WHOI-99-09

Woods Hole Oceanographic Institution

A Shallow-Water Model for Hydraulically Transcritical Flows

by

A.M. Rogerson

July 1999

Technical Report

Funding was provided by the Office of Naval Research under Grant Number N00014-93-1-1369.

Approved for public release; distribution unlimited.

19990907 106

DTIC QUALITY INSPECTED 4

WHOI-99-09

A Shallow-Water Model for Hydraulically Transcritical Flows

by

A.M. Rogerson

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

Technical Report

Funding was provided by the Office of Naval Research under Grant No. N00014-93-1-1369.

Reproduction in whole or in part is permitted for any purpose of the United States Government. This report should be cited as Woods Hole Oceanog. Inst. Tech. Rept., WHOI-99-09.

Approved for public release; distribution unlimited.

Approved for Distribution:

Terrence M. Joyce, Chair

Department of Physical Oceanography

Abstract

This document describes a numerical model that was developed to study two-dimensional, reduced-gravity, shallow-water flows. When the dynamics of these flows is strongly nonlinear, the flow may become hydraulically supercritical and discontinuities in the flow field may arise. The presence of discontinuities in the flow field requires a special numerical treatment in order to maintain both accuracy and stability in the numerically-approximated solution. In this model, a shock-capturing scheme called the Essentially Non-Oscillatory (ENO) scheme is implemented. The ENO scheme is a high-order, adaptive-stencil, finite-difference, characteristic-based scheme for hyperbolic equations that has been applied widely to flows governed by the Euler equations of gas dynamics. The model described in this document was developed for geophysical applications, and therefore includes the effects of rotation (constant Coriolis parameter), forcing (time dependent and/or spatially varying), and bottom drag (linear or nonlinear). The presentation includes the mathematical formulation of the model as well as instructions on how to prepare and execute model runs.

Contents

1 Introduction **1**

2 Model Equations **4**

3 The Essentially Non-Oscillatory (ENO) Scheme **7**

3.1 ENO Primer 8

3.1.1 Basic ENO-Roe Algorithm 11

3.1.2 Entropy Fix; the ENO-LLF Algorithm 12

3.1.3 Hybrid ENO; Biased Stencil Selection 15

3.1.4 Implementation Issues 16

3.1.5 Implemented ENO Algorithm 17

3.2 Time-stepping Scheme 19

4 ENO Scheme for the Shallow Water System **21**

5 The "Right-hand Side" **26**

5.1 Coriolis Term 26

5.2 Pressure Forcing Term 26

5.3 Bottom Stress Term 27

5.4 Grid Metric Term 28

6 Time-stepping Scheme **29**

7 Boundary Conditions **30**

7.1 Periodic BCs 31

7.2 Radiation BCs 31

7.3 Walled BCs for Non-rotating Flows 32

7.4 Walled BCs for Rotating Flows 32

8 Model Preparation and Execution **37**

8.1 Header File `enores.h` 37

8.2 Header File `gpath.h` 38

8.3 Common File `enoswcom.f` 38

8.4 Initial Condition Data File `enosw_in.dat` 41

8.5 Model Parameter File `enosw.parms` 42

8.6 Execution 43

8.7 Output files 43

Appendix: Curvilinear Grid-Generation Program `swgrid.f` **45**

References **49**

List of Figures

1 The portion of the difference table that could possibly be utilized in the approximation of $\hat{\boldsymbol{F}}_{i+1/2}$. 23

2 Schematic of the time advancement of the Riemann invariant along the eastern wall, ${}^{(1)}R_w^n$, and into the wall from the interior, ${}^{(2)}R_w^n$, at each step in the 3rd-order Runge-Kutta. 36

1 Introduction

The numerical model described in this document was developed as part of a research effort conducted by Roger Samelson and myself on orographically-modified winds in the coastal marine atmospheric boundary layer, funded by the ONR Coastal Meteorology Accelerated Research Initiative, grant N00014-93-1-1369.

The model was developed to investigate hydraulically transcritical, two-dimensional, reduced-gravity shallow-water flows (Rogerson 1999). The dynamics of these flows is strongly nonlinear, and when the flow velocity exceeds the gravity-wave phase speed the flow becomes hydraulically supercritical and discontinuities in the flow field may arise. The presence of discontinuities in the flow field requires a special numerical treatment in order to maintain both accuracy and stability in the numerically-approximated solution. In this model, the Essentially Non-Oscillatory (ENO) shock-capturing scheme is implemented. The ENO scheme is a high-order, adaptive-stencil, finite-difference, characteristic-based scheme for hyperbolic equations that has been applied widely to non-rotating flows, including, for example, flows governed by the 2-D Euler equations of gas dynamics. For the present application to the reduced-gravity shallow-water system, the effects of forcing (time dependent and/or spatially varying), bottom drag (linear or nonlinear), and rotation (constant Coriolis parameter) have been included.

The model is designed to approximate a solution on a user-specified orthogonal curvilinear grid. The original application involved channel-like domains with varying side-wall geometries which could be discretized onto an orthogonal curvilinear grid (using a conformal mapping program developed by Wilkin and modified successively by Signell, by Samelson, and by Rogerson). In this case, variations in the side-wall geometry can lead to hydraulic criticality in the two-dimensional flow. In contrast, problems in classic hydraulic flow theory typically involve a rectilinear channel geometry (or one-dimensional geometry) with varying

bottom topography. The presence of bottom topography is not currently included in the model, but could be incorporated easily *to simulate flows in which the layer interface never intersects the bottom.* For cases in which the layer depth goes to zero, significant modification would be required and it is possible that another approach (i.e., using a different numerical scheme entirely) would be more fruitful.

The types of boundary conditions that are currently implemented in the model reflect the fact that a channel geometry was used in the original application, although it is also possible to specify doubly-periodic boundary conditions. For the channel configuration, one wall of the channel may be "opened." All walled boundaries are free-slip. As with all numerical models, the boundary treatment is critical to the stability of the solution. Because the ENO scheme has high accuracy and low numerical dissipation, it is particularly sensitive to inappropriate and/or inaccurate boundary treatments. The presence of rotation in particular requires careful consideration in terms of the numerical treatment of walled boundaries.

This document has been created with the hope that it will serve as a useful aid to researchers who want to use and/or modify the model. The presentation is fairly technical, in that it includes a complete description of the ENO algorithm and some of the mathematical formalism behind the scheme, as well as instructions on how to prepare and execute model runs. The model equations are formulated in Section 2. In Section 3, the Essentially Non-Oscillatory (ENO) scheme is introduced, and a detailed description of the ENO algorithm as it applies to the flux in a one-dimensional scalar equation is presented as an example. The application of the ENO algorithm to the two-dimensional shallow-water system is discussed in Section 4. Finite-difference approximations for the non-conservative terms in the model equation are presented in Section 5, followed by the time-stepping scheme in Section 6. Boundary conditions and their implementations are presented in Section 7. The steps required to prepare and execute the model are outlined in Section 8. An auxiliary

program, `swgrid.f`, that can be used to generate an orthogonal curvilinear computational grid is described in the Appendix.

2 Model Equations

Consider the rotating, reduced-gravity (1-1/2 layer), shallow-water flow governed by the equations,

$$\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + f(\boldsymbol{k}\times\boldsymbol{u}) = -g'\nabla h - \frac{1}{\rho}\nabla p_a - \frac{1}{\rho h}\boldsymbol{\tau}_B \tag{1}$$

$$h_t + \nabla\cdot(\boldsymbol{u}h) = 0 \tag{2}$$

where $\boldsymbol{u} = (u, v)$ is the horizontal velocity vector, h is the depth of the layer (which is assumed to be nonzero), ρ is the density of the layer, $g' = g\Delta\rho/\rho$ is the reduced gravity, f is the Coriolis parameter, $p_a(x, y, t)$ is the imposed pressure forcing, and $\boldsymbol{\tau}_B$ is the stress at the bottom of the layer. The bottom stress is typically parameterized, and in the present case takes the form,

$$\boldsymbol{\tau}_B = \rho C_D|\boldsymbol{u}|\boldsymbol{u}.$$

Equations (1)–(2) are nondimensionalized using length, velocity, time, and layer depth scales L^*, U^*, $t^* = L^*/U^*$, and D^*, respectively, to yield,

$$\boldsymbol{u}_t + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + f_0(\boldsymbol{k}\times\boldsymbol{u}) = -F_r^{-2}\nabla h - \nabla P - \frac{r}{h}|\boldsymbol{u}|\boldsymbol{u} \tag{3}$$

$$h_t + \nabla\cdot(\boldsymbol{u}h) = 0 \tag{4}$$

where $f_0 \equiv fL^*/U^*$ is the inverse Rossby number, $F_r^{-2} \equiv g'D^*/U^{*2} = c^{*2}/U^{*2}$ is the squared inverse of the scaling Froude number, $\nabla P = (\nabla p/\rho)(L^*/U^{*2})$ is the nondimensional pressure gradient divided by the density of the layer, and $r = C_D L^*/D^*$ is the nondimensional drag coefficient.

Equations (3)–(4) can be generalized to any orthogonal curvilinear coordinate system. If (ζ, η) are the coordinates in the orthogonal curvilinear system, then the change in the position vector $\boldsymbol{x} = (x, y)$ in the Cartesian system can be written as,

$$\delta\boldsymbol{x} = m_1\delta\zeta\,\hat{\boldsymbol{\zeta}} + m_2\delta\eta\,\hat{\boldsymbol{\eta}}$$

where m_1 and m_2 are the coordinate metrics given by

$$m_1 = \sqrt{\left(\frac{\partial x}{\partial \zeta}\right)^2 + \left(\frac{\partial y}{\partial \zeta}\right)^2} \tag{5}$$

$$m_2 = \sqrt{\left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2} \tag{6}$$

(Batchelor 1967). It follows that the gradient of a scalar ϕ is,

$$\nabla\phi = \left(\frac{\hat{\boldsymbol{\zeta}}}{m_1}\frac{\partial}{\partial\zeta} + \frac{\hat{\boldsymbol{\eta}}}{m_2}\frac{\partial}{\partial\eta}\right)\phi$$

the divergence of the vector $\boldsymbol{v} = (v_1, v_2)$ is,

$$\nabla\cdot\boldsymbol{v} = \frac{1}{m_1 m_2}\left[\frac{\partial(m_2 v_1)}{\partial\zeta} + \frac{\partial(m_1 v_2)}{\partial\eta}\right]$$

and the gradient of $\boldsymbol{v}$ in the direction of $\boldsymbol{n}$ is,

$$\begin{aligned}\boldsymbol{n}\cdot\nabla\boldsymbol{v} &= \hat{\boldsymbol{\zeta}}\left[\boldsymbol{n}\cdot\nabla v_1 + \frac{v_2}{m_1 m_2}\left(n_1\frac{\partial m_1}{\partial\eta} - n_2\frac{\partial m_2}{\partial\zeta}\right)\right] \\ &+ \hat{\boldsymbol{\eta}}\left[\boldsymbol{n}\cdot\nabla v_2 - \frac{v_1}{m_1 m_2}\left(n_1\frac{\partial m_1}{\partial\eta} - n_2\frac{\partial m_2}{\partial\zeta}\right)\right].\end{aligned}$$

Therefore, in the (ζ, η)-coordinate system Equations (3)–(4) become,

$$\begin{aligned}u_t + \frac{1}{m_1}uu_\zeta + \frac{1}{m_2}vu_\eta + \frac{1}{m_1 m_2}v(um_{1_\eta} - vm_{2_\zeta}) - f_0 v = \\ -\frac{1}{m_1}(F_r^{-2}h_\zeta + P_\zeta) - \frac{r}{h}|\boldsymbol{u}|u\end{aligned} \tag{7}$$

$$\begin{aligned}v_t + \frac{1}{m_1}uv_\zeta + \frac{1}{m_2}vv_\eta - \frac{1}{m_1 m_2}u(um_{1_\eta} - vm_{2_\zeta}) + f_0 u = \\ -\frac{1}{m_2}(F_r^{-2}h_\eta + P_\eta) - \frac{r}{h}|\boldsymbol{u}|v\end{aligned} \tag{8}$$

$$h_t + \frac{1}{m_1 m_2}\left[(m_2 uh)_\zeta + (m_1 vh)_\eta\right] = 0 \tag{9}$$

where now u and v are the velocity components in the $\hat{\boldsymbol{\zeta}}$ and $\hat{\boldsymbol{\eta}}$ directions, respectively.

