_j. \tag{5.10}$$

The error has two contributing factors, Δ_j^3 and U_j. The goal is to limit the error committed on any time step to a certain level. Since U_j is beyond our control, use Δ_j to control the error. Then if $e_j > Tolerence$ refine the grid and reduce the error. $Tolerence$ is a user defined variable which controls how accurate the Hybrid Adaptive ENO scheme should approximate the fine grid ENO scheme. For example if $Tolerence = .001$, then the local truncation error for each step should be less than .001.

5.2 Adding Points

Let $\mathcal{U}_p(\bar{x}; y_{-2}, y_{-1}, y_0, y_1, y_2)$ be an approximation to $u^{(p)}(\bar{x})$ using the points $y_{-2}, y_{-1}, y_0, y_1, y_2$. The computation of $\mathcal{U}_p$ must be a quick computation because it will be used often.

Define $i_a(j)$ as an estimate to $u^{(iv)}(4)$ using the points $x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+2}$

$$i_a(j) = \left|\mathcal{U}_4(x_j; x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+2})\right|. \tag{5.11}$$

Then define $I_a(j)$ as

$$I_a(j) = \frac{5\Delta_j^3}{18}\left[\max_{k=j-\eta(a),j+\eta(a)} \{i_a(k)\}\right] \tag{5.12}$$

for some $\eta(a)$. If $\eta(a) = 2$, then $I_a(j)$ will bound the truncation error at x_j.

In order to anticipate the movement of data, $\eta(a)$ is chosen larger than the value of 2 required to estimate e_j [approx. 5]. This leads to a simple rule for adding points to the **grid**. If $I_a(j) > TOL_a$, then refine the **grid** by adding the points **child**$(left, j)$ and **child**$(right, j)$.

5.3 Removing Points

The second algorithm deletes **grid** points so that the method uses as few a possible. For the add algorithm above, the grid was tested at the point x_j and if required the children of the point x_j were added. For the remove algorithm, the grid is tested at the point x_j and if required the children of x_j are removed.

The indicator for removal uses x_j and a set of 4 points nearby the children of x_j, which are the points x_{j-1} and x_{j+1}. In determining whether to remove the points x_{j-1} and x_{j+1}, avoid using them to compute $i_r(j)$. Define

$$i^-(j) = |\mathcal{U}_0(x_{j-1}; x_{j-3}, x_{j-2}, x_j, x_{j+2}, x_{j+3}) - u(x_{j-1}, t)| \tag{5.13a}$$

$$i^+(j) = |\mathcal{U}_0(x_{j+1}; x_{j-3}, x_{j-2}, x_j, x_{j+2}, x_{j+3}) - u(x_{j+1}, t)| \tag{5.13b}$$

$$i_r(j) = i^-(j) + i^+(j). \tag{5.13c}$$

$\mathcal{U}_0(x_{j-1})$ is an approximation of $u(x_{j-1}, t)$ and $\mathcal{U}_0(x_{j+1})$ is a similar approximation of $u(x_{j+1}, t)$. Therefore $i_r(j)$ is the total approximation error at the two points x_{j-1} and x_{j+1} if they are not elements of the computational **grid**.

Again to anticipate $u(x, t)$, define $I_r(j)$ as

$$I_r(j) = \max_{k=j-\eta(r), j+\eta(r)} \{i_r(k)\} \tag{5.14}$$

for some $\eta(r)$ [approx 5].

The two children of x_j will not be removed if either x_{j-1} or x_{j+1} has children, or the resulting grid fails to satisfy all the **Grid Rules**.

5.3.1 ENO and the Fine Grid

Necessary to Hybrid Adaptive ENO is ensuring that the **grid** changing algorithms will always refine and never coarsen near discontinuities. In order to ensure this, show that both $i_a(j)$ and $i_r(j)$ will be large near discontinuities. If $i_a(j)$ and $i_r(j)$ are large near discontinuities, then both $I_a(j)$ and $I_r(j)$ will be large near discontinuities and the **grid** will be refined and not coarsened near discontinuities. The increased values of $\eta(a)$ and $\eta(r)$ will expand the fine grid regions and ensure that the shocks will lie within the ENO regions.

Harten [6, 9] showed for a function with a discontinuity in the q^{th} derivative, the finite difference approximation to the p^{th} derivative behaves like $\mathcal{O}(\delta^{q-p})$.

Then for this approximation,

$$i_a(j) \; = \mathcal{O}(\delta^{q-4}) \qquad i_r(j) = \mathcal{O}(\delta^{q-5}). \tag{5.15}$$

Near discontinuities in the solution, $i_a(j) = \mathcal{O}(\delta^{-4})$ and $i_r(j) = \mathcal{O}(\delta^{-5})$. Additionally, for discontinuities in the first derivative $i_a(j) = \mathcal{O}(\delta^{-3})$ and $i_r(j) = \mathcal{O}(\delta^{-4})$. Therefore both $I_a(j)$ and $I_r(j)$ will be large and ensure that a fine grid is used near shocks.

5.4 Systems of equations

If solving a system of equations, these indicators need to be modified only slightly. $\mathcal{U}_p$ will now be a vector valued approximation $d^{(p)}\mathbf{u}/dx^{(p)}$. Then (5.11) and (5.13) need to be replaced by a suitable norm of the vectors in each equation. While the l_1 or l_∞ might seem to be logical norms to use, the best choice is l_2 because it is quick to compute. Both l_1 and l_∞ require *if-then* statements.

6 Special Changes

It is not sufficient to have an algorithm which can find the best **grid** for the present time step. The algorithms must be able to predict which **grid** will be required. This involves two different methods to predict the **grid** for the function $u(x, t+\tau)$. The first is easily applied to systems of equations, but the second one requires considerable extra work and may not be efficient.

6.1 Directional Insight

Consider the equation

$$u_t = a(x,t)\, u_x \tag{6.1}$$

If $a(x,t) < 0$, then the waves move to the right. Similarly if $a(x,t) > 0$, then the waves move to the left. Of course the waves will move different directions within the same problem. But at any one point it can easily be determined which direction things are moving. Recall the original equation

$$u_t(x,t) = f_x(u(x,t)) \tag{6.2}$$

and rewrite this in the non-conservative form

$$u_t(x,t) = f'(u)\, u_x(x,t). \tag{6.3}$$

Use this equation to determine which direction things are moving.

