A finite element approach to open channel flow

Michael J Bagstad
University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/400
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
A FINITE ELEMENT APPROACH
TO OPEN CHANNEL FLOW

by

Michael J. Bagstad

A Thesis
Submitted in partial fulfillment of the requirements
for the degree of

Master of Science
in
Civil and Environmental Engineering

Department of Civil and Environmental Engineering
University of Nevada, Las Vegas
December 1994
The Thesis of Michael J. Bagstad for the degree of Master of Science in Civil and Environmental Engineering is approved.

Chairperson, James A. Cardle, Ph. D.

Examiner Committee Member, Richard Wyman, Ph. D.

Examiner Committee Member, William Culbreth, Ph. D.

Graduate Faculty Representative, Evangelos A. Yfantis, Ph. D.

Dean of the Graduate College, Ronald W. Smith, Ph. D.

University of Nevada, Las Vegas
December 1994
ABSTRACT

A dissipative Galerkin scheme has been developed for the study of the propagation of hydraulic jumps and bores in open channels. It uses the basis function to weight the temporal terms and an asymmetric weighting function to weight the convection terms. The scheme has been improved with routines that both apply the asymmetric weighting function only in the region of the shock and proportion the use of the asymmetric weighting function according to the magnitude of the local spurious oscillations. The introduced variability in the model improved the performance of the model in all examples tested. This scheme was compared to a Galerkin scheme with a symmetric weighting function instead of the basis function weighting the temporal terms and without the improved routines for applying the weighting functions. The scheme with the routines for proportionally applying the spacial weighting functions compared favorably.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>LIST OF SYMBOLS</td>
<td>vi</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>CHAPTER 2 LITERATURE REVIEW</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 3 THEORETICAL DEVELOPMENT</td>
<td>6</td>
</tr>
<tr>
<td>CHAPTER 4 METHODOLOGY</td>
<td>13</td>
</tr>
<tr>
<td>CHAPTER 5 RESULTS OF TEST MODELING</td>
<td>16</td>
</tr>
<tr>
<td>CLASSICAL GALERKIN FORMULATION</td>
<td>16</td>
</tr>
<tr>
<td>FIRST PETROV-GALERKIN SCHEME</td>
<td>17</td>
</tr>
<tr>
<td>Moving Surge</td>
<td>17</td>
</tr>
<tr>
<td>Stationary Jump</td>
<td>25</td>
</tr>
<tr>
<td>Negative Wave</td>
<td>29</td>
</tr>
<tr>
<td>Table 1 Negative Wave Crest Location</td>
<td>31</td>
</tr>
<tr>
<td>SECOND PETROV-GALERKIN SCHEME</td>
<td>32</td>
</tr>
<tr>
<td>CHAPTER 6 CONCLUSION</td>
<td>38</td>
</tr>
<tr>
<td>APPENDIX I BASIC PETROV-GALERKIN SCHEME</td>
<td>40</td>
</tr>
<tr>
<td>APPENDIX II VARIABLE ALPHA ROUTINE</td>
<td>56</td>
</tr>
<tr>
<td>APPENDIX III EPSILON ROUTINE</td>
<td>58</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>60</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1  Classical Galerkin Formulation, Alpha equal to 0.0 and Δt/Δx .............. 16
Figure 2  Moving Surge Alpha Variations, Theta equal to 1.0 at node 14 .............. 18
Figure 3  Moving Surge Alpha Variations, Theta equal to 0.75 at node 14 ......... 19
Figure 4  Moving Surge Alpha Variations, Theta equal to 0.55 at node 14 .......... 19
Figure 5  Moving Surge Theta Variations, Fₙ equal to 1.49 at node 14 .......... 20
Figure 6  Moving Surge Theta Limits at node 14 ............................................ 20
Figure 7  Moving Surge Time Step Variations, Fₙ equal to 1.49 at node 14 ....... 21
Figure 8  Moving Surge Time Step Variations, Fₙ equal to 2.45 at node 14 ....... 22
Figure 9  Moving Surge Theta Variations, Fₙ equal to 2.45 at node 14 .......... 22
Figure 10 Moving Surge Froude Number Variations, e equal to 0.50 at node 14.... 23
Figure 11 Moving Surge Epsilon Variations, Fₙ equal to 1.49 at node 14 .......... 24
Figure 12 Moving Surge Epsilon Variations, Fₙ equal to 2.45 at node 14 .......... 24
Figure 13 Moving Surge Froude Number Variations, e equal to 0.25 at node 14.... 25
Figure 14 Stationary Jump Upstream Epsilon, Fₙ equal to 2.0 ...................... 26
Figure 15 Stationary Jump Downstream Epsilon, Fₙ equal to 2.0 ................. 27
Figure 16 Stationary Jump Theta Variations ................................................. 28
Figure 17 Stationary Jump Time Step Variations ......................................... 28
Figure 18 Stationary Jump Theta Limits ....................................................... 29
Figure 19 Negative Wave Optimum Values .................................................. 30
Figure 20 Negative Wave Minimum Dampening .......................................... 32
Figure 21 Moving Surge First Scheme, Fₙ equal to 1.49 at node 14 ............... 34
Figure 22 Moving Surge Second Scheme, Fₙ equal to 1.49 at node 14 .......... 35
Figure 23 Stationary Jump First Scheme, Fₙ equal to 1.25 ......................... 36
Figure 24 Stationary Jump Second Scheme, Fₙ equal to 1.25 ....................... 37
LIST OF SYMBOLS

\( \alpha \)  Alpha, constant used to control degree of symmetry in the weighting function
\( \hat{\varphi}(\epsilon) \)  Approximating function for the partial differential equation
\( \phi \)  Basis function
\( \beta \)  Beta, constant used to control magnitude of symmetric weighting function
\( \Delta t \)  Time increment
\( \Delta x \)  Distance increment
\( \varepsilon \)  Epsilon, constant used to control application of \( \alpha \)
\( \xi \)  Natural coordinate system -1 to 1 over which \( \phi \) is defined
\( \theta \)  Theta, used to weight between present and future time steps
\( \varepsilon_1 \)  Epsilon used in control of \( \alpha \) upstream
\( \phi_1 \)  Linear basis function slopes from 1 to 0 over the natural coordinate system
\( \alpha_1 \)  Lower limit of \( \alpha \)
\( \varepsilon_2 \)  Epsilon used in control of \( \alpha \) downstream
\( \phi_2 \)  Linear basis function slopes from 0 to 1 over the natural coordinate system
\( \alpha_2 \)  Upper limit of \( \alpha \)
\( A \)  Matrix of coefficients
\( B \)  Matrix containing known values
\( C \)  Matrix containing convection terms
\( c \)  Velocity of a shallow wave
\( C_n \)  Courant Number
F_n  Froude Number

\( g \)  Acceleration constant, gravity

\( j \)  Current node location for computation

\( j+1 \)  Node to the right of the current node location for computation

\( j-1 \)  Node to the left of the current node location for computation

\( M \)  Matrix containing mass terms

\( \text{PDE} \)  Partial differential equation

\( q \)  Flow rate per unit width of channel

\( r \)  Current time step

\( r+1 \)  Next time step

\( \text{R(*)} \)  Residual equal to difference between the function and it's approximation

\( S \)  Matrix containing friction terms

\( s_f \)  Friction slope

\( s_0 \)  Channel bottom slope

\( t \)  Time

\( \text{U} \)  Solution matrix

\( v \)  Velocity of flow

\( \text{V} \)  Volume

\( w_i \)  Weighting function

\( w_{si} \)  Asymmetric weighting function for the spatial terms

\( w_{ti} \)  Symmetric weighting function for the temporal terms

\( x \)  Distance

\( y \)  Depth of flow

\( y^* \)  Depth of flow normalized with the upstream depth, \( y_1 \), or the downstream depth, \( y_2 \)
CHAPTER 1

INTRODUCTION

Open channel flow problems have long been modeled with finite difference techniques. While results have been satisfactory, versatility has been less than satisfactory. Grid mesh spacing is fixed causing models to be either too fine or too gross. Long grid spacing improves computational speed in long uniform reaches but hurts accuracy where channel configuration changes in short distances. When short grid spacing is used, accuracy in channel bends and structures is improved, but overall computation time is lengthened. In addition, the two-dimensional finite difference grids do not fit the complex geometries often faced in modeling situations.

A new family of numerical methods is emerging which addresses these problems. These methods are based on the method of weighted residuals and stem from the Galerkin technique. The Galerkin technique uses the same function as both the basis function and the weighting function. This technique is unstable in the vicinity of a shock or flow discontinuity. The Petrov-Galerkin technique uses a weighting function that is different from the basis function. This thesis studies two variations of the Petrov-Galerkin technique in which the weighting functions applied to the temporal terms are different from the weighting functions applied to the spacial terms. In addition, the spacial weighting functions are asymmetric allowing for "upwinding" of the convection terms. Techniques are utilized to smooth the flow profile and fit the shock into the solution. Numerical dampening is judiciously applied.
in an attempt to eliminate the effects of spurious oscillations while retaining the form of the theoretical solution.

