A Fifth Order Flux Implicit WENO Method

Sigal Gottlieb* and Julia S. Mullen† and Steven J. Ruuth‡

April 3, 2005

Keywords: implicit, weighted essentially non-oscillatory, time-discretizations.

Abstract

The weighted essentially non-oscillatory method (WENO) is an excellent spatial discretization for hyperbolic partial differential equations with discontinuous solutions. However, the time-step restriction associated with explicit methods may pose severe limitations on their use in applications requiring large scale computations. An efficient implicit WENO method is necessary. In this paper, we propose a prototype flux-implicit WENO (iWENO) method. Numerical tests on classical scalar equations show that this is a viable and stable method, which requires appropriate time stepping methods. Future study will include the examination of such methods as well as extension of iWENO to systems and higher dimensional problems.

1 Introduction

Explicit weighted essentially non-oscillatory (WENO) methods have proved very popular in numerical simulations of discontinuous solutions of hyperbolic problems. However, the numerical CFL condition leads to time-step restrictions which may be prohibitive for large scale computations. This is a problem frequently seen in finite difference methods and is dealt with by treating such methods implicitly. However, there are significant logistical problems in constructing a stable and efficient implicit WENO method. WENO is highly nonlinear, even for a linear problem, and the computational cost required for an implicit step, both in the line searches and (if desired) in the computation of the Jacobian, is prohibitive. We alleviate these problems by considering a prototype flux-implicit WENO (iWENO) method. In this paper we describe the development of the new iWENO method, validate it and test it for accuracy and stability on classical scalar test cases, and sketch out future directions for the study of the iWENO method.

*Department of Mathematics, UMASS-Dartmouth North Dartmouth, MA and Division of Applied Mathematics, Brown University, Providence, RI (sg@cfm.brown.edu). The work of this author supported by NSF grant DMS-0106743

†Worcester Polytechnic Institute, Worcester, MA (jsm@wpi.edu)

‡Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, V5A 1S6 Canada (sruuth@sfu.ca). The work of this author was partially supported by a grant from NSERC Canada.

2 The explicit WENO method

To approximate, in a physically correct way, ([6]) the solution to a conservation law of the form

$$u_t + f(u)_x = 0,$$

we use a conservative finite difference scheme

$$u_t = -\frac{1}{\Delta x}(\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}).$$

The term $\hat{f}_{j+\frac{1}{2}} = \hat{f}(u_{j-k}, ..., u_{j+l})$ is the *numerical flux*, and the points $x_{j-k}, ..., x_{j+l}$ constitute the *stencil*. To be a reasonable approximation, the numerical flux must be (at least) Lipschitz continuous and consistent with the physical flux f, *i.e.* $\hat{f}(u, ..., u) = f(u)$. The numerical flux determines the numerical method and its properties. Any differences between conservative numerical semi-discretizations are a result of differences in the numerical flux.

ENO ([3], [4], [9]) schemes search for the locally smoothest stencil and use that stencil to calculate the numerical fluxes, in order to prevent the shock from being crossed. On the left of the shock and the right of the shock we have smooth regions, and in those regions linear stability is enough to ensure nonlinear stability. Oscillations arise when we take points on opposite sides of the shock to evaluate the derivative at a given point. These oscillations at the shock location propagate to the smooth regions, destroying the stability of the solution. The idea behind ENO schemes is stencil switching in order to eliminate oscillations. The ENO scheme evaluates the smoothness of several stencils near the point of interest, and picks the smoothest stencil for evaluating the derivative at that point. This means that the method should avoid picking stencils that cross the shock, and so avoid stability problems. Liu, Osher and Chan [7] improved upon the ENO method and developed the WENO method. Each stencil is assigned a weight, which depends on its smoothness. Then all approximations from all the candidate stencils are summed, each with the weight assigned to it. The weights are chosen so that in smooth regions we obtain higher order accuracy whereas near discontinuities the ENO scheme is imitated by assigning near-zero weights to the stencils that contain discontinuities. To get an rth order ENO scheme, a total of $2r-1$ points are examined for each flux. Since the WENO scheme uses all the candidate stencils, a clever choice of weights [5] results in a WENO scheme which is of order $2r-1$ in smooth regions [8].

To solve

$$u_t + f(u)_x = 0$$

we approximate the spatial derivative using WENO $L(u) = -f(u)_x$, and then use a time-stepping method to solve the resulting system of ODEs. We split the flux into the positive and negative parts

$$f(u) = f^+(u) + f^-(u).$$

This can be accomplished in a variety of ways. In this paper we consider the Lax-Friedrichs flux splitting

$$f^+(u) = \frac{1}{2}\left(f(u) + mu\right) \qquad f^-(u) = \frac{1}{2}\left(f(u) - mu\right),$$

where $m = \max|f'(u)|$. In this way we ensure that $\frac{df^+}{du} \geq 0$ and $\frac{df^-}{du} \leq 0$.