Equations (7)–(9) can be rewritten in flux form in terms of the state vector,

$$\boldsymbol{q} = \begin{pmatrix} uh \\ vh \\ h \end{pmatrix} = \begin{pmatrix} U \\ V \\ h \end{pmatrix}$$

yielding

$$\boxed{\boldsymbol{q}_t + \tfrac{1}{m_1}[\boldsymbol{F}(\boldsymbol{q})]_\zeta + \tfrac{1}{m_2}[\boldsymbol{G}(\boldsymbol{q})]_\eta = \boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{P}} + \boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{M}}} \tag{10}$$

where

$$\boldsymbol{F} = \begin{pmatrix} U^2/h + F_r^{-2}h^2/2 \\ UV/h \\ U \end{pmatrix}, \qquad \boldsymbol{G} = \begin{pmatrix} UV/h \\ V^2/h + F_r^{-2}h^2/2 \\ V \end{pmatrix}$$

are the fluxes in the $\hat{\boldsymbol{\zeta}}$ and $\hat{\boldsymbol{\eta}}$ directions, respectively, and

$$\boldsymbol{\mathcal{C}} = \begin{pmatrix} f_0 V \\ -f_0 U \\ 0 \end{pmatrix}, \quad \boldsymbol{\mathcal{P}} = \begin{pmatrix} -hP_\zeta/m_1 \\ -hP_\eta/m_2 \\ 0 \end{pmatrix}, \quad \boldsymbol{\mathcal{D}} = \begin{pmatrix} -r|\boldsymbol{U}|U/h^2 \\ -r|\boldsymbol{U}|V/h^2 \\ 0 \end{pmatrix}$$

$$\boldsymbol{\mathcal{M}} = \frac{1}{m_1 m_2} \begin{pmatrix} -\alpha_1 V/h - \alpha_2 U/h \\ \alpha_1 U/h - \alpha_2 V/h \\ -\alpha_2 \end{pmatrix}$$

are the terms resulting from the Coriolis, pressure forcing, bottom stress, and grid-metric contributions, respectively. In the expression for $\boldsymbol{\mathcal{M}}$,

$$\alpha_1 \equiv U m_{1_\eta} - V m_{2_\zeta}, \qquad \alpha_2 \equiv U m_{2_\zeta} + V m_{1_\eta}.$$

3 The Essentially Non-Oscillatory (ENO) Scheme

The Essentially Non-Oscillatory (ENO) scheme is a numerical method for hyperbolic conservation laws of the form,

$$\frac{\partial \boldsymbol{u}}{\partial t} + \sum_{i=1}^{d} \frac{\partial \boldsymbol{f}_i(\boldsymbol{u})}{\partial \boldsymbol{x}_i} = 0 \quad (\text{or} = \boldsymbol{r}(\boldsymbol{u}, \boldsymbol{x}, t), \text{ a residual or forcing term}) \tag{11}$$

$$\boldsymbol{u}(\boldsymbol{x}, t = 0) = \boldsymbol{u}_0(\boldsymbol{x}) \tag{12}$$

where $\boldsymbol{u}(\boldsymbol{x}, t) = (u_1, u_2, \ldots, u_m)^T$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)^T$, d is the spatial dimension of the problem, m is the number of independent variables, and t is time. The system is hyperbolic in the sense that any linear combination of the Jacobian matrices,

$$\sum_{i=1}^{d} \gamma_i \frac{\partial \boldsymbol{f}_i}{\partial \boldsymbol{u}}$$

for real $\gamma_i = (\gamma_1, \gamma_2, \ldots, \gamma_d)$, always has m real eigenvalues and a complete set of eigenvectors.

It is well known that the solution to this equation can develop discontinuities even when the initial condition $\boldsymbol{u}_0(\boldsymbol{x})$ is smooth. Traditional finite-difference techniques will provide poor numerical approximations of the solution in this case. In general, shock-capturing schemes aim to:

- achieve high accuracy in regions where the solution is smooth;
- maintain sharp profiles of discontinuities (i.e., avoid excessive numerical dissipation);
- avoid spurious oscillations in the vicinity of discontinuities;
- accurately represent the location and speed of discontinuities; and
- avoid non-physical solutions (e.g., entropy-violating expansion shocks).

The ENO scheme satisfies these criteria. In addition, the ENO scheme is *globally* high-order, losing only one order of accuracy in discontinuous regions compared to smooth regions.

A good review of numerical methods for conservation laws can be found in the book by LeVeque (1990). A more complete description of the ENO scheme can be found in the papers by Shu and Osher (1988 and 1989), and references contained therein.

3.1 ENO Primer

To illustrate the fundamental principles of the ENO scheme and simplify the discussion, consider the one-dimensional scalar equation,

$$u_t + [f(u)]_x = 0 \tag{13}$$

for some function $f(u)$.

A numerical approximation to (13) is called *conservative* if it is of the form,

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}\left(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}\right) \tag{14}$$

where $\hat{f}_{i+1/2} = \hat{f}(u_{i-l}, \ldots, u_{i+k})$ for some $l, k > 0$ (LeVeque 1990). The key to shock-capturing schemes hinges on how the numerical fluxes at the "half" grid points, $\hat{f}_{i+1/2}$, are estimated. By using a conservative method, the Lax–Wendroff theorem guarantees that *if* the numerical scheme converges to a numerical solution, then the numerical solution does indeed approximate a weak solution of the partial differential equation (Lax and Wendroff 1960). Convergence of the numerical scheme can often be proved if the scheme is total-variation diminishing (TVD) or total-variation bounded (TVB) (Harten 1984). The total variation is defined as,

$$TV(u) = \sum_i |u_{i+1} - u_i|$$

and the scheme is termed TVD if

$$TV(u^{n+1}) \leq TV(u^n)$$

for all n. The scheme is termed TVB in $0 \leq t \leq T$ if

$$TV(u^n) < B$$

for some fixed B that depends only on $TV(u^0)$, and for all n and Δt such that $0 \leq n\Delta t \leq T$. The ENO scheme is a descendent of TVD and TVB schemes, but is unique in that it uses *adaptive* stencils to compute the numerical approximations of the flux, $\hat{f}_{i+1/2}$. High accuracy is achieved by approximating $\hat{f}_{i+1/2}$ and $\hat{f}_{i-1/2}$ to very high order in such a way that the difference yields a high-order estimate for the derivative $\partial f/\partial x$ (as opposed to seeking a high-order approximation for $\partial f/\partial x$ directly). The use of an $(r+1)$-point adaptive stencil yields $(r+1)$-order accuracy in smooth regions of the flow and r-order accuracy right up to discontinuities, and is formally r-order accurate. A high-order interpolating polynomial is constructed at each time step to approximate the flux from information at the surrounding grid points, avoiding regions where discontinuities are present. For example, to compute $\hat{f}_{i+1/2}$ using a 4-point stencil in a smooth region of the flow, the centered stencil

$$\hat{f}_{i+1/2} = \hat{f}(u_{i-1}, u_i, u_{i+1}, u_{i+2})$$

would be utilized, while in the presence of a local discontinuity, say located near x_{i-1}, the stencil would be shifted to the right to obtain information from the smoother region, e.g.,

$$\hat{f}_{i+1/2} = \hat{f}(u_i, u_{i+1}, u_{i+2}, u_{i+3}).$$

The fact that the scheme involves an adaptive stencil application has hindered progress towards a convergence theory for the ENO scheme. Nevertheless, numerical convergence has been demonstrated empirically in a number of challenging problems in gas dynamics including Riemann problems, shock-wave interactions, and shock-turbulence interactions (see, for example, Hannappel, Hauser and Friedrich 1995).

To aid the stencil selection process and the construction of the interpolating polynomial, *divided differences* are computed for the flux. The divided differences of a function $w = w(x)$ are defined recursively as,

$$w[x_i] \quad = \quad w(x_i)$$

$$w[x_i, \ldots, x_{i+k}] = \frac{w[x_{i+1}, \ldots, x_{i+k}] - w[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

where $w[x_i, \ldots, x_{i+k}]$ denotes the divided difference of w of order k. If w is smooth (i.e., w is infinitely differentiable; $w \in C^\infty$) in the interval $[x_i, x_{i+k}]$, then

$$w[x_i, \ldots, x_{i+k}] = \frac{1}{k!}\frac{d^k w(\zeta)}{dx^k}, \qquad \zeta \in [x_i, x_{i+k}],$$

but if w is discontinuous in the p-th derivative ($0 \leq p \leq k$), then

$$w[x_i, \ldots, x_{i+k}] = O\left(\Delta x^{-k+p}[w^{(p)}]\right)$$

where $[w^{(p)}]$ denotes the jump in the p-th derivative (Harten et al. 1987). Therefore, divided differences can be used to detect discontinuities. Now consider the function $h(x)$ such that,

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

It follows easily that if H is the primitive function of h, i.e.,

$$H(x) = \int_{-\infty}^{x} h(\zeta)d\zeta$$

then

$$H(x_{i+1/2}) = \int_{-\infty}^{x_{i+1/2}} h(\zeta)d\zeta = \sum_{k=-\infty}^{i}\int_{x_{k-1/2}}^{x_{k+1/2}} h(\zeta)d\zeta = \sum_{k=-\infty}^{i}(x_{k+1/2} - x_{k-1/2})f(u_k)$$

and

$$H(x_{i+1/2}) - H(x_{i-1/2}) = (x_{i+1/2} - x_{i-1/2})f(u_i).$$

Therefore, the divided differences of H can be obtained directly from the divided differences of f,

$$\begin{aligned} H[x_{i-1/2}, x_{i-1/2+1}] &= f[u_i] \\ H[x_{i-1/2}, \ldots, x_{i-1/2+k}] &= \frac{1}{k}f[u_i, \ldots, u_{i+k-1}]. \end{aligned}$$

3.1.1 Basic ENO-Roe Algorithm

We seek a solution to

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

in which $[f(u)]_x$ is approximated via,

$$\frac{1}{\Delta x}\left(\hat{f}_{i+1/2} - \hat{f}_{i-1/2}\right).$$

The numerical fluxes $\hat{f}$ are computed to r-th order using the algorithm outlined below. To simplify the notation, we will denote the k-th divided difference of H at $x_{i-1/2}$ as,

$$H^k_{i-1/2} \equiv H[x_{i-1/2}, \ldots, x_{i-1/2+k}].$$

The "ENO-Roe" algorithm (Shu and Osher 1989) for $\hat{f}_{i+1/2}$ is:

1. Compute the divided differences,

$$\begin{aligned} H^1_{i-1/2} &= f[u_i] = f(u_i) \\ H^k_{i-1/2} &= \frac{1}{k} f[u_i, \ldots, u_{i+k-1}], \qquad k = 2, \ldots, r+1 \end{aligned}$$

2. Estimate the local sign of df/du at $x_{i+1/2}$ by computing the Roe speed (Roe 1981),

$$a_{i+1/2} = \frac{f(u_{i+1}) - f(u_i)}{u_{i+1} - u_i}$$

3. Let $s(k)$ denote the starting stencil point at stage k in the selection process. Select the first stencil point ($k = 1$) in the upwind direction,

$$s(1) = \begin{cases} i & \text{if } a_{i+1/2} \geq 0 \\ i+1 & \text{otherwise} \end{cases}$$

4. At each stage thereafter, add an additional point to the stencil from the "smoother" region, using the difference table for the comparison;

$$s(k+1) = \begin{cases} s(k) - 1 & \text{if } H^k_{s(k)-1/2} \geq H^k_{s(k)-1/2-1} \\ s(k) & \text{otherwise} \end{cases}$$

When $H^k_{s(k)-1/2} \geq H^k_{s(k)-1/2-1}$, the starting stencil location is backed up; we add the point to the "left" of previous starting point. When $H^k_{s(k)-1/2} < H^k_{s(k)-1/2-1}$, the starting stencil location is unchanged; we effectively add a point to the "right-hand" end of the stencil.