6.2 Biased Add Indicator

Define a Biased Add Indicator, $I_b(j)$ as

$$I_b(j+k) = \max_{k\in\mathcal{K}_j}(I_a(j+k), I_a(j)) \tag{6.4a}$$

$$\mathcal{K}_j = \begin{cases} \{0\} & \text{if } x_j \text{ is not a fine grid point} \\ \{0\dots\eta(b)\} & \text{if } x_j \text{ is a fine grid point \& } f'(u(x_j,t)) \le 0 \\ \{-\eta(b)\dots 0\} & \text{if } x_j \text{ is a fine grid point \& } f'(u(x_j,t)) \ge 0 \end{cases} \tag{6.4b}$$

for some $\eta(b)$ [approx 5]. This procedure spreads the influence of a high value of $I_a(j)$ downstream if the point x_j is a fine grid point. This will ensure that additional fine grid points will be added ahead of any shock and that "switching [from ENO to finite difference] is prohibited too close to shocks."

6.2.1 Systems of Equations

To adapt this for systems, recall that ENO schemes must be applied to the characteristic field. The characteristic field is also exactly what is required to apply the Biased Add Indicator.

Let $A_{j+1/2}$ become "average" Jacobian of $\mathbf{f}(\mathbf{u})$ at $x_{j+1/2}$. An example is $A_{j+1/2} = (\partial f/\partial u)|_{x=(u_j+u_{j+1})/2}$ [11, page 43]. Let $\lambda^{(p)}_{j+1/2}$ be the p^{th} eigenvalue of $A_{j+1/2}$. Along with the eigenvectors of $A_{j+1/2}$, the eigenvectors are used in the ENO computations on the fine grid points. These eigenvalues tell which way information is traveling in each characteristic field.

Therefore redefine (6.4) as

$$I_b(j+k) = \max_{k\in\mathcal{K}_j}(I_a(j+k), I_a(j)) \tag{6.5a}$$

$$\mathcal{K}_j = \bigcup_p \mathcal{K}_j^p \tag{6.5b}$$

$$\mathcal{K}_j^p = \begin{cases} \{0\} & \text{if } x_j \text{ is not a fine grid point} \\ \{0\dots\eta(b)\} & \text{if } x_j \text{ is a fine grid point \& } \lambda^{(p)}_{j+1/2} \le 0 \\ \{-\eta(b)\dots 0\} & \text{if } x_j \text{ is a fine grid point \& } \lambda^{(p)}_{j-1/2} \ge 0 \end{cases} \tag{6.5c}$$

6.3 Predictive Add Indicator

Again recall the simple equation

$$u_t(x,t) = a(x,t)\, u_x(x,t) \tag{6.6}$$

Freeze $a(x,t)$ at (x_0, t_0) and consider the characteristic approximation

$$u(x_0 - a(x_0,t_0)\,\tau, t_0+\tau) = u(x_0,t_0). \tag{6.7}$$

Revert to the non-conservative form of the equation

$$u_t(x,t) = f'(u(x,t))\, u_x(x,t). \tag{6.8}$$

Freeze this at x_j,t_0. Now it is expected that at the point $\tilde{x}_j = x_j - \tau f'(u(x_j,t_0))$, the value of $u(\tilde{x}_j, t_0+\tau)$ is $u(x_j,t_0)$. This is only a rough estimate, but it provides a predictive guess of the function at $t = t_0 + \tau$.

The previous Add Indicator used the values of $u(x)$ at $x_{j-2}, x_{j-1}, x_j, x_{j+1}$ and x_{j+2}. Now use predicted values by defining

$$y_k = x_{j+k} - \tau f'(u(x_{j+k},t_0)) \quad \text{for } k = -2,-1,0,1,2 \tag{6.9}$$

and

$$u(y_k, t_0+\tau) \approx u(x_{j+k},t_0) \quad \text{for } k = -2,-1,0,1,2 \tag{6.10}$$

and then redefine $i_a(j)$ using not the points x_k, but the points y_k

$$i_a(j) = |\mathcal{U}_4(x_j; y_{-2}, y_{-1}, y_0, y_1, y_2)| \tag{6.11a}$$

$$\tau = \delta_t \cdot \eta(\tau) \tag{6.11b}$$

for some $\eta(\tau)$[approx. 8] and δ_t is the step size. This will predict the regularity of $u(x,t)$, $\eta(\tau)$time steps in the future.

The benefit of using this Predictive Add Indicator is increased computational speed because the add and remove algorithms can be applied less often. This indicator also has a drawback. Application of this to a system of equations will be difficult. (6.9) requires the Jacobian which is not known except at the ENO points. Instead of computing the Jacobian, simple approximations can be used to predict the regularity of $u(x,t)$ using values of $u(x_j,t)$ at two different time steps.

7 Parameters

The underlying **mesh** for the Hybrid Adaptive ENO computations has 320 points, with $N = 20$ and uses 4 levels of refinement and a third order TVD Runge-Kutta scheme used for the time evolution.

The time step used was chosen based on the smallest distance between grid points. Therefore once the grid was refined to the finest level possible, the time step was chosen to be the same as a regular uniform 320 point ENO computation. This ensured stability for the finest grid, but was overly small for the course grid. Berger [2] suggests a method for improving over this cautious time stepping. She suggest that intermediate time steps be taken on fine grid points. This will in effect cause every data point to use it's maximum time step.

The values of $\eta(a)$, $\eta(r)$, and $\eta(b)$ are straight forward. Their assigned values are consistent with how they were defined. The values of $\eta(\tau)$ and the intervals between application of the add and remove algorithms is less straight forward. These values were determined through trial and error, but seem to work for a number of different problems including non-convex problems.

Two different sets of values for $\eta(\tau)$ and algorithm intervals were chosen. The first versions sets $\eta(\tau) = 0$ and does not use the Predictive Add Indicator. The second sets $\eta(\tau) = 16$ and uses the Predictive Add Indicator to reduce how often the add algorithm is applied.