To calculate the numerical fluxes $\hat{f}^+_{j+\frac{1}{2}}$ and $\hat{f}^-_{j+\frac{1}{2}}$, we begin by calculating the smoothness measurements to determine if a shock lies within the stencil. For our fifth order scheme, these are:

$$\begin{aligned} IS_0^+ &= \frac{13}{12}\left(f^+_{j-2} - 2f^+_{j-1} + f^+_j\right)^2 + \frac{1}{4}\left(f^+_{j-2} - 4f^+_{j-1} + 3f^+_j\right)^2 \\ IS_1^+ &= \frac{13}{12}\left(f^+_{j-1} - 2f^+_j + f^+_{j+1}\right)^2 + \frac{1}{4}\left(f^+_{j-1} - f^+_{j+1}\right)^2 \\ IS_2^+ &= \frac{13}{12}\left(f^+_j - 2f^+_{j+1} + f^+_{j+2}\right)^2 + \frac{1}{4}\left(3f^+_j - 4f^+_{j+1} + f^+_{j+2}\right)^2 \end{aligned}$$

and

$$\begin{aligned} IS_0^- &= \frac{13}{12}\left(f^-_{j+1} - 2f^-_{j+2} + f^-_{j+3}\right)^2 + \frac{1}{4}\left(3f^-_{j+1} - 4f^-_{j+2} + f^-_{j+3}\right)^2 \\ IS_1^- &= \frac{13}{12}\left(f^-_j - 2f^-_{j+1} + f^-_{j+2}\right)^2 + \frac{1}{4}\left(f^-_j - f^-_{j+2}\right)^2 \\ IS_2^- &= \frac{13}{12}\left(f^-_{j-1} - 2f^-_j + f^-_{j+1}\right)^2 + \frac{1}{4}\left(f^-_{j-1} - 4f^-_j + 3f^-_{j+1}\right)^2 . \end{aligned}$$

Next, we use the smoothness measurements to calculate the stencil weights

$$\alpha_0^\pm = \frac{1}{10}\left(\frac{1}{\epsilon + IS_0^\pm}\right)^2 \qquad \alpha_1^\pm = \frac{6}{10}\left(\frac{1}{\epsilon + IS_1^\pm}\right)^2 \qquad \alpha_2^\pm = \frac{3}{10}\left(\frac{1}{\epsilon + IS_2^\pm}\right)^2$$

and

$$\omega_0^\pm = \frac{\alpha_0^\pm}{\alpha_0^\pm + \alpha_1^\pm + \alpha_2^\pm} \qquad \omega_1^\pm = \frac{\alpha_1^\pm}{\alpha_0^\pm + \alpha_1^\pm + \alpha_2^\pm} \qquad \omega_2^\pm = \frac{\alpha_2^\pm}{\alpha_0^\pm + \alpha_1^\pm + \alpha_2^\pm}.$$

Finally, the numerical fluxes are

$$\begin{aligned} \hat{f}^+_{j+\frac{1}{2}} &= \omega_0^+\left(\frac{2}{6}f^+_{j-2} - \frac{7}{6}f^+_{j-1} + \frac{11}{6}f^+_j\right) + \omega_1^+\left(-\frac{1}{6}f^+_{j-1} + \frac{5}{6}f^+_j + \frac{2}{6}f^+_{j+1}\right) \\ &+ \omega_2^+\left(\frac{2}{6}f^+_j + \frac{5}{6}f^+_{j+1} - \frac{1}{6}f^+_{j+2}\right) \end{aligned}$$

and

$$\begin{aligned} \hat{f}^-_{j+\frac{1}{2}} &= \omega_2^-\left(-\frac{1}{6}f^-_{j-1} + \frac{5}{6}f^-_j + \frac{2}{6}f^-_{j+1}\right) + \omega_1^-\left(\frac{2}{6}f^-_j + \frac{5}{6}f^-_{j+1} - \frac{1}{6}f^-_{j+2}\right) \\ &+ \omega_0^-\left(\frac{11}{6}f^-_{j+1} - \frac{7}{6}f^-_{j+2} + \frac{2}{6}f^-_{j+3}\right). \end{aligned}$$

Thus, each one of these can be written as:

$$\hat{f}^+_{j+\frac{1}{2}} = \sum_{i=0}^{4} q_i^+(j) f^+_{j-2+i} \text{ and } \hat{f}^-_{j+\frac{1}{2}} = \sum_{i=0}^{4} q_i^-(j) f^+_{j-1+i}.$$

Since each of these expressions is simply a weighted sum of the numerical fluxes, we may rewrite

$$f^+(u)_x \approx \frac{1}{\Delta x}\left(\hat{f}^+_{j+\frac{1}{2}} - \hat{f}^+_{j-\frac{1}{2}}\right) = -W^+(u)f^+(u)$$

and

$$f^-(u)_x \approx \frac{1}{\Delta x}\left(\hat{f}^-_{j+\frac{1}{2}} - \hat{f}^-_{j-\frac{1}{2}}\right) = -W^-(u)f^-(u),$$

for matrices of weights $W^+(u)$ and $W^-(u)$, defined by a trivial rearrangement and differencing of the q_is. Thus, we can write the WENO approximation

$$L(u) \approx -f(u)_x$$

where $L(u) = W^+(u)f^+(u) + W^-(u)f^-(u)$.

3 Complexities of implicit WENO methods

Often, a finite difference method can be made implicit in a very simple and straightforward way. While the WENO method may be viewed, in some sense, as a finite difference method, the nonlinear weights imply that each point in space and time will use a different finite difference discretization. In other words, in standard finite difference methods the differentiation matrices are constant, while in WENO, they depend on u^n. The primary hurdle in developing an efficient implicit WENO method is the high computational cost associated with computing the highly nonlinear weights. Approximating a derivative using explicit WENO requires costly computations of the weights, and a fully implicit WENO requires significantly more of these computations within the nonlinear solver. This nonlinear solver is typically Jacobian based, and the expensive step is usually the inversion of the Jacobian. However, in the case of implicit WENO, the mere computation of the Jacobian matrix is very costly. Repeated computations of the Jacobian are necessary for each line search, and this cost can become prohibitive. This problem could be alleviated by choosing a Jacobian-free nonlinear solver; however, this approach could lead to problems if the Jacobian is approximated using differencing which crosses the shock. The WENO method is based on the premise that to preserve stability, we used stencils which are weighted heavily in a nonlinear manner on one side of the shock or the other. Therefore, the Jacobian-free method could lead to instability by violating this rule.