5. After the $(r+1)$-point stencil has been selected, we construct a high-order interpolating polynomial for the primitive function,

$$\begin{aligned} Q^{(r+1)}(x) &= H^1_{s(1)-1/2}(x - x_{s(1)-1/2}) \\ &+ H^2_{s(2)-1/2}(x - x_{s(1)-1/2})(x - x_{s(1)-1/2+1}) \\ &+ H^3_{s(3)-1/2}(x - x_{s(2)-1/2})(x - x_{s(2)-1/2+1})(x - x_{s(2)-1/2+2}) + \cdots \\ &= H^1_{s(1)-1/2}(x - x_{s(1)-1/2}) + \sum_{k=2}^{r+1} \left\{ H^k_{s(k)-1/2} \left[\prod_{\alpha=0}^{k-1} (x - x_{s(k-1)-1/2+\alpha}) \right] \right\} \end{aligned}$$

whose derivative is,

$$\begin{aligned} \frac{d}{dx} Q^{(r+1)}(x) &= H^1_{s(1)-1/2} \\ &+ H^2_{s(2)-1/2}[(x - x_{s(1)-1/2}) + (x - x_{s(1)-1/2+1})] + \cdots \\ &= H^1_{s(1)-1/2} + \sum_{k=2}^{r+1} \left\{ H^k_{s(k)-1/2} \left[\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (x - x_{s(k-1)-1/2+\beta}) \right] \right\} \end{aligned}$$

The interpolating polynomial for $\hat{f}_{i+1/2}$ is then,

$$\begin{aligned} \hat{f}_{i+1/2} &= \left. \frac{d}{dx} Q^{(r+1)} \right|_{x_{i+1/2}} \\ &= H^1_{s(1)-1/2} \\ &+ \sum_{k=2}^{r+1} \left\{ H^k_{s(k)-1/2} \left[\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (x_{i+1/2} - x_{s(k-1)-1/2+\beta}) \right] \right\}. \end{aligned} \tag{15}$$

3.1.2 Entropy Fix; the ENO-LLF Algorithm

The ENO-Roe scheme described above does admit a non-physical entropy-violating expansion shock but the problem can be easily remedied (Shu and Osher 1989). If $f'(u)$ does

not change sign between u_i and u_{i+1}, then we compute the numerical flux, $\hat{f}_{i+1/2}$, according to Equation (15) in the ENO-Roe fashion. If $f'(u)$ does change sign between u_i and u_{i+1}, then we compute the numerical flux in a slightly different fashion based on the local Lax–Friedrichs flux (the "ENO-LLF" scheme) described below.

The flux, $f(u)$, can be split into two parts

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

where

$$\begin{aligned} f^+(u) &= \frac{1}{2}[f(u) + \alpha u] \\ f^-(u) &= \frac{1}{2}[f(u) - \alpha u] \end{aligned}$$

with

$$\alpha = \max |f'(u)|$$

so that

$$\begin{aligned} \frac{\partial f^+}{\partial u} &> 0 \\ \frac{\partial f^-}{\partial u} &< 0. \end{aligned}$$

The numerical flux is similarly split in the Lax–Friedrichs fashion,

$$\hat{f}_{i+1/2} = \hat{f}^+_{i+1/2} + \hat{f}^-_{i+1/2}$$

and ENO approximations are computed for each component as follows:

1. Compute the divided differences for ${}^+H$ and ${}^-H$,

$$\begin{aligned} {}^{\pm}H^1_{i-1/2} &= \frac{1}{2}(f[u_i] \pm \alpha_{i+1/2} u[x_i]) \\ {}^{\pm}H^k_{i-1/2} &= \frac{1}{k}\frac{1}{2}(f[u_i, \ldots, u_{i+k-1}] \pm \alpha_{i+1/2} u[x_i, \ldots, x_{i+k-1}]), \qquad k = 2, \ldots, r+1 \end{aligned}$$

2. Define

$$\alpha_{i+1/2} = \max_{u_i \le u \le u_{i+1}} |f'(u)|$$

3. Select the first stencil point for the + and − components in the upwind direction with respect to the half-grid point $i + 1/2$. Since

$$\left.\frac{\partial f^+}{\partial u}\right|_{x_{i+1/2}} = \tfrac{1}{2}\,\alpha_{i+1/2} \quad > 0$$
$$\left.\frac{\partial f^-}{\partial u}\right|_{x_{i+1/2}} = -\tfrac{1}{2}\,\alpha_{i+1/2} \quad < 0$$

the first stencil points are chosen as,

$$\begin{aligned} s^+(1) &= i \\ s^-(1) &= i+1 \end{aligned}$$

4. Select the rest of the stencil in the ENO fashion,

$$s^{\pm}(k+1) = \begin{cases} s^{\pm}(k) - 1 & \text{if } {}^{\pm}H^k_{s^{\pm}(k)-1/2} \geq^{\pm} H^k_{s^{\pm}(k)-1/2-1} \\ s^{\pm}(k) & \text{otherwise} \end{cases}$$

5. Form the interpolating polynomials for $\hat{f}^+_{i+1/2}$ and $\hat{f}^-_{i+1/2}$,

$$\begin{aligned} \hat{f}^{\pm}_{i+1/2} &= {}^{\pm}H^1_{s^{\pm}(1)-1/2} \\ &+ \sum_{k=2}^{r+1} \left\{ {}^{\pm}H^k_{s^{\pm}(k)-1/2} \left[\sum_{\gamma=0}^{k-1} \prod_{\beta=0,\gamma\neq\beta}^{k-1} (x_{i+1/2} - x_{s^{\pm}(k-1)-1/2+\beta}) \right] \right\} \end{aligned} \tag{16}$$

6. The numerical flux computed using the ENO-LLF scheme is then,

$$\hat{f}_{i+1/2} = \hat{f}^+_{i+1/2} + \hat{f}^-_{i+1/2} \tag{17}$$

To prevent entropy-violating expansion shocks in the ENO-Roe scheme, the numerical flux must be computed according to Equations (16)–(17) if $f'(u)$ changes sign between u_i and u_{i+1}, and not in accordance with Equation (15).

Since the entropy fix for the ENO-Roe scheme requires the implementation of an additional scheme, ENO-LLF, one might wonder if it would be better to compute all of the fluxes using the ENO-LLF scheme in the first place. Employing the ENO-LLF scheme globally would certainly be simpler (algorithmically) than ENO-Roe with entropy fix, and in the shallow-water model the user has the option to select either scheme. However, the numerical dissipation associated with the ENO-Roe scheme is less than that of ENO-LLF (Shu and Osher 1989), so there is less shock smearing and better overall accuracy with ENO-Roe. In general, I have found the ENO-Roe (with entropy fix) solutions to be superior to those generated by ENO-LLF.

3.1.3 Hybrid ENO; Biased Stencil Selection

Adaptive stencil selection is the key feature of the ENO scheme. It allows an interpolating polynomial to be constructed using a stencil that avoids discontinuous regions of the flow. It is inevitable that in this process linearly unstable stencils will be selected. In general, however, the selection of linearly unstable stencils does not lead to numerical instability since rapid stencil switching is often observed (Harten et al. 1987). However, in *smooth* regions of the flow, the use of linearly unstable stencils (and the rapid stencil switching that accompanies it) can degrade the convergence rate of the solution (Rogerson and Meiburg 1990). The error reduction during mesh refinement is not uniform and in some cases grid refinement can produce an increase in the truncation error. This degeneration in accuracy can be remedied by using fixed linearly stable stencils in smooth regions of the solution and adaptive stencils where strong gradients are present. A simple modification to the basic ENO algorithm combines the use of fixed and adaptive stencils, creating a "hybrid" ENO scheme that restores the desired accuracy (Shu 1990). In the stencil selection process (item 4 in ENO-Roe and ENO-LLF algorithms) we simply replace,

$$s(k+1) = \begin{cases} s(k)-1 & \text{if } H^k_{s(k)-1/2} \geq H^k_{s(k)-1/2-1} \\ s(k) & \text{otherwise} \end{cases}$$

with

$$
\begin{aligned}
&\text{if } s(k) > c(k) \text{ then} \\
&\quad s(k+1) = \begin{cases} s(k)-1 & \text{if } 2H^k_{s(k)-1/2} \geq H^k_{s(k)-1/2-1} \\ s(k) & \text{otherwise} \end{cases} \\
&\text{else} \\
&\quad s(k+1) = \begin{cases} s(k)-1 & \text{if } H^k_{s(k)-1/2} \geq 2H^k_{s(k)-1/2-1} \\ s(k) & \text{otherwise} \end{cases}
\end{aligned}
\tag{18}
$$

where $c(k)$ is the leftmost grid point in the centered stencil. The weighting factor of 2 in Equation (18) is used for the reasons provided by Shu (1990). Restated, the modified algorithm is,

> if the stencil is to the right of the centered stencil [i.e., $s(k) > c(k)$] then
> favor adding a point on the left [i.e., $s(k+1) = s(k) - 1$]
> else
> favor adding a point on the right [i.e., $s(k+1) = s(k)$].

The modified algorithm biases the stencil selection towards the linearly stable centered stencil in smooth regions where $H^k_{s(k)-1/2}$ and $H^k_{s(k)-1/2-1}$ are the same order of magnitude.

3.1.4 Implementation Issues

Notice that on an equally-spaced grid, Equation (15) becomes,

$$
\hat{f}_{i+1/2} = H^1_{s(1)-1/2} + \sum_{k=2}^{r+1} \left\{ H^k_{s(k)-1/2} (\Delta x)^{k-1} \left[\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (i - s(k-1) + 1 - \beta) \right] \right\}. \tag{19}
$$

Therefore, if we compute the *undivided* differences,

$$
\begin{aligned}
\mathcal{H}^1_{i-1/2} &= f(u_i) \\
\mathcal{H}^k_{i-1/2} &= \mathcal{H}^{k-1}_{i+1/2} - \mathcal{H}^{k-1}_{i-1/2}, \qquad k = 2, \ldots, r+1
\end{aligned}
$$

(19) becomes,

$$
\hat{f}_{i+1/2} = \mathcal{H}^1_{s(1)-1/2} + \sum_{k=2}^{r+1} \left\{ \mathcal{H}^k_{s(k)-1/2} \left[\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (i - s(k-1) + 1 - \beta) \right] \right\}. \tag{20}
$$

Since the coefficients in the summation,

$$
\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (i - s(k-1) + 1 - \beta), \qquad k = 2, \ldots, r+1
$$

depend on $(i - s(k))$ (the difference between the grid point in question and the left-most stencil point) and not on i itself, the set of possible coefficients can be precomputed for use in Equation (20). A similar simplification can be applied to the split numerical fluxes in the ENO-LLF algorithm (Equation (16)).

When the grid is not uniform, we make a change of variables, e.g., $x \to \zeta$, and reformulate the governing equation,

$$u_t + \frac{1}{m}[f(u)]_\zeta = 0$$

where $m = \partial x / \partial \zeta$ is the grid metric. The approximation for the flux, $\hat{f}_{i+1/2}$, then proceeds on the ζ grid for which $\Delta\zeta$ is constant.

3.1.5 Implemented ENO Algorithm

The ENO algorithm that is implemented in the model is a hybrid ENO-Roe scheme with the "entropy fix" (Shu and Osher 1988; Shu and Osher 1989; Rogerson and Meiburg 1990; Shu 1990). Below, we recapitulate the algorithm for clarity as it applies to the 1-D scalar equation (13).

1. Compute the undivided differences for f and u,

$$\begin{aligned}
\mathcal{H}^1_{i-1/2} &= f[u_i] = f(u_i) \\
\mathcal{U}^1_{i-1/2} &= u[x_i] = u(x_i) \\
\mathcal{H}^k_{i-1/2} &= \frac{1}{k} f[u_i, \ldots, u_{i+k-1}] \\
\mathcal{U}^k_{i-1/2} &= \frac{1}{k} u[x_i, \ldots, x_{i+k-1}], \qquad k = 2, \ldots, r+1
\end{aligned}$$

2. Compute the Roe speed,

$$a_{i+1/2} = \frac{f(u_{i+1}) - f(u_i)}{u_{i+1} - u_i}$$

and

$$\alpha_{i+1/2} = \max_{u_i \le u \le u_{i+1}} |f'(u)|$$

3. Form the undivided difference tables for the split local Lax–Friedrichs flux,

$$^{\pm}\mathcal{H}^k_{i-1/2} = \frac{1}{2}(\mathcal{H}^k_{i-1/2} \pm \alpha_{i+1/2}\,\mathcal{U}^k_{i-1/2}), \qquad k = 1, \ldots, r+1$$

4. If $f'(u)$ does not change sign between u_i and u_{i+1}, then construct $\hat{f}_{i+1/2}$ using the ENO-Roe algorithm.