The final two parameters are the tolerances which dictate whether to remove or add points.

$$\begin{array}{ll} \text{if } I_b > TOL_a & \text{then add the children of } x_j \\ \text{if } I_r < TOL_r & \text{then remove the children of } x_j \end{array}$$

							Intervals		Tolerances	
Version	order	CFL	$\eta(a)$	$\eta(r)$	$\eta(b)$	$\eta(\tau)$	Add	Remove	TOL_a	TOL_r
1	3	.25	5	5	7	0	4	10	.00001	.001
2	3	.25	5	5	7	16	8	15	.00001	.001

Table 6: User Parameters

8 Numerical Results

I considered the solution to the inviscid Burger's equation

$$\frac{\partial}{\partial t}u(x,t) = \frac{1}{2}\frac{\partial}{\partial x}(u^2(x,t)) \tag{8.1}$$

with two different initial conditions,

$$u(x,0) = \frac{1}{2}\sin(x) + \frac{1}{2} \tag{8.2}$$

$$u(x,0) = \frac{1}{2}\cos(x) + \sin(2x + .2) + \frac{3}{2}. \tag{8.3}$$

The first leads to a single shock in the data, while the second leads to two shocks traveling at different speeds which join together into one shock.

I compared my results to standard ENO routines using 128 points, 256 points and 320 points. These were chosen because 128 points takes approximately the same time to compute, and the second two give approximately the same accuracy as the Hybrid Adaptive ENO near the shock.

8.1 Single Shock

This example is used to demonstrate accuracy in smooth areas and shock speed. Table 7 shows the results of this computation. The second and third columns detail the average number of points which used an ENO approximation and the average number of **grid** points used overall. The errors measured are the pointwise error of points which are sufficiently removed from the discontinuity. The shock width measures the smearing of the shock by the numerical method. Shock width is defined as the distance from the shock to where the computed error drops below .001. For the Hybrid Adaptive ENO scheme, the solution error is less than .001 for all points which are more than .05655 away from the shock. The shock width is larger for both 128 point and 256 point computations. The Hybrid Adaptive ENO shock width is the same as the 320 point computation.

	Avg. # of Points		Error			Shock	CPU
Scheme	ENO	Total	L_1	L_2	L_∞	Width	Time
Hybrid ENO	16.0	64.9	.00004160	.00004789	.0001928	.05655	144s
128 Points	128	128	.00001200	.00007930	.0007891	.1252	154s
256 Points	256	256	.000004076	.00002975	.0002824	.07196	694s
320 Points	320	320	.000002696	.00002504	.0003529	.05655	1081s

Table 7: Single Shock – Errors and Computational Times

Figures 4 and 5 show the solution and error of the single shock problem at $t = 2$, while figures 6 and 7 show the solution and error of the single shock problem at $t = 8$. Noticeable in figures 5 and 7 is that the Hybrid Adaptive ENO scheme is less accurate than the standard ENO schemes away from the shock, but more accurate close to the shock. This is not surprising since the algorithm uses more points close to the shock and fewer points away from the shock. Remember, the goal is not to have the highest accuracy everywhere, but to have approximately the same accuracy everywhere. The errors at $t = 8$ show that Hybrid Adaptive ENO provides a uniform accuracy.

Figure 8 plots the difference between the Hybrid Adaptive ENO solution and a standard ENO scheme using 320 points. The figure shows that away from the shock, the Hybrid Adaptive ENO scheme resolves the solution within .0001 of the 320 points solution. This is to be expected since points are removed when $I_r < .001$. Therefore the Hybrid Adaptive ENO scheme approximates the 320 points ENO scheme within .0001 away from the shock using only 17% the computational time.

Figure 9 shows which **grid** points are used. Figure 10 displays the points where an ENO approximation was used instead of finite differences. Notice that they are surrounding the shock location, with additional ENO points ahead of of the shock.

8.2 Two Shocks

For this problem I compared two different sets of user parameters. Table 8 shows the results from using the different parameters and standard ENO schemes. The error results are similar to the single shock example. Version 1 uses the same parameters as the single shock problem. Notice version 1 takes slightly longer to compute than standard ENO using 128 points and substantially quicker than using 256 points. Version 2 takes 15% less time than version 1, and is quicker than the ENO computation with 128 points. Both versions have small errors and narrow shock widths. The difference between versions is that version 2 uses the Predictive Add Indicator to change the **grid**.

	Avg. # of Points		Error			Shock	CPU
Scheme	ENO	Total	L_1	L_2	L_∞	Width	Time
Hybrid ENO (1)	42.2	83.8	.00007057	.00009807	.0003543	.07853	424s
Hybrid ENO (2)	42.2	84.5	.00004936	.00007454	.0003258	.07853	358s
128 Points	128	128	.00001444	.00008630	.0006660	.1767	378s
256 Points	256	256	.000005265	.00004380	.0004645	.1030	1205s
320 Points	320	320	.000006023	.00006008	.0009528	.07853	2393s

Table 8: Two Shocks – Errors and Computational Times

Figures 11, 12, 13, and 14 compare the solutions using both versions of the Hybrid Adaptive ENO scheme along with two standard ENO computations. Version 1 and version 2 are Hybrid Adaptive ENO computations that are virtually indistinguishable from each other. As before the Hybrid Adaptive ENO schemes are less accurate away from the shock, but more accurate near the shock.

Figures 15 and 16 again show which data points are used and which points used an ENO approximation to the derivative rather than a finite difference approximation.

9 Conclusion

The Hybrid Adaptive ENO scheme is an essentially non-oscillatory high-order scheme for conservative problems. The method uses an adaptive grid to maintain accuracy away from discontinuities while minimizing the smearing of shocks.

One possible change would be a combination of Harabetian and Pego's work [3] with this work. This would involve using a switching method to switch from ENO to finite difference for points on the finest grid and would slightly reduce the number of points were ENO computations are made.

The obvious question is how can this be applied to higher dimensions. The first consideration is computation of the derivatives required. Since conservative equations have no cross terms and therefore can be written as

$$u_t(x,y,t) + F_x(u(x,y,t)) + G_y(u(x,y,t)) = 0. \tag{9.1}$$

Computation of F_x and G_y use the same technique for computation of derivatives in each direction separately. In doing so, each line would be treated as a separate problem and where to use ENO and where to use finite difference would be decided by the data points.

The second consideration is the general make-up of the computational grid. The easiest way is to test along each vertical and horizontal line in the data whether more or less data points are required. Computational tests by myself and others using this simple method leads to grids similar to sparse grids. For example three neighboring parallel slices of the data can have drastically different number of points on each slice. One line might have 128 points while its neighbors might only have 16. This is not because of peculiarities in the data, it's just that the grid when refined completely defaults to a sparse grid.