These problems were encountered in [1], where an implicit WENO was developed for direct convergence to steady state. In that case, there was no time-stepping, only an implicit solution of the steady state equations. This class of methods solved the steady-state problem

$$f(u)_x = 0 \ ,$$

by using the WENO approximation to obtain

$$L(u) = 0 \ .$$

This system of equations was solved iteratively, using a damped Newton method. The experience gained from that approach indicated that in the nonlinear solver, the computation of the WENO weights and the analytic computation of the Jacobian was extremely slow, due to the nonlinearity of WENO.

The idea behind the flux-implicit iWENO method is to treat the differentiation matrices explicitly and the flux implicitly. In this way, each implicit-solver iWENO step has a computational cost equivalent to that in a standard finite difference scheme. We anticipate that

the combination of the flux-implicit approach and the relaxed time step restriction of the implicit solver will make the implicit method cost-effective relative to the explicit method.

4 The flux-implicit approach

In the following, we treat the scalar conservation law

$$u_t + f(u)_x = 0$$

as described above, where the spatial differencing is accomplished via the WENO approximation

$$L(u) = W^+(u)f^+(u) + W^-(u)f^-(u).$$

We suggest at this point thinking of the iWENO operator as a product of two operators, $W(u)F(u)$. Clearly, that is not exactly correct in this context. However, for simplicity, we will use this formulation at present, and discuss the generalization later. We consider $W(u)$ to be the differentiation matrix containing the WENO weights and $F(u)$ to be some combination of the numerical fluxes.

A simple first order prototype of the flux implicit iWENO method can be viewed as a predictor-corrector scheme:

$$\begin{aligned} \tilde{u} &= u^n + \Delta t W(u^n)F(u^n) \\ u^{n+1} &= u^n + \Delta t W(\tilde{u})F(u^{n+1}) \quad . \end{aligned}$$

The quantity $\tilde{u}$ is computed in order to predict the weights in W. The value of $\tilde{u}$ is obtained by an explicit forward-Euler time step. The corrector step is then an implicit Euler method, which is implicit only in F, but not in the highly nonlinear W. Both the predictor and the corrector steps are of the same order; however, the use of the implicit corrector step may allow a larger time step for stability than a purely explicit method would allow. Now the implicit step in iWENO is of the same cost as the implicit step of any usual finite difference scheme. However, the matrix W may not have the nice features, such as symmetry, associated with standard finite difference methods.

In general, this predictor-corrector approach involves using a known explicit method (either Runge-Kutta or multi step) of the desired order to evaluate $\tilde{u}$, and then calculate the matrix of weights $W(\tilde{u})$. To step forward in time, we use any implicit method of the desired order, but we now treat $W(\tilde{u})$ as a constant matrix.

A flux-implicit approach which does not involve a separate predictor is also possible. For example, the simplest first order prototype would be

$$u^{n+1} = u^n + \Delta t W(u^n)F(u^{n+1}) \quad .$$

In this case, the weights are not predicted at time level t^{n+1}, but rather the weights at time t^n are used. There is the danger, in this type of method, that the stencil determined by the weights at time t^n will cross the shock at time t^{n+1} and lead to instability.

To extend these ideas, one may consider any M-stage Runge-Kutta method of the form

$$\begin{aligned} u^{(0)} &= u^n \\ u^{(l)} &= \sum_{j=0}^{l} \alpha_{lj} u^{(j)} + \Delta t \sum_{j=0}^{l} \beta_{lj} W(u^{(i_j)}) F(u^{(j)}), \quad \text{where each } i_j < l \text{ and } l = 1, .., M \\ u^{n+1} &= u^{(M)} \ . \end{aligned}$$

As long as $W(\cdot)$ is based on an already-computed quantity, this method has the advantages of a flux-implicit iWENO. The constants α_{mj} and β_{mj} need to be chosen carefully to ensure the method is of the correct order.

The formulations of the methods in this paper are such that the general case

$$L(u) = W^+(u)F^+(u) + W^-(u)F^-(u)$$

can be implemented as easily as the simplified case $L(u) = W(u)F(u)$, by simply adding in the corresponding terms. More complicated formulations are possible, however, one must confirm that the general case can be implemented.

In the numerical section, we solve linear and nonlinear scalar equations with several types of predictor-corrector iWENO methods. The aim of these implicit methods is to raise the value of the time step which is allowed for stability, while reducing the cost of the implicit WENO step to that of a standard implicit finite difference scheme.

5 Numerical Results

For the numerical experiments, we considered two classical scalar test cases, and time-stepping methods of order up to three.

Example 1: The first set of examples is a linear wave equation

$$u_t + u_x = 0 \quad 0 \le x \le 2$$

with periodic boundary conditions and two sets of initial conditions. We use the smooth initial condition $u(x, 0) = \sin(\pi x)$ to verify the spatial and temporal order of the schemes, and the step-function initial condition

$$u(x, 0) = \begin{cases} 1 & \text{if } 0 \le x < 1 \\ 0 & \text{otherwise} \end{cases}$$

to observe the effect of iWENO on a discontinuous problem.