   - Select the first stencil point in the upwind direction,

$$s(1) = \begin{cases} i & \text{if } a_{i+1/2} \geq 0 \\ i+1 & \text{otherwise} \end{cases}$$

   - Select the remaining stencil points. Bias the stencil selection towards the linearly stable centered stencil, $c(k)$, in smooth regions.

$$\begin{aligned} &\text{if } s(k) > c(k) \text{ then} \\ &\qquad s(k+1) = \begin{cases} s(k)-1 & \text{if } 2\mathcal{H}^k_{s(k)-1/2} \geq \mathcal{H}^k_{s(k)-1/2-1} \\ s(k) & \text{otherwise} \end{cases} \\ &\text{else} \\ &\qquad s(k+1) = \begin{cases} s(k)-1 & \text{if } \mathcal{H}^k_{s(k)-1/2} \geq 2\mathcal{H}^k_{s(k)-1/2-1} \\ s(k) & \text{otherwise} \end{cases} \end{aligned}$$

   - Compute the interpolating polynomial for $\hat{f}_{i+1/2}$,

$$\begin{aligned} \hat{f}_{i+1/2} &= \mathcal{H}^1_{s(1)-1/2} \\ &+ \sum_{k=2}^{r+1} \left\{ \mathcal{H}^k_{s(k)-1/2} \left[\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (i - s(k-1) + 1 - \beta) \right] \right\} \end{aligned}$$

5. If $f'(u)$ changes sign between u_i and u_{i+1}, then construct $\hat{f}_{i+1/2}$ using the ENO-LLF algorithm.

   - Select the first stencil point in the upwind direction,

$$\begin{aligned} s^+(1) &= i \\ s^-(1) &= i+1 \end{aligned}$$

- Select the remaining stencil points, $s^+(k)$ and $s^+(k)$, in the hybrid ENO fashion,

$$\text{if } s^{\pm}(k) > c(k) \text{ then}$$
$$s^{\pm}(k+1) = \begin{cases} s^{\pm}(k) - 1 & \text{if } 2({}^{\pm}\mathcal{H}^k_{s(k)-1/2}) \geq^{\pm} \mathcal{H}^k_{s(k)-1/2-1} \\ s^{\pm}(k) & \text{otherwise} \end{cases}$$
$$\text{else}$$
$$s^{\pm}(k+1) = \begin{cases} s^{\pm}(k) - 1 & \text{if } {}^{\pm}\mathcal{H}^k_{s(k)-1/2} \geq 2({}^{\pm}\mathcal{H}^k_{s(k)-1/2-1}) \\ s^{\pm}(k) & \text{otherwise} \end{cases}$$

- Form the interpolating polynomials for $\hat{f}^+_{i+1/2}$ and $\hat{f}^-_{i+1/2}$,

$$\begin{aligned} \hat{f}^{\pm}_{i+1/2} &= {}^{\pm}\mathcal{H}^1_{s^{\pm}(1)-1/2} \\ &+ \sum_{k=2}^{r+1} \left\{ {}^{\pm}\mathcal{H}^k_{s^{\pm}(k)-1/2} \left[\sum_{\alpha=0}^{k-1} \prod_{\beta=0,\alpha\neq\beta}^{k-1} (i - s^{\pm}(k-1) + 1 - \beta) \right] \right\} \end{aligned}$$

- Sum the split fluxes to obtain the interpolating polynomial for the numerical flux,

$$\hat{f}_{i+1/2} = \hat{f}^+_{i+1/2} + \hat{f}^-_{i+1/2}$$

6. Repeat the previous steps to compute $\hat{f}_{i-1/2}$ and approximate $(\partial f/\partial u)_i$ as,

$$\left.\frac{\partial f}{\partial u}\right|_i \doteq \frac{1}{\Delta x}(\hat{f}_{i+1/2} - \hat{f}_{i-1/2})$$

3.2 Time-stepping Scheme

ENO spatial approximations are typically paired with TVD time-stepping schemes (see Shu and Osher (1988) for background). Shu and Osher (1988) formulated several Runge-Kutta time-stepping schemes that have optimal (i.e., large) CFL restrictions. For the equation,

$$\frac{\partial u}{\partial t} = L(u)$$

the 2nd-order and 3rd-order Runge-Kutta methods are,

$$u^a = u^n + \Delta t L(u^n) \tag{21}$$

$$u^{n+1} = \frac{1}{2}u^n + \frac{1}{2}u^a + \frac{1}{2}\Delta t L(u^a) \tag{22}$$

and

$$u^a = u^n + \Delta t L(u^n) \tag{23}$$

$$u^b = \frac{3}{4}u^n + \frac{1}{4}u^a + \frac{1}{4}\Delta t L(u^a) \tag{24}$$

$$u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}u^b + \frac{2}{3}\Delta t L(u^b). \tag{25}$$

The theoretical CFL coefficient for both schemes is 1. In practice, the recommended maximal CFL coefficient is 0.6 when $L(u)$ is approximated with an ENO algorithm (C.-W. Shu, personal communication), i.e.,

$$\frac{\Delta t}{\Delta x} \max_u |f'(u)| \leq 0.6. \tag{26}$$

4 ENO Scheme for the Shallow Water System

Reconsider the shallow-water system in flux form on a curvilinear grid (see Equation (10), Section 2),

$$\boldsymbol{q}_t + \frac{1}{m_1}[\boldsymbol{F}(\boldsymbol{q})]_\zeta + \frac{1}{m_2}[\boldsymbol{G}(\boldsymbol{q})]_\eta = \boldsymbol{\mathcal{Q}} \tag{27}$$

where

$$\boldsymbol{q} = \begin{pmatrix} uh \\ vh \\ h \end{pmatrix} = \begin{pmatrix} U \\ V \\ h \end{pmatrix}$$

$$\boldsymbol{F} = \begin{pmatrix} U^2/h + F_r^{-2}h^2/2 \\ UV/h \\ U \end{pmatrix}, \qquad \boldsymbol{G} = \begin{pmatrix} UV/h \\ V^2/h + F_r^{-2}h^2/2 \\ V \end{pmatrix}$$

and $\boldsymbol{\mathcal{Q}} = \boldsymbol{\mathcal{C}}+\boldsymbol{\mathcal{P}}+\boldsymbol{\mathcal{D}}+\boldsymbol{\mathcal{M}}$ is the sum of all of the terms on the right-hand side of Equation (10).

For systems of equations, the ENO algorithm is applied to each local characteristic field, not to each state variable. To illustrate how the fluxes in the $\hat{\zeta}$ direction are computed, consider the one-dimensional conservation equation,

$$\boldsymbol{q}_t + \frac{1}{m_1}[\boldsymbol{F}(\boldsymbol{q})]_\zeta = 0 \tag{28}$$

or

$$\boldsymbol{q}_t + \frac{1}{m_1}\boldsymbol{A}\boldsymbol{q}_\zeta = 0 \tag{29}$$

where

$$\boldsymbol{A} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}} = \begin{pmatrix} 2u & 0 & -u^2 + F_r^{-2}h \\ v & u & -uv \\ 1 & 0 & 0 \end{pmatrix}.$$

The matrix $\boldsymbol{A}$ has eigenvalues and left and right eigenvectors,

$$\begin{array}{lllllllllllll}
\lambda^{(1)} & = & u & \boldsymbol{l}^{(1)} & = & & (& 0 & 1 & -v &) & \boldsymbol{r}^{(1)} & = (\; 0 \quad 1 \quad 0 \;)^T \\
\lambda^{(2)} & = & u+c & \boldsymbol{l}^{(2)} & = & -\frac{1}{2c} & (& -1 & 0 & u-c &) & \boldsymbol{r}^{(2)} & = (\; u+c \quad v \quad 1 \;)^T \\
\lambda^{(3)} & = & u-c & \boldsymbol{l}^{(3)} & = & \frac{1}{2c} & (& -1 & 0 & u+c &) & \boldsymbol{r}^{(3)} & = (\; u-c \quad v \quad 1 \;)^T
\end{array}$$

where $c = \sqrt{F_r^{-2}h}$. Equation (29) can be projected onto the eigenspace via

$$\boldsymbol{S}^{-1}\boldsymbol{q}_t + \frac{1}{m_1}(\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S})\boldsymbol{S}^{-1}\boldsymbol{q}_\zeta = 0$$

where

$$\boldsymbol{S} = \begin{pmatrix} \boldsymbol{r}^{(1)} & \boldsymbol{r}^{(2)} & \boldsymbol{r}^{(3)} \end{pmatrix}, \qquad \boldsymbol{S}^{-1} = \begin{pmatrix} \boldsymbol{l}^{(1)} \\ \boldsymbol{l}^{(2)} \\ \boldsymbol{l}^{(3)} \end{pmatrix}$$

yielding,

$$\boldsymbol{R}_t + \frac{1}{m_1}\boldsymbol{\Lambda}\boldsymbol{R}_\zeta = 0$$

where

$$\boldsymbol{R} = \begin{pmatrix} v \\ u + 2c \\ u - 2c \end{pmatrix}, \qquad \boldsymbol{\Lambda} = \boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S} = \begin{pmatrix} \lambda^{(1)} & & \\ & \lambda^{(2)} & \\ & & \lambda^{(3)} \end{pmatrix}.$$

As in the 1-D scalar case (see Equation (14), Section 3.1), the spatial derivative in Equation (28) is computed using the simple difference formula,

$$\left.\frac{\partial \boldsymbol{F}(\boldsymbol{q})}{\partial \zeta}\right|_i \doteq \frac{(\hat{\boldsymbol{F}}_{i+1/2} - \hat{\boldsymbol{F}}_{i-1/2})}{(\zeta_{i+1/2} - \zeta_{i-1/2})} = \hat{\boldsymbol{F}}_{i+1/2} - \hat{\boldsymbol{F}}_{i-1/2}$$

where $\hat{\boldsymbol{F}}_{i+1/2}$ and $\hat{\boldsymbol{F}}_{i-1/2}$ are high-order approximations to the flux obtained from an adaptive stencil that avoids discontinuous regions of the flow. To compute $\hat{\boldsymbol{F}}_{i+1/2}$, the algorithm outlined in Section 3.1.5 for the 1-D scalar case is generalized. First, undivided difference tables are computed for each component of the flux and the state vector, as in step 1 in Section 3.1.5, i.e.,

$$\begin{aligned}
\mathcal{H}^1_{i-1/2} &= \boldsymbol{F}[\boldsymbol{q}_i] \\
\mathcal{U}^1_{i-1/2} &= \boldsymbol{q}[\zeta_i] \\
\mathcal{H}^k_{i-1/2} &= \frac{1}{k}\boldsymbol{F}[\boldsymbol{q}_i, \ldots, \boldsymbol{q}_{i+k-1}] \\
\mathcal{U}^k_{i-1/2} &= \frac{1}{k}\boldsymbol{q}[\zeta_i, \ldots, \zeta_{i+k-1}], \qquad k = 2, \ldots, r+1.
\end{aligned}$$

The difference tables are then projected onto the eigenspace using the left eigenvectors of $\boldsymbol{A}$,

$$\tilde{\mathcal{H}}_{i-1/2} = (\boldsymbol{S}^{-1})_{i+1/2}\mathcal{H}_{i-1/2}.$$

Only the portion of the difference table that might be utilized in the approximation for $\hat{\boldsymbol{F}}_{i+1/2}$ is projected onto the eigenspace (see Figure 1). For each projected component, p,

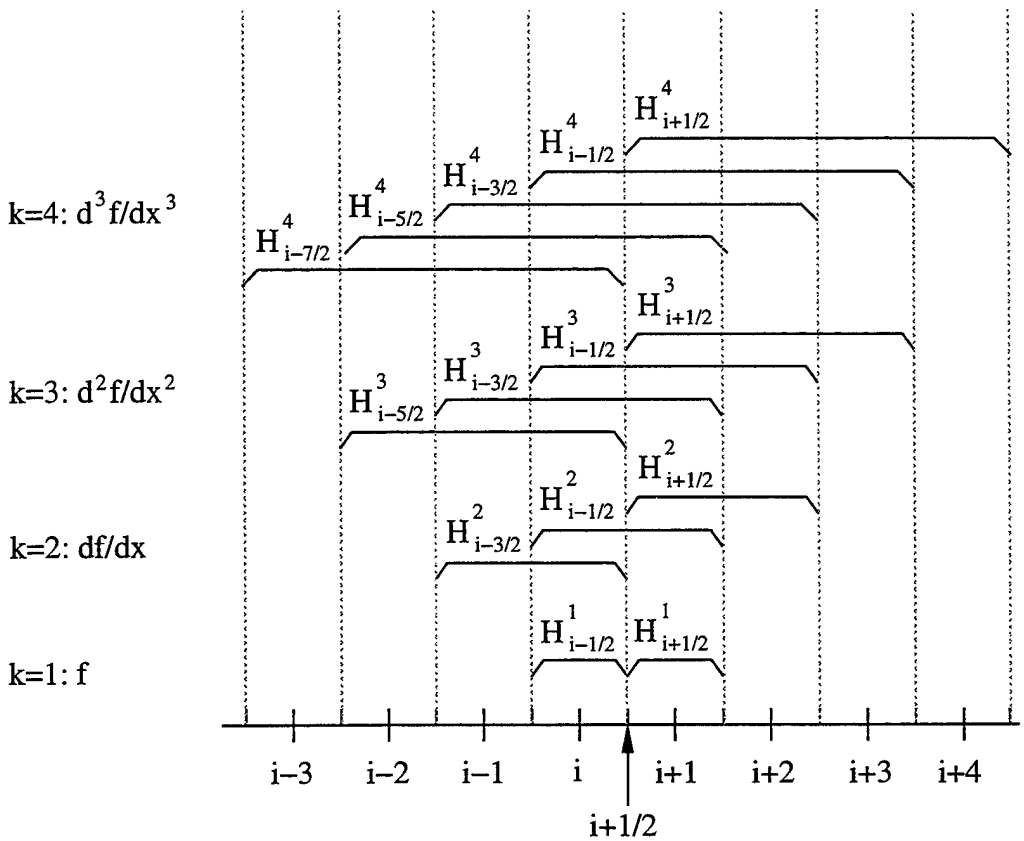

Figure 1: The portion of the difference table that could possibly be utilized in the approximation of $\hat{F}_{i+1/2}$.

an appropriate stencil is selected in the ENO fashion and the interpolating polynomial, $\tilde{F}^{(p)}_{i+1/2}$, is formed, as outlined in steps 2–5 in Section 3.1.5. The interpolating polynomials in the eigenspace are then projected back using the right eigenvectors of $\boldsymbol{A}$, i.e.,

$$\hat{\boldsymbol{F}}_{i+1/2} = \boldsymbol{S}_{i+1/2}\tilde{\boldsymbol{F}}_{i+1/2}.$$

The Roe speeds in this case are the eigenvalues,

$$a^{(p)}_{i+1/2} = \lambda^{(p)}_{i+1/2}, \qquad p = 1, 2, 3$$

and the local Lax–Friedrichs estimate is,

$$\alpha^{(p)}_{i+1/2} = \max_{\boldsymbol{q}_i \le \boldsymbol{q} \le \boldsymbol{q}_{1+1}} \lambda^{(p)}(\boldsymbol{q}), \qquad p = 1, 2, 3.$$

Since the Roe speeds and projection matrices are evaluated at the half-grid points, a suitable average must be computed for u, v, and h. The appropriate averages are the Roe-averaged quantities (Roe 1981) given by,

$$\begin{aligned} u_{i+1/2} &= \frac{u_i\sqrt{h_i} + u_{i+1}\sqrt{h_{i+1}}}{\sqrt{h_i} + \sqrt{h_{i+1}}} \\ v_{i+1/2} &= \frac{v_i\sqrt{h_i} + v_{i+1}\sqrt{h_{i+1}}}{\sqrt{h_i} + \sqrt{h_{i+1}}} \\ h_{i+1/2} &= \frac{1}{2}(h_i + h_{i+1}). \end{aligned}$$

The ENO scheme is easily generalized to multi-dimensions since the approximations to the fluxes $\boldsymbol{F}(\boldsymbol{q})$ and $\boldsymbol{G}(\boldsymbol{q})$ are computed separately. Equation (27) is therefore approximated as

$$\boldsymbol{q}_t \doteq -\frac{1}{m_1}(\hat{\boldsymbol{F}}_{i+1/2,j} - \hat{\boldsymbol{F}}_{i-1/2,j}) - \frac{1}{m_2}(\hat{\boldsymbol{G}}_{i,j+1/2} - \hat{\boldsymbol{G}}_{i,j-1/2}) + \boldsymbol{Q}_{ij}. \tag{30}$$

The computation of $\hat{\boldsymbol{G}}_{i,j+1/2}$ is analogous to that of $\hat{\boldsymbol{F}}_{i+1/2,j}$. For completeness, it is noted that the matrix

$$\boldsymbol{B} = \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{q}} = \begin{pmatrix} v & u & -uv \\ 0 & 2v & -v^2 + F_r^{-2}h \\ 0 & 1 & 0 \end{pmatrix}$$

has eigenvalues and left and right eigenvectors,

$$
\begin{array}{lllllllllllll}
\lambda^{(1)} & = & v & \boldsymbol{l}^{(1)} & = & & \begin{pmatrix} 1 & 0 & -u \end{pmatrix} & \boldsymbol{r}^{(1)} & = & \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}^T \\
\lambda^{(2)} & = & v+c & \boldsymbol{l}^{(2)} & = & -\frac{1}{2c} & \begin{pmatrix} 0 & -1 & v-c \end{pmatrix} & \boldsymbol{r}^{(2)} & = & \begin{pmatrix} u & v+c & 1 \end{pmatrix}^T \\
\lambda^{(3)} & = & v-c & \boldsymbol{l}^{(3)} & = & \frac{1}{2c} & \begin{pmatrix} 0 & -1 & v+c \end{pmatrix} & \boldsymbol{r}^{(3)} & = & \begin{pmatrix} u & v-c & 1 \end{pmatrix}^T.
\end{array}
$$

5 The "Right-hand Side"

This section describes the finite-difference approximations for the non-conservative terms on the right-hand side of Equation (10),

$$\boldsymbol{q}_t + \frac{1}{m_1}[\boldsymbol{F}(\boldsymbol{q})]_\zeta + \frac{1}{m_2}[\boldsymbol{G}(\boldsymbol{q})]_\eta = \boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{P}} + \boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{M}}$$

where

$$\boldsymbol{q} = \begin{pmatrix} uh \\ vh \\ h \end{pmatrix} = \begin{pmatrix} U \\ V \\ h \end{pmatrix}$$

and $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{D}}$, and $\boldsymbol{\mathcal{M}}$ represent the Coriolis, pressure forcing, bottom stress, and grid-metric contributions, respectively. The approximations for $\boldsymbol{\mathcal{C}}$, $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{D}}$, and $\boldsymbol{\mathcal{M}}$ are all straightforward.

5.1 Coriolis Term

The approximation for the Coriolis term is simply,

$$\boldsymbol{\mathcal{C}}_{ij} = \begin{pmatrix} f_0 V_{ij} \\ -f_0 U_{ij} \\ 0 \end{pmatrix}.$$

5.2 Pressure Forcing Term

The pressure forcing term is,

$$\boldsymbol{\mathcal{P}} = \begin{pmatrix} -hP_\zeta/m_1 \\ -hP_\eta/m_2 \\ 0 \end{pmatrix}.$$

The model is currently set up to read from an input file the steady component of the (nondimensional) pressure-gradient forcing,

$$\nabla P^0 = \left(\frac{1}{m_1}\frac{\partial P^0(\zeta,\eta)}{\partial \zeta}, \frac{1}{m_2}\frac{\partial P^0(\zeta,\eta)}{\partial \eta} \right)$$

along with the initial flow fields. If there is a time-dependent component to the pressure gradient,

$$\nabla P' = \left(\frac{1}{m_1}\frac{\partial P'(\zeta,\eta,t)}{\partial \zeta}, \frac{1}{m_2}\frac{\partial P'(\zeta,\eta,t)}{\partial \eta} \right)$$

it should be specified in `subroutine Forcing`. (In the current version, $\nabla P'$ corresponds to a propagating low-pressure anomaly.) The approximation for the pressure term is then simply,

$$\boldsymbol{\mathcal{P}}_{ij} = \begin{pmatrix} -h\,(P^0_\zeta + P'_\zeta)/m_1 \\ -h\,(P^0_\eta + P'_\eta)/m_2 \\ 0 \end{pmatrix}_{ij} .$$

5.3 Bottom Stress Term

The bottom stress term is,

$$\boldsymbol{\mathcal{D}} = \begin{pmatrix} -r|\boldsymbol{U}|U/h^2 \\ -r|\boldsymbol{U}|V/h^2 \\ 0 \end{pmatrix} .$$

where $r = C_D L^*/D^*$ is the nondimensional drag coefficient. The appropriate formula for C_D is, of course, application specific. For the original application to the marine atmospheric boundary layer flow, for example, two formulas for C_D for the air-sea interface were coded. The first is the 10-m neutral drag coefficient given by Large and Pond (1981),

$$10^3 C_D = \begin{cases} 1.14 & u_{10} \leq 10 \text{ m s}^{-1} \\ 0.49 + 0.065 u_{10} & u_{10} > 10 \text{ m s}^{-1}. \end{cases}$$

The second is the drag coefficient obtained from the Coastal Waves 96 field experiment (Edwards and Rogerson, in preparation),

$$10^3 C_D = \begin{cases} 2.43 - 0.261 u_{10} & u_{10} \leq 6.1 \text{ m s}^{-1} \\ 0.44 + 0.065 u_{10} & u_{10} > 6.1 \text{ m s}^{-1}. \end{cases}$$

In both cases, the 10-m winds in the formula, u_{10}, must be related to the model's nondimensional layer-averaged wind speed, $|\boldsymbol{u}|$. This relation is currently specified as,

$$u_{10} = 0.75|\boldsymbol{u}|U^* .$$

Note that, to obtain the nondimensional drag coefficient, r, the length, velocity, and depth scales (L^*, U^*, and D^*) must be specified as well.

A linear bottom stress may also be modeled. In this case,

$$\mathcal{D} = \begin{pmatrix} -\tilde{r}U/h \\ -\tilde{r}V/h \\ 0 \end{pmatrix}$$

where $\tilde{r}$ is a (constant) nondimensional linear drag coefficient provided as model input.

5.4 Grid Metric Term

The grid metric term is,

$$\mathcal{M} = \frac{1}{m_1 m_2} \begin{pmatrix} -\alpha_1 V/h - \alpha_2 U/h \\ \alpha_1 U/h - \alpha_2 V/h \\ -\alpha_2 \end{pmatrix}$$

where

$$\alpha_1 \equiv U m_{1_\eta} - V m_{2_\zeta}, \qquad \alpha_2 \equiv U m_{2_\zeta} + V m_{1_\eta}.$$

The metric derivatives,

$$\frac{\partial m_1}{\partial \eta}, \qquad \frac{\partial m_2}{\partial \zeta},$$

are computed during the initialization phase of the model using simple first-order differences. The approximation for $\mathcal{M}_{ij}$ during the model integration follows directly from the algebraic expression above.

6 Time-stepping Scheme

The numerical solution to

$$q_t = L(q) = -\frac{1}{m_1}[F(q)]_\zeta - \frac{1}{m_2}[G(q)]_\eta + \mathcal{Q} \tag{31}$$

is advanced in time with a TVD Runge-Kutta scheme (see Section 3.1 for the definition of TVD). In the shallow-water model, the user can specify either 2nd-order or 3rd-order TVD Runge-Kutta time-stepping in the form,

$$q^a = q^n + \Delta t L(q^n) \tag{32}$$

$$q^{n+1} = \frac{1}{2}q^n + \frac{1}{2}q^a + \frac{1}{2}\Delta t L(q^a) \tag{33}$$

or

$$q^a = q^n + \Delta t L(q^n) \tag{34}$$

$$q^b = \frac{3}{4}q^n + \frac{1}{4}q^a + \frac{1}{4}\Delta t L(q^a) \tag{35}$$

$$q^{n+1} = \frac{1}{3}q^n + \frac{2}{3}q^b + \frac{2}{3}\Delta t L(q^b) \tag{36}$$

respectively. As mentioned in Section 3.2, the theoretical CFL coefficient for both schemes is 1. In practice, however, the recommended maximal CFL coefficient is 0.6, i.e.,

$$\Delta t \max_q \left(\frac{1}{m_1 \Delta\zeta}|F'(q)| + \frac{1}{m_2 \Delta\eta}|G'(q)| \right) \leq 0.6 \tag{37}$$

when $L(u)$ is approximated with an ENO algorithm (C.-W. Shu, personal communication). The use of lower CFL coefficients (e.g., 0.1 or 0.2) is frequently quoted in the literature as well. For the shallow-water model, I have typically selected CFL coefficients in the range 0.4–0.5 and would categorize the use of CFL coefficients in the range 0.1–0.2 as conservative.

7 Boundary Conditions

The types of boundary conditions that are currently implemented in the model correspond to two basic geometries: a channel-like domain and a doubly-periodic domain. For the channel configuration, the along-channel direction is assumed to be the η direction. In the following discussion, the η direction (or y direction in the rectilinear case) is also referred to as the north–south direction, while the ζ (or x) direction is associated with the east–west direction. The user currently has the following options with regard to boundary conditions:

<u>ζ (or x) direction</u>

- periodic
- east wall/west wall
- east wall/west open

<u>η (or y) direction</u>

- periodic
- open (north and south)

Free-slip no-normal-flow boundary conditions are applied at the walls. Gravity-wave radiation is approximated at the open boundaries.

In general, boundary conditions can be treated in one of two ways; (i) one can appropriately assign values to points "outside" the computational domain and apply the same algorithm used in the interior, or (ii) one can apply a different algorithm at the boundary. The treatment for walled boundaries in a rotating flow follows the second approach, while the others use the first approach. Recall that the r-th order ENO scheme requires an $r+1$ adaptive stencil, so to follow the first approach $r+1$ points outside the domain must be assigned.

In the discussion that follows, consider a computational grid that is discretized into $M \times N$ grid points,

$$\zeta_i, \qquad i = 0, \ldots, M-1$$

$$\eta_j, \qquad j = 0, \ldots, N-1.$$

The western and eastern boundaries are located at ζ_0 and ζ_{M-1}, respectively. The southern

and northern boundaries are located at η_0 and η_{N-1}, respectively.

7.1 Periodic BCs

The implementation of periodic boundary conditions is trivial. In the η direction, for example, we simply set

$$\begin{aligned} \boldsymbol{q}_{-j} &= \boldsymbol{q}_{N-j} \\ \boldsymbol{q}_{N-1+j} &= \boldsymbol{q}_{j-1}, \qquad \text{for } j = 1, \ldots, r+1. \end{aligned}$$

7.2 Radiation BCs

Radiation boundary conditions are easily implemented in the model since the ENO scheme is a characteristic-based scheme. In the eigenspace, the sign of the eigenvalue indicates the direction of wave propagation along the characteristic. When the wave propagation is directed out of the domain, the radiation treatment calls for extrapolation of the Riemann invariant to the grid points "outside" the domain. When the wave propagation is directed into the domain, the value of the Riemann invariant outside the domain is prescribed. In the current model implementation, the prescribed values for the incoming waves are the initial conditions along the boundary.

In the η direction, for example, we obtain the Riemann invariants by projecting $\boldsymbol{q}$ using the left eigenvectors of $\boldsymbol{B} = \partial \boldsymbol{G}/\partial \boldsymbol{q}$, i.e.,

$$\boldsymbol{\mathcal{R}} = \boldsymbol{P}^{-1}\boldsymbol{q}, \qquad \boldsymbol{P}^{-1} = \begin{pmatrix} 1 & 0 & -u \\ 0 & \frac{1}{2c} & -\frac{1}{2c}(v-c) \\ 0 & -\frac{1}{2c} & \frac{1}{2c}(v+c) \end{pmatrix}.$$

The eigenvalues of $\boldsymbol{B}$,

$$(\lambda^{(1)}, \lambda^{(2)}, \lambda^{(3)}) = (v, v+c, v-c)$$

reveal the direction of wave propagation at the northern and southern boundaries and determine how the value of the Riemann invariants outside the domain will be set. At the

southern boundary (at grid point $j = 0$), for example, the algorithm for the p-th Riemann invariant is,

$$\begin{aligned}
&\text{if } \lambda^{(p)} > 0 \text{ then} \\
&\qquad \mathcal{R}^{(p)}(\eta_{-j}, t) = \mathcal{R}^{(p)}(\eta_0, t = 0), \qquad j = 1, \ldots, r+1 \\
&\text{else} \\
&\qquad \mathcal{R}^{(p)}(\eta_{-j}, t) = \mathcal{R}^{(p)}(\eta_0, t), \qquad j = 1, \ldots, r+1.
\end{aligned}$$

The value of the state vector outside the domain is then set by projecting back to physical space via the right eigenvectors of $\boldsymbol{B}$, i.e.,

$$\boldsymbol{q} = \boldsymbol{P}\boldsymbol{\mathcal{R}}, \qquad \boldsymbol{P} = \begin{pmatrix} 1 & u & u \\ 0 & v+c & v-c \\ 0 & 1 & 1 \end{pmatrix}.$$

7.3 Walled BCs for Non-rotating Flows

In the absence of rotation, $u = 0$ implies $h_\zeta = 0$ and $v_\zeta = 0$ (see Equation (7)). Therefore the no-normal-flow condition at the western boundary (grid point $i = 0$), for example, can be satisfied by simply setting,

$$\begin{aligned}
u_0 &= 0, \qquad \text{and} \\
u_{-i} &= -u_i \\
v_{-i} &= v_i \\
h_{-i} &= h_i, \qquad \text{for } i = 1, \ldots, r+1.
\end{aligned}$$

In this scenario, the points outside the domain are typically referred to as *image points* or *ghost points.*

7.4 Walled BCs for Rotating Flows

While the use of ghost points works well in the non-rotating case, an alternative approach is necessary when rotation is present. In rotating flows, $u = 0$ implies

$$f_0 v = \frac{1}{m_1} F_r^{-2} h_\zeta.$$

Even though we could still anti-image u at the wall (e.g., $u_{-i} = -u_i$), we do not know the functional form of h (or v), so we cannot effectively assign values to h and v at the ghost points that would maintain the geostrophic relationship to a high order of accuracy. Extrapolation from the interior to the ghost points was tested (up to 4th-order extrapolation), but a mismatch in the truncation error between the ghost point and the wall point resulted in an error in the flux at the wall that eventually contaminated the numerical solution. For rotating flows, we have not found a satisfactory method to update the flow field at the wall using ghost points. Instead, we have implemented a boundary treatment that locally solves a Riemann problem to update the grid points at the wall using only interior information.

Consider the model equations (see Equation (10)) rewritten as,

$$\boldsymbol{q}_t + \frac{1}{m_1}[\boldsymbol{F}(\boldsymbol{q})]_\zeta = \tilde{\boldsymbol{\mathcal{Q}}}$$

where

$$\tilde{\boldsymbol{\mathcal{Q}}} = -\frac{1}{m_2}[\boldsymbol{G}(\boldsymbol{q})]_\eta + \boldsymbol{\mathcal{C}} + \boldsymbol{\mathcal{P}} + \boldsymbol{\mathcal{D}} + \boldsymbol{\mathcal{M}}.$$

Applying the projection matrix to the system, i.e.,

$$\boldsymbol{S}^{-1}\boldsymbol{q}_t + \frac{1}{m_1}(\boldsymbol{S}^{-1}\boldsymbol{A}\boldsymbol{S})\boldsymbol{S}^{-1}\boldsymbol{q}_\zeta = \boldsymbol{S}^{-1}\tilde{\boldsymbol{\mathcal{Q}}}$$

$$\boldsymbol{A} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}}, \qquad \boldsymbol{S}^{-1} = \begin{pmatrix} \boldsymbol{l}^{(1)} \\ \boldsymbol{l}^{(2)} \\ \boldsymbol{l}^{(3)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & -v \\ \frac{1}{2c} & 0 & -\frac{1}{2c}(u-c) \\ -\frac{1}{2c} & 0 & \frac{1}{2c}(u+c) \end{pmatrix}$$

yields the characteristic form,

$$\boldsymbol{R}_t + \frac{1}{m_1}\boldsymbol{\Lambda}\boldsymbol{R}_\zeta = \boldsymbol{Q}$$

where

$$\boldsymbol{R} = \begin{pmatrix} v \\ u+2c \\ u-2c \end{pmatrix}, \qquad \boldsymbol{\Lambda} = \begin{pmatrix} u & & \\ & u+c & \\ & & u-c \end{pmatrix}$$

and

$$Q = \frac{1}{h}\begin{pmatrix} \tilde{\mathcal{Q}}^{(2)} - v\tilde{\mathcal{Q}}^{(3)} \\ \tilde{\mathcal{Q}}^{(1)} - (u-c)\tilde{\mathcal{Q}}^{(3)} \\ \tilde{\mathcal{Q}}^{(1)} - (u+c)\tilde{\mathcal{Q}}^{(3)} \end{pmatrix}$$

(see Section 4). Since $u = 0$ at a wall, we have

$$\begin{aligned} R_t^{(1)} &= Q^{(1)} \\ R_t^{(2)} + \frac{c}{m_1} R_\zeta^{(2)} &= Q^{(2)} \\ R_t^{(3)} - \frac{c}{m_1} R_\zeta^{(3)} &= Q^{(3)}. \end{aligned}$$

Note that since $R^{(3)} = -R^{(2)}$ when $u = 0$, the flow at the wall can be obtained from the interior flow fields, and no information outside the domain is necessary. For the eastern wall at $\zeta = \zeta_{M-1}$, we solve for $R^{(1)}$ and $R^{(2)}$; for the western wall at $\zeta = \zeta_0$, we solve for $R^{(1)}$ and $R^{(3)}$.

Consider the wall at the eastern boundary, $\zeta = \zeta_{M-1}$, which we now denote by ζ_w. The characteristic quantities at the wall at time t^n, ${}^{(1)}R_w^n$ and ${}^{(2)}R_w^n$, are computed by locating the characteristic that intersects the wall at the required time level, as depicted in Figure 2. (The vector-component index has been moved to the left of the vector variable for notational clarity.) The characteristic quantities at the wall are advanced in time in a manner that is consistent with the time-stepping scheme used in the interior. For example, when the 3rd-order TVD Runge-Kutta is used in the interior (see Section 6), the characteristic quantities at the wall are advanced according to,

$$\begin{aligned} {}^{(1)}R_w^a &= {}^{(1)}R_w^n + \Delta t\, {}^{(1)}Q_w^n \\ {}^{(2)}R_w^a &= {}^{(2)}R_*^n + \Delta t\, {}^{(2)}Q_w^n \end{aligned}$$

$$\begin{aligned} {}^{(1)}R_w^b &= {}^{(1)}R_w^a + \frac{\Delta t}{4}\left({}^{(1)}Q_w^n + {}^{(1)}Q_w^a\right) \end{aligned}$$

$$
\begin{aligned}
{}^{(2)}R_w^b &= {}^{(2)}R_*^a + \frac{\Delta t}{4}\left({}^{(2)}Q_w^n + {}^{(2)}Q_w^a\right)
\end{aligned}
$$

$$
\begin{aligned}
{}^{(1)}R_w^{n+1} &= {}^{(1)}R_w^b + \frac{\Delta t}{6}\left({}^{(1)}Q_w^n + {}^{(1)}Q_w^a + 4{}^{(1)}Q_w^b\right) \\
{}^{(2)}R_w^{n+1} &= {}^{(2)}R_*^b + \frac{\Delta t}{6}\left({}^{(2)}Q_w^n + {}^{(2)}Q_w^a + 4{}^{(2)}Q_w^b\right).
\end{aligned}
$$

(It is easily verified that this form of the Runge-Kutta is equivalent to that defined by Equations (34)–(36).) The computation of ${}^{(2)}R_w$ involves the evaluation of ${}^{(2)}R_* = {}^{(2)}R(\zeta_*)$. Here ζ_* is the interior location of the characteristic that intersects the wall at the next partial time step (see Figure 2) and is given by,

$$
\begin{aligned}
\zeta_*^n &= \zeta_w - \frac{1}{m_1}\Delta t c_w^n \\
\zeta_*^a &= \zeta_w - \frac{1}{m_1}\frac{\Delta t}{4}(c_w^n + c_w^a) \\
\zeta_*^b &= \zeta_w - \frac{1}{m_1}\frac{\Delta t}{6}(c_w^n + c_w^a + 4c_w^b).
\end{aligned}
$$

Four-point Lagrange interpolation is then used to evaluate ${}^{(2)}R_*$ from ${}^{(2)}R_w$, ${}^{(2)}R_{w-1}$, ${}^{(2)}R_{w-2}$, and ${}^{(2)}R_{w-3}$. The new values of $\boldsymbol{R}$ are projected back using the right eigenvectors of $\boldsymbol{A} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{q}}$ to update the physical flow variables $\boldsymbol{q}$.

Since the values at the wall are updated by this alternative method, the flux at the half-grid point adjacent to the wall does not have the same truncation error as fluxes computed at points farther in the interior using the ENO algorithm, and therefore high-order accuracy of the derivative is not guaranteed. To help alleviate this slight mismatch, the first two interior points adjacent to the wall are weakly smoothed after the flow has been advanced to the new time level,

$$
\boldsymbol{q}_{w-i}^{n+1} = \frac{s_1\boldsymbol{q}_{w-i-1}^{n+1} + s_2\boldsymbol{q}_{w-i}^{n+1} + s_3\boldsymbol{q}_{w-i+1}^{n+1}}{s_1 + s_2 + s_3}, \qquad \text{for } i = 1, 2.
$$

For the original application of the model, I typically used the smoothing coefficients,

$$
(s_1, s_2, s_3) = (1, 38, 1).
$$

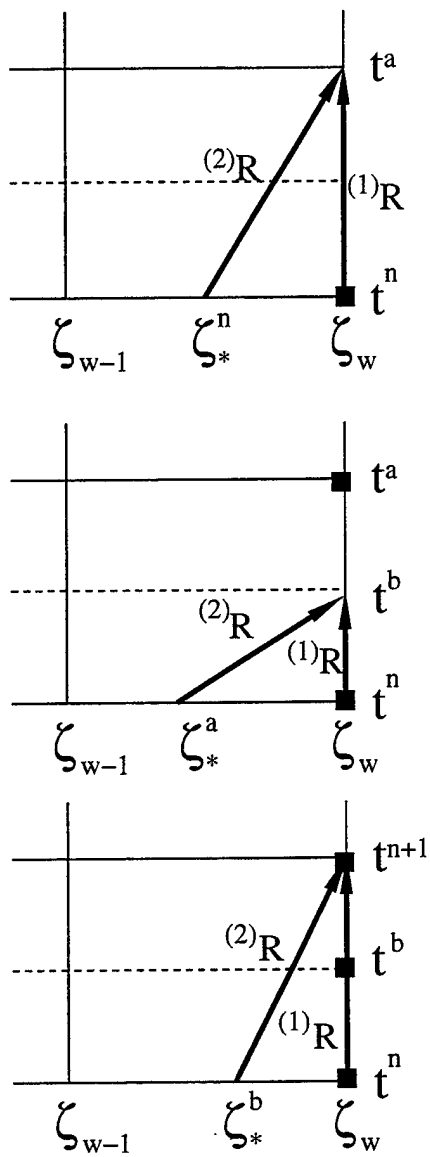

Figure 2: Schematic of the time advancement of the Riemann invariant along the eastern wall, ${}^{(1)}R_w^n$, and into the wall from the interior, ${}^{(2)}R_w^n$, at each step in the 3rd-order Runge-Kutta.

8 Model Preparation and Execution

In this section, instructions are provided on how to prepare and execute model runs. In the discussion that follows, the reader may find it helpful to refer to the source code since it contains comments as well.

The model code is written in Fortran 77 and uses C-directives (i.e., `#define` and `#ifdef` constructs) to organize the source code corresponding to various model options. The source code is divided into four files: two header files, `enores.h` and `gpath.h`, a common file, `enoswcom.f`, and the main program, `enosw.F`.

8.1 Header File `enores.h`

Within the header file `enores.h`, the user must set the parameter specifications for the grid size and the numerical accuracy of the scheme.

- `Q1` specifies the spatial order of accuracy for the ENO scheme. The ENO scheme is intended to be a high-order numerical scheme. For instance, one would not specify `Q1=1` (specifying a 2-point stencil) since this would greatly restrict the stencil selection process which forms the cornerstone of the ENO algorithm. The setting `Q1=3` (4-point stencil) is more typical.

- `Q2` specifies the temporal order of accuracy for the Runge-Kutta scheme. Runge-Kutta schemes for `Q2=2` and `Q2=3` are implemented (see Equations (32)–(33) and Equations (34)–(36)).

- `M` and `N` specify the grid size.

- `IRSIZ=2*M*N` currently specifies the record length for the direct-access unformatted (binary) I/O of one double-precision `MxN` model data field.

8.2 Header File gpath.h

Within the header file gpath.h, the user must provide information regarding the grid metrics. If the grid is rectilinear with constant grid spacing in the x ($= \zeta$) and y ($= \eta$) directions, then Δx and Δy need to be provided by setting the variables dx0 and dy0, respectively, e.g.,

```
dx0 = 0.1
dy0 = 0.1
```

The variable gpath in this case is inactive and is used only in the creation of a log file at the end of the model run (see below).

If a more general orthogonal curvilinear grid is to be used, then the character variable gpath specifies the path to the grid metric data. The grid metric data are expected in a direct-access unformatted (binary) file called swgrid.met, in record 1. The read statement within subroutine Init of enosw.F is,

```
c     * Get grid metrics
      open(unit=10,file=gpath(1:lnblnk(gpath))//'swgrid.met',
     .     form='unformatted',access='direct',recl=2*IRSIZ,status='old')
      read(10,rec=1) ((h1(i,j), h2(i,j),  i=0,M-1),  j=0,N-1)
      close(10)
```

in which the arrays h1(i,j) and h2(i,j) store the value of the grid metrics m_1 and m_2 (see Equations (5)–(6)).

8.3 Common File enoswcom.f

The file enoswcom.f defines the common blocks and contains all of the C-directives controlling the various model options, described below. The user sets the various C-directives by un-commenting the #define statements corresponding to the desired options and commenting out the others.

- *ENO algorithm.* As discussed in Section 3.1, the Hybrid ENO-Roe algorithm is the recommended algorithm. Therefore, the recommended settings are

```
#define HYBRID
#define ROE
```

The Hybrid ENO-LLF algorithm might also be considered,

```
#define HYBRID
#define LLF
```

but in my experience, the ENO-Roe solutions have been superior to those produced using ENO-LLF.

- *Computational grid type.* If the grid is uniform, un-comment the statement,

```
#define UNIGRID
```

When `UNIGRID` is "on", the grid spacing must be specified in the header file `gpath.h` (see above). If the grid is not uniform, it is expected that the grid-metric input file can be found in the location indicated within `gpath.h` (see above).

- *Bottom stress parameterization.* A linear bottom stress is selected by turning on `LSTRESS`. If `LSTRESS` is on, the linear drag coefficient $\tilde{r}$ will be taken from the parameter file `enosw.parms` (see below). At present, two nonlinear drag coefficients are coded. If `BSTRESS` is on, the drag coefficient derived by Large and Pond will be used; if `CWSTRESS` is on, the formula derived from the Coastal Waves 96 data will be used. (See Section 5.)

- *Forcing.* During the initialization phase of the model, the steady component of the forcing field (see Section 5) is read from the input file `enosw_in.dat`, described below. If the C-directive `STDYFORC` is on, the forcing is assumed to be steady (i.e., the steady component is the total field).

If there is a time-dependent component, it should be coded in subroutine `Forcing` of `enosw.F` and the C-directive `STDYFORC` should be turned off. In the original application of the model, time-dependent forcing was used and the computational grid was curvilinear. The time-dependent pressure gradient forcing was defined on the x-y grid and then mapped to the ζ-η grid. Therefore in the current version, if `STDYFORC` is on and `UNIGRID` is off, the program will attempt to read the curvilinear grid points and the (x, y)-to-(ζ, eta) transformation matrix (see Section 8.7) in subroutine `Init` of `enosw.F` with the statements,

```
c     * Get the grid for the HARD-CODED pressure forcing function
      open(unit=10,file=gpath(1:lnblnk(gpath))//'swgrid.pts',
     .     form='unformatted',status='old')
      read(10) ((x(i,j), y(i,j),  i=0,M-1),  j=0,N-1)
      close(10)

c     * Get the (x,y)->(zeta,eta) mapping matrix to map grad(P)
      open(unit=10,file=gpath(1:lnblnk(gpath))//'swgrid.map',
     .     form='unformatted',access='direct',recl=4*IRSIZ,status='old')
      read(10,rec=2) ((roti11(i,j),roti12(i,j),
     .                roti21(i,j),roti22(i,j),i=0,M-1), j=0,N-1)
      close(10)
```

- *Boundary conditions.* A number of boundary conditions have been implemented. Refer to Section 7 for a complete description. For example, to specify a periodic channel in a rotating system, the C-directives should be turned on and off as,

```
c#define XPERIODIC
c#define EastGHOST
c#define WestGHOST
#define EastWALL
#define WestWALL
c#define WestCHAR
#define YPERIODIC
c#define YCHAR
c#define YORLANSKI
```

For a channel in a non-rotating system with radiation conditions at the channel ends, the settings would be,

```
c#define XPERIODIC
#define EastGHOST
#define WestGHOST
c#define EastWALL
c#define WestWALL
c#define WestCHAR
c#define YPERIODIC
#define YCHAR
c#define YORLANSKI
```

Note that the Orlanski radiation treatment is implemented for comparative purposes only. The `YCHAR` option should be used for radiation boundary conditions.

- *Smoothing for* `EastWALL` *or* `WestWALL`.

  If `EastWALL` or `WestWALL` is on, the C-directive `SMOOTHI` should also be on (see Section 7). The smoothing coefficients are provided in the input parameter file `enosw.parms`, described below.

8.4 Initial Condition Data File `enosw_in.dat`

It is assumed that the initial flow fields (i.e., the flow velocities, u and v, and the layer thickness h) and the steady component of the pressure-gradient forcing (i.e., ∇P^0) reside in a direct-access unformatted file called `enosw_in.dat`. The read statement within subroutine `Init` of `enosw.F` is,

```
c     * Get initial u,v,h, and forcing fields
      open(unit=10,file='enosw_in.dat',form='unformatted',
     .     access='direct',recl=5*IRSIZ,status='old')
      read(10) ((u(i,j,0),v(i,j,0),h(i,j,0),pxf0(i,j),pyf0(i,j),
     .                                          i=0,M-1), j=0,N-1)
```

in which the following correspondence is made,

$$\texttt{u(i,j,0)} \leftarrow u(\zeta_i, \eta_j, t=0)$$

$$\texttt{v(i,j,0)} \leftarrow v(\zeta_i, \eta_j, t=0)$$

$$\texttt{h(i,j,0)} \leftarrow h(\zeta_i, \eta_j, t=0)$$

$$\texttt{pxf0(i,j)} \leftarrow \frac{1}{m_1}\frac{\partial P^0(\zeta,\eta)}{\partial \zeta}$$
$$\texttt{pyf0(i,j)} \leftarrow \frac{1}{m_2}\frac{\partial P^0(\zeta,\eta)}{\partial \eta}.$$

After reading the initial flow fields, the variables `u(,,)` and `v(,,)` are reassigned to hold the *flux* variables $U = uh$ and $V = vh$, respectively.

8.5 Model Parameter File `enosw.parms`

A run-time parameter file, `enosw.parms`, is also supplied to the model. An example of this ASCII file is shown below,

```
10.  10.  500.        L* (km), U* (m/s), D* (m)
0.1 1.0               f0=1/Rossby=fL/U,  Fr-2=1/(Froude^2)
0.0                   r_linear (ignored if BSTRESS/CWSTRESS is on)
0.01                  dt
0. 200. 50.           tstart, tfinal, tdump
1. 38. 1.             smoothing coefficients (s1,s2,s3)
```

- Line 1 specifies the length, velocity and depth scales that will be used in the hard-coded formulas that parameterize the nonlinear bottom stress in subroutine `Friction` when either `BSTRESS` or `CWSTRESS` is on (see Section 5).

- Line 2 specifies the inverse Rossby number, $f_0 \equiv fL*/U*$, and scaling inverse-squared Froude number, $F_r^{-2} \equiv g'D^*/U^{*2}$ (see Section 2).

- Line 3 specifies the linear (nondimensional) drag coefficient if the C-directive `LSTRESS` is on. If a nonlinear bulk formula is specified (i.e., C-directive `BSTRESS` or `CWSTRESS` is on), then the velocity-dependent value for the drag coefficient is hard-coded in subroutine `Friction` and the value of the linear drag coefficient in `enosw.parms` is ignored (see Section 5).

- Line 4 specifies the nondimensional time step, Δt (see Section 6).

- Line 5 specifies the time associated with the beginning of the simulation (`tstart`), the time associated with the end of the simulation (`tfinal`), and the time interval at which the model data will be output (`tdump`).

- Line 6 specifies the smoothing coefficients that are required when characteristic-based walled boundary conditions are in effect (C-directive `EastWALL` and/or `WestWALL`) (see Section 7).

8.6 Execution

To conduct model runs,

1. Select/generate a computational grid and edit `gpath.h` appropriately.

2. Set the parameters in `enores.h`.

3. Set the C-directives that control the model options in `enosw.parms`.

4. Generate `enosw_in.dat`, the initial-condition data file.

5. Edit `enosw.parms` as necessary.

6. Make `enosw`.

7. Execute `enosw`. (There are no command-line arguments.)

8.7 Output files

The executable, `enosw`, generates the output files, `u.dat`, `v.dat`, and `h.dat`, containing the data for u, v, and h, respectively. If the C-directive `STDYFORC` is off, two additional output files, `px.dat` and `py.dat`, are created for the time-dependent pressure-gradient forcing field. Each file is a direct-access unformatted file. For example, `u.dat` is opened with the statement,

```
      open(unit=20,file='u.dat',form='unformatted',
     .     access='direct',recl=IRSIZ)
```

The initial data are placed in record 1, and the remaining records correspond to time `tdump`, time 2*`tdump`, time 3*`tdump`, etc., where `tdump` is specified in `enosw.parms`. For example, the u-velocity at time $t = 0$ is output with the statement,

```
write(20,rec=1) ((u(i,j,0)/h(i,j,0), i=0,M-1), j=0,N-1)
```

(Recall that the variable `u(,,)` holds the flux $U = uh$.)

A log file, `enosw.log`, is also created to catalogue the input parameters that were specified for the model run. An example is,

```
(M,N)=(100,100)    (Q1,Q2)=(3,3)    dt=0.01000

(t0,tmax,tdump)=(  0.000,200.000,50.000)

L*(km)= 10.  U*(m/s)= 10.  D*(m)=500.

f0= 0.100  r=0.0000  Fi= 1.000

Grid:uniform
(dx,dy)=(  0.100,  0.100)

Smoothing coefficients:  1. 38.  1.

ROE-EF   HYBRID   EastWALL WestWALL YCHAR    LSTRESS  STDYFORC SMOOTHI
```

Appendix: Curvilinear Grid-Generation Program `swgrid.f`

The shallow-water model is designed to approximate a solution on a user-specified orthogonal curvilinear grid. If the desired grid is rectilinear with uniformly-spaced grid points in the x and y direction, then the user need only specify Δx and Δy in addition to the number of grid points in each direction. If a more general orthogonal grid is to be used, the user must provide the (spatially-varying) grid metrics to the shallow-water model.

One orthogonal curvilinear grid generation program, `swgrid.f`, is included in the model package as an example. In this section, we review the basics of coordinate transformations and provide a brief description of `swgrid.f` and how it interfaces with the shallow-water model.

If (ζ, η) are the coordinates in the orthogonal curvilinear system, then the change in the position vector $\boldsymbol{x} = (x, y) = (X(\zeta, \eta), Y(\zeta, \eta))$ in the Cartesian system can be written as,

$$\delta \boldsymbol{x} = m_1 \delta\zeta \, \hat{\boldsymbol{\zeta}} + m_2 \delta\eta \, \hat{\boldsymbol{\eta}}$$

where m_1 and m_2 are the coordinate metrics given by

$$m_1 = \sqrt{\left(\frac{\partial x}{\partial \zeta}\right)^2 + \left(\frac{\partial y}{\partial \zeta}\right)^2} = \sqrt{X_\zeta^2 + Y_\zeta^2} \tag{38}$$

$$m_2 = \sqrt{\left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2} = \sqrt{X_\eta^2 + Y_\eta^2}. \tag{39}$$

A vector (u, v) on the (x, y)-grid can be transformed to the (ζ, η)-grid via,

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} X_\zeta/(m_1 A) & X_\eta/(m_2 A) \\ Y_\zeta/(m_1 B) & Y_\eta/(m_2 B) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \tag{40}$$

where

$$A = \left[\left(\frac{X_\zeta}{m_1}\right)^2 + \left(\frac{X_\eta}{m_2}\right)^2\right]^{1/2}$$

$$B = \left[\left(\frac{Y_\zeta}{m_1}\right)^2 + \left(\frac{Y_\eta}{m_2}\right)^2\right]^{1/2}.$$

Similarly, a vector $(\tilde{u}, \tilde{v})$ on the (ζ, η)-grid can be transformed to the (x, y)-grid via,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{X_\zeta Y_\eta - X_\eta Y_\zeta} \begin{pmatrix} Y_\eta m_1 A & -X_\eta m_1 B \\ -Y_\zeta m_2 A & X_\zeta m_2 B \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix}. \tag{41}$$

The metrics m_1 and m_2 (Equations (38)–(39)) must be provided to the shallow-water model. The coordinate transformation (40) can be used during the model initialization to map flow fields specified on a rectilinear (e.g., north–south, east–west) coordinate system to the model's curvilinear coordinate system. During the post-processing phase, the inverse mapping (41) can be applied to visualize the model output in the rectilinear coordinate system if so desired.

The grid-generation program `swgrid.f` was originally developed by Wilkin and modified successively by R. Signell, by R. Samelson and by A. Rogerson. In short, the user specifies the desired boundary and provides an initial distribution of grid points along the boundary. The orthogonal curvilinear grid is obtained by iteratively applying a conformal mapping algorithm to the gridded domain. Detailed aspects of the algorithmic approach and the implementation will not be discussed here. Rather, a brief description of how to use this program is provided below.

Several parameters must be set in `swgrid.f`. The first two specify the grid size and are set by the parameter statement

```
parameter (L=100,M=200)
```

located at the head of `program swgrid` and the subroutines `spline`, `cofx`, and `cofy`. The third is the number of iterations to be performed to obtain the grid, set by the data statement

```
data itmax /15/
```

in the first portion of `swgrid`. Following immediately after the data statement for `itmax` is the data statement

```
data kb2/4/
```

This parameter specifies which of the four boundaries will maintain its original distribution of grid points after the mapping.

The user specifies the geometric shape and the initial distribution of grid points for the western, southern, eastern, and northern boundaries within subroutine `z1`, and entry points `z2`, `z3`, and `z4`, respectively. The boundary position data are stored in a single complex-valued vector variable `z(i)=(x(i),y(i))`, which holds the position of the grid point at the northwest corner of the domain in `z(1)` and the remaining data points in successive storage locations, proceeding counter-clockwise around the boundary. In subprograms `z1`, `z2`, `z3`, and `z4`, the boundary data are defined parametrically through the variable `s` which varies from zero to one along each portion of the boundary. In the current setup, the domain is 30 units long in the x direction and 50 units long in the y direction (variables `XL` and `YL` in subroutine `z1`) and is discretized into $L \times M = 60 \times 100$ grid points. The eastern boundary consists of a series of bends; `phi(j)` are the bend angles, measured from due south, `ybend(j)` are the y positions of the bends, `rc(j)` are the radii of curvature for the bend. The formula for the eastern boundary points is obtained after a little bit of trigonometry. The western, northern, and southern boundaries are straight. Grid points along the southern boundary are equally-spaced while those along the western and northern boundaries are clustered non-uniformly. The clustering in the current implementation is achieved by mapping the parametric variable `s` to a piecewise continuous cubic polynomial. Additional details are provided by the comments within the source code.

When the deformation of the boundary is severe, the grid that is generated may not be orthogonal near the edges of the domain. Two additional parameters have been introduced at the beginning of program `swgrid` to clip the grid at the northern and/or southern ends during output,

```
parameter (jclipn=0, jclips=5)
```

In this case, the grid points indexed by $j \leq 5$ are eliminated.

Program `swgrid.f` produces five output files:

- `swgrid.met`, a direct-access unformatted (binary) file containing the grid metrics, m_1 and m_2, on the Sadourney C grids. The metrics on the h, u, and v grids are stored in records 1, 2, and 3, respectively. The shallow-water model accesses the grid metrics on the h grid (record 1).

- `swgrid.map`, a direct-access unformatted file containing the coordinate transformation metrics that appear in Equations (40) and (41). The (x, y)-to-(ζ, η) transformation matrix is stored in record 1. The (ζ, η)-to-(x, y) transformation matrix is stored in record 2.

- `swgrid.pts`, an unformatted file containing the grid points $(x, y) = (X(\zeta, \eta), Y(\zeta, \eta))$.

- `swgrid.bdry`, an ASCII file containing the boundary points.

- `mesh.dat`, an ASCII file to ease the plotting of the grid mesh.

Note that for the direct-access binary files, the record length is the minimum required for double-precision output.

References

Batchelor, G. K., 1967: *An Introduction to Fluid Dynamics*. Cambridge University Press, Cambridge, 615 pp.

Harten, A., 1984: On a class of high resolution total-variation-stable finite-difference schemes. *SIAM J. Numer. Anal.*, **21**, 1–23.

Harten, A., B. Engquist, S. Osher and S. R. Chakravarthy, 1987: Uniformly high order accurate essentially non-oscillatory schemes, III. *J. Comput. Phys.*, **71**, 231–303.

Large, W. G., and S. Pond, 1981: Open ocean momentum flux measurements in moderate to strong winds. *J. Phys. Oceanogr.*, **11**, 324–336.

Lax, P. D., and B. Wendroff, 1960: Systems of conservation laws. *Commun. Pure Appl. Math.*, **13**, 217–237.

LeVeque, R. J., 1990: *Numerical Methods for Conservation Laws*. Lectures in Mathematics, Birkhäuser Verlag, Basel, 214 pp.

Roe, P. L., 1981: Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comput. Phys.*, **43**, 357–372.

Rogerson, A. M., 1999: Transcritical flows in the coastal marine atmospheric boundary layer. *J. Atmos. Sci.*, **56**(8), 2761–2779.

Rogerson, A. M., and E. Meiburg, 1990: A numerical study of the convergence properties of ENO schemes. *J. Sci. Comp.*, **5**(2), 151–167.

Shu, C. -W., 1990: Numerical experiments on the accuracy of ENO and modified ENO schemes. *J. Sci. Comp.*, **5**(2), 127–149.

Shu, C. -W., and S. Osher, 1988: Efficient implementation of essentially non-oscillatory shock-capturing schemes. *J. Comput. Phys.*, **77**(2), 439–471.

Shu, C. -W., and S. Osher, 1989: Efficient implementation of essentially non-oscillatory shock-capturing schemes, II. *J. Comput. Phys.*, **83**(1), 32–78.

DOCUMENT LIBRARY

Distribution List for Technical Report Exchange – July 1998

University of California, San Diego
SIO Library 0175C
9500 Gilman Drive
La Jolla, CA 92093-0175

Hancock Library of Biology & Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Research Library
U.S. Army Corps of Engineers
Waterways Experiment Station
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological Observatory
Columbia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Building 1003 South
1002 Balch Blvd.
Stennis Space Center, MS, 39522-5001

Library
Institute of Ocean Sciences
P.O. Box 6000
Sidney, B.C. V8L 4B2
CANADA

National Oceanographic Library
Southampton Oceanography Centre
European Way
Southampton SO14 3ZH
UK

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

IFREMER
Centre de Brest
Service Documentation - Publications
BP 70 29280 PLOUZANE
FRANCE

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-99-09	2.	3. Recipient's Accession No.
4. Title and Subtitle A Shallow-Water Model for Hydraulically Transcritical Flows			5. Report Date
			6.
7. Author(s) A.M. Rogerson			8. Performing Organization Rept. No. WHOI-99-09
9. Performing Organization Name and Address Woods Hole Oceanographic Institution Woods Hole, Massachusetts 02543			10. Project/Task/Work Unit No.
			11. Contract(C) or Grant(G) No. (C) N00014-93-1-1369 (G)
12. Sponsoring Organization Name and Address Office of Naval Research			13. Type of Report & Period Covered Technical Report
			14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-99-09.

16. Abstract (Limit: 200 words)

This document describes a numerical model that was developed to study two-dimensional, reduced-gravity, shallow-water flows. When the dynamics of these flows is strongly nonlinear, the flow may become hydraulically supercritical and discontinuities in the flow field may arise. The presence of discontinuities in the flow field requires a special numerical treatment in order to maintain both accuracy and stability in the numerically-approximated solution. In this model, a shock-capturing scheme called the Essentially Non-Oscillatory (ENO) scheme is implemented. The ENO scheme is a high-order, adaptive-stencil, finite-difference, characteristic-based scheme for hyperbolic equations that has been applied widely to flows governed by the Euler equations of gas dynamics. The model described in this document was developed for geophysical applications, and therefore includes the effects of rotation (constant Coriolis parameter), forcing (time dependent and/or spatially varying), and bottom drag (linear or nonlinear). The presentation includes the mathematical formulation of the model as well as instructions on how to prepare and execute model runs.

17. Document Analysis a. Descriptors

shallow-water
numerical model
shock-capturing

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Approved for public release; distribution unlimited.	UNCLASSIFIED	59
	20. Security Class (This Page)	22. Price

(See ANSI-Z39.18) *See Instructions on Reverse* OPTIONAL FORM 272 (4-77) (Formerly NTIS-35) Department of Commerce