This thesis investigates the amount of numerical dampening needed and the manner in which it should be applied. It was found that very small amounts were sufficient to dampen out spurious oscillations. It was advantageous to apply the asymmetric function in direct proportion to relative magnitude of the oscillation.
CHAPTER 2

LITERATURE REVIEW

Much work has been done in the field of finite difference modeling for open channel varied flow situations since the advent of the digital computer. Within the limits of our ability to define the geometrical boundaries of the problem, we have been able to predict and replicate the flow situations that were only understood intuitively a matter of a few decades ago.

Sod (1978) reviewed several finite difference methods as applied to gas flow and introduced artificial viscosity terms which dampened out spurious oscillations behind the shock of flows with discontinuities. Several techniques for artificial dampening were employed matching the technique best suited to the particular finite difference scheme. The artificial viscosity term was not applied to the mass equation in some schemes to minimize mass diffusion across the discontinuity, and the term was also not applied in smooth regions. The artificial dampening was found to improve the accuracy of the solution when applied. All methods but one, Glimm’s random choice method, showed smearing of the discontinuity across several grid points. Glimm’s random choice method did show some tendency to shift the position of the solution indicating that some sort of smoothing technique could improve the results.

Fennema and Chaudhry (1989) used an implicit finite difference scheme for
modeling two-dimensional unsteady free-surface flows. They were able to model sharp discontinuities while dampening spurious oscillations in the vicinity of a bore. Since discontinuities are not isolated in their scheme, however, they may be smeared over several mesh points.

A limitation to this modeling ability is the difficulty caused by boundary conditions with anything but simple geometric shapes. Recently work has been done in the field of finite element modeling for open channel flow problems which do not share these difficulties to the same degree.

While the finite element methods facilitate the modeling of more complicated geometrical problems, they have been found to be unstable in the vicinity of flow discontinuities. The Galerkin formulation has outstanding computational characteristics but its nondissipative nature leads to spurious oscillations in the vicinity of flow discontinuities. Research has been done into techniques to cause an appropriate amount of dampening in such regions of the flow.

Gray and Lynch (1977) studied the finite element Galerkin method and found that spurious oscillations with approximate length $2\Delta x$, remain in the solution. Several time stepping procedures were studied. At the expense of accuracy of the solution, methods like the stable Lax-Wendroff scheme were found to dampen the $2\Delta x$ wave.

Selmin et al (1985) developed a finite element method for non-linear conservation law equations which couple the accurate spatial discretization provided by the Galerkin method with high-order accurate time-stepping schemes derived by Taylor-series expansion in the time increment. The spatial oscillations noted in the solutions of mixed initial boundary value problems were found to be of tolerably small amplitude, to the credit of the effective matching of the time integration schemes with the spatial discretization method.
Morton (1985) discusses characteristic Galerkin methods as the natural progress from earlier Petrov-Galerkin and Taylor-Galerkin improvements to the simple Galerkin finite element technique. An Euler Characteristic Galerkin algorithm is presented based on the $L^2$ projection and forward Euler time-stepping, piecewise constants as the basic elements and recovery procedures. The use of linear piecewise constants, however, was found to be inadequate in the vicinity of discontinuities. Appropriate recovery algorithms for smoothing spurious oscillations were not found, and a retreat to discontinuous piecewise linear elements was also unfruitful.

Katopodes (1984) developed a dissipative Galerkin scheme with some success. Katopodes utilized a highly selective dissipative interface which applied an asymmetric discontinuous weighting function to dampen out the numerically-generated high-frequency parasitic waves. The weighting function was composed of the basis function plus a lower order function multiplied by the convective velocity. The weighting functions were applied to both the time and spatial derivatives based on analytical techniques. The scheme utilized spatial optimization of dampening and a lumped mass matrix. Spurious oscillations were dampened while the accuracy of the model compared favorably to the theoretical solution.

Cardle (1993) used a modification of the Galerkin formulation in the study of the one dimensional convection diffusion equation. Cardle's scheme applied asymmetric weighting functions to the convective and diffusion terms and a symmetric weighting function to the temporal terms. He showed that the resulting scheme emulated the finite element form of the Lax-Wendroff equations known for their ability to handle discontinuous flow situations in the case of purely convective flows.
CHAPTER 3
THEORETICAL DEVELOPMENT

As described in the literature, the Galerkin finite element method has been used successfully when applied to systems of hyperbolic partial differential equations (PDE). When used to analyze the equations of motion of a liquid with a free surface, additional techniques must be employed in the vicinity of a flow discontinuity to suppress the generation and propagation of spurious oscillations in the solution. The technique used in this study is that used by Cardle (1994), a system of asymmetric weighting functions applied to the spatial components of the flow. This technique is applied to a system of hyperbolic equations, the one dimensional open channel flow equations. The method of applying that technique is further studied and improved.

The equations of continuity and momentum are the familiar one dimensional St. Venant equations for shallow open channel flow:

\[
\frac{\partial y}{\partial t} + \frac{\partial q}{\partial x} = 0 \tag{1a}
\]

\[
\frac{\partial q}{\partial t} + \left(\frac{2q}{y}\right) \frac{\partial q}{\partial x} + \left(gy - \frac{q^2}{2y^2}\right) \frac{\partial y}{\partial x} = g (s_o - s_f) \tag{1b}
\]

Where \(y\) is the depth of flow, \(q\) is the flow rate per unit width of channel, \(s_o\) is the channel bottom slope, \(s_f\) is the friction slope, \(x\) is the distance, \(t\) is time, and \(g\) is the acceleration due to gravity constant.
These can be expressed in vector form as follows:

$$\frac{\partial U}{\partial t} + A \frac{\partial U}{\partial x} = B$$

(2)

Where:

$$U = \begin{bmatrix} y \\ q \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 \\ gy - \frac{q^2}{y^2} & \frac{2q}{y} \end{bmatrix}, \quad \text{and} \quad B = \begin{bmatrix} 0 \\ gy(s_0 - s_f) \end{bmatrix}$$

(3)

The Galerkin technique is a particular form of the method of weighted residuals (MWR) as described by Lapidus and Pinder (1982). The desired function \(v(*)\) is replaced by a finite series approximation \(\hat{v}(*)\) such that:

$$v(*) = \hat{v}(*) = \sum_{j=1}^{N} V_j \phi_j (x)$$

(4)

The basis function \(\phi_j\) is made to be a function of space and the undetermined coefficients \(V_j\) are made to be a function of time. Using this approximation for the flow condition leads to:

$$\hat{q} = \sum Q_j(t) \phi_j(x), \quad \text{and} \quad \hat{y} = \sum Y_j(t) \phi_j(x)$$

so that

$$\hat{U} = \begin{bmatrix} \hat{y} \\ \hat{q} \end{bmatrix}$$

(5)

Because the finite approximation will not match exactly the function, substitution of \(\hat{v}(*)\) into the PDE, \(Lv - f = 0\), results in the equation \(L\hat{v} - \hat{f} = R(*)\). Forcing the residual \(R(*)\) to be orthogonal to a series of orthogonal weight functions \(w_i(*)\) will result in \(N\) equations for the \(N\) unknowns and the basic MWR equation:
\[ \int_T \int_V R(\cdot) w_i(\cdot) \, dV \, dt = 0, \ i = 1, 2, \ldots, N \]  \hspace{1cm} (6)

Where \( T \) is time and \( V \) is volume.

In the classic Galerkin formulation, the weight function and the basis function are the same. As indicated previously, this formulation leads to spurious oscillations in the solution in the vicinity of large discontinuities in the flow such as a surge or bore. In the Petrov-Galerkin method, the weight function and the basis function are different. Two variations of the Petrov-Galerkin method are applied to the open channel flow equations in this study. The variations studied herein represent a departure from the usual Petrov-Galerkin scheme since the weighting functions applied to the temporal and spacial terms are different. The introduction of a separate weight function equation and the ability to weight temporal and spacial terms differently leads to a solution to the instability problem in the vicinity of large discontinuities.

The first variation of the Petrov-Galerkin method modeled in this study retains the basis function as the weighting function for the temporal terms and introduces a different function as the weighting function for the spacial terms. The following MWR equation is produced:

\[ \int_x \phi_i \frac{\partial \tilde{U}}{\partial t} \, dx + \int_x \left( w_s \left( \tilde{A} \frac{\partial \tilde{U}}{\partial x} - \tilde{B} \right) \right) \, dx = 0, \ i = 1, 2, \ldots, N \]  \hspace{1cm} (7)

The second variation introduces a second separate weighting function for the temporal terms different from both the basis function and the weighting function applied to the spacial terms. The following MWR equation is produced:
\[ \int_{x} w_{i} \frac{\partial \hat{U}}{\partial t} \, dx + \int_{x} \left( w_{s_{i}} \left( \hat{A} \frac{\partial \hat{U}}{\partial x} - \hat{B} \right) \right) \, dx = 0, \ i = 1, 2, \ldots, N \] (8)

Linear basis functions are used over each element. These are of the form:

\[ \phi_{i}^{e} = \frac{1}{2} (1 - \xi) \quad \text{and} \quad \phi_{2}^{e} = \frac{1}{2} (1 + \xi) \] (9)

where \( \xi \) is the natural coordinate \(-1 \leq \xi \leq +1 \), so that

\[ \phi_{i}^{e} = \begin{pmatrix} \phi_{i}^{e} & 0 \\ 0 & \phi_{2}^{e} \end{pmatrix} \quad \text{for} \quad i = 1 \text{ and } 2. \] (10)

With the basic MWR equations defined for study, the various weight functions need to be defined. The introduction of separate weight functions for the spacial and temporal terms leads to a solution to the spurious oscillation problem inherent with the classical Galerkin formulation of the equations of motion in the vicinity of a large discontinuity. One solution to this problem is to give more weight to the convection terms on the upstream side. This can be done as suggested by Lapidus and Pinder with the use of an asymmetric weighting term. They reference Huyakorn:

\[ w_{1}(\xi) = \phi_{1}^{e} - 3\alpha \phi_{1}^{e} \phi_{2}^{e} \] (11)

\[ w_{2}(\xi) = \phi_{2}^{e} + 3\alpha \phi_{1}^{e} \phi_{2}^{e} \]

The variable \( \alpha \), controls the upstream weighting. As can be seen, when \( \alpha \) equals zero, the weighting functions reduce to the basis function. Writing this in matrix form and introducing an additional element, the matrix \( A \) from equation (3), as
suggested by Cardle, yields two weighting functions for the spacial terms. The following weighting functions are used in both Petrov-Galerkin model variations analyzed in this study:

\[
\begin{align*}
 w_{s1} &= \begin{bmatrix} \phi^e_1 & 0 \\ 0 & \phi^e_2 \end{bmatrix} - 3\alpha A \begin{bmatrix} \phi^e_1 \phi^e_2 & 0 \\ 0 & \phi^e_1 \phi^e_2 \end{bmatrix} \\
 w_{s2} &= \begin{bmatrix} \phi^e_1 & 0 \\ 0 & \phi^e_2 \end{bmatrix} + 3\alpha A \begin{bmatrix} \phi^e_1 \phi^e_2 & 0 \\ 0 & \phi^e_1 \phi^e_2 \end{bmatrix}
\end{align*}
\]  

(12)

For the second Petrov-Galerkin model variation, a separate weighting function is used for the temporal terms which is different from both the basis function and the weighting function applied to the spacial terms:

\[
\begin{align*}
 w_{t1} &= \begin{bmatrix} \phi^e_1 & 0 \\ 0 & \phi^e_2 \end{bmatrix} - 10\beta A 2\xi \begin{bmatrix} \phi^e_1 \phi^e_2 & 0 \\ 0 & \phi^e_1 \phi^e_2 \end{bmatrix} \\
 w_{t2} &= \begin{bmatrix} \phi^e_1 & 0 \\ 0 & \phi^e_2 \end{bmatrix} + 10\beta A 2\xi \begin{bmatrix} \phi^e_1 \phi^e_2 & 0 \\ 0 & \phi^e_1 \phi^e_2 \end{bmatrix}
\end{align*}
\]  

(13)

Where \(\beta\) is a constant that controls the magnitude of the symmetric temporal weighting term. Over each element equation (7) becomes:

\[
\left[ \int_{-1}^{+1} \Phi^e_i \Phi^e_j \frac{dx}{d\xi} d\xi \right] \frac{dU^e_j}{dt} + \left[ \int_{-1}^{+1} w^e_{s1} A_j \frac{d\phi^e_j}{d\xi} d\xi \right] U^e_j \\
- \int_{-1}^{+1} w^e_{s1} B^e \frac{dx}{d\xi} d\xi = 0
\]

(14)
And, over each element equation (8) becomes:

\[
\left[ \int_{-1}^{+1} w_i^e \frac{d x}{d \xi} d \xi \right] \frac{d U_j^e}{d t} + \left[ \int_{-1}^{+1} w_i^e A_j \frac{d \phi_j^e}{d \xi} d \xi \right] U_j^e
\]

\[
- \int_{-1}^{+1} w_i^e B_i \frac{d x}{d \xi} d \xi = 0
\]

Equations (14) and (15) can be written as:

\[
M_{ij}^e \frac{d U_j^e}{d t} + C_{ij} U_j^e + S_j^e = 0
\]

Where:

\[
M_{ij}^e = \frac{1}{2} \int_{-1}^{+1} \phi_i^e \phi_j^e d \xi, \quad \text{or} \quad M_{ij}^e = \frac{1}{2} \int_{-1}^{+1} w_i^e \phi_j^e d \xi,
\]

\[
C_{ij} = \frac{1}{\Delta x} \int_{-1}^{+1} w_i^e A_j \frac{d \phi_j^e}{d \xi} d \xi, \quad \text{and} \quad S_{ij} = \int_{-1}^{+1} w_i^e B_i d \xi
\]

for i and j equal to 1 and 2. Assembling equations and grouping terms at the same time level yields the following equation for the interior nodes:
\[ M_{21}^s (U_{j+1}^r - U_{j-1}^r) + (M_{22}^s + M_{11}^{s+1})(U_j^{r+1} - U_j^r) + M_{12}^{s+1} (U_{j+1}^r - U_{j+1}^r) \]

\[ + \theta \Delta t \left[ C_{21}^s U_{j-1}^{r+1} + (C_{22}^s + C_{11}^{s+1}) U_j^{r+1} + C_{12}^{s+1} U_{j+1}^{r+1} \right] \]

\[ + (1 - \theta) \Delta t \left[ C_{21}^s U_{j-1}^r + (C_{22}^s + C_{11}^{s+1}) U_j^r + C_{12}^{s+1} U_{j+1}^r \right] \]

\[ + \theta \Delta t \left[ S_{21}^s + 2 (S_{22}^s + S_{11}^{s+1}) + S_{12}^{s+1} \right]^{r+1} \]

\[ + (1 - \theta) \Delta t \left[ S_{21}^s + 2 (S_{22}^s + S_{11}^{s+1}) + S_{12}^{s+1} \right]^r = 0 \]

Where \(e\) and \(r\) denote the time step, \(j-1\) denotes the node to the left, and \(j+1\) denotes the node to the right.

The variable \(\theta\), is used to give weight alternately to the spatial terms of the present and future time steps. When \(\theta\) is equal to 1, the equation operates similar to an implicit finite difference scheme. When \(\theta\) is equal to 0, terms at the new time step are not used and the equation operates like an explicit finite difference scheme. When \(\theta\) equals 1/2, equal weight is given to both present and future time steps.

Stability for \(\theta\) less than 1/2 dictated the time step be equal to the distance step divided by the square root of three times the absolute value of the velocity plus the speed of a shallow wave. For values of \(\theta\) greater than or equal to 1/2, there is no restriction on the time step.
CHAPTER 4

METHODOLOGY

Solution of equation (18) requires evaluation of the integrals in the coefficients of the C, M and S matrices, representing convection, mass and source, in equations (16) and (17). The integrals are evaluated numerically with Gauss quadrature which has the following general form:

\[ \int_{-1}^{+1} f(x) \, dx = \sum H_i f(x_i) \]  

(19)

The 2N-4 equations with 2N unknowns are coupled with two upstream and two downstream boundary conditions. The resulting system of nonlinear equations is solved iteratively with the Newton-Rhapson method. The Jacobian is approximated with a forward difference method (FDJAC) developed by Dennis and Schnabel (1983). The estimated Jacobian is then solved as a system of linear equations with an IMSL, Inc. routine, LSLRG. The solution matrix generated by LSLRG is a set of corrections that are then added to the equation variables. The corrections are compared to some reference values, and when the corrections no longer exceed the reference values, the solution at the new time step is considered complete.

As indicated previously, \( \alpha \) controls the amount of upstream weighting. It introduces dampening into the equations for control of spurious oscillations. As shown by Cardle, when the value of \( \alpha \) is set equal to \( \Delta t/\Delta x \) (where \( \Delta t \) is the time step
and Δx equals the grid distance), the numerical technique shows characteristics similar to the Lax-Wendroff finite difference scheme. For the analyses studied here, this value is used for α as its maximum value.

In addition to varying the ratio of weighting between upstream and downstream, the distance both upstream and downstream at which weighting is to be applied can also be varied. This is accomplished with the regulation of the value of α with an additional variable epsilon, ε. The variable ε is input by the user and is used by the program to determine what value of α to use. ε is compared to the relative difference in depth at the adjacent nodes. The larger the relative difference in depth at the adjacent nodes, the larger the value of α.

ε is an input to the model as are the lower and upper limiting values of α. Two model variations were created; one that toggles between the lower and upper α values, and one that interpolates an α value based on the current value for the change in water surface slope at a node and its position between zero and the value for ε.

For the moving surge model, ε is input as a single value. The change in depth between adjacent nodes, Δy, is compared to the difference in depth upstream and downstream of the surge according to the following equation:

\[ Δy = \frac{|y_{i+1} - y_i|}{(y_{ds} - y_{us})} \] (20)

If this change is greater than some user specified limit, ε, then α is set at its maximum value. If Δy is less than ε, then α is set by interpolation between its upper and lower limits according to the following equation:

\[ α = α_1 + (α_2 - α_1) \frac{Δy}{ε} \] (21)

For the stationary jump, two separate values of ε were used, ε1 for the region upstream of the jump, and ε2 for the region downstream of the jump. It was found to
be advantageous to introduce the dampening effects of $\alpha$ in a much more rapid fashion in the supercritical region of flow upstream of the jump.

The depth variation comparison was also refined for the stationary jump model. The change in depth for a node was redefined as the cumulative change in depth at the node when compared to both the upstream and downstream nodes as shown in the following equation:

$$\Delta y = \left( |y_{i+1} - y_i| + |y_i - y_{i-1}| \right) / (y_{ds} - y_{us})$$  \hfill (22)

It was also found to be advantageous to increase the overall magnitude of $\alpha$ by a factor of 5. The modification to equation (21) is shown in the following equation:

$$\alpha = \alpha_1 + 5 (\alpha_2 - \alpha_1) \left( \Delta y / \varepsilon_{1,2} \right)$$  \hfill (23)
CHAPTER 5

RESULTS OF TEST MODELING

CLASSICAL GALERKIN SCHEME

The classical Galerkin scheme uses the basis function as the weighting function. While it has exhibited accuracy in predicting the solutions to the unsteady one dimensional equations of motion, it typically allows spurious oscillations to contaminate the solution in the vicinity of a flow discontinuity. This is exhibited in the solution set of a standing jump with an upstream Froude Number, $F_n$, equal to 2.0 (Figure 1).

**Figure 1** Classical Galerkin Formulation, Alpha equal to 0.0 and $\Delta t/\Delta x$
The variation in which \( \alpha \) is set equal to zero reduces the weighting function to the basis function essentially removing any dampening from the routine. The other variation shown sets \( \alpha \) equal to an optimal value, \( \Delta t/\Delta x \), introducing an appropriate amount of dampening. The variation without dampening shows extreme contamination of the solution after 46 time steps while the dampened variation closely approximates the analytic solution after as many time steps.

**FIRST PETROV-GALERKIN SCHEME**

The first Petrov-Galerkin scheme studied is defined in equation (7). This uses the basis function as the weighting function for the temporal terms and the asymmetric weighting function (12) for the spatial terms. Variations in upstream and downstream weighting are investigated.

**Moving Surge**

A moving surge was analyzed with the following properties: upstream \( F_n \) of 1.49, a Courant Number \( (C_n) \) of 0.577, an upstream depth of 6.0 ft., a downstream depth of 10.0 ft., and a distance interval \( \Delta x \) of 500 ft. The Courant Number set the relationship between the time and distance steps according to equation (24) where \( v \) is the velocity of flow and \( c \) is the velocity of a shallow wave.

\[
C_n = \frac{|v| + c}{|v| + c} \frac{\Delta t}{\Delta x}
\]  

(24)

A node was analyzed in the mid point of the channel reach over time as the surge passed. The model was tested with two variations in the use of the variable \( \alpha \). In each case the maximum \( \alpha \) was 0.3 and it was used or not used according to a limiting value of \( \varepsilon \) equal to 0.5. One test model, the fixed model, either set \( \alpha \) equal to
0.0 or the maximum value of 0.3. The other test model, the variable model, varied $\alpha$ between 0.0 and 0.3 as $\epsilon$ varied between 0.0 and its limiting value of 0.5.

Both fixed and variable $\alpha$ models were analyzed for three different values of $\theta$, 0.55, 0.75 and 1.0. The results are shown in Figures 2, 3, and 4 which show normalized depth plotted over time at a midway point in the channel. In all cases, the variable $\alpha$ model showed less initial variation from theoretical and more tendency to dampen out spurious oscillations over time.

Higher values of $\theta$ were more effective at dampening out spurious oscillations, and with the fixed $\alpha$ model, lower values of $\theta$ caused the growth of spurious oscillations in the downstream region. Figure 5 shows the three $\theta$ plots for the variable $\alpha$ model. It is clear to see in this plot that the higher values of $\theta$ produce the best results. The analysis was extended to even lower values of $\theta$. Figure 6 shows the variable $\alpha$ model with values of $\theta$ equal to 0.5 and 0.0. The variable $\alpha$ model remains stable at a $\theta$ of 0.5, but is unstable in the vicinity of the surge at a $\theta$ of 0.0.

**Figure 2** Moving Surge Alpha Variations, Theta equal to 1.0 at node 14.
Moving Surge Alpha Variations
Fixed vs. Variable Alpha at Theta = 0.75

Figure 3 Moving Surge Alpha Variations, Theta equal to 0.75 at node 14.

Moving Surge Alpha Variations
Fixed vs. Variable Alpha at Theta = 0.55

Figure 4 Moving Surge Alpha Variations, Theta equal to 0.55 at node 14.
Moving Surge Theta Variations
Variable Alpha at Theta = 0.55, 0.75, and 1.0

Figure 5 Moving Surge Theta Variations, \( F_n \) equal to 1.49 at node 14.

Moving Surge Theta Limits
Variable Alpha at Theta = 0.0 and 0.5

Figure 6 Moving Surge Theta Limits at node 14.
The effect of the size of the time step was analyzed by setting $C_n$ equal to 0.577, 1.0 and 2.0 for the variable $\alpha$ model with a $\theta$ of 0.75, a $F_n$ of 1.49, an $\alpha$ of $\Delta t/\Delta x$, and an $\epsilon$ of 0.5. The larger time step corresponding to the larger $C_n$ resulted in more spurious oscillations in the region upstream of the surge relative to the spurious oscillations in the region downstream of the surge (Figure 7). Similarly results, although not as pronounced, are noticed at higher $F_n$ ($F_n = 2.45$) as shown in Figure 8.

The effect of changing the upstream $F_n$ was analyzed. A moving surge was analyzed with the following properties: an upstream $F_n$ of 2.45, a $C_n$ of 0.577, an $\alpha$ of $\Delta t/\Delta x$, an $\epsilon$ of 0.5, an upstream depth of 6.0 ft., a downstream depth of 18.0 ft., and a distance interval $\Delta x$ of 500 ft. When $\theta$ was varied between 1.0 and 0.55, the same characteristics as previously noted for the lower $F_n$ were noted. The program was consistently more accurate at the higher values of $\theta$ as shown in Figure 9.
Moving Surge Time Step Variations

Figure 8 Moving Surge Time Step Variations, $F_n$ equal to 2.45 at node 14.

Moving Surge Theta Variations

Figure 9 Moving Surge Theta Variations, $F_n$ equal to 2.45 at node 14.
For the flow characteristics studied in this example, however, the technique exhibited greater instability at the higher $F_n$. This is particularly true in the supercritical region upstream of the surge as shown in a normalized plot shown in Figure 10. The solutions for a midrange value of $\theta$ equal to 0.75 were plotted for both values of $F_n$ studied. At the higher value of $F_n$, the relative variation from theoretical was greater while for the lower value of $F_n$, the relative variation from theoretical was not as great.

The effect of varying $\epsilon$ was studied. $\epsilon$ was varied between 0.50 and 0.25 for both values of $F_n$ analyzed. The smaller value of $\epsilon$ was found to have a slightly more salutary effect on spurious oscillations (Figures 11 and 12). This was expected as $\epsilon$ is the numerical models sensitivity check against oscillations in the flow. The lower $\epsilon$ value will cause dampening at smaller variations in the solution.

![Moving Surge Froude Number Variations](image)

**Figure 10** Moving Surge Froude Number Variations, $\epsilon$ equal to 0.50 at node 14.
Moving Surge Epsilon Variations

\( \epsilon = 0.5 \) and \( \epsilon = 0.25 \) with \( F_n = 1.49 \)

[Graph showing time step vs. \( y/y_1 \) for \( \epsilon = 0.50 \) and \( \epsilon = 0.25 \).]

**Figure 11** Moving Surge Epsilon Variations, \( F_n \) equal to 1.49 at node 14.

Moving Surge Epsilon Variations

\( \epsilon = 0.5 \) and \( \epsilon = 0.25 \) with \( F_n = 2.45 \)

[Graph showing time step vs. \( y/y_1 \) for \( \epsilon = 0.50 \) and \( \epsilon = 0.25 \).]

**Figure 12** Moving Surge Epsilon Variations, \( F_n \) equal to 2.45 at node 14.
The lower $\varepsilon$ value also caused an increase number of iterations. This was particularly true of the lower $F_n$ value where the number of iterations doubled. As in the study case with the higher $\varepsilon$ value (Figure 10), the higher $F_n$ value showed more instability in the study case with the lower $\varepsilon$ value (Figure 13). And again, this was particularly true in the region upstream of the discontinuity.

![Moving Surge Froude Number Variations](image)

**Figure 13** Moving Surge Froude Number Variations, $\varepsilon$ equal to 0.25 at node 14.

**Stationary Jump**

A stationary jump tested had the following characteristics: an upstream depth of 6.0 feet, a downstream depth of 14.4 feet, a constant flow of 166.8 cubic feet per second (cfs) per foot width of channel, and an upstream $F_n$ of 2.0.

For the stationary jump model an additional variable was added. $\varepsilon$ was expanded to include two values, one for upstream and one for downstream of the surge. It had been noticed that varying $\varepsilon$ had different effects for the regions upstream
and downstream of the surge. In the region upstream of the surge an optimum value for $\epsilon$ was found to be 0.01 ($\epsilon$ is denoted as $\epsilon_1$ or epsilon1 for the region upstream of the surge, nodes 0 to 14, see Figure 14). Decreasing its value further did not further enhance the program's salutary effect. The plot is normalized with the upstream depth upstream of the jump ($y^* = y/y_1$) and with the downstream depth downstream of the jump ($y^* = y/y_n$) to accentuate the differences noted.

**Figure 14** Stationary Jump Upstream Epsilon, $F_n$ equal to 2.0.

In the region downstream of the jump, a higher range for $\epsilon$ was found to produce optimal results. The optimum value was found to be 0.75 ($\epsilon$ is denoted as $\epsilon_2$ or epsilon2 for the region downstream of the surge, nodes 15 to 35, see Figure 15). Increasing its value further reduced the salutary effects of the program on the spurious oscillations.
Figure 15 Stationary Jump Downstream Epsilon, $F_n$ equal to 2.0.

With the optimum values for $\epsilon$ for both upstream and downstream regions of the surge, $\theta$ was varied as with the moving surge setting it equal to 1.0, 0.75, and 0.55. All values produce similar results in the region upstream of the surge (Figure 16). Values for $\theta$ of 1.0 and 0.75 produce similar results downstream of the surge while a value of 0.55 was less effective in dampening out spurious oscillations.

As with the moving surge, the effect of the size of the time step was analyzed by varying $C_n$ from a value of 0.577 to a value of 1.0. The same results were noted with the stationary surge that were noted for the moving surge. The larger time step corresponding to the larger $C_n$ resulted in more spurious oscillations in the region just upstream of the surge relative to the region downstream of the surge (Figure 17).
Stationary Jump Theta Variations

Theta = 0.55, 0.75, and 1.0

Figure 16 Stationary Jump Theta Variations.

Stationary Jump Time Step Variations

Cn = 0.57 and 1.0

Figure 17 Stationary Jump Time Step Variations.
In the final test of the stationary surge, \( \theta \) was set equal to 0.0 with a corresponding value for \( C_n \) to ensure stability. The results were compared to the results when \( \theta \) is equal to 1.0 (Figure 18). Spurious oscillations are much greater when \( \theta \) equals 0.0 for regions upstream and immediately downstream of the surge. The spurious oscillations die out sooner when \( \theta \) equals 0.0 in the region that is further downstream of the surge.

![Stationary Jump Theta Limits](image)

**Figure 18** Stationary Jump Theta Limits.

**Negative Wave**

The propagation of a negative wave was studied. The case simulated was a moving stream with a falling gate or a landslide causing a complete blockage of flow. The flow characteristics prior to blockage consisted of an initial depth of flow, \( y_0 \), of 14.4 feet, a unit width flow rate of 166.8 cfs/ft, and an initial velocity, \( v_0 \), of 11.6 fps. The program parameters set were similar to those found to give optimal results for the stationary surge, an \( \alpha \) of \( 5\Delta t/\Delta x \), an \( \theta \) of 0.75, an \( \epsilon_1 \) of 0.01, and an \( \epsilon_2 \) of 0.75. The
The depth of flow immediately downstream from the landslide, \( y_{x=0} \), was calculated to be 7.7 ft with the aid of the following equation:

\[
y_{x=0} = \frac{1}{g} \left( \sqrt{g y_0} - \frac{1}{2} v_0 \right) \tag{25}
\]

The initial simulation (shown in Figure 19) gave less than satisfactory results. The negative surge is shown at four different times after beginning the simulation period. The location of the numerical wave crest is estimated from the graph by visually locating the point at which the wave front becomes asymptotic with the downstream water surface level (Table 1).

**Negative Wave Optimum Values**

![Figure 19 Negative Wave Optimum Values](image)

Additionally, for each elapsed time of simulation, the wave crest location was calculated from its theoretical velocity, \( v_w \), which is calculated from the initial depth of flow and the initial velocity with the following equation:
\[ v_o + c_o = v_o + \sqrt{gY_o} \] (26)

The numerical wave crest is shown to travel further than what would be theoretically predicted. The location of the wave crest is reported as the nearest node. A difference of two nodes, therefore, would mean that the model prediction and the theoretical calculation for the location of the wave crest differ by approximately 1,000 feet. The calculations are summarized in the following table:

Table 1
Negative Wave Crest Location

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Measured from Graph</th>
<th>Theoretical Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>509.2</td>
<td>55</td>
<td>34</td>
</tr>
<tr>
<td>1027.0</td>
<td>90</td>
<td>68</td>
</tr>
<tr>
<td>1717.5</td>
<td>140</td>
<td>114</td>
</tr>
<tr>
<td>2407.9</td>
<td>190</td>
<td>160</td>
</tr>
</tbody>
</table>

It was hypothesized that the strong dampening mechanisms needed to minimize spurious oscillations in the vicinity of a surge or jump as studied in the previous sections may not be required when approximating the more gentle curvature of a negative wave and that the dampening mechanism could be smearing an otherwise clean solution at the extremities of the discontinuity. Three variations were
attempted to minimize any smearing of the solution due to numerical dampening. In the first variation both $\varepsilon_1$ and $\varepsilon_2$ were set equal to 0.95 so that dampening would be delayed both upstream and downstream of the wave. In the second variation $\alpha$ was reduced by a magnitude of 5. In the third variation $\alpha$ was set equal to 0 to essentially eliminate dampening. None of the variations improved the accuracy of the numerical solution. The last variation is shown in Figure 20. It can be seen that not only is the accuracy not improved, but dampening has been removed to the point of instability of the solution. No explanation can be given for the apparent inaccuracy of the numerical routine to track the crest of the negative wave.

**Figure 20** Negative Wave Minimum Dampening.

**SECOND PETROV-GALERKIN SCHEME**

The second Petrov-Galerkin scheme defined in equation (8) was also studied. This scheme uses the asymmetric weighting function (12) for the spatial terms and the symmetric weighting function (13) for the temporal terms.
The computational accuracy of the first scheme was compared to the second scheme. Both schemes use the same formulation of the Galerkin MWR equations and asymmetric weighting functions. One of the major differences between the two methods is the weighting function used on the temporal terms. The first scheme uses the basis function to weight the temporal terms and the second scheme uses a symmetric modification of the spatial asymmetric weighting function to weight the temporal terms. The other major difference between the two schemes is the manner in which dampening is applied to the routine in the vicinity of large shock disturbances. Unlike the second scheme, the first scheme has the capability of regulating the relative magnitude of upstream versus downstream weighting.

Both schemes set $\alpha$ equal to $\Delta t/\Delta x$. The first scheme varies the magnitude of $\alpha$ between 0 and $\Delta t/\Delta x$ depending on the proximity to the shock and the magnitude of the variation in the depth of flow between adjacent nodes. Additionally, the first scheme applies $\alpha$ differently on the upstream side of the shock from the downstream side by use of the variables $e_1$ and $e_2$. The second scheme applies $\alpha$ uniformly equal to $\Delta t/\Delta x$ in the same manner on both sides of the shock. The first scheme, then, can be considered an attempt to control dampening in a more efficient manner than the second scheme.

Both schemes were used to analyze two similar shock situations, one a moving surge and one a stationary jump. The moving surge had an upstream $F_n$ of 1.49 and a downstream depth to upstream depth ratio of 1.67. The surge occurred in a wide rectangular smooth flat bottom channel. Simulation results were recorded for a point midway in the channel. The period of simulation allowed the surge to pass the midpoint.

Figures 21 and 22 show the recorded depth as a ratio of depth to upstream depth of flow at the channel midpoint over the time of simulation. Both schemes show considerable agitation in the upstream or supercritical region of flow. The
second scheme in addition shows considerable agitation in the downstream or subcritical region of flow. The first scheme on the other hand exhibits agitation in the downstream region of a greatly subdued nature. The second scheme shows an overall variation in the region of the surge from 0.85 to 1.77 while the first scheme shows an overall variation from 0.91 to 1.68. The second scheme felt reverberations from the surge over a simulation period for more than twice as long as the first scheme. This demonstrated the increased sensitivity to flow variations of the first scheme and its increased ability over the second scheme to dampen out spurious oscillations.

Moving Surge First Scheme

\[ F_n = 1.49 \]

Figure 21 Moving Surge First Scheme, \( F_n \) equal to 1.49 at node 14.
The first scheme routines to selectively and optimally apply dampening are further illustrated by the analysis of a stationary jump. A stationary jump with a $F_n$ of 2.0 and a downstream to upstream depth ratio of 2.4 was tested in both schemes. The second scheme was unable to withstand the severe variations introduced by such a strong discontinuity and the solution oscillated out of control in just two iterations. The first scheme on the other hand was able to dampen out spurious oscillations associated with the strong jump.

A stationary jump with a $F_n$ of 1.25 and a downstream to upstream depth ratio of 1.3 was analyzed with both schemes. The results are shown in Figures 23 and 24. The first scheme was only marginally better in dampening out the spurious oscillations with a magnitude of the $y/y_1$ ratio that varied from 0.94 to 1.39 while the second scheme $y/y_1$ ratio varied from 0.91 to 1.42. The second scheme showed the variations in depth due to the jump over approximately 9 nodes while the first scheme showed

**Figure 22** Moving Surge Second Scheme, $F_n$ equal to 1.49 at node 14.
the variations in depth due to the jump over approximately 8 nodes. In relatively mild stationary shock situations, the second scheme and the first scheme converged to essentially the same approximate solution.

Figure 23 Stationary Jump First Scheme, $F_n$ equal to 1.25.
Stationary Jump Second Scheme

$F_n = 1.25$

Figure 24 Stationary Jump Second Scheme, $F_n$ equal to 1.25.
CHAPTER 6

CONCLUSION

This thesis explores two modified Petrov-Galerkin schemes for studying the propagation of bores and hydraulic jumps in open channels. Attention has more recently been focused on such techniques and the problems encountered simulating accurately flow variables in the vicinity of large discontinuities. The method studied by this thesis attempts to regulate more judiciously the application of dampening in the region of the shock to improve model accuracy by eliminating spurious oscillations.

When a moving surge was analyzed, it was found that a better solution was obtained when $\alpha$ was varied between a high and low value rather than simple toggled between the high and low value. This suggests that the variable $\alpha$ can be tailored to improve solution accuracy. It was also noted that making the routine more sensitive to $\alpha$ by reducing the size of $\epsilon$ improved accuracy. It was noted of the Galerkin scheme in general that the implicit like scheme ($\Theta$ equal to 0.75 or 1.0) gives the best overall solution, that the scheme works better at larger time steps and not as well with increased upstream Froude Number.

It was discovered during the analysis of a stationary jump, that different amounts of dampening were optimal depending on the location of the jump. In the region upstream of the jump, a very small value for $\epsilon$ was optimal suggesting that sensitivity to shock conditions is important and the introduction of dampening as soon
as shock conditions are approaching gives the best results. On the other hand in the region downstream of the shock, larger values for $\varepsilon$ were optimal. This suggests that less dampening is necessary to give the best solution. With optimal values of $\varepsilon$ and $\alpha$ set, all values for $\theta$ tested greater than 0.5 gave similar results in the region upstream of the shock with higher values marginally better in the downstream region.

With optimal values for the variables that control dampening in the first scheme, the first and second schemes were compared. Figures 21 and 22 show the results of an analysis of a moving surge with the first scheme and the second scheme respectively. Figures 23 and 24 show the results of an analysis of a stationary jump again with the first scheme and the second scheme respectively. In all cases the first scheme produced more accurate results and less spurious oscillations in the vicinity both upstream and downstream of the surge. The most dramatic results of the comparison are noted in the downstream region of the moving shock where the first scheme practically eliminates unwanted variations in the solution plot. These comparisons indicate that control of dampening in a Petrov-Galerkin scheme can lead to beneficial results.

Less success was noted when attempting to simulate the propagation of a negative wave. More work is warranted in this area. Additional work is suggested as well with the positive surge waves. The apparent success of the modeling effort in the downstream region of the moving surge should be replicated in the upstream region of the moving surge and in both upstream and downstream regions of the stationary jump.
APPENDIX I

BASIC PETROV-GALERKIN SCHEME

C open channel flow eq's
C galerkin formulation
C asymmetric weighting functions on space derivatives
C file = gmatx2.f
C output written to facilitate graphic manipulation

Program mike
implicit real*4(a-h,o-z)
common/blkl/nn,nq,t,i,gr
common/blk2/fo(400),yi,qi,th,h,yn,qn
common/blk3/fc(400),xc(400),kount
common/blk4/al(400),dx,ali,al2,beta,cr
common/blk5/ldfjac,jjac(400,400),epsfcn,dxc(400),m,n,$xscale(400)
common/blk6/ym,y,yp,qm,q,qp,j
common/blk7/knt
common/blk8/first
common/blk9/xn,so
logical first
open (unit=7,file='gmatx2.out',status='old')
open (unit=8,file='frl49m4.dat',status='old')
write(7,48)
48 format(5x,'gmatx2' )
write(7,50)
50 format(5x,'frl49m4'/)
write(7,56)
read(8,*),node,knt,ldfjac,gr,ali,al2,beta,cr,nw
nq=2*node-4
nn=node
read(8,*),dx,th,tmax,yi,qi,xn,so,yn,qn
read(8,*),(j,xc(2*j-3),xc(2*j-2),ali(i),i=1,nn-2)
write(7,51) node, knt, ldfjac, gr, al1, al2, beta, cr, nw
write(7,52) dx, th, t, tmax, yi, qi, xn, so, yn, qn
write(7,53) (i, xc(2*i-1), xc(2*i), al(i), i=1, nn-2)
write(7,55)
51 format(5x, 'node= ', i3, '; knt= ', i2, '; ldfjac= ', i4,
$'; gr= ', f4.2, '; al1= ', f4.3, '; al2= ', f4.3, '; beta= ', f4.3,
$'; Cr= ', f5.3, '; nw= ', i3)
52 format(5x, 'dx= ', f5.1, '; th= ', f4.2,
$'; t= ', f3.1, '; tmax= ', f6.1, '; yi= ', f3.1, '; qi= ', f5.1,
$'; xn= ', f5.4, '; so= ', f5.4, '; yn= ', f3.1, '; qn= ', f5.1)
53 format(5x, 'j; y; q; alpha= ', i2, f6.1, f7.1, f7.3)
54 format(//)
55 format(//)
56 format('******************************************************/)
c m=nq
n=nq
first=.true.
do 5 i=1, nq
   xscale(i)=1.0
5 continue
c dx is delta x
c h is delta t
c th is theta
c t is elapsed time
c alpha is
c fcn presents the continuity and momentum eq's.
c fdjac solves for the jacobian of the coefficients
c lslrg is linear eq solver
c output writes to an output file
c time checks the accumulated time and increments
c reset makes the new values the old values
do 200 i=1, nq
   fc(i)=0.0
200 continue
sq3=sqrt(3.0)
h=dx/(sq3*(abs(xc(2)/xc(1))+sqrt(gr*xc(1))))*cr
do 40 i=4, nq, 2
   ht=dx/(sq3*(abs(xc(i)/xc(i-1))+sqrt(gr*xc(i-1))))*cr
   if(ht.lt.h) h=ht
40 continue
h=0.99*h
al2=h/dx
call reset
c kntw is used to write every 10 time steps
c kntw=0
10 knt=knt+1
c kntw=kntw+1
call iterate
if (kount.ge.40) then
    write(7,12)
    12 format(/'*** error in converging ***', go to 25
    endif
    call xtime
    al2=h/dx
    if (kntw.lt.nw) go to 16
    kntw=0
    15 call output2
    16 call reset
    if (xc(7).lt.(yi+yn)/2) go to 10
    25 stop
end

subroutines

subroutine iterate
    implicit real*4(a-h,o-z)
    common/blkl/nn,nq,t,i,gr
    common/blk2/fo(400),yi,qi,th,h,yn,qn
    common/blk3/fc(400),xc(400),kount
    common/blk4/al(400),dx,al1,al2,beta,cr
    common/blk5/ldfjac,fjac(400,400),epsfcn,dxc(400),m,n,$xscale(400)
    common/blk6/ym,yp,qm,q,qn,qp,j
    common/blk7/knt
    common/blk8/first
    external fcn
    logical first
    epsfcn=0.0
    kount=1
    10 call fcn(m,n,xc,fc)
    first=.false.
    call fdjac(fcn,m,nq,xc,xscale,fc,epsfcn,fjac,ldfjac)
    ipath=1
    call lslrg(nq,fjac,ldfjac,fc,ipath,dxc)
    do 20 jj=1,nq
        xc(jj)=xc(jj)-dxc(jj)
    20 continue
    eps=0.0
    do 30 jj=1,nq
        eps=amax1(eps,abs(dxc(jj)/xc(jj)))
    30 continue
kount=kount+1
if (kount.ge.40) go to 40
if (eps.gt.0.00001) go to 10
40 return
end

subroutine fcn(m,n,xc,fc)
implicit real*4(a-h,o-z)
common/biki/nn,nq,t,i,gr
common/bik2/fo(400),yi,qi,th,h,yn,qn
common/bik4/al(400),dx,al,al2,beta,cr
common/bik6/ym,y,yp,qm,q,qp,j
common/bik7/knt
common/bik8/first
common/bik9/xn,so
dimension fc(m), xc(m)

logical first
external g,ytc,ytcm,ytc,ytcp,qxm,qxm,qxm,qxmp,ym,xym,xym,yym,yym,yym,yym,
yym,yym,yym,xym,xym,xym

nm=nq/2

First do node 2
i=l
ym=yi
qm=qi
y=xc(1)
q=xc(2)
yp=xc(3)
qp=xc(4)

if (first) then
  fl=(g(ytcm)*ym+g(ytc)*y+g(ytcp)*yp)
f2=(1-th)*(g(qxcm)*qm+g(qxc)*q+g(qxcp)*qp)*h
f3=(1-th)*(g(yxcm)*ym+g(yxc)*y+g(yxcp)*yp)*h
  fo(1)=f2+f3-fl
f4=(g(qxmm)*qm+g(qxm)*q+g(qxmp)*qp)*h
f5=(1-th)*(g(qxmm)*qm+g(qxm)*q+g(qxmp)*qp)*h
f6=(1-th)*(g(yxmm)*ym+g(yxm)*y+g(yxmp)*yp)*h
  fo(2)=f5+f6-f4
endif

f1=(g(ytcm)*ym+g(ytc)*y+g(ytcp)*yp)
f2=th*(g(qxcm)*qm+g(qxc)*q+g(qxcp)*qp)*h
f3=th*(g(yxcm)*ym+g(yxc)*y+g(yxcp)*yp)*h
fc(1)=f1+f2+f3+fo(1)
f4=(g(qxmm)*qm+g(qxm)*q+g(qxmp)*qp)*h
f5=th*(g(qxmm)*qm+g(qxm)*q+g(qxmp)*qp)*h
\[ f_6 = (1 - \theta_n) \ast (g(y_{xmm}) \ast y_m + g(y_{xmp}) \ast y_p) \ast h \]
\[ f_0(nq) = f_5 + f_6 - f_4 \]
\[ f_1 = (g(y_{tcn}) \ast y_m + g(y_{tcp}) \ast y_p) \]
\[ f_2 = \theta \ast (g(q_{xcm}) \ast y_m + g(q_{xcp}) \ast y_p) \ast h \]
\[ f_3 = \theta \ast (g(y_{xcm}) \ast y_m + g(y_{xcp}) \ast y_p) \ast h \]
\[ f_{c(nq-1)} = f_1 + f_2 + f_3 + f_0(nq-1) \]
\[ f_4 = (g(q_{tmm}) \ast q_m + g(q_{tmp}) \ast q_p) \ast h \]
\[ f_5 = \theta \ast (g(q_{xmm}) \ast q_m + g(q_{xmp}) \ast q_p) \ast h \]
\[ f_6 = \theta \ast (g(y_{xmm}) \ast y_m + g(y_{xmp}) \ast y_p) \ast h \]
\[ f_{c(nq)} = f_5 + f_6 + f_4 + f_0(nq) \]

**SUBROUTINE OUTPUT**

```fortran
implicit real*4(a-h,o-z)
common/blkl/nn,nq,t,gr
common/blk2/fo(400),yi,qi,th,h,yn,qn
common/blk3/fc(400),xc(400),kount
common/blk4/al(400),dx,all,al2,beta,cr
common/blk7/knt

the following code writes to a file that will be
easy to manipulate in a graphics program
```

```fortran
write(7,60)knt
60 format(5x,i3)
write(7,70)
70 format(/)
write(7,65)(xc(i-1),xc(i),al(i/2),i=2,nq,2)
65 format(5x,3f15.3)
write(7,70)
return
end
```

**SUBROUTINE OUTPUT2**

```fortran
implicit real*4(a-h,o-z)
common/blkl/nn,nq,t,gr
common/blk2/fo(400),yi,qi,th,h,yn,qn
common/blk3/fc(400),xc(400),kount
common/blk4/al(400),dx,all,al2,beta,cr
common/blk7/knt

c the following code writes the output of node 14
to be folled over time
```

```fortran
write(7,65)xc(27),xc(28),al(14)
65 format(5x,3f15.3)
return
end
```
subroutine xtime
  implicit real*4(a-h,o-z)
  common/blkl/nn,nq,t,i,gr
  common/blk2/fo(400),yi,qi,th,h,yn,qn
  common/blk3/fc(400),xc(400),kount
  common/blk4/al(400),dx,all,al2,beta,cr
  h=(dx/(abs(xc(2)/xc(1))+sqrt(gr*xc(1))))*cr
  do 40 i=4,nq,2
    ht=(dx/(abs(xc(i)/xc(i-1))+sqrt(gr*xc(i-1))))*cr
    if (ht.lt.h) h=ht
  40 continue
  h=0.99*h
  t=t+h
  return
end

subroutine reset
  implicit real*4(a-h,o-z)
  common/blkl/nn,nq,t,i,gr
  common/blk2/fo(400),yi,qi,th,h,yn,qn
  common/blk3/fc(400),xc(400),kount
  common/blk4/al(400),dx,all,al2,beta,cr
  common/blk8/first
  logical first
  first=.true.
  nm=nq/2

  check change in slope routine for alpha
  i=1
  chsl=abs((xc(3)-yi)/(yn-yi))
  if(chsl.ge.beta)then
    al(1)=al2
  else
    al(1)=all
  endif
  write(7,15)i,chsl,al(i)
  15 format(5x,i3,2f10.4)
  do 20 i= 2,nm-1
    j=2*i
    chsl=abs((xc(j+1)-xc(j-3))/(yn-yi))
    if(chsl.ge.beta)then
      al(i)=al2
    else
      al(i)=all
    endif
    write(7,15)i,chsl,al(i)
  20 continue
  i=nm
  chsl=abs((yn-xc(nq-3))/(yn-yi))
if(chsl.ge.beta)then
   al(nn-2)=a12
else
   al(nn-2)=a11
endif

write(7,15)i,chsl,al(i)
return
end

definitions

function g(func)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,all,a12,beta,cr
  gl=func(-0.9061798)+func(0.9061798)
  g2=func(-0.5384693)+func(0.5384693)
  g=0.2369269*gl+0.4786287*g2+0.5688889*func(0.0)
return
end

function php(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,all,a12,beta,cr
  php=0.5*(1+x)
return
end

function phm(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,all,a12,beta,cr
  phm=0.5*(1-x)
return
end

function qtmm(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,all,a12,beta,cr
  qtmm=0.5*dx*phm(x)*php(x)
return
end

function qtm(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,all,a12,beta,cr
  qtm=0.5*dx*(php(x)*php(x)+phm(x)*phm(x))
return
end

function qtmn(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,all,a12,beta,cr
  qtmn=0.5*dx*(php(x)*php(x))
return
end
function qtmp(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,al1,al2,beta,cr
qtmp=0.5*dx*php(x)*phm(x)
return
end

function qxmm(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

c set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
yjm=ym*phmx+y*phpx
um=qjm/yjm
c2m=gr*yjm

c q1=2*um*phpx
q2=3*al(i)*phmx*phpx*(c2m+3*um**2)
qxmm=-0.5*(q1+q2)
return
end

function qxm(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

c set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
qjp=q*phmx+qp*phpx
yjm=ym*phmx+y*phpx
yjp=y*phmx+yp*phpx
um=qjm/yjm
up=qjp/yjp
c2m=gr*yjm
c2p=gr*yjp

c q1=2*um*phpx
q2=3*al(i)*phmx*phpx*(c2m+3*um**2)
q3=2*up*phmx
q4=-3*al(i)*phmx*phpx*(c2p+3*up**2)
qxm=0.5*(q1+q2)-0.5*(q3+q4)
return
end

function qxmp(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjp=q*phmx+qp*phpx
ypj=y*phmx+y*p*phpx
up=qjp/ypj
c2p=gr*yjp

q1=2*up*phmx
q2=-3*al(i)*phmx*phpx*(c2p+3*up**2)
qxmp=0.5*(q1+q2)
return
end

function qxmn(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
ymj=ym*phmx+y*phpx
um=qjm/ymj
c2m=gr*yjm

q1=2*um*phpx
q2=3*al(i)*phmx*phpx*(c2m+3*um**2)
qxmn=0.5*(q1+q2)
return
end

function yxmm(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
yjm=ym*phmx+y*phpx
um=qjm/yjm

c2m=gr*yjm

sfm=(xn**2*qjm*abs(qjm))/yjm**(10/3)

c
yl=phpx+6*al(i)*um*phmxphpx
y2=-0.5*(c2m-um**2)
y3=0.5*gr*dx*(sfm-so)
yxmm=yl*(y2+y3)
return
end

c

function yxm(x)
imPLICIT real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

c
set up variables for use in coefficients

c
phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
qjp=q*phmx+qp*phpx
yjm=ym*phmx+y*phpx
yjp=y*phmx+yp*phpx
um=qjm/yjm
up=qjp/yjp
c2m=gr*yjm
c2p=gr*yjp

sfm=(xn**2*qjm*abs(qjm))/yjm**(10/3)
sfp=(xn**2*qjp*abs(qjp))/yjp**(10/3)

c

yl=phpx+6*al(i)*um*phmxphpx
y2=0.5*(c2m-um**2)
y3=0.5*gr*dx*(sfm-so)
yxmp=yl*(y2+y3)+y4*(y5+y6)
return
end


c

function yxmp(x)
imPLICIT real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so


c
set up variables for use in coefficients

c
phmx=phm(x)
phpx=php(x)
qjp=q*phmx+qp*phpx
yjp = y*phmx + yp*phpx
up = qjp/yjp
c2p = gr*yjp
sfp = (xn**2*qjp*abs(qjp))/yjp**(10/3)

yl = phmx - 6*al(i)*up*phmx*phpx
y2 = 0.5*(c2p-up)**2
y3 = 0.5*gr*dx*(sfp-so)
yxmp = yl*(y2+y3)
return
end

function yxmn(x)
implicit real*4(a-h, o-z)
common/blkl/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

set up variables for use in coefficients
phmx = phm(x)
phpx = php(x)
qjm = qm*phmx + q*phpx
yjm = ym*phmx + y*phpx
um = qjm/yjm
c2m = gr*yjm
sfm = (xn**2*qjm*abs(qjm))/yjm**(10/3)

yl = phpx + 6*al(i)*um*phmx*phpx
y2 = 0.5*(c2m-um)**2
y3 = 0.5*gr*dx*(sfm-so)
yxmn = yl*(y2+y3)
return
end

function yxcm(x)
implicit real*4(a-h, o-z)
common/blkl/nn,nq,t, i, gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

set up variables for use in coefficients
phmx = phm(x)
phpx = php(x)
qjm = qm*phmx + q*phpx
yjm = ym*phmx + y*phpx
um = qjm/yjm
c2m = gr*yjm
sfm = (xn**2*qjm*abs(qjm))/yjm**(10/3)

yl = -0.5*(c2m-um)**2
y2 = 0.5*gr*dx*(sfm-so)
yxcm = 3.*al(i)*phmx*phpx*(y1+y2)
return
end

c  function yxc(x)
  implicit real*4(a-h,o-z)
  common/blk1/nn,nq,t,i,gr
  common/blk4/al(400),dx,all,al2,beta,cr
  common/blk6/ym,y,yp,qm,q,qp,j
  common/blk9/xn,so
  
  c  set up variables for use in coefficients
  
c    phmx=phm(x)
    phpx=php(x)
    qjm=qm*phmx+q*phpx
    qjp=q*phmx+qp*phpx
    yjm=ym*phmx+y*phpx
    yjp=y*phmx+yp*phpx
    um=qjm/yjm
    up=qjp/yjp
    c2m=gr*yjm
    c2p=gr*yjp
    sfm=(xn**2*qjm*abs(qjm))/yjm**(10/3)
    sfp=(xn**2*qjp*abs(qjp))/yjp**(10/3)
    yl=0.5*(c2p-up**2)
    y2=0.5*gr*dx*(sfm-sfp)
    y3=0.5*(c2m-um**2)
    yxc= 3.*al(i)*phmx*phpx*(-yl+y2+y3)
  return
end

c  function yxcp(x)
  implicit real*4(a-h,o-z)
  common/blk1/nn,nq,t,i,gr
  common/blk4/al(400),dx,all,al2,beta,cr
  common/blk6/ym,y,yp,qm,q,qp,j
  common/blk9/xn,so
  
  c  set up variables for use in coefficients
  
c    phmx=phm(x)
    phpx=php(x)
    qjp=q*phmx+qp*phpx
    yjp=y*phmx+yp*phpx
    up=qjp/yjp
    c2p=gr*yjp
    sfp=(xn**2*qjp*abs(qjp))/yjp**(10/3)
    yl=0.5*(c2p-up**2)
    y2=0.5*gr*dx*(sfp-so)
    yxc= -3.*al(i)*phmx*phpx*(yl+y2)
  return
end

c  function yxcn(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,y,yp,qm,q,qp,j
common/blk9/xn,so

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
yjm=ym*phmx+y*phpx
um=qjm/yjm
c2m=gr*yjm
sfm=(xn**2*qjm*abs(qjm))/yjm**(10/3)

yi=0.5*gr*dx*abs(sfm-so)
y2=0.5*(c2m-um**2)
yxcn = 3.*al(i)*phmx*phpx*(yi+y2)
return

function ytcm(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,al1,al2,beta,cr
ytcm=0.5*dx*phm(x)*php(x)
return
end

function ytc(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,al1,al2,beta,cr
phmx=phm(x)
phpx=php(x)
ytc=0.5*dx*(phpx*phpx+phmx*phmx)
return
end

function ytcn(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,al1,al2,beta,cr
phpx=php(x)
ytcn=0.5*dx*(php(x)*php(x))
return
end

function ytcp(x)
implicit real*4(a-h,o-z)
common/blk4/al(400),dx,al1,al2,beta,cr
ytcp=0.5*dx*php(x)*phm(x)
return
end

function qxcm(x)
implicit real*4(a-h,o-z)
common/blk1/nn,nq,t,i,gr
common/blk4/al(400),dx,al1,al2,beta,cr
common/blk6/ym,yp,qm,q,qp,j

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
yjm=ym*phmx+y*phpx
um=qjm/yjm

qxc=0.5*(phpx+6*al(i)*um*phmx*phpx)
return
end

c

function qxcm(x)
imPLICIT REAL*4(a-h,o-z)
common/blkl/nn,nq,t,i,gr
c

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjm=qm*phmx+q*phpx
qjp=q*phmx+qp*phpx
yjm=ym*phmx+y*phpx
yjp=y*phmx+yp*phpx
um=qjm/yjm
up=qjp/yjp

q1=0.5*(phpx+6*al(i)*um*phmx*phpx)
q2=0.5*(phmx-6*al(i)*up*phmx*phpx)
qxcm=q1+q2
return
end

function qxcp(x)
imPLICIT REAL*4(a-h,o-z)
common/blkl/nn,nq,t,i,gr
c

set up variables for use in coefficients

phmx=phm(x)
phpx=php(x)
qjp=q*phmx+qp*phpx
yjp=y*phmx+yp*phpx
up=qjp/yjp

qxcp=0.5*(phmx-6*al(i)*up*phmx*phpx)
return
end
function qxcn(x)
  implicit real*4(a-h,o-z)
  common/bkl1/nn,nq,t,i,gr
  common/bkl4/al(400),dx,all,al2,beta,cr
  common/bkl6/ym,y,yp,qm,q,qp,j
  c
  c set up variables for use in coefficients
  c
  phmx=phm(x)
  phpx=php(x)
  qjm=qm*phmx+q*phpx
  yjm=ym*phmx+y*phpx
  um=qjm/yjm
  c
  qxcn=0.5*(phpx+6*al(i)*um*phmx*phpx)
  return
end
APPENDIX II

VARIABLE ALPHA ROUTINE

The following routine differs from the first basic Petrov-Galerkin scheme shown in Appendix I in the way in which $\alpha$ is calculated. In the basic scheme (called the fixed $\alpha$ model) $\alpha$ is toggled between high and low values depending on the magnitude of the variation in the water surface at two adjacent nodes. In the modified scheme (called the variable $\alpha$ model) shown below, the value of $\alpha$ is interpolated between high and low values depending on the ratio of the variation in the water surface at two adjacent nodes to the variation in the water surface at the upstream and downstream ends. The same variable, $e$, called beta in the computer code, used to trigger the use of the appropriate value of $\alpha$ is also used here to interpolate between the high and low values of $\alpha$.

```
c
subroutine reset
  implicit real*4(a-h,o-z)
  common/blkl/nn,nq,t,i,gr
  common/blk2/f0(400),yi,qi,th,h,yn,qn
  common/blk3/fc(400),xc(400),kount
  common/blk4/al(400),dx,all,al2,beta,cr
  common/blk8/first
  logical first
  first=.true.
  c
  nm=nq/2
  c
  check change in slope routine for alpha
  c
```
i=1
chsl=abs((xc(3)-yi)/(yn-yi))
if(chsl.ge.beta) then
  al(1)=a12
else
  al(1)=all+(a12-all)*(chsl/beta)
endif
c
write(7,15)i,chsl,al(i)
15 format(5x,i3,2f10.4)
do 20 i= 2,nm-1
  j=2*i
  chsl=abs((xc(j+1)-xc(j-3))/(yn-yi))
  if(chsl.ge.beta) then
    al(i)=a12
  else
    al(i)=all+(a12-all)*(chsl/beta)
  endif
c
write(7,15)i,chsl,al(i)
20 continue
i=nm
chsl=abs((yn-xc(nq-3))/(yn-yi))
if(chsl.ge.beta) then
  al(nn-2)=a12
else
  al(nn-2)=all+(a12-all)*(chsl/beta)
endif
c
write(7,15)i,chsl,al(i)
return
end
The following routine is a further variation the first Petrov-Galerkin schemes shown in Appendices I and II. In this routine the variable epsilon, ε, is divided into two variables, ε₁ and ε₂, one for upstream of the wave and one for downstream of the wave. The premise was that different degrees of damping would be optimal in the subcritical and supercritical regions of flow.

In the computer routine, ε is represented by the variable beta, beta1, and beta2 respectively. ε is set equal to ε₁ or ε₂ depending on the overall depth of flow in the channel. If the overall depth of flow is closer to the downstream depth, ε is set equal to ε₁. If the overall depth of flow is closer to the upstream depth, ε is set equal to ε₂. Closer for this exercise is defined as 75 percent of the difference between the upstream and downstream depths of flow.

```fortran

subroutine reset
    implicit real*4(a-h,o-z)
    common/blkl/nn,nq,t,i,gr
    common/blk2/fo(400),yi,qi,th,h,yn,qn
    common/blk3/fc(400),xc(400),kount
    common/blk4/al(400),dx,al1,al2,beta,cr
    common/blk8/first,betal,beta2
    logical first
    first=.true.
    nm=nq/2
```

58
c  check change in slope routine for alpha
  
  ys=yn-yi
  d2=.75*ys
  i=1
  chsl=(abs(xc(3)-xc(1))+abs(xc(1)-yi))/ys
  beta=betal
  if(chsl.ge.beta) then
    al(1)=5*al2
  else
    al(1)=al(1)+5*(al2-al1)*(chsl/beta)
  endif
  write(7,15)i,chsl,al(i)
  15 format(5x,i3,2f10.4)

  do 20 i= 2,nm-1
    j=2*i
    chsl=(abs(xc(j+1)-xc(j-1))+abs(xc(j-1)-xc(j-3)))/ys
    dl=abs(xc(j-3)-yi)
    if(dl.le.d2)then
      beta=betal
    else
      beta=beta2
    endif
    if(chsl.ge.beta) then
      al(i)=5*al2
    else
      al(i)=al(i)+5*(al2-al1)*(chsl/beta)
    endif
  write(7,15)i,chsl,al(i)
  20 continue

  i=nm
  chsl=(abs(yn-xc(nq-1))+abs(xc(nq-1)-xc(nq-3)))/ys
  beta=betal
  if(chsl.ge.beta) then
    al(nm-2)=5*al2
  else
    al(nm-2)=al(1)+5*(al2-al1)*(chsl/beta)
  endif
  write(7,15)i,chsl,al(i)
  return
end
BIBLIOGRAPHY