1. We begin with two first order methods:
   1a) A first order flux-implicit Euler method where the weights are computed explicitly:

$$u^{n+1} = u^n + \Delta t W(u^n) F(u^{n+1})$$

and
1b) A method consisting of an explicit first order forward Euler predictor and first order implicit Euler corrector:

$$\begin{aligned} \tilde{u} &= u^n + \Delta t W(u^n) F(u^n) \\ u^{n+1} &= u^n + \Delta t W(\tilde{u}) F(u^{n+1}) \end{aligned}$$

Both methods show first order accuracy in time using the smooth initial condition. We tested the stability of the method by solving the linear equation with step function initial conditions up to final time $T_f = 5.0$ and for number of points N up to 200. Both methods were stable for all values of $\Delta t = c\frac{\Delta x}{m}$, for $c \leq 10$ and $m = \max |f'(u)|$. It became apparent that the methods would continue to be stable as we raise the step size, since both methods are very diffusive and the step function became increasingly smeared as Δt became larger.

The first method did not use a predictor to approximate the weights for the implicit method. We feared this may lead to instability because of the possibility that the stencil defined at time t^n will cross the shock at time t^{n+1}. However, in our computations we observe that both methods behave very much the same. We attribute this to the strong diffusivity of the methods which, in convecting the discontinuous initial condition, smeared it to the extent that in only a few time steps the shock was completely smeared so that the solution profile was essentially smooth.

2. We proceed with the second order methods:
2a) A first order explicit Euler predictor and a second order implicit Crank-Nicolson corrector ([2]):

$$\begin{aligned} \tilde{u} &= u^n + \Delta t W(u^n) F(u^n) \\ u^{n+1} &= u^n + \frac{1}{2}\Delta t W(u^n) F(u^n) + \frac{1}{2}\Delta t W(\tilde{u}) F(u^{n+1}) \end{aligned}$$

and
2b) A second order explicit Runge-Kutta predictor [9] and second order implicit Crank-Nicolson corrector [2]:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t W(u^n) F(u^n) \\ \tilde{u} &= \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \frac{1}{2}\Delta t W(u^{(1)}) F(u^{(1)}) \\ u^{n+1} &= u^n + \frac{1}{2}\Delta t W(u^n) F(u^n) + \frac{1}{2}\Delta t W(\tilde{u}) F(u^{n+1}) \ . \end{aligned}$$

and
2c) A first order flux-implicit Euler predictor and a second order implicit Crank-Nicolson corrector [2]:

$$\begin{aligned} \tilde{u} &= u^n + \Delta t W(u^n) F(\tilde{u}) \\ u^{n+1} &= u^n + \frac{1}{2}\Delta t W(u^n) F(u^n) + \frac{1}{2}\Delta t W(\tilde{u}) F(u^{n+1}) \end{aligned}$$

Although the first and third method use only a first order predictor, the predicted terms enter into the final formula multiplied by Δt, and thus we can afford to use a predictor of one order less than the corrector. Accuracy tests on the linear problem with smooth initial conditions demonstrate that the iWENO method is indeed fifth order in space (tested with method(2c), see Table 1) and that methods (2a), (2b) and (2c) give very similar results and that these implicit methods are second order in time (Table 2).

In terms of stability considerations, the two methods (2a) and (2b) behave similarly. Both methods show non-oscillatory results for CFL values $c < \frac{1}{2}$. Interestingly, the methods develop an oscillation on either side of the shock for CFL values $\frac{1}{2} \leq c \leq 1$. However, in tests up to final time $T_f = 4.0$ and number of points $N = 200$, this oscillation remains the same or decays and does not develop into instability (Figure 1). As the figure shows, the oscillation in method (2b) is smaller than that of method (2a) and decays over time. The third method (2c), which features a flux-implicit predictor, is more stable – no oscillation appears for values $c < 1$. Instability is observed for all three methods at CFL values $c > 1$.

A comparison with the explicit second order Runge-Kutta method SSP(2,2) [9]:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t W(u^n) F(u^n) \\ u^{n+1} &= \frac{1}{2}u^n + \frac{1}{2}u^{(1)} + \frac{1}{2}\Delta t W(u^{(1)}) F(u^{(1)}) \end{aligned}$$

shows that this method is stable for values of $c < \frac{1}{2}$, and becomes gradually unstable afterwards. For this example, using the iWENO method with these implicit second order methods does not confer a significant advantage in terms of allowed Δt for stability. Figure 2 shows a comparison of all four methods for $c = \frac{1}{2}$, $c = \frac{3}{4}$, $c = 1$, and $c = \frac{5}{4}$, with $N = 40$ points, with $\Delta t = c\Delta x$ and convected over 100 time steps.

3. Finally, we consider third order methods:
3a) A second order explicit Adams-Bashforth predictor and a third order implicit Adams-Moulton corrector [2]:

$$\begin{aligned} \tilde{u} &= u^n + \frac{3}{2}\Delta t W(u^n) F(u^n) - \frac{1}{2}\Delta t W(u^{n-1}) F(u^{n-1}) \\ u^{n+1} &= u^n + \Delta t \left(\frac{5}{12} W(\tilde{u}) F(u^{n+1}) + \frac{8}{12} W(u^n) F(u^n) - \frac{1}{12} W(u^{n-1}) F(u^{n-1}) \right) . \end{aligned}$$

and
3b) A second order extrapolated BDF predictor and a third order BDF corrector [2]:

$$\begin{aligned} \tilde{u} &= \frac{4}{3}u^n - \frac{1}{3}u^{n-1} + \Delta t \left(\frac{4}{3} W(u^n) F(u^n) - \frac{2}{3}\Delta t W(u^{n-1}) F(u^{n-1}) \right) \\ u^{n+1} &= \frac{18}{11}u^n - \frac{9}{11}u^{n-1} + \frac{2}{11}u^{n-2} + \frac{6}{11}\Delta t W(\tilde{u}) F(u^{n+1}) . \end{aligned}$$

Accuracy tests show, as expected, third order accuracy (Table 3). The results from these methods were very interesting (Figure 3). Method 3b was stable for CFL values

$c \leq 0.5$, but showed oscillations or signs of instability for larger values. Method 3a was stable for $c \leq 1$.

In comparison,
3c) the explicit third order SSP Runge Kutta [9]:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t W(u^n) F(u^n) \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t W(u^{(1)}) F(u^{(1)}) \\ u^{n+1} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t W(u^{(2)}) F(u^{(2)}) \end{aligned}$$

was also stable for $c \leq 1$. A comparison of the three methods (3a), (3b) and (3c) is presented in Figure 3.

3d) the explicit third order Runge-Kutta predictor and implicit third order Adams-Moulton corrector:

$$\begin{aligned} u^{(1)} &= u^n + \Delta t W(u^n) F(u^n) \\ u^{(2)} &= \frac{3}{4}u^n + \frac{1}{4}u^{(1)} + \frac{1}{4}\Delta t W(u^{(1)}) F(u^{(1)}) \\ \tilde{u} &= \frac{1}{3}u^n + \frac{2}{3}u^{(2)} + \frac{2}{3}\Delta t W(u^{(2)}) F(u^{(2)}) \\ u^{n+1} &= u^n + \Delta t \left(\frac{5}{12} W(\tilde{u}) F(u^{n+1}) + \frac{8}{12} W(u^n) F(u^n) - \frac{1}{12} W(u^{n-1}) F(u^{n-1}) \right) . \end{aligned}$$

gave significantly more stable results. Figure 4 shows a comparison of methods (3a), (3c) and (3d). It is evident that in terms of computational cost the explicit SSP Runge-Kutta still outperforms all three of the implicit methods.

The linear test cases demonstrated the spatial order of accuracy of the iWENO and the time-accuracy of the time-stepping methods. Although the linear test cases showed that iWENO is reliable, the stability results were heavily dependent on the time stepping method, and were discouraging. We did not observe any significant advantages of the implicit methods in terms of step size for order two or higher. The nonlinear example provides a clearer picture of the need for appropriate time stepping:

Example 2: The second example is the nonlinear Burgers' equation

$$u_t + \left(\frac{u^2}{2} \right)_x = 0$$

with periodic boundary conditions, and initial condition $u(x, 0) = \frac{1}{2} + \frac{1}{4}\sin(\pi x)$.

This problem was solved using methods (1b), (2a), (2b) and (2c) above. In the case of a nonlinear equation, a nonlinear solver must be used to solve the system resulting from the implicit step. Whenever possible we use a predicted value $\tilde{u}$ as the initial guess for the nonlinear solver. This is possible in the case of (1b), (2a) and (2b). However, in the case of (2c), the predictor itself is implicit and requires an initial guess. In this case we use u^n as

the initial guess for the solver used for $\tilde{u}$, and $\tilde{u}$ as the initial guess for the solver used for u^{n+1}. Although there is the possiblility that the nonlinear solver for $\tilde{u}$ will not converge due to the initial guess crossing the shock, this was not observed in our simulations.

We used the first order time stepping method, consisting of the first order forward Euler predictor and first order implicit Euler corrector (1b) to solve the nonlinear equation. As for the linear example, this implicit solver is stable for all step sizes Δt which we tested (corresponding to a CFL value up to $c = 10$). We compared these results to those obtained from using an explicit forward Euler method for time-stepping. As expected, the explicit solver is only stable for values $c \leq 0.5$. The first order implicit method is extremely stable and extremely diffusive; the price of the stability is the smearing of the solution. The implicit methods (2a), (2b), and (2c) are stable for values $c \leq 1$. The allowable timestep is doubled by using the implicit methods; however, when considering the cost of the implicit methods it is evident that they do not outperform the explicit Runge-Kutta method SSP(2,2). These numerical results indicate that the flux-implicit iWENO is a viable approach, and that the first order method performs as desired. Further work is required to find higher order time-stepping method which preserves the stability properties of the iWENO coupled with implicit Euler.

6 Conclusions and future directions

The main conclusion of this paper is that the success or failure of the flux-implicit iWENO spatial discretization depends heavily on the time-discretization used, as well as the numerical problem. When coupled with an appropriate time discretization, the iWENO method performs beautifully. Thus, we can turn our attention in future work to the task of finding an efficient and stable combination of time-discretization and iWENO. There is an inevitable trade-off between stability and diffusion, and a good balance may be difficult to strike.

In this context, future work shall consider different choices of predictor-corrector pairs, the effect of strong-stability-preserving methods, and diagonally split Runge-Kutta methods. Further study is also needed to evaluate the choice of nonlinear solvers. In this work, we used MATLAB's Jacobian based solver. It is possible to let MATLAB approximate the Jacobian using finite differences, however the stability implications of such a procedure are unclear and need to be examined closely. Finally, based on our positive results with scalar one dimensional problems, the iWENO idea clearly should be extended to systems and multidimensional problems.

Acknowledgements
The authors wish to thank Colin Macdonald for his thorough work in verifying the numerical results, and for his expert bug-catching.

References

[1] S. Gottlieb and J. S. Mullen, *An Implicit WENO Scheme for Steady-State Computation of Scalar Hyperbolic Equations*, in **Computational Fluid and Solid Mechanics 2003 (ed. K.J. Bathe)**.

[2] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I – Nonstiff Problems*. Second edition, Springer Series in Comput. Math. 8, Springer, 1993.

[3] Harten A., Engquist B., Osher S., Chakravarthy S., *Uniformly High Order Essentially Non-Oscillatory Schemes I*. SIAM Journal on Numerical Analysis 1987; 24:279-309.

[4] Harten A., Engquist B., Osher S., Chakravarthy S., *Uniformly High Order Essentially Non-Oscillatory Schemes III*. Journal of Computational Physics 1987; 71:231-303.

[5] G.-S. Jiang and C.-W. Shu, *Efficient Implementation of Weighted ENO Schemes*, J. of Comp. Physics, **vol. 126**, 1996, pp.202-228.

[6] LeVeque R.J., *Numerical Methods for Conservation Laws*, (Lectures in Mathematics). Basel: Birkhauser, 1992.

[7] X.-D. Liu, S. Osher and T. Chan, *Weighted Essentially Non-Oscillatory Schemes*, Journal of Computational Physics, **v. 115**, 1994, p.200.

[8] C.-W. Shu, High order finite difference and finite volume WENO schemes and discontinuous Galerkin methods for CFD , International Journal of Computational Fluid Dynamics, v17 (2003), pp.107-118.

[9] C.-W. Shu and S. Osher, *Efficient Implementation of Essentially Non-Oscillatory Shock-Capturing Schemes*, J. Comput. Phys. **v. 77**, 1988, pp.439-471.

Table 1: Verification of the spatial order of accuracy for of iWENO, in conjunction with method (2c). $\Delta t = 0.0001$ to final time $T_f = 0.01$.

	L^2 error	order
N = 20	1.18×10^{-5}	
N = 30	1.41×10^{-6}	5.24
N = 40	2.98×10^{-7}	5.40
N = 60	3.70×10^{-8}	5.14
N = 80	8.55×10^{-9}	5.09

Table 2: L^2 errors and order of accuracy for the second order methods with $\Delta t = 0.4\Delta x$, computed to final time $T_f = 1.0$.

	method 2a		method 2b		method 2c	
	error	order	error	order	error	order
N = 20	3.83×10^{-3}		3.83×10^{-3}		3.83×10^{-3}	
N = 40	8.51×10^{-4}	2.17	8.51×10^{-4}	2.17	8.51×10^{-3}	2.17
N = 60	3.63×10^{-4}	2.10	3.63×10^{-4}	2.10	3.63×10^{-4}	2.10
N = 80	1.99×10^{-4}	2.08	1.99×10^{-4}	2.08	1.99×10^{-4}	2.08
N = 100	1.25×10^{-4}	2.07	1.25×10^{-4}	2.07	1.25×10^{-4}	2.07
N = 120	8.63×10^{-5}	2.06	8.63×10^{-5}	2.06	8.63×10^{-5}	2.06

Table 3: L^2 errors and order of accuracy for the third order methods (3a) and (3b) with $\Delta t = \frac{1}{2}\Delta x$, computed to final time $T_f = 1.0$.

	method 3a		method 3b	
	error	order	error	order
N = 20	8.80×10^{-4}		7.58×10^{-4}	
N = 40	1.30×10^{-5}	6.07	1.11×10^{-4}	2.76
N = 60	3.83×10^{-6}	3.01	3.66×10^{-5}	2.74
N = 80	2.10×10^{-6}	2.08	1.60×10^{-5}	2.86
N = 100	1.21×10^{-6}	2.48	8.40×10^{-6}	2.91
N = 120	7.45×10^{-7}	2.67	4.92×10^{-6}	2.93

List of Captions

1. Oscillation forms on either side of the shock. Algorithms (2a) on left and (2b) on right. Using $\Delta t = \Delta x$, for $N = 100$. Note that the oscillation in method (2b) is smaller than that of method (2a), and decays over time. The codes were run for 40 steps (dashed line), 120 steps (dotted line) and 160 steps (solid line).

2. Linear problem: algorithms 2a (dashed line), 2b (dash-dot), 2c (solid line), and SSP(2,2) (dotted line) using $\Delta t = c\Delta x$, for $N = 40$, over 100 time steps. Top left: $c = 0.5$, top right: $c = 0.75$, bottom left: $c = 1.$, bottom right: $c = 1.25$ (instability)

3. Linear problem: algorithms 3a (solid line), 3b (dashed line) and 3c (dotted line), using $\Delta t = c\Delta x$, for $N = 100$, over 40 time steps. Top left $c = 0.5$ all methods are stable. Top right $c = 0.75$ methods 3a and 3c are stable. Bottom left $c = 1.0$, method 3a shows an oscillation near the shock, method 3b is unstable, and method 3c is starting to exhibit excessive smearing. Bottom right $c = 1.5$ all methods are unreliable.

4. Linear problem: algorithms 3a (dotted line), 3c (dashed line), 3d (solid line), using $\Delta t = c\Delta x$, for $N = 100$, over 20 time steps. Left $c = 0.5$, middle $c = 1.0$, right $c = 1.5$.

Figure 1: Oscillation forms on either side of the shock. Algorithms (2a) on left and (2b) on right. Using $\Delta t = \Delta x$, for $N = 100$. Note that the oscillation in method (2b) is smaller than that of method (2a), and decays over time. The codes were run for 40 steps (dashed line), 120 steps (dotted line) and 160 steps (solid line).

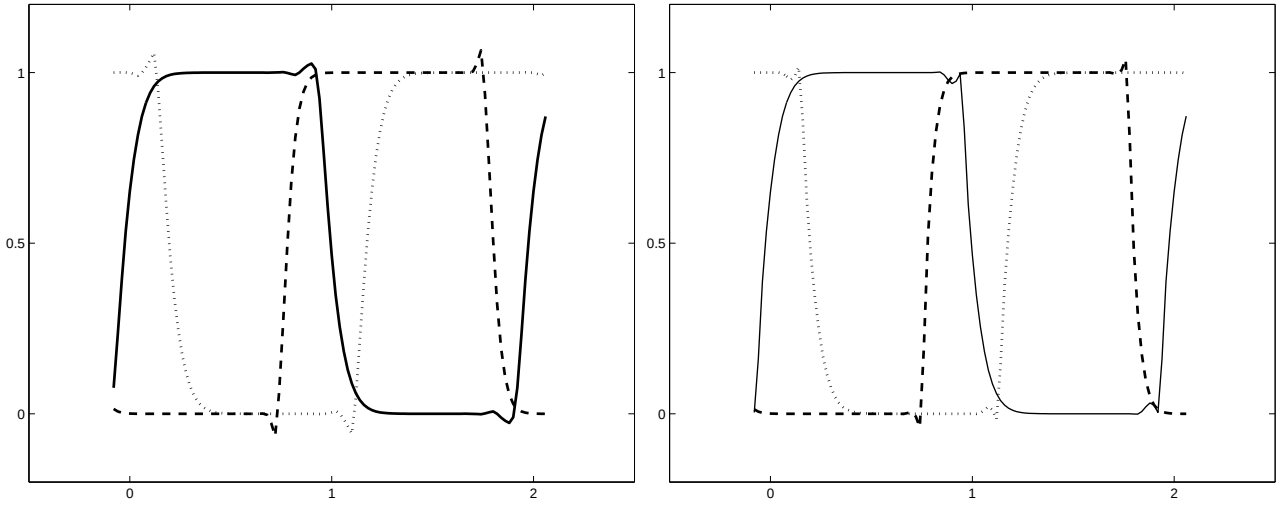

Figure 2: Linear problem: algorithms 2a (dashed line), 2b (dash-dot), 2c (solid line), and SSP(2,2) (dotted line) using $\Delta t = c\Delta x$, for $N = 40$, over 100 time steps. Top left: $c = 0.5$, top right: $c = 0.75$, bottom left: $c = 1.$, bottom right: $c = 1.25$ (instability)

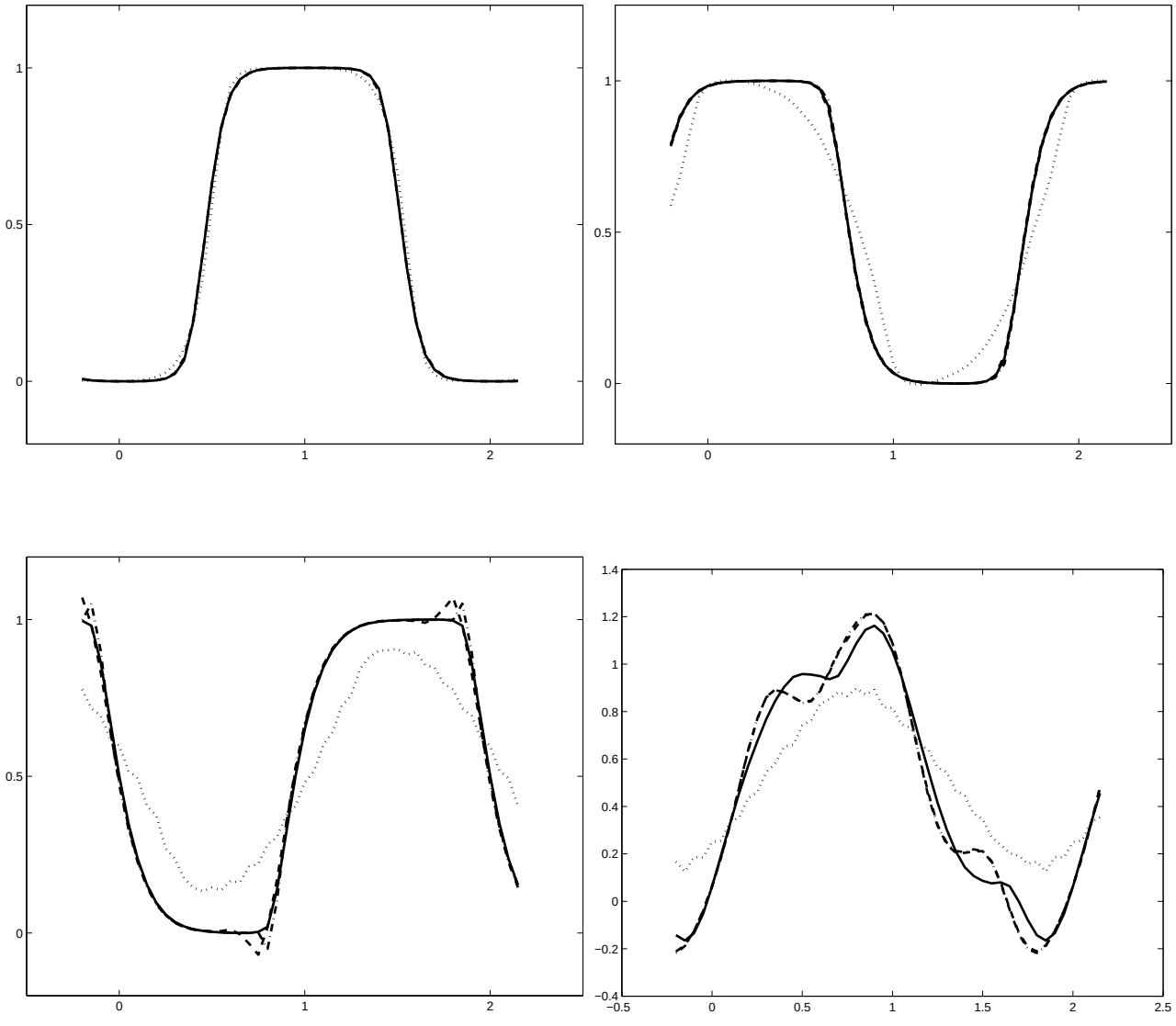

Figure 3: Linear problem: algorithms 3a (solid line), 3b (dashed line) and 3c (dotted line), using $\Delta t = c\Delta x$, for $N = 100$, over 40 time steps. Top left $c = 0.5$ all methods are stable. Top right $c = 0.75$ methods 3a and 3c are stable. Bottom left $c = 1.0$, method 3a shows an oscillation near the shock, method 3b is unstable, and method 3c is starting to exhibit excessive smearing. Bottom right $c = 1.5$ all methods are unreliable.

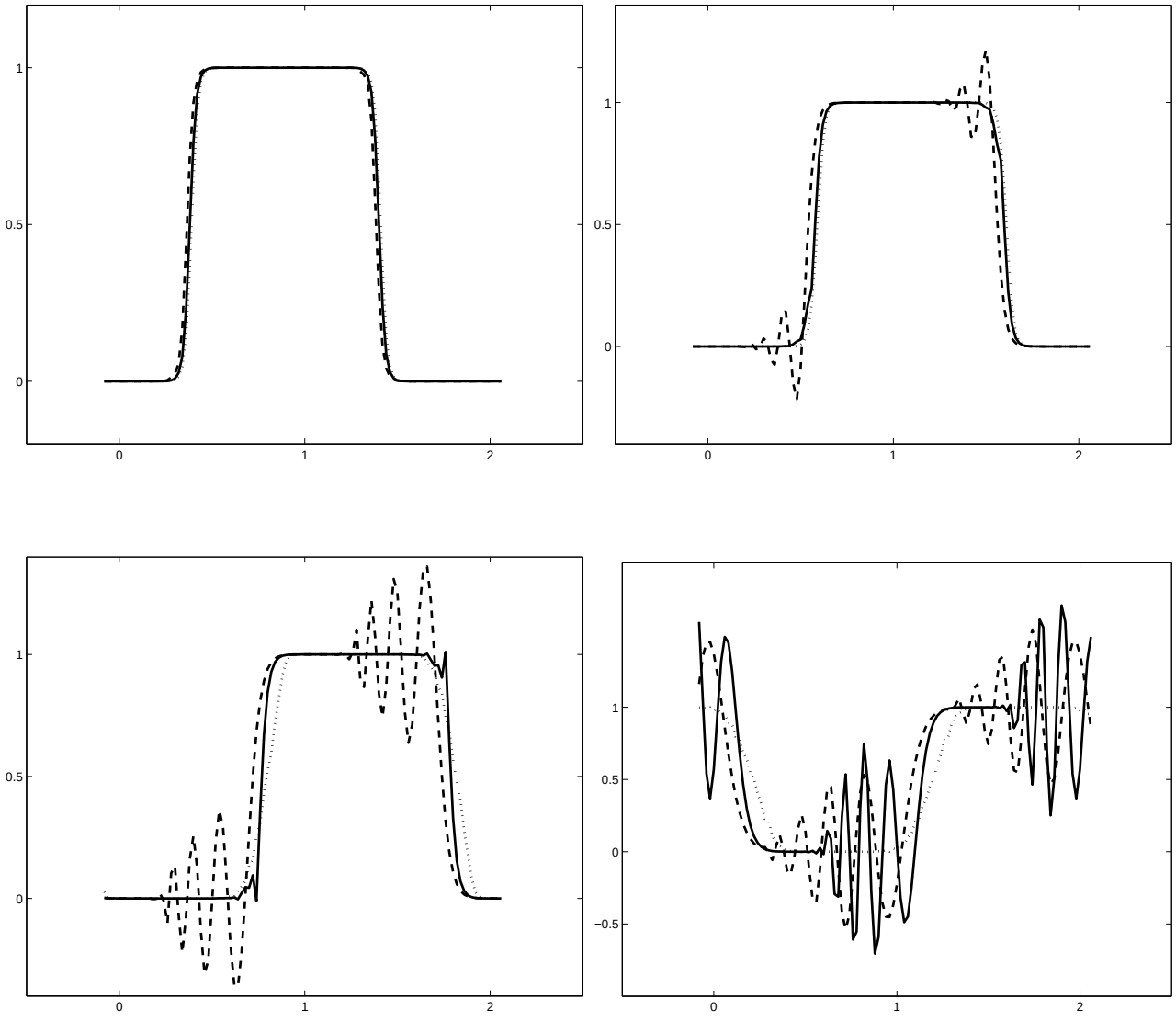

Figure 4: Linear problem: algorithms 3a (dotted line), 3c (dashed line), 3d (solid line), using $\Delta t = c\Delta x$, for $N = 100$, over 20 time steps. Left $c = 0.5$, middle $c = 1.0$, right $c = 1.5$.