In researching how to avoid these problems, I have found a solution is staggering each of the grid lines slightly. This will somewhat increase the number of data points, but will avoid sparse grids.

The final consideration is how to change the computational grid. Before this problem is totally attacked, the above problem with the higher dimension adaptive computational grid needs to be resolved.

Acknowledgments: I would like to thank Lee Jameson for our long discussions, my wife Jennifer for her through editing, and the reviewers for their constructive comments.

A Fifth Order Example

For the fifth order approximation, use stencils with the points $x_{j-2}, x_{j-1}, x_j, x_{j+1}, x_{j+2}$ and x_{j+3}. Recall that the number of changes in w_j within one stencil was restricted to one. Therefore there are only 9 possible stencils for the fifth order method. These can be found in table 9.

The fifth order approximation for $u'(x)$ is

$$G_j = \sum_{k=-2}^{3} c_k u(x_k, t) \tag{A.1}$$

where c_k are chosen to ensure fifth order accuracy and depend on the grid spacing. These can be computed for each of the possible stencils in table 10.

For this fifth order example the local truncation error is

$$e_j = \frac{1}{6!} \left| \sum_{k=-2}^{3} c_k (x_{j+k} - x_j)^6 u^{(vi)}(\xi_k) \right| \tag{A.2}$$

for some values $\xi_k \in [x_{j-2}, x_{j+3}]$.

Assume a bound on $u^{(vi)}(\xi)$ for $\xi \in [x_{j-2}, x_{j+3}]$,

$$U_j = max_{\xi \in [x_{j-2}, x_{j+3}]} \left| u^{(vi)}(\xi) \right|. \tag{A.3}$$

Thus using this bound and a knowledge of the grid spacing allows a bound placed on the local truncation error

$$e_j \le \frac{U_j \delta^6}{6!} \sum_{k=-2}^{3} |c_k| \left(\frac{x_{j+k} - x_j}{\delta} \right)^6 \tag{A.4}$$

$$\le \frac{U_j \delta^6}{6!} B(w_{-2}, w_{-1}, w_0, w_1, w_2). \tag{A.5}$$

The values for $B(w_{-2}, w_{-1}, w_0, w_1, w_2)$ are also found in table 10. The maximum value of $B()$ is 92. Therefore the bound on the local truncation error is

$$e_j \le \frac{23 \Delta_j^5}{144} U_j. \tag{A.6}$$

Stencil #	w_{-2}	w_{-1}	w_0	w_1	w_2
-4	1	1	0	0	0
-3	1	0	0	0	0
-2	0	0	0	-1	-1
-1	0	0	0	0	-1
0	0	0	0	0	0
+1	-1	0	0	0	0
+2	-1	-1	0	0	0
+3	0	0	0	0	+1
+4	0	0	0	+1	+1

Table 9: Stencils for fifth order approximations

Stencil #	c_{-2}	c_{-1}	c_0	c_1	c_2	c_3	$B(\ldots)$
-4	$1/140$	$-1/10$	$-13/12$	$8/5$	$-1/2$	$8/105$	$624/5$
-3	$1/120$	$-3/8$	$-1/2$	$9/8$	$-3/10$	$1/24$	$1143/20$
-2	$1/28$	$-2/5$	$-2/3$	2	$-128/105$	$1/4$	$242/7$
-1	$5/108$	$-10/21$	$-2/5$	$10/9$	$-5/12$	$128/945$	$450/7$
0	$1/20$	$-1/2$	$-1/3$	1	$-1/4$	$1/30$	45
+1	$64/315$	$-3/4$	$-1/6$	$9/10$	$-3/14$	$1/36$	$531/14$
+2	$1/4$	$-64/35$	$7/6$	$1/2$	$-1/10$	$1/84$	$111/7$
+3	$1/18$	$-8/15$	$-1/4$	$8/9$	$-1/6$	$1/180$	$192/5$
+4	$1/14$	$-5/8$	$-1/30$	$5/8$	$-1/24$	$1/280$	92

Table 10: Coefficients and Bounds for fifth order stencils

References

[1] Robert Bruce Bauer. An efficient adaptive grid ENO algorithm. *Houston Journal of Mathematics*, pages 329–346, 1996. Proceeding of the Third International Conference on Spectral and High Order Methods (ICOSAHOM ‘95).

[2] Marsha J. Berger. On conservation at grid interfaces. *SIAM J. Num. Anal*, 24(5):967–984, 1987.

[3] Eduard Harabetian and Robert Pego. Nonconservative hybrid shock capturing schemes. *Journal of Computational Physics*, 105:1–13, 1993.

[4] Ami Harten. Multiresolution analysis for ENO schemes. *ICASE Report*, 77(NASA CR-189043), 1991.

[5] Ami Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws, 1993. preprint.

[6] Ami Harten and Sukumar R. Chakravarthy. Uniformly high order accurate essentially non–oscillatory schemes, III. *Journal of Computational Physics*, 71(2):231–303, 1987.

[7] Ami Harten and Sukumar R. Chakravarthy. Multiresolution analysis ENO schemes. *ICASE Report*, 76(NASA CR-187637), 1991.

[8] Ami Harten, B. Engquist, Stanley Osher, and Sukumar R. Chakravarthy. Uniformly high order accurate essentially non–oscillatory schemes, II. *Journal of Computational Physics*, 83:231–303, August 1987.

[9] Ami Harten and James M. Hyman. Self adjusting grid methods for one–dimensional hyperbolic conservation laws. *Journal of Computational Physics*, 50(2):235–269, May 1983.

[10] Lee Jameson. On the wavelet-optimized finite difference method. *ICASE Report*, 9(NASA CR-191601), 1994. Submitted to Journal of Computation Physics.

[11] S. Osher and C. W. Shu. Efficient implementation of essentially non-oscillatory shock capturing schemes. *Journal of Computational Physics*, 77:439–471, 1988.

[12] S. Osher and C. W. Shu. Efficient implementation of essentially non-oscillatory shock capturing schemes II. *Journal of Computational Physics*, 83(1):32–78, 1989.

[13] Bram Van Leer. Towards the ultimate conservative difference scheme. II. monotonicity and conservagtion combined in a second-order scheme. *Journal of Computational Physics*, 14:361–370, 1974.

List of Figures

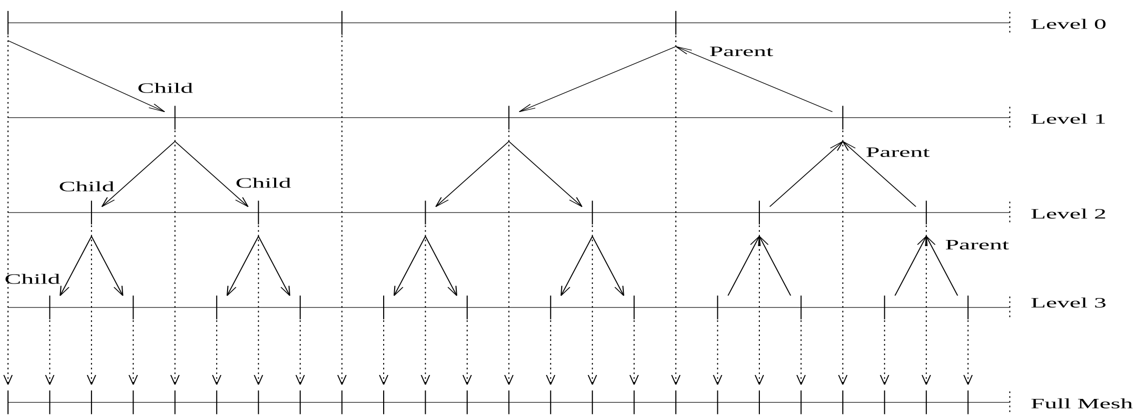
Level 0
Parent
Child
Level 1
Parent
Child
Child
Level 2
Parent
Child
Level 3
Full Mesh

Figure 1:

Sample Mesh

Total Mesh

Mesh Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Mesh Level	0	3	2	3	1	3	2	3	0	3	2	3	1	3	2	3	0	3	2	3	1	3	2	3	0	3	2	3	1	3	2	3
Parent	-	2	4	2	8	6	4	6	-	10	12	10	8	14	14	14	-	18	20	18	24	22	20	22	-	26	28	26	24	30	28	30
Children	-	-	1,3	-	2,6	-	5,7	-	4,12	-	9,11	-	10,14	-	13,15	-	-	-	17,19	-	18,22	-	21,23	-	20,28	-	25,27	-	26,30	-	29,31	-
Sibling	8	3	6	1	12	7	2	5	0	11	14	9	4	15	10	13	24	19	22	17	28	23	18	21	16	27	30	25	20	31	26	29
Next(Left)	24	31	30	1	28	3	2	5	0	7	6	9	4	11	14	13	8	15	14	17	12	19	18	21	16	23	22	25	20	27	26	29
Next(Right)	8	3	6	5	12	7	10	9	16	11	14	13	20	15	18	17	24	19	22	21	28	23	26	25	0	27	30	29	4	31	2	1

Figure 2:

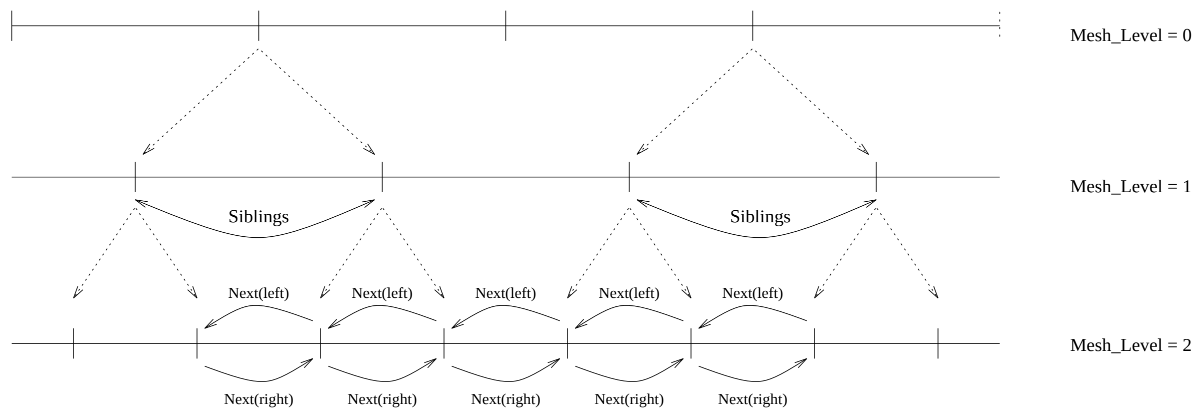

Figure 3:

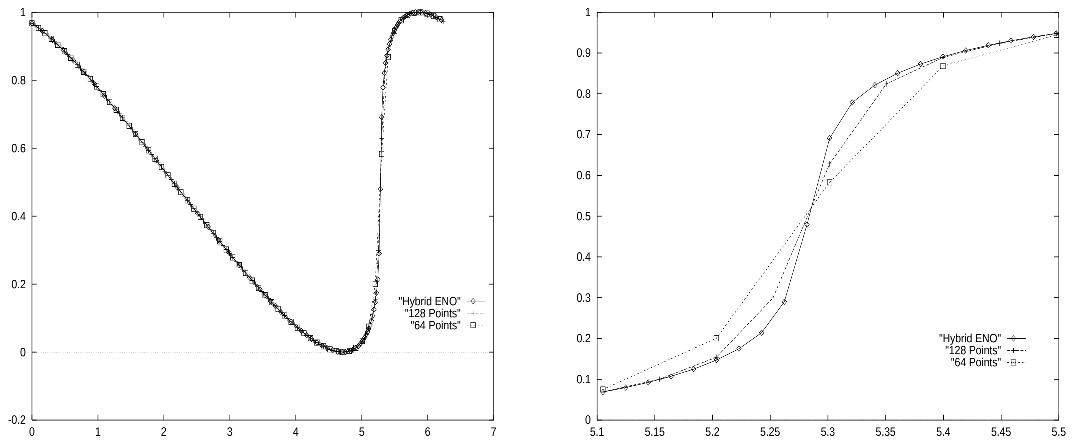

Figure 4:

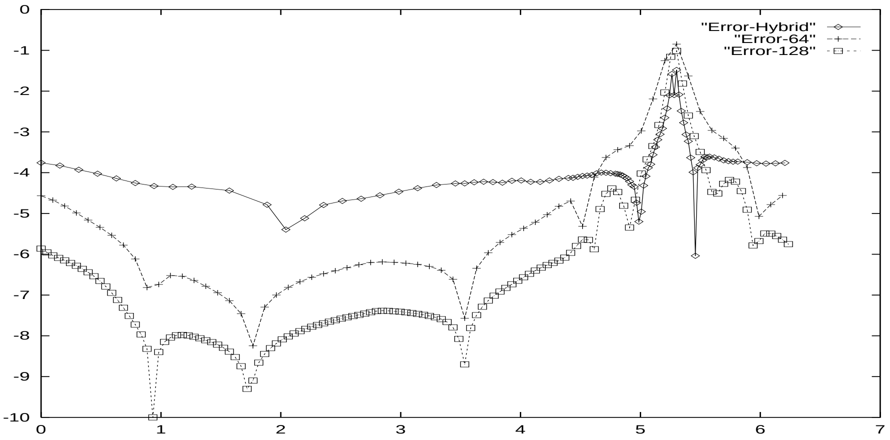

Figure 5:

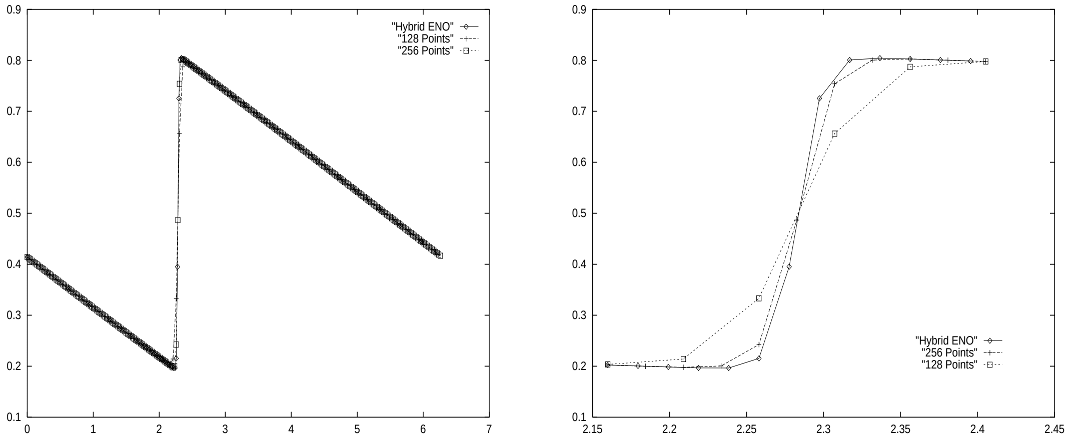

Figure 6:

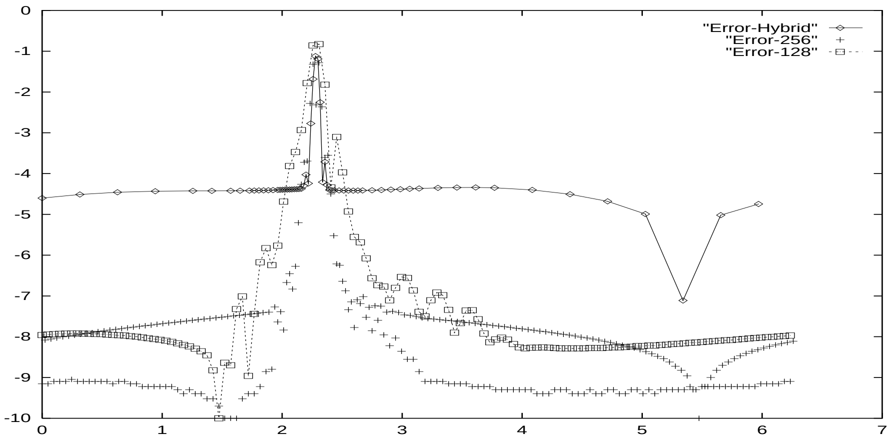

Figure 7:

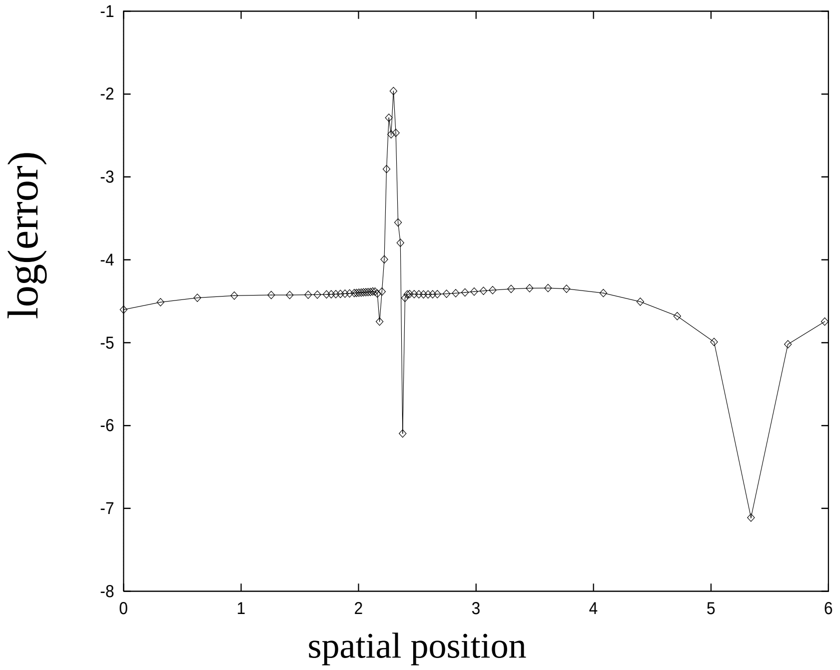

Figure 8:

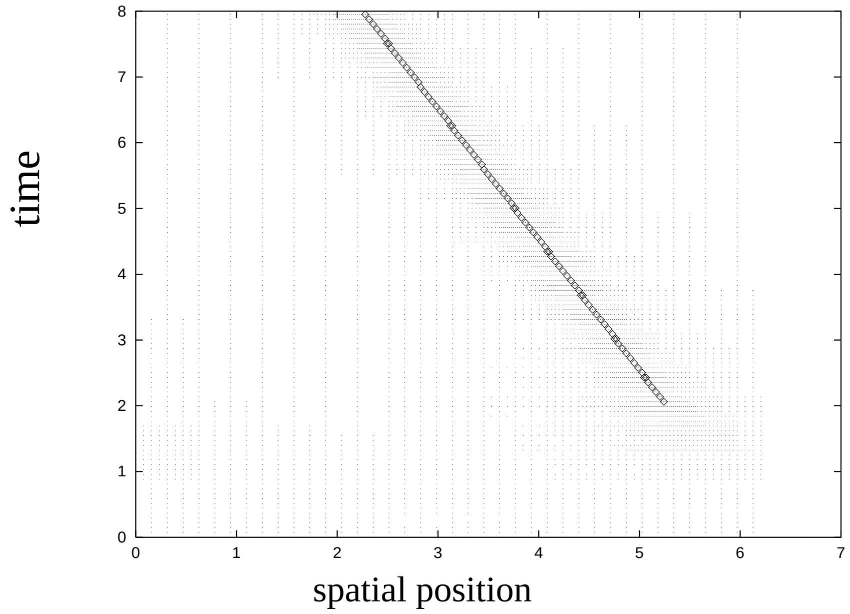

Figure 9:

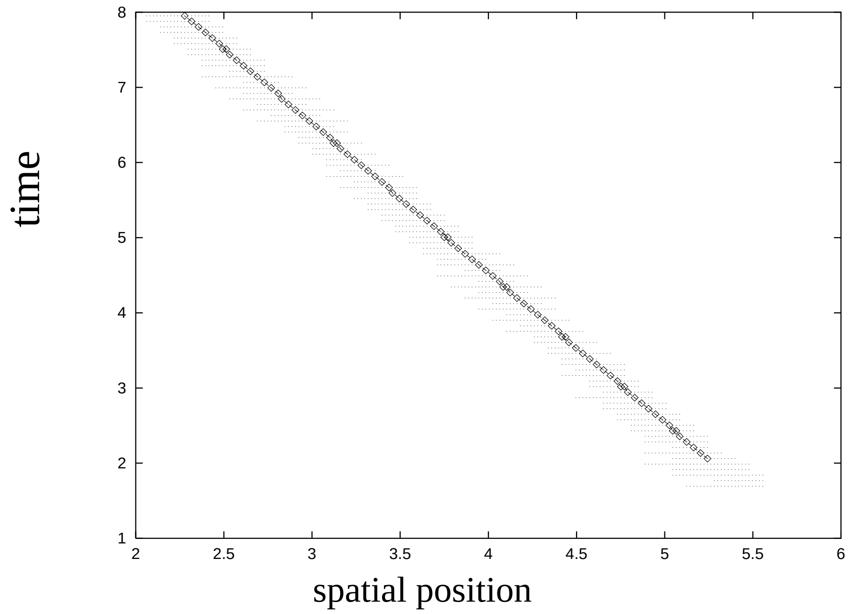

Figure 10:

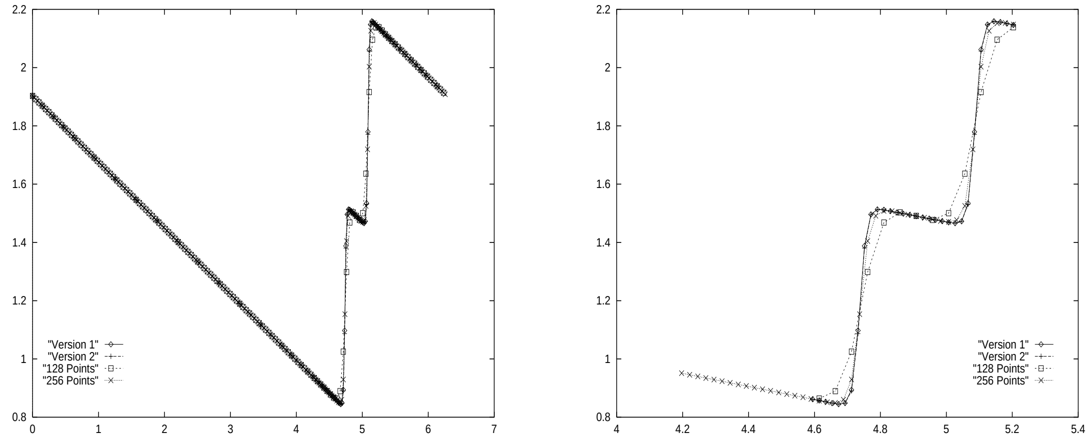

Figure 11:

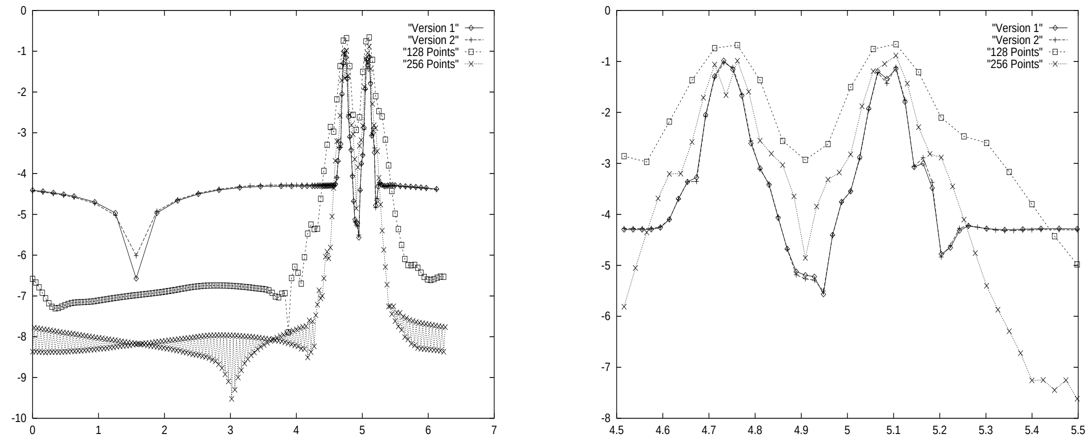

Figure 12:

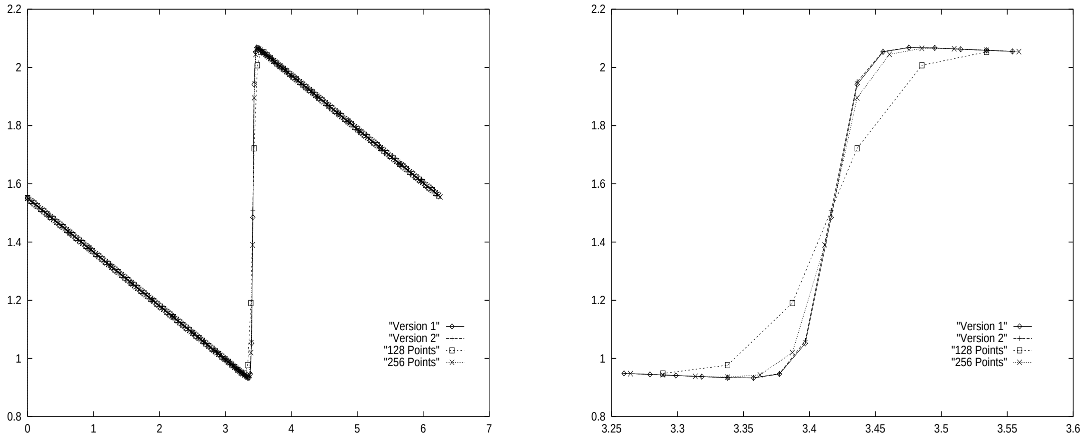

Figure 13:

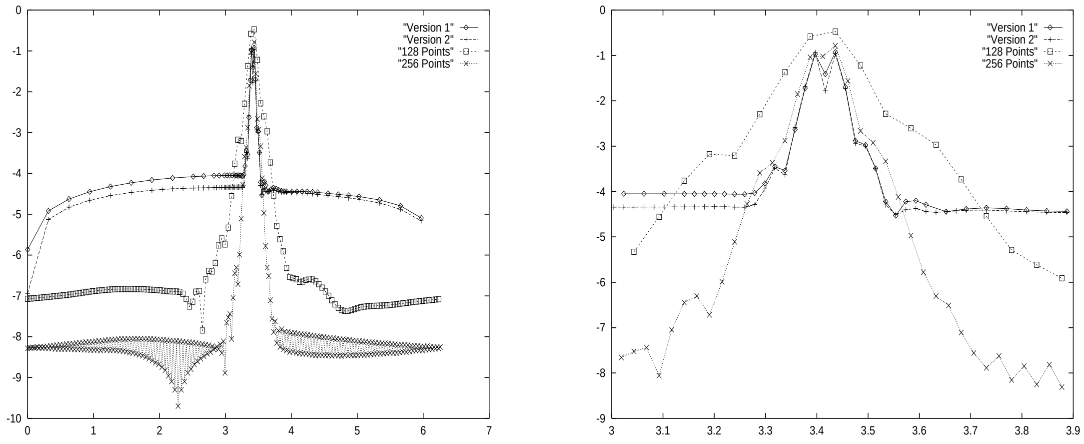

Figure 14:

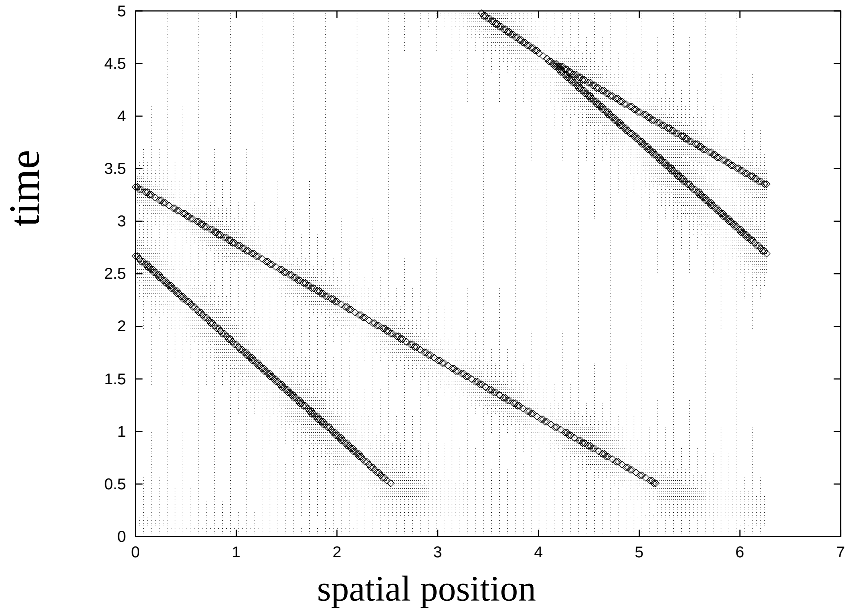

Figure 15:

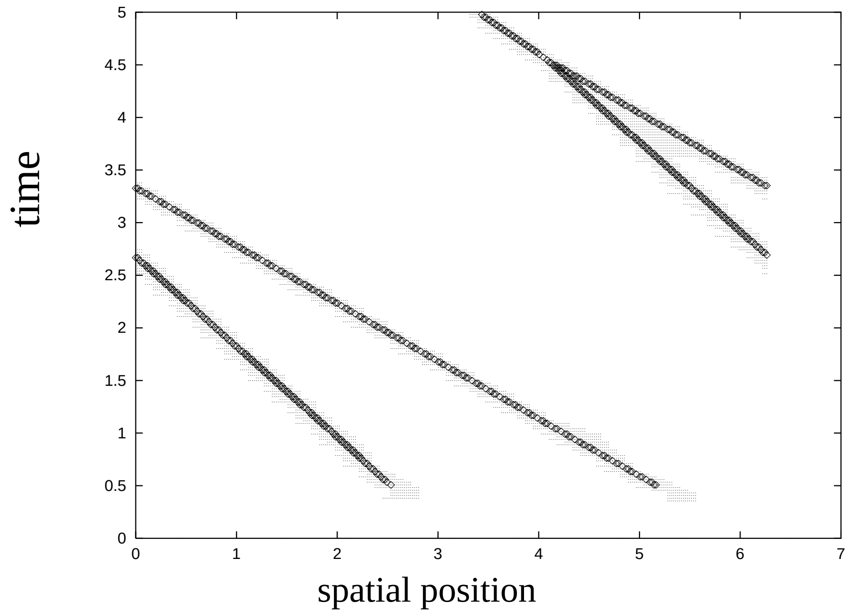

Figure 16: