Computers & Geosciences Vol. 13, No. 3, pp. 255–285, 1987
Printed in Great Britain.

0098-3004/87 $3.00 + 0.00

FEUDX: A TWO-STAGE, HIGH-ACCURACY, FINITE-ELEMENT FORTRAN PROGRAM FOR SOLVING SHALLOW-WATER EQUATIONS

I. M. NAVON*

Supercomputer Computations Research Institute, Florida State University, Tallahassee, FL 32306-4052, U.S.A.

(*Received* 19 *February* 1986; *revised* 30 *October* 1986)

Abstract—A FORTRAN computer program is presented and documented which implements a new, two-stage finite-element Numerov–Galerkin method for integrating the nonlinear shallow-water equations on a β-plane limited-area domain. In this method high accuracy is obtained by combining the Galerkin product with a high-order compact (hence the name Numerov) difference approximation to derivatives in the nonlinear advection operator. Conservation of integral invariants is obtained by nonlinear constrained optimization using the Augmented-Lagrangian method, allowing perfect conservation of the integral invariants for long-term integrations.

Program options include the use of a weighted selective lumping scheme in the finite-element method, use of either a Gauss-Seidel or a successive overrelaxation (S.O.R.) iterative method for solving the resulting systems of linear equations, a line-printer plot of the fields contours and finally, determination at each time-step of the values of three integral invariants of the shallow-water equations. A solver for periodic pentadiagonal matrices resulting from the application of the high-order difference approximation is included. Long-term numerical integrations (10–20 days) have been performed using this program. Small-scale noise was eliminated using a Shuman filter, periodically applied to one component of the velocity field.

The method was determined to exhibit a consistently higher accuracy than the single-stage finite-element method and can be use to advantage by meteorologists and oceanographers. Due to the code being modular and flexible it can be changed easily to suit the aims of different researchers. A vectorized version of the code, operative on a CYBER-205 also is available.

Key Words: Finite-element methods, Shallow-water equations, Augmented-Lagrangian methods.

INTRODUCTION

It has become customary in developing new numerical methods for numerical weather prediction or oceanography to study first the simpler nonlinear shallow-water equations system, which possesses the same mixture of slow and fast-moving waves as the more complex barocline three-dimensional primitive equations of motion. One of the issues associated with numerically solving these equations is how to treat the nonlinear advective terms (Cullen and Morton, 1980). In this paper a two-stage Galerkin method combined with a high-accuracy compact (Numerov) approximation to the first derivative is presented. For more theoretical background the reader is referred to Cullen and Morton (1980) and Navon (1979a, 1979b, 1983).

The finite-element method when applied to meteorological and oceanographic problems gives an accurate phase propagation and also handles nonlinearities well. The Galerkin finite-element is conservative and therefore avoids aliasing errors associated with nonlinear terms. It has the advantage over the finite-difference method of being flexibile in the treatment of irregular domains and to allow a variable resolution, thus permitting a focus on regions of interest.

This model can be used to model the upwelling problem in oceanography, to study current patterns in coastal waters, and in meteorology it can be applied for studying the dynamics of observed large-scale waves in the earth's atmosphere.

In the first section of this paper the finite-element Galerkin solution of the shallow-water equations is reviewed and a brief survey of previous research on this topic in meteorology and oceanography is given. The shallow-water equations describe the dynamics of a shallow rotating layer of homogeneous, incompressible, and inviscid fluid with a free surface. The shallow-water equations model is capable of describing important aspects of atmospheric and oceanic motions. The derivation of the two-stage Numerov–Galerkin method for the advective terms of the shallow-water equations is given in another part of this paper and the remainder is devoted to a description of the finite-element code and specifications for its use. Typical run outputs are provided to illustrate each stage of the calculations. A listing of the FORTRAN IV source code of the program FEUDX is included in the Appendix. (A vectorized version of the program FEUDX, termed FEUDX1, is available for interested users.) The typical outputs illustrating the compact storage method, the printer-plotted maps of the height field as well as the constrained minimization

*Supported by the Florida State University Supercomputer Computations Research Institute which is partially funded by the U.S. Department of Energy through Contract No. DE-FC05-85ER250000.

CAGEO 13:3-D

procedure also are included. The code itself is documented amply with a wealth of comments—allowing first-time users of a finite-element code to understand, and if so wished, to modify, the finite-element program.

THE FINITE-ELEMENT GALERKIN SOLUTION OF SHALLOW-WATER EQUATIONS

Derivation of the basic finite-element algorithm

The barotropic nonlinear shallow-water equations on a limited-area domain of a rotating earth (using the β-plane assumption) have the following form:

$$\left.\begin{aligned} u_t + uu_x + vu_y + \phi_x - fv &= 0 \\ v_t + uv_x + vv_y + \phi_y + fu &= 0 \\ \phi_t + (\phi u)_x + (\phi v)_y &= 0 \end{aligned}\right\} \quad (1)$$

$$0 \leqslant x \leqslant L, \quad 0 \leqslant y \leqslant D, \quad t = 0.$$

Here u and v are the velocity components in the x and y directions, respectively; f is the Coriolis parameter by the β-plane approximation:

$$f = \hat{f} + \beta\left(y - \frac{D}{2}\right) \quad (2)$$

where $\hat{f}$ and β are constants, L and D are the dimensions of the rectangular domain of integration, $\phi = gh$ is the geopotential, h is the depth of the fluid and g the acceleration of gravity.

The beta plane assumption in a model consists of a model in which the effect of the earth's sphericity is modeled by a linear variation in the Coriolis parameter $f = 2\Omega \sin \theta$ about a mean latitude θ_0, in an otherwise planar geometry. This is termed the β-plane assumption. Here Ω is the angular velocity of the earth's rotation and θ is latitude. $f = 2\Omega \sin \theta_0$ $\beta_0 = (2\Omega/r_0)\cos\theta_0$ where r_0 is the radius of the earth.

Periodic boundary conditions are assumed in the x-direction, whereas rigid boundary conditions

$$v(x, o, t) = v(x, D, t) = 0 \quad (3)$$

are imposed in the y direction.

In the finite-element Galerkin discretization we use linear piecewise polynomials on triangular elements where for a given triangular element each variable is represented as a linear sum of interpolating functions, for example

$$u_{el} = \sum_{j=1}^{3} u_j(t) V_j(x, y) \quad (4)$$

where $u_j(t)$ represents the scalar nodal value of the variable u at the mode of the triangular element, whereas V_j is the basis function (interpolation function) which can be defined by the coordinates of the modes. Within each triangle any point is affected by only the three basis functions that have nodal points at the three vertices of the triangle.

In what follows the Galerkin formulation with the Einsteinian notation is used, that is a repeated index implies summation with respect to that index.

The notation used is

$$\langle f(x, y), V_i \rangle = \sum_{\text{elements}}^{m} \int\int f(x, y) V_i \, dx dy = \iint_{\text{global}} f(x, y) V_i \, dx dy \quad (5)$$

which defines the inner product when a function is multiplied by a trial function. A convenient procedure for evaluating integrals for each triangle is given in Zienkiewicz (1978) and Desai and Abel (1972). It involves introducing triangular coordinates differing linearly across each triangle in the same way as the basis function. The integrals then can be evaluated analytically using the following formula for area integrals

$$\iint_A L_1^a L_2^b L_3^c \, dx dy = \frac{a!b!c!}{(a + b + c + 2)!} \quad (6)$$

where a, b, and c are integers, L_i, $(i = 1, 2, 3)$ are the basic functions for the triangular linear element as well as the natural coordinate variables.

$$L_i = \frac{1}{2A}(a_i y + b_i x + c_i) \quad i = 1, 2, 3$$

$$A\text{—area of the triangle} \quad (7)$$

$$a_i = y_j - y_k, \quad b_i = x_k - x_j, \quad c_i = x_j y_k - x_k y_j$$

$$i, j, k \text{ cyclically permuted } (i, j, k = 1, 2, 3). \quad (8)$$

The derivatives of the shape functions L_i are:

$$\frac{\partial L_i}{\partial x} = \frac{b_i}{2A}, \frac{\partial L_i}{\partial y} = \frac{c_i}{2A} \quad i = 1, 2, 3. \quad (9)$$

A time extrapolated Crank-Nicolson time-differencing scheme was applied for integrating in time the system of ordinary differential equations resulting from the application of the Galerkin finite-element method (Navon, 1979a, 1979b).

Upon introducing the time discretization in the continuity equation, which is the first to be solved at a given time-step, we obtain:

$$M(\phi_j^{n+1} - \phi_j^n) - \frac{\Delta t}{2} K_1(\phi_j^{n+1} + \phi_j^n) = 0. \quad (10)$$

Here n is the time level $(t_n = n\Delta t)$, Δt the time-step, M is the mass matrix given by

$$M = \iint_A V_j V_i \, dA \quad (11)$$

where the element-mass matrix (3×3) is

$$M_{el} = \frac{A}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \quad (12)$$

obtained by using the integration Equation (6). K_1 gives also rise to a (3×3) element matrix (Navon

and Muller, 1979) and only after the assembly process are the global ($N \times N$) matrices obtained.

$$K_1 = \int\int_A V_i V_k u_k^* \frac{\partial V_j}{\partial x} \, dA + \int\int_A V_i V_k v_k^* \frac{\partial V_j}{\partial y} \, dA. \tag{13}$$

In our notation we use already the global matrices and u^* and v^* are given by

$$\begin{aligned} u^* &= u^{n+1/2} = \frac{3}{2} u^n - \frac{1}{2} u^{n-1} + 0(\Delta t^2) \\ v^* &= v^{n+1/2} = \frac{3}{2} v^n - \frac{1}{2} v^{n-1} + 0(\Delta t^2) \end{aligned} \tag{14}$$

and result from the time extrapolated Crank-Nicolson method (see Douglas and Dupont, 1970; Wang and others, 1972; etc.). This method is used to quasilinearize the nonlinear advective terms.

After an amount of algebra the u and v-momentum equations are obtained with the following form:

$$M(u_i^{n+1} - u_i^n) + \frac{\Delta t}{2} K_2^*(u_i^{n+1} + u_i^n) + \frac{\Delta t}{2}(K_{21}^{n+1} + K_{21}^n) + \Delta t P_2 = 0, \tag{15}$$

$$M(v_i^{n+1} - v_i^n) + \frac{\Delta t}{2} K_3^*(v_i^{n+1} + v_i^n) + \frac{\Delta t}{2}(K_{31}^{n+1} + K_{31}^n) + \Delta t P_3 = 0, \tag{16}$$

where the following matrix definitions have been used

$$K_2^* = \int\int_A u_K^* V_K V_i \frac{\partial V_j}{\partial x} \, dA + \int\int_A v_K^* V_K V_i \frac{\partial V_j}{\partial y} \, dA, \tag{17}$$

$$K_{21}^{n+1} = \int\int \phi_K^{n+1} \frac{\partial V_K}{\partial x} V_i \, dA, \tag{18}$$

$$P_2 = \int\int_A f v_K^* V_K V_i dA, \tag{19}$$

$$K_3^* = \int\int_A u_K^{n+1} V_K \frac{\partial V_j}{\partial x} \, dA + \int\int v_K^* V_K \frac{\partial V_j}{\partial x} \, dA, \tag{20}$$

$$K_{31}^{n+1} = \int\int_A \phi_K^{n+1} \frac{\partial V_K}{\partial y} V_i \, dA, \tag{21}$$

$$P_3 = \int\int_A f u_K^{n+1} V_K V_i dA \tag{22}$$

and similar definitions for K_{31}^n and K_{21}^n, respectively.

Brief review of previous finite-element solutions of shallow-water equations in meteorology and oceanography

Interest in solving the shallow-water equations using the finite-element method had been increasing during the last few years. The shallow-water equations have been used for a wide variety of coastal phenomena such as tide-currents, pollutant dispersion, storm-surges, tsunami wave propogation, upwelling, drift-sand transport, etc. A comprehensive review of the application of the finite-element method for solving shallow-water equations describing coastal as well as oceanographic phenomena was provided by Kawahara (1980). He compared the different finite-element methods as far as time integration procedures, selection of interpolating elements, and finally different applications are concerned.

Brebbia and Partridge (1976a, 1976b) as well as Connor and Brebbia (1976) described shallow-water finite-element models applied to modeling tidal effects and current patterns in coastal waters. Hua and Thomasset (1980, 1984) applied a finite-element scheme for the problem of coastal upwelling induced by winds in the sea waters using a two-layer shallow-water equations model and a semi-implicit time-integration scheme. Their model was built following that of O'Brien and Hulbert (1972). For an implementation of the Hua and Thomasset method for studying effects of coastline geometry on upwellings see Crépon, Richez, and Chartier (1984). Foreman (1983, 1984) analyzed the accuracy of finite-element methods which solve linearized shallow-water equations including group-velocity analysis as well as phase-velocity error analysis. This analysis however is limited by the linearity assumption as well as by the assumption of periodic boundary conditions. In his 1984 paper, Foreman (1984) analyzes the wave equation finite-element method developed by Gray and Lynch (1978) who transform the continuity equation in the shallow-water equations model to a second-order partial differential equation. He also analyzes the Thacker (1978) irregular grid finite difference technique—and draws conclusions as far as accuracy and computational costs are concerned. Amongst his conclusions is that Crank-Nicolson is the best time-stepping method to use. He also recommends use of equilateral triangles to represent accurately phase group velocity. Malone and Kuo (1981) used a semi-implicit finite-element method for computing low-frequency, low wave motions driven by tides and wind on continental shelves. Earlier work on the same topics was carried out by Fix (1975) and Taylor and Davis (1975; see also Praagman, 1979).

In meteorology, the first application of the finite-element method to the shallow-water equations was by Wang and others (1972) solving the one-dimensional shallow-water equations using a Crank-Nicolson time-discretization. Cullen (1973, 1974) solved the shallow-water equations both on a β-plane and on the sphere and pointed out some problem areas. Cullen and Hall (1979) provided a clear exposition of the finite-element method and the analysis of the spatial evolutionary error in different finite-element schemes. Hinsman (1975) and Hinsman and Archer (1976) used linear equilaterial triangles defined on an icosahedral

mesh for solving the shallow-water equations on the sphere using an extrapolated Crank-Nicolson time differencing scheme. Kelley and Williams (1976), Older (1981), and Woodward (1981) used shallow-water equations finite-element models using differently shaped triangles (right-angled and equilateral ones) as well as vorticity-divergence formulations coupled with a semi-implicit scheme. An article by Hinsman, Williams, and Woodward (1982) sums up the results. Williams (1981) examined staggered and unstaggered finite-element formulations for both primitive and vorticity-divergence forms of the shallow-water equations. Williams and Zienkiewicz (1981) proposed a mixed-order type element on a staggered grid for linearized 1-D shallow-water equations and prove that the staggered formulation should be superior. Navon (1977) reviewed the application of finite-element methods to quasilinear fluid flow, whereas Navon (1979a, 1979b) as well as Navon and Muller (1979) solved the shallow-water equations in a channel on the β-plane using different mass-matrix formulation concluding that a mixed mass formulation defined by the average of consistent and lumped mass matrices gave the best results (see also Navon and Riphagen, 1979 and Navon and de Villiers, 1983).

Sasaki and Reddy (1980) studied the advection of a circular vortex using bilinear square elements. They were the first to propose a constrained variational method for enstrophy conservation noting that it improved the long-term numerical integration results. Cullen and Hall (1979) extended the finite-element method comparisons to 3-D general circulation simulations.

Staniforth and Mitchell (1977) proposed a finite-element method based on two-dimensional Châpeau basis function having a nice separability property and applied it to the shallow-water equations on the sphere. The method proved to be efficient for storage and was extended to limited area 3-D by Staniforth and Daley (1979) in a nested baroclinic model simulation. Staniforth (1982, 1984) provided a comprehensive and lucid review of finite-element method applications to meteorological simulations covering the field up to and including 1982.

Navon (1982, 1983) introduced the Numerov–Galerkin finite-element method for the shallow-water equations with an Augmented-Lagrangian constrained-optimization method to enforce integral invariants conservation. Similar work was done by Zienkiewicz and Heinrich (1979) with a finite-element penalty method, and by Zienkiewicz and others (1984).

Boundary condition implementation

I have adopted an approach suggested by Huebner (1975). In this approach one modifies the diagonal terms of the global matrix associated with the nodal variables by multiplying them by a large number, say 10^{16} (selected with a view toward the significant digits available on the local computer facility and the size of the field variables) whereas the corresponding term in the right-hand side vector R in the linear system of equations

$$KX = R \tag{23}$$

where K is the global matrix, is replaced by the specified boundary nodal variable multiplied by the same large number time the diagonal term. This procedure is repeated for all the boundary nodal variables.

If for instance, in the matrix K the boundary condition

$$X_r = \beta_r \tag{24}$$

is to be implemented then its implementation is

$$\begin{bmatrix} k_{11} & k_{12} & \cdots & \cdots & k_{1N} \\ \vdots & \ddots & & & \vdots \\ k_{r1}\,k_{r2}\cdots & \cdots & k_{rr}\cdot 10^{16} & \cdots\cdots & k_{rN} \\ \vdots & & & & \vdots \\ k_{N1}\,k_{N2}\cdots & \cdots & \cdots & \cdots & k_{NN} \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_r \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ \beta_r \cdot K_{rr} 10^{16} \\ \vdots \\ R_N \end{bmatrix} \tag{25}$$

THE TWO-STAGE NUMEROV–GALERKIN SCHEME

The two-stage Galerkin method (see Cullen and Morton, 1980) is applied to the nonlinear advective terms of form $v\partial v$. If we consider the advective operator

$$L(u, v) = u\frac{\partial v}{\partial x} \tag{26}$$

then as shown by Cullen and Morton (1980) we can consider two methods.

A direct Galerkin approximation

$$\text{Using two functions } u = \exp^{(ik\xi)}, \quad v = \exp^{(il\eta)} \quad \text{with} \quad \xi = kh, \quad \eta = lh \tag{27}$$

where h is a positive mesh length one can show the asymptotic truncation error of $u(\partial v/\partial x)$ is (by assuming Fourier modes)

$$|T.E.| \sim \frac{[4\eta^4 + 8\eta^2\xi + 7\eta^2\xi^2 - 2\eta\xi^3]}{720} \quad (28)$$

and if $\xi = \eta$

$$|T.E.| \sim \frac{17}{720}\eta^4. \quad (29)$$

The two-stage Galerkin approximation

In this approach one calculates the Galerkin approximation to $\partial v/\partial x$ which we term Z:

$$\frac{1}{6}Z_{j-1} + \frac{2}{3}Z_j + \frac{1}{6}Z_{j-1} = \frac{1}{2}h^{-1}(V_{j+1} - V_{j-1}). \quad (30)$$

Then we calculate the product

$$W = u\frac{\partial v}{\partial x} \quad (31)$$

$$\frac{1}{6}W_{j+1} + \frac{2}{3}W_j + \frac{1}{6}W_{j-1} = \frac{1}{12}(U_{j-1}Z_{j-1} + U_{j-1}Z_j + U_jZ_{j-1} + U_jZ_{j+1} + U_{j+1}Z_j + U_{j+1}Z_{j+1}) + \frac{1}{2}U_jZ_j. \quad (32)$$

This can be shown to give an algorithm with an asymptotic truncation error of

$$|T.E.|_{\text{Two-stage N G}} \sim \frac{[2\xi^3\eta + 3\xi^2\eta^2 + 2\xi\eta^3 - 4\eta^4]}{720} \quad (33)$$

and if $\xi = \eta$

$$|T.E.|_{\text{Two-stage N G}} \sim \frac{3}{720}\eta^4 \quad (34)$$

that is, an error at least six times smaller than Equation (29). In our approach we combine the two-stage Galerkin method with a high-order compact implicit difference approximation to the first derivative.

This approximation has a truncation-error of $0(h^{4l})$ and uses a finite-difference stencil of $2l + 1$ grid points—at the price of solving a $2l + 1$ banded matrix (see Swartz and Wendroff, 1974; and Navon and Riphagen, 1979). The compact Numerov $0(h^8)$ approximation to $\partial v/\partial x$ is given by

$$\frac{1}{70}\left[\left(\frac{\partial u}{\partial x}\right)_{i+2} + 16\left(\frac{\partial u}{\partial x}\right)_{i+1} + 36\left(\frac{\partial u}{\partial x}\right)_i + 16\left(\frac{\partial u}{\partial x}\right)_{i-1} + \left(\frac{\partial u}{\partial x}\right)_{i-2}\right]$$

$$= \frac{1}{84h}[-5u_{i-2} - 32u_{i-1} + 32u_{i+1} + 5u_{i+2}]$$

$$h = \Delta x = \Delta y. \quad (35)$$

The estimation of $\partial v/\partial x$ necessitates solving a pentadiagonal system of the form:

$$\frac{1}{70}\begin{bmatrix} 36 & 16 & 1 & & 0 \\ 16 & \ddots & \ddots & \ddots & \\ 1 & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & \ddots & 16 \\ 0 & & 1 & 16 & 36 \end{bmatrix}\begin{bmatrix} v \\ \partial v/\partial x \end{bmatrix}$$

$$= \frac{1}{84h}\begin{bmatrix} 0 \\ -5v_0 - 32v_1 + 32v_3 + 5v_4 \\ -5v_1 - 32v_2 + 32v_4 + 5v_5 \\ \vdots \\ -5v_{N_x-4} - 32v_{N_x-2} + 32v_{N_x} + 5v_{N_x+1} \\ -5v_{N_x-3} - 32v_{N_x-2} + 32v_{N_x} + 5v_{N_x+1} \\ 0 \end{bmatrix}$$

$$j = 1, 2, \ldots, N_y. \quad (36)$$

Here we interpolate v_0 and v_{N_x+1} using

$$v_0 = 4v_1 - 6v_2 + 4v_3 - v_4$$
$$v_{N_x+1} = 4v_{N_x} - 6v_{N_x-1} + 4v_{N_x-2} - v_{N_x-3} \quad (37)$$

whereas for the intermediate expression Z we have:

$$Z_1 = \left(\frac{\partial v}{\partial y}\right)_1 = \frac{-25v_1 + 48v_2 - 36v_3 - 16v_4 - 3v_5}{12h} + 0(h^4)$$

$$Z_{N_y} = \left(\frac{\partial v}{\partial y}\right)_{N_y} = \frac{3v_{N_y-4} - 16v_{N_y-3} + 36v_{N_y-2} - 48v_{N_y-1} + 25v_{N_y}}{12h} + 0(h^4). \quad (38)$$

For the second stage of the finite-element-Numerov-Galerkin we solve a tridiagonal system of the form

$$\frac{1}{6}\begin{bmatrix} 4 & 1 & & \\ 1 & \ddots & \ddots & 0 \\ & \ddots & \ddots & 1 \\ 0 & & 1 & 4 \end{bmatrix}[w_j] = \frac{1}{12}\begin{bmatrix} v_{j-1}Z_{j-1} + v_jZ_{j-1} + v_{j-1}Z_j + \\ v_{j+1}Z_j + v_jZ_{j+1} + v_{j+1}Z_{j+1} + 6v_jZ_j \end{bmatrix}. \quad (39)$$

In the second stage we interpolate the values of Z_0 and Z_{N_x+1} in a way similar to Equation (38). A pentadiagonal and a cyclic pentadiagonal matrix solver (necessitated due to periodic boundary conditions) were developed following Von-Rosenberg (1969) and generalizing Ahlberg, Nielson, and Walsh (1967), re-

spectively. A detailed account of the pentadiagonal solver will be published elsewhere.

COMPUTATIONAL ECONOMY DUE TO THE NUMEROV–GALERKIN METHOD

The u and v momentum Equations (15) and (16) undergo changes due to the use of the Numerov–Galerkin finite-element method. Denoting

$$\frac{\partial u}{\partial x} = Z_{xu}, \quad \frac{\partial V}{\partial v} = Z_{xv} \tag{40}$$

the intermediate Numerov approxiamtion representing the first-stage derivatives $\partial_x u$ and $\partial_x v$ respectively and similar notation Z_{yu}, Z_{yv} for the y derivatives corresponding to the intermediate stage of the Numerov–Galerkin. We get the following modified matrix u-momentum equation

$$M\{(u_j^{n+1} - u_j^n) + \Delta t\,[(uZ_{xu})_j^* + (vZ_{yu})_j^* - f_j v_j^*]\} = \Delta t \bar{K}_{21} \tag{41}$$

and in a similar manner we obtain the modified v-momentum equation

$$M\{(v_j^{n+1} - v_j^n) + \Delta t\,[(vZ_{xv})_j^* + (u^{n+1}Z_{xv})_j + f_j u_j^{n+1}]\} = \Delta t \bar{K}_{11} \tag{42}$$

where

$$\bar{K}_{21} = \frac{1}{2}(K_{21}^{n+1} + K_{21}^n); \quad \bar{K}_{11} = \frac{1}{2}(K_{11}^{n+1} + K_{11}^n). \tag{43}$$

Compared to the single-stage Galerkin finite-element method we observe that Equations (41) and (42) result in a computational economy as the mass-matrix M is time-independent and is calculated only once. Thus, the solution process is simplified compared to the single-stage Galerkin where we have to solve the matrix equations

$$\left(M + \frac{\Delta t}{2} K_2^*\right)(u_j^{n+1} - u_j^n) = \Delta t(\bar{K}_{21} + P_2 + K_2^* u_j^n). \tag{44}$$

PROGRAM FEUDX

Grid geometry

In our situation a cylindrical channel is used simulating a latitude belt around the earth (see also Hinsman, 1975; Kelley and Williams, 1976; Older, 1981; and Woodward, 1981). We have north-south walls and we use triangular right-angled elements to subdivide our domains. Cyclic continuity is assumed in the x direction to simulate flow around the earth. In the y direction we have the north-south walls; see also discussion about implementation of the boundary-conditions.

Initial conditions and test problem

The test problem used is one for the nonlinear shallow-water equations in a channel on the rotating earth, that is the initial height field condition No. 1 used by Grammeltvedt (1969), which has been tested by different researchers (Cullen and Morton, 1980; Gustafsson, 1971; etc.). This initial condition can be written as

$$gh(x, y) = \phi = g\left\{H_0 + H_1 \tan h\left(\frac{9(D/2 - y)}{2D}\right) + H_2 \sec h^2\left(\frac{9(D/2 - y)}{D}\right)\frac{\sin 2\pi x}{L}\right\}. \tag{45}$$

The initial velocity fields were derived from the initial field via the geostrophic relationship, that is

$$fu = -\phi_y \quad \text{or} \quad u = \left(\frac{-g}{f}\right)\frac{\partial h}{\partial y},$$
$$fv = \phi_x \quad \text{or} \quad v = \left(\frac{g}{f}\right)\frac{\partial h}{\partial x}. \tag{46}$$

The constants used were:

$$\begin{aligned} L &= 6000\,\text{km}, & g &= 10\,\text{m sec}^{-2} \\ D &= 4400\,\text{km}, & H_0 &= 2000\,\text{m}, \\ f &= 10^{-4}\,\text{sec}^{-1}, & H_1 &= -220\,\text{m}, \\ \beta &= 1.5 \times 10^{-11}\,\text{sec}^{-1}\,\text{m}^{-1}, & H_2 &= 133\,\text{m}. \end{aligned} \tag{47}$$

The time and space increments

$$\begin{aligned} h = \Delta x = \Delta y &= 400\,\text{km} \\ \Delta t &= 1800\,\text{sec} \end{aligned} \tag{48}$$

(For stability considerations see Navon, 1979). The model was tested mainly on a (15 × 12) regular grid domain, but the code allows for any resolution.

Computer implementation

Program philosophy and architecture. The program is modular and is complemented with easily reachable switches controlling print and plot options and the display of intermediate results. The program is documented amply, the function of each module being described in a short phrase.

Input specifications. The input to the program consists of a single data card of format (F5.0,515) containing the following six parameters:

DT—the time step in seconds.

NLIMIT—total number of time-steps.

MF—a parameter controlling output operations of the program, that is specifying that after MF time steps subroutine OUT is to be called.

NOUTU—a parameter controlling printout of the u-component of the velocity field. If NOUTU = 0 the u-velocity field is printed by subroutine OUT.

NOUTV—a parameter controlling printout of the v-component of the velocity field. If NOUTV = 0 no printout is obtained, and if NOUTV $\neq$ 0 the v-velocity is printed by subroutine OUT.

NPRINT—if NPRINT $\neq$ 0, the global nodal numbers of each triangular element as well as the indices of all the nonzero entries of the global matrix along with the node coordinates are printed out. If NPRINT = 0, none of the mentioned information is printed out.

The main program FEUDX

The main program initializes all variables and then reads the only data card of the program. It then proceeds to index and label the nodes and the elements, thus setting up the integration domain. This is done by subroutine NUMBER. (We have 16 × 12 = 192 nodes and 330 elements, for the test case.)

Subroutine CORRES determine the nonzero locations in the global matrix and stores them in array LOCAT. The initial fields of height and velocity are set up by subroutine INCOND. The derivatives of the shape functions are calculated in AREAA. A compact storage scheme for the banded and sparse global matrices is implemented in subroutine ASSEM. This method was devised initially by Hinsman (1975) and used by Kelley and Williams (1976), Navon (1979a, 1979b, 1983), as well as by Older (1981) and Woodward (1981; see also Hinsman, Williams, and Woodward, 1982). The method is based on the fact that the maximum number of triangles supporting any node is six. Each row k in the $N \times N$ global matrix represents the equations written as point k and, in the global matrix, each row would have at most seven entries. Thus we have only $(N \times 7)$ nonzero entries. To reduce the $(N \times N)$ global matrix into an $(N \times 7)$ condensed matrix, a correlation address matrix, also of size $(N \times 7)$, storing pointers for each of the seven points involved in any row of the global matrix, has to be saved also (for details, see Hinsman, 1975 or Woodward, 1981).

Four different types of element matrices (3 × 3) will be required for assembly in the global matrices.

$$\text{(a)}\quad M = \iint_A V_i V_j \mathrm{d}A = \frac{A}{12}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{49}$$

where A is the area of the triangular element.

$$\text{(b)}\quad \iint_A V_K \frac{\partial V_i}{\partial x}\mathrm{d}A = \iint_A V_k \frac{b_i}{2A}\mathrm{d}A = \frac{1}{6}\begin{pmatrix} b_1 & b_2 & b_3 \\ b_1 & b_2 & b_3 \\ b_1 & b_2 & b_3 \end{pmatrix} \tag{50}$$

$$b_i = x_k - x_j \tag{51}$$

where x_k and x_j are the Cartesian coordinates for a given triangle.

$$\text{(c)}\quad \iint V_K p_j \frac{\partial V_j}{\partial x} V_i \mathrm{d}A$$

where p_j stands for either u_j, v_j, or ϕ_j.

Then

$$\begin{aligned} \iint V_K p_j \frac{\partial V_j}{\partial x} V_i \mathrm{d}A &= \iint V_K p_j \frac{b_j}{2A} V_i \mathrm{d}A \\ &= \frac{1}{2A} p_j b_j \iint V_i V_K \mathrm{d}A \\ &= \frac{1}{2A} \sum p_j b_j \frac{A}{12}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \\ &= \frac{1}{24} \sum p_j b_j \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \end{aligned} \tag{52}$$

where p_j stands for either u_j, v_j or ϕ_j.

$$\text{(d)}\quad \iint_A V_j p_j V_K \frac{\partial V_i}{\partial x}\mathrm{d}A = \iint_A V_j p_j V_K \frac{b_i}{2A}\mathrm{d}A$$

$$= \frac{b_i}{2A}\left(\sum_{\substack{j=1 \\ j\neq k}} \frac{A}{12} p_i + \frac{A}{6} p_K\right) = \frac{1}{24}$$

$$\times \begin{bmatrix} (2p_1 + p_2 + p_3)b_1 & (2p_1 + p_2 + p_3)b_2 & (2p_1 + p_2 + p_3)b_3 \\ (p_1 + 2p_2 + p_3)b_1 & (p_1 + 2p_2 + p_3)b_2 & (p_1 + 2p_2 + p_3)b_3 \\ (p_1 + p_2 + 2p_3)b_1 & (p_1 + p_2 + 2p_3)b_2 & (p_1 + p_2 + 2p_3)b_3 \end{bmatrix}. \tag{53}$$

A switch, denoted NSWITCH is set for selecting between the different types of element matrices. After setting up the time independent global matrices the program proceeds to the main do-loop which performs the time-integration and which is executed once for every new time-step.

As the solution of the nonlinear constrained optimization problem of enforcing conservation of the nonlinear integral invariants requires scaling of the variables so that the scaled variables should be of some order of magnitude and order unity in the region of interest, the scaling is performed in the main program as well as in subroutine INCOND.

The scaling also should result in the nonlinear equality constraints being of the same order of magnitude to avoid one constraint dominating the others.

In the main integration loop the simulation time is set up and adjusted and then the subroutines ASSEM

and MAMULT set up and assemble the global matrices which then are added up in a matrix equation, first for the continuity Equation (10) and in a similar manner for the u and v-momentum Equations (41) and (42).

Subroutine SOLVER then is called to solve the resulting system of linear equations (of block tridiagonal form) either by a Gauss–Seidel or S.O.R. method, with different parameters for the continuity and the u and v-momentum equations.

The new field values for the geopotential and velocities, ϕ_{ij}^{n+1}, u_{ij}^{n+1}, v_{ij}^{n+1} respectively, are used immediately as obtained in solving the coupled shallow-water equations system. For the u and v-momentum equations, the new two-stage Numerov–Galerkin scheme is implemented. Separate routines are set up for the x and y-derivatives advection terms, DX and DY respectively. The role of these subroutines will be discussed in detail when individual subroutines are described.

The boundary conditions are implemented by subroutine BOUND. Periodically, a Schuman filtering procedure is implemented for the v-component of velocity only, by calling subroutine SMOOTH. The integral invariants are calculated at each time-step by calling subroutine LOOK.

If the variations in the integral invariants exceed the allowable limits δ_E, δ_H, or δ_Z, the Augmented-Lagrangian nonlinear constrained optimization procedure is activated. The unconstrained optimization uses the conjugate-gradient subroutine E04DBF of the NAG(1982) scientific library, which determines an unconstrained minimum of a function of several variables, using first derivatives, by the conjugate-gradient method.

One sets up an Augmented-Lagrangian function, with initial multipliers UH, UZ, and UE corresponding to the constraints of total mass, potential enstrophy, and total energy, respectively; also initial penalties PNLTH, PNLTZ, and PNLTE are set up. An initial value for the parameter ETA (scaled) is also set up.

An array XC contains the variables before the constrained minimization adjustment. The NAG library conjugate-gradient unconstrained minimizer solver E04DBF is used to minimize the Augmented-Lagrangian. The unconstrained minimization is considered to be accomplished once a threshold accuracy dependent on ETA is attained.

Thereafter, the Lagrange multipliers, penalties and the parameter ETA are updated (see Navon and de Villiers, 1983) and another cycle of Augmented-Lagrangian minimization is carried out.

The process is set to stop either when the nonlinear equality constraints are satisfied within a preset accuracy or when ETA becomes too small, that is when the number of Augmented-Lagrangian minimization cycles exceeds a preset limit. Practically 4–5 Augmented-Lagrangian minimization cycles were determined to be sufficient.

Subroutine E04DBF calls a user-supplied subroutine FUNCT which evaluates the function value and its gradient vector as well as subroutine MONIT whose purpose is merely to print out different minimization parameters. After a predetermined number of steps, subroutine OUT is called, which in turn calls upon the subroutines LOOK and MAPPA to calculate the integral invariants and to produce a line-printer plot of height or of velocity fields contours.

Detailed description of the various subroutines

Subroutine INCOND(PSI,U,V,H,F,NODE, NROW,NCOL). Subroutine INCOND sets up the initial height field and geopotential field and then using Equation (45), calculates the initial velocity field components u and v as well as the Coriolis parameter f. It prints out the initial fields and calls upon LOOK and MAPPA to generate initial-time integral invariants as well as a lineprinter plot of the geopotential field.

Parameters of INCOND:

PSI—real array of dimension NODE (geopotential).

U,V—real arrays of dimension NODE. Contain on exit the u and v wind components.

F—real array of dimension NODE. Contains on exit the Coriolis parameter.

NODE—integer-total number of nodes (180).

NROW—integer, number of nodes-1 in x direction.

NCOL—integer, number of nodes-1 in the y-direction.

Subroutine NUMBER (NPRINT). Subroutine NUMBER labels the elements in a direction selected to minimize the maximum bandwidth, numbers the nodes, and stores the global node numbers of the three vertices of each triangle (element). In addition, the global coordinates of each node are calculated and the cyclic boundary conditions implemented. IF NPRINT = 0 no printout occurs. Otherwise the array NOD which stores the global node numbers of each element along with the arrays X and Y which contain the coordinates of all the nodes is printed.

Subroutine AREAA This subroutine calculates the x and y-derivatives of the shape functions and stores them in the arrays AT and BT of dimension (3,300). There are 330 elements (15 × 11 × 2). It also calculates the area of the triangular element.

Subroutine BOUND (LEFT,RIGHT,BX,BY, NBX,NBY,NODE,JC,JR). This subroutine implements the boundary conditions (as detailed in the section on Boundary Condition Implementation), after the system of linear equations AX = R has been obtained.

The parameters of subroutine BOUND are:

LEFT—real array of dimension (7,NODE). On entry LEFT contains the nonzero elements of matrix A. On exit it contains the modified matrix A through implementation of the boundary conditions.

RIGHT—real array of dimension (NODE). On entry contains vector R. On exit R is modified according to the boundary conditions.

BX,BY—real arrays of dimensions (2,JC), (2,JR) respectively containing the boundary values.

NBX,NBY—integer arrays of dimensions (2,JC), (2,JR) containing the numbers of boundary nodes.

NODE—already defined.

JC,JR—number of boundary nodes in the x and y-directions respectively.

Subroutine OUT (U,V,PSI,JC,JR,NODE, NOUTU,NOUTV,TIME,NTIME,F). Subroutine OUT is an output routine which, when called upon, prints out the height field as well as the U and V fields. OUT calls subroutines MAPPA and LOOK.

Parameters of OUT (not defined previously):

NOUTU,NOUTV—integers specifying printout options for U and V fields.

TIME—real, the real time.

NTIME—integer, specifies number of time-steps.

Subroutine SOLVER (COMA,RIGHT,XSOLV, NODE,EPS,ITERMAX). This subroutine is dedicated to the solution of the resulting system of linear algebraic equations of the form

$$AX = R \tag{54}$$

by iterative methods. Two versions are available. One using a Gauss–Seidel iterative method whereas the other uses a successive under-relaxation method with different relaxation parameters for the systems resulting from the continuity and the u and v-momentum equations respectively. An under-relaxation was determined to perform best in the situation.

The parameters are:

COMA—a real array of dimension (7,NODE). On entry the array COMA contains the nonzero entries of the matrix A. Unchanged on exit.

RIGHT—real array of dimension (NODE). On entry the array RIGHT will contain the elements of the right-hand side vector R. Unchanged on exit.

NODE—number of nodes which is also the number of equations.

XSOLV—real array of dimension NODE. On entry XSOLV contains a first guess for the solution vector. On exit XSOLV contains the solution vector obtained by the iterative procedure.

EPS—relative error controlling the iterative process, that is if

$$\frac{\|x^{(k)} - x^{(k-1)}\|}{\|x^{(k)}\|} < \text{EPS} \tag{55}$$

we stop successfully the iterations, the procedure has been completed successfully.

ITERAMAX—maximal number of iterations. If convergence has not been reached after ITERAMAX iterations the procedure is completed unsuccessfully and it will print "NO CONVERGENCE".

Subroutine MAMULT (COMA,VECTOR, RIGHT,NODE). Subroutine MAMULT multiplies a matrix stored in compact form (i.e., only nonzero entries) by a vector V. Here COMA contains the real array of the compact matrix of dimension (7,NODE).

VECTOR—real array of dimension NODE.

RIGHT—real array of dimension NODE. On exit it will contain the product R. (COMA·V = R).

Subroutine ASSEM (COMA,STI,NODE,NELE, NSWITCH,PSIUV,CODI,AREA). Subroutine ASSEM assembles the local 3 × 3 element matrices for each element of the domain and stores the nonzero coefficients in compact form in the global matrix COMA. Here NODE is the integer number of nodes and NELE the integer number of elements.

NSWITCH—is an integer which determines which type of local element matrix must be used for the assembly into a global matrix. [Type (a), (b), (c), or (d) following Eqns. (49)–(53)].

STI—a real array of dimension (3,3) serving as working space used for generating the different (3 × 3) element matrices.

PSIUV—a real array of dimension (NODE). On entry contains the values of either U, V, or PSI fields depending upon the type of global matrix to be assembled. Unchanged on exit.

CODI—is a real array of dimension (3,NELE) containing on entry the x or y derivative of the shape functions. Unchanged on exit.

ASSEM also uses the array LOCAT of dimension (6,NODE) which is a pointer-address matrix containing all the indices for nonzero elements in the global matrix. For each node LOCAT is giving a connectivity list of the adjacent nodes plus the node itself. In total, 7 for interior nodes or 5 for boundary nodes. For each triangular element, ASSEM is searching for a correspondence between the 3 node numbers of the element and the nodes connectivity list to determine the position in the global matrix where the contribution is to be added.

Subroutine CORRES(NPRINT). Subroutine CORRES locates all the nonzero locations of the global matrix—by establishing the connectivity list —and stores these indices in the array LOCAT of dimension (6,NODE).

The parameter NPRINT when different from zero will cause the array LOCAT to be printed (see example of such a printout).

Subroutine MAPPA (PSI,C,NX,NZ,G). This subroutine provides a visual display of the height (geopotential) field by lineprinting an isoline contour plot of the height field for every 50 m. The parameter PSI is the forecast field to be contoured, whereas the parameter C is the inverse of the contour interval multiplied in this instance by the relevant scaling factors. Here NX is the number of nodes + 1 in the x direction and NZ is the number of nodes + 1 in the y direction.

Subroutine LOOK (UU,VV,PSIPSI,NX,NY, TIME,NTIME,G,NODE,F). This subroutine calculates, at each time-step it is called the total energy, potential enstrophy and mean height (total mass) which are the integral invariants of the shallow-water

1ELEMENT NUMBER	NUMBER OF NODES		
1	1	13	2
2	14	2	13
3	2	14	3
4	15	3	14
5	3	15	4
6	16	4	15
7	4	16	5
8	17	5	16
9	5	17	6
10	18	6	17
11	6	18	7
12	19	7	18
13	7	19	8
14	20	8	19
15	8	20	9
16	21	9	20
17	9	21	10
18	22	10	21
19	10	22	11
20	23	11	22
21	11	23	12
22	24	12	23
23	13	25	14
24	26	14	25
25	14	26	15
26	27	15	26
27	15	27	16
28	28	16	27
29	16	28	17
30	29	17	28
31	17	29	18
32	30	18	29
33	18	30	19
34	31	19	30
35	19	31	20
36	32	20	31
37	20	32	21
38	33	21	32
39	21	33	22
40	34	22	33
41	22	34	23
42	35	23	34
43	23	35	24
44	36	24	35
45	25	37	26
46	38	26	37
47	26	38	27
48	39	27	38
49	27	39	28
50	40	28	39
51	28	40	29
52	41	29	40
53	29	41	30
54	42	30	41
55	30	42	31
56	43	31	42
57	31	43	32
58	44	32	43
59	32	44	33
60	45	33	44
61	33	45	34
62	46	34	45
63	34	46	35
64	47	35	46
65	35	47	36

Figure 1. Typical element and nodes numbering for triangular finite element on rectangular domain.

1 NODE	NEIGHBOURING NODES					
1	13	2	169	170	0	0
2	1	13	14	3	170	171
3	2	14	15	4	171	172
4	3	15	16	5	172	173
5	4	16	17	6	173	174
6	5	17	18	7	174	175
7	6	18	19	8	175	176
8	7	19	20	9	176	177
9	8	20	21	10	177	178
10	9	21	22	11	178	179
11	10	22	23	12	179	180
12	11	23	24	180	0	0
13	1	2	14	25	0	0
14	2	13	3	15	25	26
15	3	14	4	16	26	27
16	4	15	5	17	27	28
17	5	16	6	18	28	29
18	6	17	7	19	29	30
19	7	18	8	20	30	31
20	8	19	9	21	31	32
21	9	20	10	22	32	33
22	10	21	11	23	33	34
23	11	22	12	24	34	35
24	12	23	35	36	0	0
25	13	14	26	37	0	0
26	14	25	15	27	37	38
27	15	26	16	28	38	39
28	16	27	17	29	39	40
29	17	28	18	30	40	41
30	18	29	19	31	41	42
31	19	30	20	32	42	43
32	20	31	21	33	43	44
33	21	32	22	34	44	45
34	22	33	23	35	45	46
35	23	34	24	36	46	47
36	24	35	47	48	0	0
37	25	26	38	49	0	0
38	26	37	27	39	49	50
39	27	38	28	40	50	51
40	28	39	29	41	51	52
41	29	40	30	42	52	53
42	30	41	31	43	53	54
43	31	42	32	44	54	55
44	32	43	33	45	55	56
45	33	44	34	46	56	57
46	34	45	35	47	57	58
47	35	46	36	48	58	59
48	36	47	59	60	0	0
49	37	38	50	61	0	0
50	38	49	39	51	61	62
51	39	50	40	52	62	63
52	40	51	41	53	63	64
53	41	52	42	54	64	65
54	42	53	43	55	65	66
55	43	54	44	56	66	67
56	44	55	45	57	67	68
57	45	56	46	58	68	69
58	46	57	47	59	69	70
59	47	58	48	60	70	71
60	48	59	71	72	0	0
61	49	50	62	73	0	0
62	50	61	51	63	73	74
63	51	62	52	64	74	75
64	52	63	53	65	75	76
65	53	64	54	66	76	77
66	54	65	55	67	77	78
67	55	66	56	68	78	79
68	56	67	57	69	79	80
69	57	68	58	70	80	81
70	58	69	59	71	81	82

Figure 2. Neighboring nodes for each node, that is triangles supporting given node.

equations. It also can calculate CPU-time spent between a given number of time-steps. Here UU, VV, and PSIPSI stand for the velocity field components and the geopotential field respectively.

Subroutine DX (R,S,A,NODE,NR,NC,DISTX). Subroutine DX implements the two-stage Numerov–Galerkin algorithm described previously for the advective terms in the u and v-momentum equations involving the x-derivative.

In the first stage it calculates the $O(h^8)$ accurate generalized-spline approximation to the $(\partial u/\partial x)$ first derivative by calling upon subroutine CYCPNT which solves a periodic pentadiagonal system of linear equations generated by the spline approximation.

In the second stage it implements the second part of the Numerov–Galerkin algorithm for the nonlinear advective term $u(\partial u/\partial x)$ and solves a cyclic tridiagonal system [Eqn. (39)] by calling upon subroutine CYCTRD. The final result is returned in the array R(192).

Subroutine PENTDG (U,F,NX). This subroutine solves a pentadiagonal system of linear equations of the form:

$$A*U(I-2) + B*U(I-1) + C*U(I) + D*U(I+1) + E*U(I+2)$$

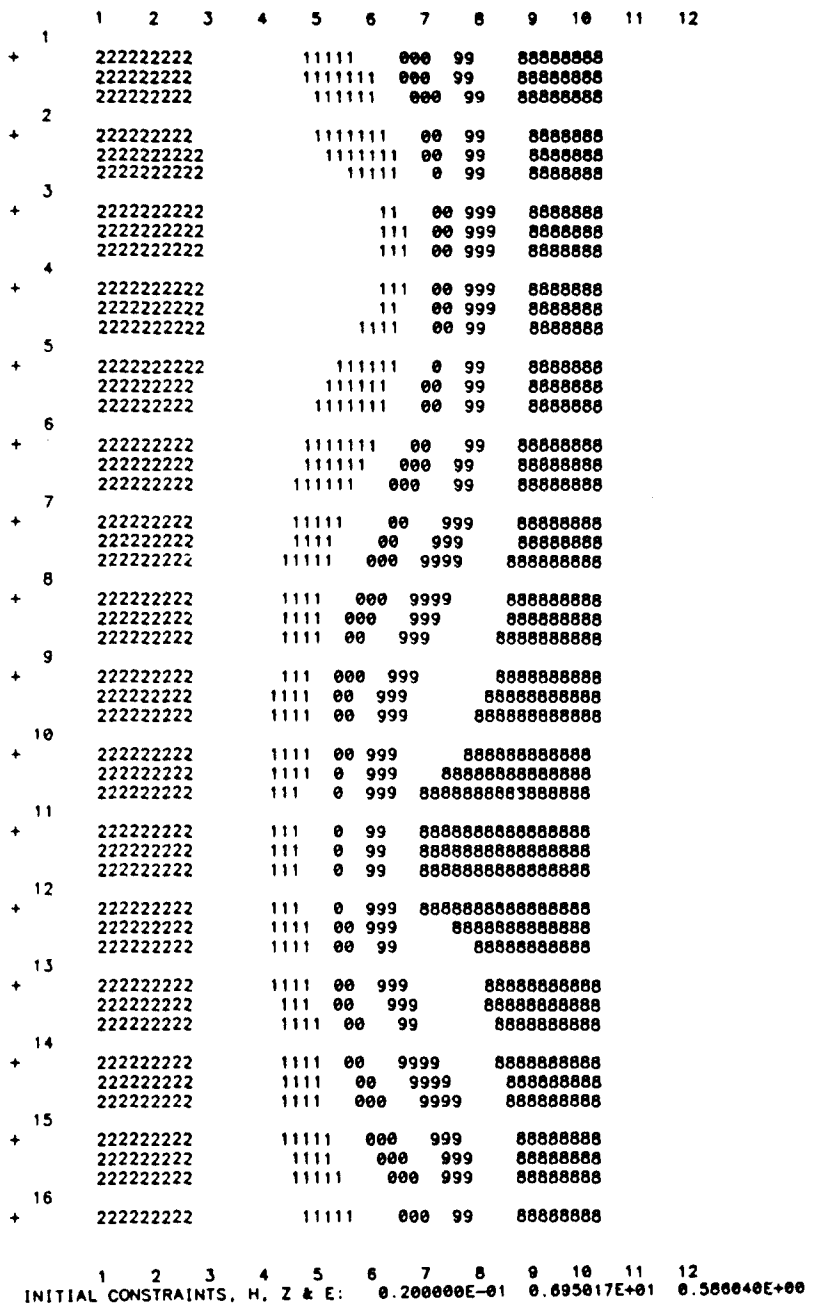

Figure 3. Initial height field using space resolution of $\Delta x = \Delta y = 400$ km, contoured in intervals of 50 m, from 1800 to 2200 m. Scaled values of total mass (H), potential enstropy (Z), and total energy (E) also are displayed. Grid of 16 × 12 points was used.

$$= F(I) \qquad 1 \leqslant I \leqslant NX. \qquad (56)$$

This is a utility subroutine.

Subroutine CYCPNT (Z,Z,NX). This routine solves a periodic pentadiagonal matrix (resulting from the periodic boundary conditions in one coordinate direction). The method used extends an algorithm due to Ahlberg, Nielson, and Walsh (1967) and it uses subroutine PENTDG for solving part of this algorithm.

Subroutine NCTRD (U,D,NX). This is a specialized tridiagonal solver routine which implements the second stage of the Numerov–Galerkin method [Eq. (39)] for the $(\partial u/\partial y)$ derivative (i.e., the one which has no periodic boundary conditions) in subroutine DY.

Subroutine CYCTRD (D,Z,NX). This subroutine is a cyclic tridiagonal solver implementing the second stage of the Numerov–Galerkin method for the $(\partial u/\partial x)$ derivative, for which we have periodic boundary conditions. It is called by subroutine DX and it needs a tridiagonal solver obtained by calling subroutine TRIDG. The method of solution is based on the algorithm of Ahlberg, Nielson, and Walsh (1967).

Subroutine TRIDG (U,D,NX). This subroutine is

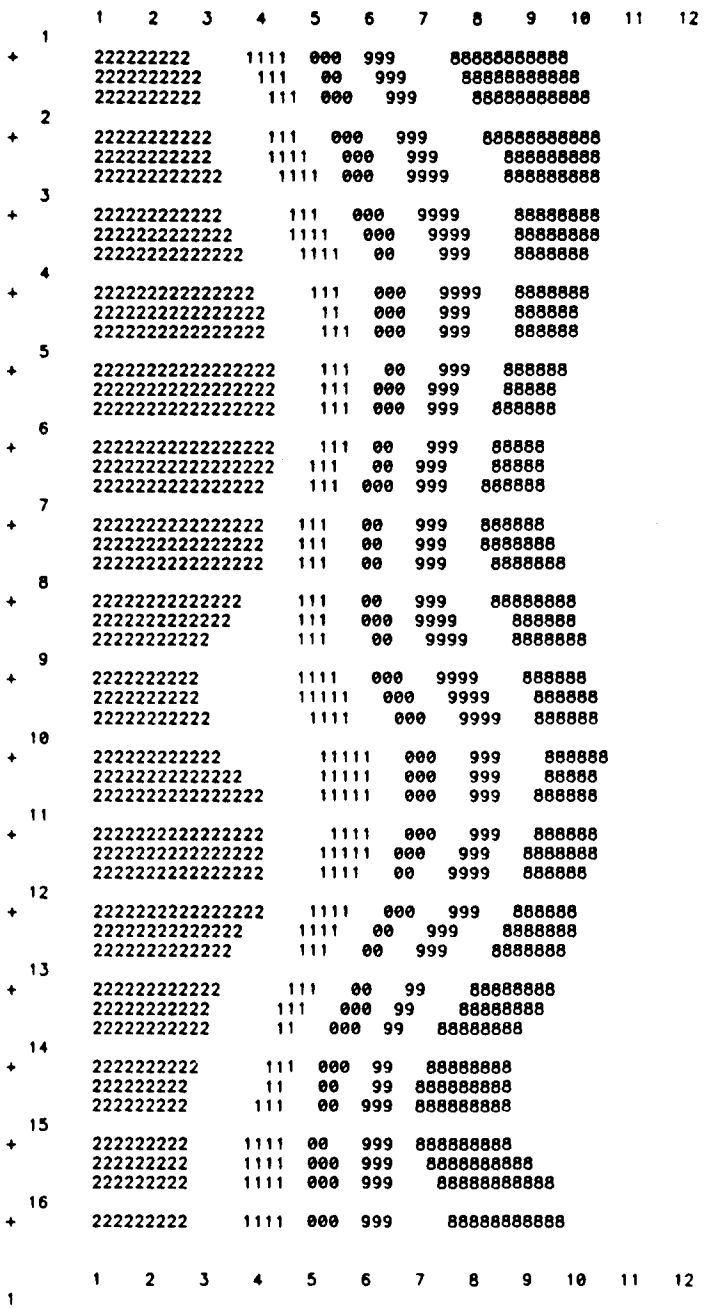

Figure 4. Five-day forecast of height field using numerical integration with Numerov–Galerkin finite-element shallow-water equations solver, depicted by isopleths drawn at 50 m intervals. Domain is covered by 16 × 12 grid.

the usual tridiagonal solver based on the familiar Thomas algorithm for tridiagonal matrices. It is called by subroutine CYCTRD.

Subroutine DY (R,S,A,NODE,NR,NC,DISTY). Subroutine DY implements the two-stage Numerov–Galerkin algorithm described previously for the advective terms in the u and v-momentum equations involving the y-derivative. In its first stage it calculates the $O(h^8)$ accurate generalized-spline approximation to the $(\partial u/\partial y)$ first derivative by calling upon subroutine PENTDG which solves the usual pentadiagonal system of linear equations generated by the generalized-spline approximation.

In the second stage subroutine DY implements the second part of the Numerov–Galerkin algorithm for the nonlinear advective term $u(\partial u/\partial y)$ and solves the Galerkin product by calling upon subroutine NCTRD to solve a special tridiagonal system. The final result is stored in array R(192).

Subroutine FUNCT (N,XC,FC,GC). This is a user-supplied routine used in conjunction with the NAG (1982) Scientific Library routine E04DBF, which is a conjugate-gradient function minimization subroutine based on the Fletcher and Reeves (1964) method. Subroutine FUNCT calculates the function to be minimized which is the Augmented-Lagrangian

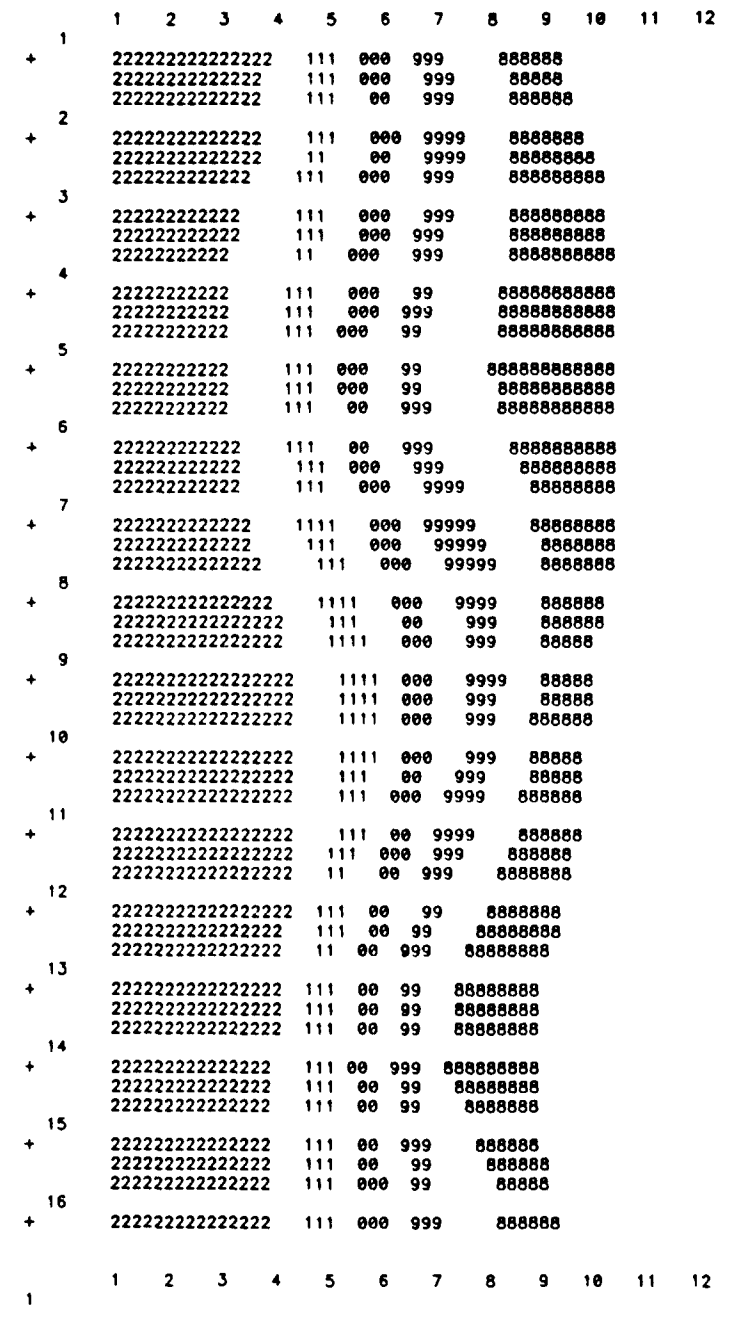

Figure 5. Ten-day forecast of height field using numerical integration with Numerov–Galerkin finite-element shallow-water equations solver, depicted by isopleths drawn at 50 m intervals. Domain is covered by 16 × 12 grid.

and stores it in array FC of dimension (3*NODE). The vector x is also of length 3*NODE and is stored in array XC, whereas the gradient of the Augmented-Lagrangian with respect to the variables u, v, and h is stored in array GC also of dimension 3*NODE.

Subroutine MONIT (N,SC,FC,GC,NCALL, TEST). This is a user-supplied monitoring routine used in conjunction with the NAG (1982) Scientific Library routine E04DBF, which is a conjugate-gradient function minimization subroutine. Subroutine MONIT tests the peformance of the conjugate-gradient routine, monitors the number of conjugate-gradient iterations (NCALL), the norm of the gradient and performs some tests for the Augmented-Lagrangian constrained minimization.

Subroutine SMOOTH (ZK,TEM,NX,NY). Subroutine SMOOTH implements a low-pass filter designed to filter out short wavelength components. It is applied periodically only to the v-momentum equation (every 12h) and follows the method of filtering of Shuman (1957). The smoothing is applied sequentially in the two coordinate directions x and y.

Examples of output

Examples of the FEUDX output are provided in order to illustrate the different options of the pro-

```
SCALED TIME:  6552.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694326E+01  0.586100E+00
SCALED TIME:  6570.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694327E+01  0.586107E+00
SCALED TIME:  6588.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694343E+01  0.586115E+00
SCALED TIME:  6606.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694368E+01  0.586121E+00
SCALED TIME:  6624.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694395E+01  0.586124E+00
SCALED TIME:  6642.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694420E+01  0.586125E+00
SCALED TIME:  6660.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694442E+01  0.586126E+00
SCALED TIME:  6678.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694460E+01  0.586129E+00
SCALED TIME:  6696.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694341E+01  0.586128E+00
SCALED TIME:  6714.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694374E+01  0.586132E+00
SCALED TIME:  6732.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694386E+01  0.586137E+00
SCALED TIME:  6750.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694367E+01  0.586139E+00
SCALED TIME:  6768.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694333E+01  0.586140E+00
SCALED TIME:  6786.00 CONSTRAINTS, H, Z & E:   0.199769E-01  0.694297E+01  0.586140E+00
ETA =   0.100000E+03
R         , H, Z & E:   0.000000E+00  0.500000E+00  0.500000E+00
U         , H, Z & E:   0.000000E+00  0.000000E+00  0.000000E+00
PENALTIES, H, Z & E:   0.000000E+00  0.100000E+01  0.100000E+01
SUMSQ AT CALL     1 =   0.0000E+00 , GNORM =   0.5324E+00
ERRORS    , H, Z & E :  -0.231449E-04 -0.720750E-02  0.100485E-03
NORM =   0.720820E-02          ETA * NORM =   0.720820E+00          MIN. VALUE =   0.720820E-01
SUMSQ AT CALL     3 =   0.1645E-05 , GNORM =   0.6093E-02
ERRORS    , H, Z & E :  -0.344887E-04 -0.275201E-03 -0.466997E-03
NORM =   0.542054E-03          ETA * NORM =   0.542054E-01          MIN. VALUE =   0.720820E+00
SCALED TIME:  6786.00 CONSTRAINTS, H, Z & E:   0.199655E-01  0.694990E+01  0.585573E+00
ETA =   0.800000E+02
R         , H, Z & E:   0.000000E+00  0.500000E+00  0.200000E+00
U         , H, Z & E:   0.000000E+00 -0.550402E-03 -0.233499E-02
PENALTIES, H, Z & E:   0.000000E+00  0.100000E+01  0.250000E+01
SUMSQ AT CALL     1 =   0.0000E+00 , GNORM =   0.3030E-01
ERRORS    , H, Z & E :  -0.344887E-04 -0.275201E-03 -0.466997E-03
NORM =   0.542054E-03          ETA * NORM =   0.433643E-01          MIN. VALUE =   0.720820E-01
SUMSQ AT CALL     3 =   0.8307E-08 , GNORM =   0.1619E-01
ERRORS    , H, Z & E :  -0.349230E-04  0.171473E-03 -0.483011E-03
NORM =   0.512545E-03          ETA * NORM =   0.410036E-01          MIN. VALUE =   0.720820E-01
SCALED TIME:  6786.00 CONSTRAINTS, H, Z & E:   0.199651E-01  0.695035E+01  0.585557E+00
ETA =   0.640000E+02
R         , H, Z & E:   0.000000E+00  0.200000E+00  0.800000E-01
U         , H, Z & E:   0.000000E+00  0.306962E-03 -0.837263E-02
PENALTIES, H, Z & E:   0.000000E+00  0.250000E+01  0.625000E+01
SUMSQ AT CALL     1 =   0.0000E+00 , GNORM =   0.9742E-01
ERRORS    , H, Z & E :  -0.349230E-04  0.171473E-03 -0.483011E-03
NORM =   0.512545E-03          ETA * NORM =   0.328029E-01          MIN. VALUE =   0.720820E-02
SUMSQ AT CALL     3 =   0.1441E-07 , GNORM =   0.4637E-01
ERRORS    , H, Z & E :  -0.336556E-04 -0.389548E-03 -0.412873E-03
NORM =   0.567637E-03          ETA * NORM =   0.363287E-01          MIN. VALUE =   0.720820E-02
SUMSQ AT CALL     5 =   0.1962E-05 , GNORM =   0.4659E-02
ERRORS    , H, Z & E :  -0.252567E-04 -0.149195E-03  0.177256E-03
NORM =   0.231687E-03          ETA * NORM =   0.148280E-01          MIN. VALUE =   0.720820E-02
SCALED TIME:  6786.00 CONSTRAINTS, H, Z & E:   0.199747E-01  0.695002E+01  0.586217E+00
ETA =   0.512000E+02
R         , H, Z & E:   0.000000E+00  0.800000E-01  0.320000E-01
U         , H, Z & E:   0.000000E+00 -0.155797E-02 -0.283337E-02
PENALTIES, H, Z & E:   0.000000E+00  0.625000E+01  0.156250E+02
SUMSQ AT CALL     1 =   0.0000E+00 , GNORM =   0.1350E+00
ERRORS    , H, Z & E :  -0.252567E-04 -0.149195E-03  0.177256E-03
NORM =   0.231687E-03          ETA * NORM =   0.118624E-01          MIN. VALUE =   0.148280E-02
SUMSQ AT CALL     4 =   0.2820E-08 , GNORM =   0.5921E-02
ERRORS    , H, Z & E :  -0.257444E-04  0.136466E-03  0.152461E-03
NORM =   0.204615E-03          ETA * NORM =   0.104763E-01          MIN. VALUE =   0.148280E-02
SCALED TIME:  6786.00 CONSTRAINTS, H, Z & E:   0.199743E-01  0.695031E+01  0.586192E+00
```

Figure 6. Typical output from Augmented-Lagrangian nonlinear constrained optimization, after detecting variation in potential enstrophy (Z). After four iterations, value of Z is restored to value of Z_0, initial enstrophy.

gram. Figure 1 shows the element numbering and the nodes numbering. In Figure 2, for each given node (grid point) the six neighboring nodes which constitute the triangles supporting a given node are shown.

The initial height field, using a space resolution of $\Delta x = \Delta y = 400$ km is shown in Figure 3. It is contoured in intervals of 50 m from 1800 to 2200 m. The values of the initial integral invariants of total mass (H), potential enstrophy (Z), and total energy (E) also are printed out. Figures 4 and 5 show the height field contours after 5 and 10 days of numerical integration with a time-step of 1800 sec.

Figure 6 shows the typical output from an Augmented-Lagrangian nonlinear constrained optimization which entered into action after detecting a variation in the potential enstrophy invariant which exceeded the allowable error δ_Z. At each Augmented-Lagrangian minimization cycle the penalties R and the modified penalties "PENALTIES" along with "U"—the Lagrange multipliers and the norm of the gradient of the Augmented-Lagrangian L, ∇L, are printed out. ETA, the scaled variable accuracy parameter changes from one Augmented-Lagrangian minimization iteration to another following the formula.

$$(ETA)^{k+1} = (ETA)^k \cdot 0.8$$

four such iterations were necessary.

Note that after four iterations in the Augmented-Lagrangian minimization the value of Z, the potential enstrophy has been restored to the initial value Z_0

with negligible changes in the values of H and E, the total mass and total energy integral invariants, respectively. The modified values of the fields of velocity and height are used for subsequent integration.

Acknowledgments—It is a pleasure to acknowledge the brilliant programming help of Mrs Rosalie de Villiers of the National Research Institute for Mathematical Sciences, Council of Scientific and Industrial Research, Pretoria, South Africa, where part of this work was done. Dr. G. Beckwith, Control Data Corporation consultant's expert help with implementing FEUDX on the CDC CYBER-205 is acknowledged thankfully. Ms. Mimi Burbank's expert typing rendered my manuscript into a legible paper.

REFERENCES

Ahlberg, H. H., Nilson, E. N., and Walsh, J. L., 1967, The theory of splines and their application: Mathematics in Science and Engineering, v. 38: Academic Press, New York, 289 p.

Brebbia, C. A., and Partridge, P. W., 1976a, Finite-element simulation of water-circulation in the North Sea: Appl. Math. Modell., v. 1, no. 2, p. 101–107.

Brebbia, C. A., and Partridge, P. W., 1976b, Finite-element models for circulation studies, *in* Brebbia, C. A., ed., Mathematical models for environmental problems: Pentech Press, London, p. 141–160.

Connor, J. J., and C. A. Brebbia, 1976, Finite-element techniques for fluid flow: Newnes-Butterworth, Publishers, London and Boston, 310 p.

Crépon, M., Richez, M. C., and Chartier, M., 1984, Effects of coastline geometry on upwellings: Jour. Phys. Ocean., v. 14, p. 365–1382.

Cullen, M. J. P., 1973, A simple finite-element method for meteorological problems: Jour. Inst. Math. Applics., v. 11, p. 15–31.

Cullen, M. J. P., 1974, A finite-element method for a nonlinear initial-value problem: Jour. Inst. Math. Applics., v. 13, p. 233–247.

Cullen, M. J. P., 1979, The finite-element method, *in* Numerical methods used in atmosphere models: WMO/GARP Publ. Ser. no. 17, v. 2, Chapt. 5, World Meteorological Organization, Geneva, Switzerland, p. 330–338.

Cullen, M. J. P., 1982, The use of quadratic finite-element methods and irregular grids in the solution of hyperbolic problems: Jour. Comput. Phys., v. 45, p. 221–245.

Cullen, M. J. P., and Hall, C. D., 1979, Forecasting and general circulation results from finite-element models: Quart. Jour. Roy. Met. Soc., v. 102, p. 571–592.

Cullen, M. J. P., and Morton, K. W., 1980, Analysis of evolutionary error in finite-element and other methods: Jour. Comput. Phys., v. 34, p. 245–267.

Desai, C. S., and Abel, F. J., 1972, Introduction to the finite element method: Van-Nostrand/Reinhold, New York, 477 p.

Douglas, J., and Dupont, T., 1970, Galerkin methods for parabolic problems: SIAM Jour. Numer. Anal., v. 7, p. 575–626.

Fix, G. J., 1975, Finite-element models for ocean-circulation problems: SIAM Jour. Appl. Math, v. 29, p. 371–387.

Fletcher, R., and Reeves, C. M., 1964, Function minimization by conjugate-gradients: Computer Jour., v. 7, p. 149–154.

Foreman, M. G. G., 1983, An analysis of two-step time-discretizations in the solution of the linearized shallow-water equations: Jour. Comput. Phys., v. 51, p. 454–483.

Foreman, M. G. G., 1984, A two-dimensional dispersion analysis of selected methods for solving the linearized shallow-water equations: Jour. Comput. Phys., v. 56, p. 287–323.

Grammeltvedt, A., 1969, A survey of finite-difference schemes for the primitive equations for a barotropic fluid: Mon. Wea. Rev., v. 97, p. 384–404.

Gray, W. R., and Lynch, D. R., 1978, Finite-element simulation of shallow-water equations with moving boundaries, *in* Brebbia, C. A., and others, eds., Proc. 2nd Conf. on Finite-Elements in Water Resources: p. 2.23–2.42.

Gustafsson, B., 1971, An alternating direction implicit method for solving the shallow-water equations: Jour. Comput. Phys., v. 7, p. 239–251.

Hinsman, D. E., 1975, Application of a finite-element method to the barotropic primitive equations: unpubl. masters thesis, Naval Postgraduate School, Monterey, California 107 p.

Hinsman, D. E., and Archer, D., 1976, A finite-element application to a barotropic primitive equation model in spherical coordinates: Proc. 6th Conf. Weather Forecasting and Analysis, Albany, New York, p. 121–124.

Hinsman, D. E., Williams, R. T., and Woodward, E., 1982, Recent advances in the Galerkin finite-element method as applied to the meteorological equations on variable resolution grids, *in* Kaway, T., ed., Finite-element flow-analysis: Univ. Tokyo Press, Tokyo, 999 p.

Hua, L., and Thomasset, F., 1980, Numerical study of coastal upwelling by a finite-element method, *in* Glowinsky, R. and Lions, J. L., eds., Computing methods in applied sciences and engineering: North Holland Publ. Co., Amsterdam, p. 237–258.

Hua, B. L., and Thomasset, F., 1984, A noise-free finite-element scheme for the two-layer shallow-water equations: Tellus, v. 36A, p. 157–165.

Huebner, K. H., 1975, The finite-element method for engineers: John Wiley & Sons, New York, 500 p.

Kawahara, M., 1980, On finite-element methods in shallow-water long-wave flow analysis, *in* Oden, Y. T., ed., Computational methods in nonlinear mechanics: North-Holland Publ. Co., Amsterdam, p. 261–287.

Kelley, R. G., and Williams, R. T., 1976, A finite-element prediction model with variable element sizes: Naval Postgraduate School Report NPS-63WVF6101, 109 p.

Malone, T. D., and Kuo, J. T., 1981, Semi-implicit finite-element methods applied to the solution of the shallow-water equations: Jour. Geophys. Res., v. 86, no. C5, p. 4029–4040.

Morton, K. W., 1977, Initial-value problems by finite-difference and other methods, *in* Jacobs, D., ed., State of the art in numerical analysis: Academic Press, New York, p. 699–756.

NAG, 1982, Numerical Algorithms Group FORTRAN Library Manuals, Vols. 1–6: NAG, Banbury Road, Oxford, OOX2-6HN, England or NAG-INC., 1131 Warren Ave., Downers Grove, Illinois, 70515.

Navon, I. M., 1977, A survey of finite-element methods in quasi-linear fluid flow problems: Wisk Report no. 140, National Research Institute for Mathematical Sciences, Pretoria, South Africa, 44 p.

Navon, I. M., 1979a, Finite-element simulation of the shallow-water equations model on a limited area domain: Appl. Math. Modell., v. 3, p. 337–348.

Navon, I. M., 1979b, Finite-element solution of the shallow-water equations on a limited area domain with three different mass matrix formulations: Proc. 4th Conf. Numer. Wea. Pred., Silver Springs, Maryland, Am. Meteorological Soc., p. 223–227.

Navon, I. M., 1982, A Numerov-Galerkin technique applied to a finite-element shallow-water equations model with exact conservation of integral invariants, *in* Kaway, T., ed. Finite-element flow-analysis: Univ. Tokyo Press, Tokyo, p. 75–86.

Navon, I. M., 1983, A Numerov-Galerkin technique applied to a finite-element shallow-water equations model with enforced conservation of integral invariants and selective lumping: Jour. Comput. Phys., v. 52, no. 2, p. 313–339.

Navon, I. M., and de Villiers, R., 1983, Combined penalty-multiplier optimization methods to enforce integral invariants conservation: Mon. Wea. Rev., v. 111, 1228–1243.

Navon, I. M., and de Villiers, R., 1986, GUSTAF, A Quasi-Newton nonlinear ADI FORTRAN IV program for solving the shallow-water equations with Augmented Lagrangians: Computers & Geosciences, v. 12, no. 2, p. 151–173.

Navon, I. M., and Muller, V., 1979, FESW—a finite-element FORTRAN IV program for solving the shallow-water equations: Adv. Eng. Software, v. 1, no. 2, p. 77–86.

Navon, I. M., and Riphagen, H. A., 1979, An implicit compact fourth-order algorithm for solving the shallow-water equations in conservation law-form: Mon. Wea. Rev., v. 111, p. 1228–1243.

O'Brien, J. J., and Hulburt, H. E., 1972, A numerical model of coastal upwelling: Jour. Phys. Oceanog., v. 2, p. 14–26.

Older, M. E., 1981, A two-dimensional finite-element advection model with variable resolution: unpubl. masters thesis, Naval Postgraduate School, Monterey, California, 84 p.

Praagman, N., 1979, Numerical solution of the shallow-water equations by a finite-element method: unpubl. doctoral dissertation, Delft Univ.

Sasaki, Y. K., and Reddy, J. N., 1980, A comparison of stability and accuracy of some numerical models of two-dimensional circulation: Intern. Jour. Numer. Meth. Eng., v. 16, p. 149–170.

Shuman, F. G., 1957, Numerical methods in weather prediction II, Smoothing and filtering: Mon. Wea. Rev., v. 85, p. 357–361.

Staniforth, A. N., 1982, A review of the application of the finite-element method to meteorological flows, *in* Kaway, T., ed., Finite-element flow-analysis: Univ. Tokyo Press, Tokyo, p. 835–842.

Staniforth, A. N., 1984, The application of the finite-element methods to meteorological simulations—a review: Intern. Jour. Numer. Methods in Fluids, v. 4, p. 1–22.

Staniforth, A. N., and Daley, R. W., 1979, A baroclinic finite-element model for regional forecasting with the primitive equations: Mon. Wea. Rev., v. 107, p. 107–121.

Staniforth, A. N., and Mitchell, H. L., 1977, A semi-implicit finite-element barotropic model: Mon. Wea. Rev., v. 105, p. 154–169.

Swartz, B. K., and Wendroff, B., 1974, The relative efficiency of finite-difference and finite-element methods, I, hyperbolic problems and splines: SIAM Jour. Numer. Anal., v. 11, no. 5, p. 979–993.

Taylor, C., and Davis, J., 1975, Tidal and long-wave propagation, a finite-element approach: Computers and Fluids, v. 3, p. 125–148.

Thacker, W. C., 1978, Comparison of finite-element and finite difference schemes, Part 2, Two-dimensional gravity-wave reaction: Jour. Phys. Oceanogr., v. 8, p. 680–689.

Von-Rosenberg, D. U., 1969, Methods for the numerical solution of partial differential equations: American Elsevier, New York, 127 p.

Wang, H. H., Halpern, P., Douglas, J., Jr., and Dupont, I., 1972, Numerical solutions of the one-dimensional primitive equations using Galerkin approximations with localized basis functions: Mon. Wea. Rev., v. 100, p. 738–746.

Williams, R. T., 1981, On the formulation of finite-element prediction models: Mon. Wea. Rev., v. 109, p. 463–466.

Williams, R. T., and Zienkiewicz, O. C., 1981, Improved finite element forms for shallow-water wave equations: Intern. Jour. Numer. Meth. Fluids, v. 1, p. 81–97.

Woodward, E. T., 1981, Development of improve finite-element formulations for shallow-water equations: unpubl. masters thesis, Naval Postgraduate School, Monterey, California, 168 p.

Zienkiewicz, O. C., 1978, The finite-element method (3rd extended ed.): McGraw Hill Book Co., Maidenhead, 787 p.

Zienkiewicz, O. C., and Heinrich, J. C., 1979, A unified treatment of steady-stage shallow-water equations and two-dimensional Navier-Stokes equations—a finite element penalty function approach: Computer Meth. Appl. Mech. and Eng., v. 17/18, p. 673–688.

Zienkiewicz, O. C., Vilotte, J. P., Nakazawa, S., and Toyoshima, S., 1984, Iterative method for constrained and mixed approximation: an inexpensive improvement of F.E.M. performance: Inst. Numerical Methods in Engineering Report C/R/489/84, Swansea, United Kingdom, 20 p.

APPENDIX

```
      PROGRAM  FEUDX  (INPUT,OUTPUT,TAPE5,TAPE3=OUTPUT,TAPE7,
     *TAPE8)
      COMMON/OLD/XO(540),H0,Z0,E0,ALPHA,BEETA,TG,F(180),NX,NY,EX,EY,
     1PNLTH,PNLTZ,PNLTE,UH,UZ,UE,ETA,EMINN
      COMMON/INVRNT/ HMEAN,ZMEAN,ENERGY
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      COMMON /DIM/ NODE , NELE , NCOL ,NROW
      COMMON /COR/ X(192),Y(192),DISTX,DISTY
      COMMON /AREA/ AT(3,330),BT(3,330),AREA
      COMMON /CONST/HH0,H1,H2,G,FST,BETA,XL,D
      DIMENSION  CHIDX(7,192),CHICHI(7,192),DUMMY(7,192),AD(7,192),
     +  CHIDY(7,192),STI(3,3)
      DIMENSION RIGHT(192),RS(192)  ,LEFT(7,192) ,DIF(192)
      DIMENSION PSI(180),U(180),V(180),USTAR(180),VSTAR(180),UNEW(180),
     +     VNEW(180),PSISTAR(180),H(180)
      DIMENSION BX(2,15),BY(2,15),NBX(2,15),NBY(2,15),R1(180),R2(180)
      DIMENSION WS(540),XTOL(540),XC(540),GC(540)
      DIMENSION HPLT(100),ZPLT(100),EPLT(100),TPLT(100)
      EQUIVALENCE (U(1),XC(1)),(V(1),XC(181)),(PSI(1),XC(361))
      EXTERNAL FUNCT,MONIT
      REAL LEFT
      DATA BX /30*0./
      DATA DIF/192*0./
      DATA XL/6.E6/,D/4.4E6/,DISTX/4.E5/,DISTY/4.E5/
      DATA NELE /330/, NNOD /3/, NODE /180/, NCOL /14/, NROW /11/
      DATA EPS/1.E-6/, ITERMAX /30/
      DATA HH0/2000./,H1/220./,H2/133./
      DATA  G/10./,FST/1.E-4/,BETA/1.5E-11/
      DATA NIN/5/
```

```
      DATA NOUT/3/
      DATA HFAC/1.E5/,UFAC/1.E3/
 2100 FORMAT (1H1,10HCONSTANTS:,4X,2HH0,G14.4,5X,2HF*,G14.4,5X,2HXL,
     +G14.4,5X,5HDISTX,G11.4,/,15X,2HH1,G14.4,5X,4HBETA,G11.4,5X,1HD,
     +G15.4,5X,5HDISTY,G11.4,/,15X,2HH2,G14.4,5X,1HG,G15.5,26X,2HDT,
     +G14.4)
 1000 FORMAT (F5.0,5I5)
      HPLT(1)=1.
      ZPLT(1)=1.
      EPLT(1)=1.
      TPLT(1)=0.
      IT1=1
C
C     READ IN TIMESTEP, NUMBER OF TIMESTEPS AND PRINTOUT OPTIONS
C
      READ (NIN,1000) DT,NLIMIT,MF,NOUTU,NOUTV,NPRINT
      DT=DT*UFAC/HFAC
      NDIM = 30
      ND=48
      DT2 = DT / 2.
      JC = NCOL +1
      JR = NROW + 1
C
C     SET UP THE GRID, NUMBER THE ELEMENTS AND THE NODES
C
      CALL NUMBER (NPRINT)
C
C     FIND NON-ZERO ENTRIES OF THE GLOBAL MATRICES
C
      CALL CORRES (NPRINT)
C
C     SET UP INITIAL FIELDS
C
      DISTX=DISTX/HFAC
      DISTY=DISTY/HFAC
      CALL INCOND (PSI,U,V,H,F,NODE,NROW,NCOL)
      PRINT 299,HMEAN,ZMEAN,ENERGY
 299 FORMAT("0INITIAL CONSTRAINTS, H, Z \& E: ",3E14.6)
      H0=HMEAN
      E0=ENERGY
      Z0=ZMEAN
      ALPHA=1.
      BEETA=1./(G*HMEAN)
      TG=G*G
      EX=DISTX
      EY=DISTY
      NX=NCOL+1
      NY=NROW+1
      EPSH=H0*5.E-3
      EPSZ=Z0*1.E-3
      EPSE=E0*1.E-3
      RH0=0.0
      RZ0=0.5
      RE0=0.5
      N=540
      FEST=0.
      IFAIL=0
C
C     CALCULATE THE DERIVATIVES OF THE SHAPE FUNCTIONS
C
      CALL AREAA
C
C
      NEL = 3 * NELE
      DO 30 K = 1,NEL
   30 NOD(K) = IABS (NOD(K))
      KK = 0
C
C     PRINT OUT CONSTANTS USED
C
      WRITE (NOUT,2100) H0,FST,XL,DISTX,H1,BETA,D,DISTY,H2,G,DT
C
C     SET UP THE INITIAL PSI-, U-, AND V- FIELDS FOR THE DIFFERENT TIM
C
      DO 5 K = 1, NODE
      PSISTAR(K) = PSI(K)
      VNEW(K) = VSTAR(K) = V(K)
    5 UNEW(K) = USTAR(K) = U(K)
C
C     ASSEMBLE GLOBAL MATRICES WHICH ARE TIME- AND PARAMETERINDEPENDE
C
      CALL ASSEM  (CHICHI,STI,NODE,NELE,3,DUMMY,AT,AREA)
      CALL ASSEM  (CHIDX,STI,NODE,NELE,1,DUMMY,BT,AREA)
      CALL ASSEM  (CHIDY,STI,NODE,NELE,1,DUMMY,AT,AREA)
C
C     FIND BOUNDARY NODES
C
      DO 10 K = 1,JC
      NBX(1,K)=(K-1)*JR+1
   10 NBX(2,K)=K*JR
      TIME = 0.
C
C
C
      DO 9999 NTIME = 1,NLIMIT
      TIME = TIME+DT
C
C     SET UP MATRIX EQUATION FOR THE CONTINUITY EQUATION
C
      CALL ASSEM (AD,STI,NODE,NELE,4,USTAR,BT,AREA)
      CALL ASSEM (DUMMY,STI,NODE,NELE,4,VSTAR,AT,AREA)
      DO 200 K=1,NODE
      DO 200 L=1,7
      LEFT(L,K)=CHICHI(L,K)/DT2 - AD(L,K) - DUMMY(L,K)
      DUMMY(L,K)=AD(L,K)+DUMMY(L,K)
  200 CONTINUE
      CALL MAMULT(DUMMY,PSI,RIGHT,NODE)
```

```
C
C     SOLVE
C
      CALL SOLVER (LEFT,RIGHT,DIF,NODE,EPS,ITERMAX)
C
C     ADJUST PARAMETERS
C
      DO 201 K = 1 , NODE
      PSISTAR(K) = PSI(K)+DIF(K)
  201 CONTINUE
C
C     SET UP MATRIX EQUATION FOR FIRST EQUATION OF MOTION (U-EQUATION)
                CALL MAMULT (CHIDX,PSISTAR,RS,NODE)
      CALL DX(R1,USTAR,USTAR,NODE,NROW,NCOL,DISTX)
      CALL DY(R2,VSTAR,USTAR,NODE,NROW,NCOL,DISTY)
      DO 203 K=1,NODE
  203 RIGHT(K)=-RS(K)*DT
C
C     SOLVE
C
      CALL SOLVER (CHICHI,RIGHT,DIF,NODE,EPS,ITERMAX)
      DO 202 K=1,NODE
  202 DIF(K)=DIF(K)-(R1(K)+R2(K)-F(K)*VSTAR(K))*DT
C
C     ADJUST PARAMETERS
C
      DO 301 K = 1 , NODE
      UNEW(K) = U(K) + DIF(K)
      USTAR(K) = .5*(UNEW(K) + U(K))
  301 CONTINUE
C
C     SET UP MATRIX EQUATION FOR SECOND EQUATION OF MOTION (V-EQUATION
                CALL MAMULT (CHIDY,PSISTAR,RS,NODE)
      CALL DY(R1,VSTAR,VSTAR,NODE,NROW,NCOL,DISTY)
      CALL DX(R2,USTAR,VSTAR,NODE,NROW,NCOL,DISTX)
      DO 205 K=1,NODE
      RIGHT(K)=-RS(K)*DT
      DO 205 L=1,7
  205 LEFT(L,K)=CHICHI(L,K)
      DO 206 KB=1,JC
      K1=NBX(1,KB)
      K2=NBX(2,KB)
      BX(1,KB)=(R1(K1)+R2(K1)+F(K1)*USTAR(K1))*DT
  206 BX(2,KB)=(R1(K2)+R2(K2)+F(K2)*USTAR(K2))*DT
C
C
C     IMPLEMENTATION OF BOUNDARY CONDITION
C
      CALL BOUND (LEFT,RIGHT,BX,BY,NBX,NBY,NODE,JC,JR)
C     SOLVE
C
C
      CALL SOLVER (LEFT,RIGHT,DIF,NODE,EPS,ITERMAX)
      DO 204 K=1,NODE
      DIF(K)=DIF(K)-(R1(K)+R2(K)+F(K)*USTAR(K))*DT
  204 VNEW(K)=V(K)+DIF(K)
      IF(MOD(NTIME,12).NE.0) GO TO 210
      CALL SMOOTH(VNEW,WS,12,15)
C
C     ADJUST PARAMETERS
C
  210 DO 207 K = 1 , NODE
      PSI(K)=PSISTAR(K)
      VSTAR(K)=1.5*VNEW(K)-.5*V(K)
      V(K)=VNEW(K)
      USTAR(K)=1.5*UNEW(K)-.5*U(K)
  207 U(K)=UNEW(K)
C       CORRECT INTEGRAL INVARIANTS IF NECESSARY
  352 CALL LOOK(U,V,PSI,JC+1,JR,TIME,NTIME,G,NODE,F)
      PRINT 300,TIME,HMEAN,ZMEAN,ENERGY
  300 FORMAT("0SCALED TIME: ",F8.2," CONSTRAINTS, H, Z \& E: ",3E14.6)
      HDF=ABS(HMEAN-H0)
      ZDF=ABS(ZMEAN-Z0)
      EDF=ABS(ENERGY-E0)
      IF(HDF.LT.EPSH) GO TO 350
      DO 351 K=1,NODE
  351 PSI(K)=PSI(K)+G*(H0-HMEAN)
      GO TO 352
  350 IF(ZDF.LT.EPSZ.AND.EDF.LT.EPSE) GO TO 302
      ETA=100.
      EMINN=ETA*SQRT(ZDF**2+EDF**2)/10.
      RH=RH0
      RZ=RZ0
      RE=RE0
      PNLTH=0.
      PNLTZ=0.5/RZ
      PNLTE=0.5/RE
      UH=0.
      UE=0.
      UZ=0.
  305 DO 306 II=1,N
  306 XO(II)=XC(II)
      PRINT 307,ETA,RH,RZ,RE,UH,UZ,UE,PNLTH,PNLTZ,PNLTE
  307 FORMAT("0ETA = ",E14.6/" R         , H, Z \& E: ",3E14.6/
     *" U         , H, Z \& E: ",3E14.6/" PENALTIES, H, Z \& E: ",3E14.6)
      CALL E04DBF(N,XC,FLOW,GC,XTOL,FEST,DUM,WS,FUNCT,MONIT,50,IFAIL)
      CALL LOOK(U,V,PSI,JC+1,JR,TIME,NTIME,G,NODE,F)
      PRINT 300,TIME,HMEAN,ZMEAN,ENERGY
      HDIF=HMEAN-H0
      ZDIF=ZMEAN-Z0
      EDIF=ENERGY-E0
      IF(ABS(ZDIF*4.).GT.ZDF) RZ=RZ*0.4
      UZ=UZ+ZDIF/RZ
      PNLTZ=0.5/RZ
      IF(ABS(EDIF*4.).GT.EDF) RE=RE*0.4
      UE=UE+EDIF/RE
```

```
      PNLTE=0.5/RE
      HDF=ABS(HDIF)
      ZDF=ABS(ZDIF)
      EDF=ABS(EDIF)
      IF(ZDF.LT.EPSZ/5..AND.EDF.LT.EPSE/5.) GO TO 310
      ETA=ETA*0.8
      IF(ETA.LT.51.1) GO TO 310
      EMINN=EMINN/10.
      GO TO 305
  310 IT1=IT1+1
      HPLT(IT1)=HMEAN/H0
      ZPLT(IT1)=ZMEAN/Z0
      EPLT(IT1)=ENERGY/E0
      TPLT(IT1)=TIME/36.
      IF(HDF.LT.EPSH) GO TO 303
      DO 353 K=1,NODE
  353 PSI(K)=PSI(K)+G*(H0-HMEAN)
      CALL LOOK(U,V,PSI,JC+1,JR,TIME,NTIME,G,NODE,F)
      PRINT 300,TIME,HMEAN,ZMEAN,ENERGY
C
C     PRINTOUT (IF REQUIRED)
C
  303 IF(NTIME.EQ.960) GO TO 9992
      IF(NTIME.GT.960) GO TO 9994
  302 IF(MOD(NTIME,MF).NE.0) GO TO 9996
      CALL OUT(U,V,PSI,JC,JR,NODE,NOUTU,NOUTV,TIME,NTIME,F)
 9996 IF(MOD(NTIME,ND).NE.0.OR.NTIME.GT.ND*20) GO TO 9999
      DO 9998 K=1,NODE
 9998 H(K)=2.*SQRT(PSI(K))
      WRITE(7) (H(K),K=1,NODE)
      WRITE(7) (V(K),K=1,NODE)
      WRITE(7) (U(K),K=1,NODE)
 9999 CONTINUE
 9992 DO 9993 K=1,NODE
 9993 H(K)=2.*SQRT(PSI(K))
      WRITE(7) (H(K),K=1,NODE)
      WRITE(7) (V(K),K=1,NODE)
      WRITE(7) (U(K),K=1,NODE)
 9994 IT=IT1-1
      TFAC=(TPLT(IT1)-TPLT(IT))/(480.-TPLT(IT))
      TPLT(IT1)=480.
      HPLT(IT1)=HPLT(IT)+(HPLT(IT1)-HPLT(IT))/TFAC
      ZPLT(IT1)=ZPLT(IT)+(ZPLT(IT1)-ZPLT(IT))/TFAC
      EPLT(IT1)=EPLT(IT)+(EPLT(IT1)-EPLT(IT))/TFAC
      WRITE(8) IT1,(HPLT(I),I=1,IT1),(ZPLT(I),I=1,IT1),
     *(EPLT(I),I=1,IT1),(TPLT(I),I=1,IT1)
C     STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                              C
C     INCOND CALCULATES THE INITIAL U,V,PSI AND H FIELD AND THE                C
C     CORIOLIS PARAMETER F AT EACH GRIDPOINT                                   C
C                                                                              C
C     ---------------------------------------------------------------          C
C                                                                              C
C                                                                              C
C     PARAMETERS:                                                              C
C                                                                              C
C     PSI         - REAL ARRAY OF DIMENSION (NODE)                             C
C                   CONTAINS ON EXIT GEOPOTENTIAL (=G*H)                       C
C     U,V         - REAL ARRAYS OF DIMENSION (NODE)                            C
C                   CONTAIN ON EXIT EASTWARD, RESP. WIND                       C
C                   COMPONENT                                                  C
C     H           - REAL ARRAY OF DIMENSION (NODE),CONTAINS                    C
C                   ON EXIT HEIGHT                                             C
C     F           - REAL ARRAY OF DIMENSION (NODE) CONTAINS ON EXIT            C
C                   CORIOLIS PARAMETER                                         C
C     NODE        - INTEGER,  TOTAL NUMBER OF NODES                            C
C     NROW        - INTEGER, NUMBER OF NODES-1  IN X-DIRECTION                 C
C     NCOL        - INTEGER, NUMBER OF NODES-1 IN Y-DIRECTION                  C
C                                                                              C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
C
      SUBROUTINE INCOND (PSI,U,V,H,F,NODE,NROW,NCOL)
      COMMON/INVRNT/ HMEAN,ZMEAN,ENERGY
      COMMON /AREA/ AT(3,330),BT(3,330),AREA
      COMMON /COR/ X(192),Y(192),DISTX,DISTY
      COMMON /CONST/ H0,H1,H2,G,FST,BETA,XL,D
      DIMENSION PSI(NODE),U(NODE),V(NODE),H(NODE),F(NODE)
      DATA UFAC/1.E03/,HFAC/1.E05/
      PI =  3.1415926535
      PI2XL = 2. * PI / XL
      D2 = D/2.
C
C     CALCULATE INITIAL HEIGHT FIELD
C
      GNEW=G*HFAC/UFAC**2
      DO 50 K=1,NODE
      F(K) = FST + BETA * (Y(K) - D2)
      GF = G / F(K)
      F(K)=F(K)*HFAC/UFAC
      B = 9. * (D2 - Y(K)) / D
      COH = COSH (B)
      COH2=COH*COH
      CO2=COSH(B/2.)
      CO22=CO2*CO2
       SI=SIN(PI2XL*X(K))
      DHX=H2*COS(PI2XL*X(K))*PI2XL/COH2
      DH1 = -4.5 * H1 / (CO22*D)
      DH2 =18.*  H2 * TANH(B) * SI / (COH2*D)
      DHY = DH1 + DH2
C
C     CALCULATE U AND V VELOCITY COMPONENTS
C
```

```
      U(K) =(-GF * DHY)/UFAC
      V (K) =(GF * DHX)/UFAC
      H   (K) =(H0 + H1 * TANH (B/2.) + H2 * SI / COH2)/HFAC
C
C     CALCULATE GEOPOTENTIAL
C
      PSI(K) = H(K) * GNEW
   50 CONTINUE
      JR = NROW + 1
      JC = NCOL + 1
C
C     SET THE V VELOCITY COMPONENT AT THE BOUNDARY IN Y DIRECTION
C     EQUAL 0.
C
      DO 60 K = 1 , JC
      KK = (K-1) * JR
      V(KK+1) = V(KK+JR) = 0.
   60 CONTINUE
      G=GNEW
C
C     PRINT INITIAL FIELDS
C
      PRINT 2002, (H(K),K=1,NODE)
      PRINT 2003, (PSI(K) ,K=1,NODE)
      PRINT 2004, (U(K) ,K=1,NODE)
      PRINT 2005, (V(K),K=1,NODE)
      PRINT 2000
      CALL LOOK (U,V,PSI,JC+1,JR,0.0,0.0,G,NODE,F)
      CALL MAPPA (PSI,2.E03,JC,JR,G)
 2000 FORMAT (1H1)
 2002 FORMAT (1H1,"INITIAL HEIGHT FIELD" /, (12F8.6))
 2003 FORMAT (10(/), " INITIAL PSI FIELD"/,(12F8.6))
 2004 FORMAT (1H1," INITIAL U FIELD"/,(12G10.3))
 2005 FORMAT (10(/)," INITIAL V FIELD"/,(12G10.3))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C     NUMBER LABELS THE ELEMENTS,NUMBERS THE NODES AND STORES THE    C
C     GLOBAL NODENUMBERS OF THE THREE VERTICES OF EACH TRIANGLE.     C
C     IN ADDITION THE GLOBAL COORDINATES OF EACH NODE ARE CALCULATED C
C--------------------------------------------------------------------C
C                                                                    C
C     PARAMETERS:                                                    C
C                                                                    C
C     NPRINT     - = 0  :  NO PRINTOUT                               C
C                  \/ 0  :  THE ARRAY NOD WHICH STORES THE GLOBAL     C
C                          NODENUMBERS OF EACH ELEMENT AND THE TWO   C
C                          ARRAYS  X  AND  Y  WHICH CONTAIN THE      C
C                          COORDINATES OF ALL THE NODES  ARE PRINTED C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE NUMBER (NPRINT)
      COMMON /DIM/ NODE , NELE , NCOL ,NROW
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      COMMON /COR/ X(192),Y(192),DISTX,DISTY
      JC = NCOL +  2
      JR = NROW + 1
      NNOD = 3
      K=1
C
C     LABEL ELEMENTS AND NUMBER NODES
C
      DO 20 J = 1, NCOL
      JJ = (J-1) * JR
      DO 20 KK = 1 , NROW
      KKK = KK + JJ
      NOD(1,K) = KKK
      NOD(1,K+1) = KKK + JR + 1
      NOD(3,K) = NUD(2 K+1) = KKK + 1
      NOD(2,K) = NOD(3,K+1) = KKK + JR
      K=K+2
   20 CONTINUE
C
C     IMPLEMENT CYCLIC BOUNDARY CONDITIONS
C
      DO 25 KK = 1,NROW
      JJ = NCOL * (NROW + 1)
      NOD(1,K) = KK + JJ
      NOD (1,K+1) =  - ( KK + 1 )
      NOD(3,K) = NOD(2,K+1) = KK + JJ + 1
      NOD(2,K) = NOD(3,K+1) =   - KK
      K = K + 2
   25 CONTINUE
C
C     CALCULATE COORDINATES OF NODES
C
      K=1
      DO 30 J = 1,JC
      XX = J * DISTX
      DO 30 KK = 1,JR
      Y(K) = (KK-1) * DISTY
      X ( K) = XX
      K = K + 1
   30 CONTINUE
      IF (NPRINT .EQ. 0) RETURN
C
C     PRINTOUT (IF REQUIRED)
C
      PRINT 2002
      DO 40 L=1,NELE
   40 PRINT 2001,     L,(NOD(K,L),K=1,NNOD)
      PRINT 2003  ,   (X(K)*1.0E-3,Y(K)*1.0E-3, K=1,NODE)
 2003 FORMAT (1H1,"COORDINATES  (KM)",/,(12G10.2))
 2002 FORMAT (1H1,14HELEMENT NUMBER, 6X, 15HNUMBER OF NODES)
 2001 FORMAT ((5X,I4,13X,3I4))
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                             C
C     AREAA CALCULATES THE DERIVATIVES OF THE SHAPEFUNCTIONS (X- AND          C
C     Y-DIRECTON)  AND  STORES  THEM  IN  THE ARRAYS  AT  AND  BT             C
C                                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE AREAA
      COMMON /AREA/ AT(3,330),BT(3,330),AREA
      COMMON /COR/ X(192),Y(192),DISTX,DISTY
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      COMMON /DIM/ NODE , NELE , NCOL ,NROW
      DATA HFAC/1.E5/
      DO 500 K=1,NELE
      N1 = NOD(1,K)
      N2 = NOD(2,K)
      N3 = NOD(3,K)
      IF (N1.GT.0.AND.N2.GT.0.AND.N3.GT.0) GOTO 100
      N1 = IABS(N1)
      N2 = IABS (N2)
      N3 = IABS(N3)
      IF (N1.LE.NROW+1) N1 = (NCOL+1)*(NROW+1) + N1
      IF (N2.LE.NROW+1) N2 = (NCOL+1)*(NROW+1) + N2
      IF (N3.LE.NROW+1) N3 = (NCOL+1)*(NROW+1) + N3
  100 CONTINUE
C
C     CALCULATE DERIVATIVES OF SHAPE FUNCTIONS
C
      AT(1,K) =(X(N3) - X(N2))/HFAC
      BT(1,K) =(Y(N2) - Y(N3))/HFAC
      AT(2,K) =(X(N1) - X(N3))/HFAC
      BT(2,K) =(Y(N3) - Y(N1))/HFAC
      AT(3,K) =(X(N2) - X(N1))/HFAC
      BT(3,K) =(Y(N1) - Y(N2))/HFAC
  500 CONTINUE
C
C     CALCULATE AREA OF ONE ELEMENT
C
      AREA  = DISTX * DISTY * .5
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                             C
C     BOUND IMPLEMENTS THE BOUNDARY CONDITIONS AFTER THE SYSTEM               C
C     OF EQUATIONS AX=R  HAS BEEN OBTAINED                                    C
C                                                                             C
C                                                                             C
C-------------------------------------------------------------------------------C
C                                                                             C
C     PARAMETERS:                                                             C
C                                                                             C
C     LEFT        - REAL ARRAX OF DIMENSION  (7, NODE), ON ENTRY              C
C                   LEFT CONTAINS MATRIX A , ON EXIT  MODIFIED                C
C                   MATRIX A.                                                 C
C     RIGHT       - REAL ARRAY OF DIMENSION (NODE), ON ENTRY CONTAINS         C
C                   VECTOR R , ON EXIT  R  IS MODIFIED ACCORDING              C
C                   BOUNDARY CONDITION.                                       C
C     BX,BY       - REAL ARRAYS OF DIMENSION (2,JC) RESP(2,JR),               C
C                   CONTAIN BOUNDARY VALUES                                   C
C     NBX,NBY     - INTEGER ARRAYS OF DIMENSION (2,JC),RESP.(2,JR),           C
C                   CONTAIN NUMBERS  OF  BOUNDARY NODES                       C
C     NODE        - INTEGER, TOTAL NUMBER OF NODES                            C
C     JC          - NUMBER OF BOUNDARY NODES IN X-DIRECTION                   C
C     JR          - NUMBER OF BOUNDARY NODES IN Y-DIRECTION                   C
C                                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE BOUND (LEFT,RIGHT,BX,BY,NBX,NBY,NODE,JC,JR)
      REAL LEFT
      DIMENSION LEFT(7,NODE),RIGHT(NODE),BX(2,JC),NBX(2,JC),BY(2,JR),
     +       NBY(2,JR)
      DO 10 K = 1,JC
      KK = NBX(1,K)
      LL = NBX(2,K)
      LEFT(7,KK) = LEFT(7,KK) * 1. E 15
      LEFT(7,LL) = LEFT(7,LL) * 1. E 15
      RIGHT(KK) = BX(1,K)*LEFT(7,KK)
      RIGHT(LL) = BX(2,K)*LEFT(7,LL)
   10 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                             C
C     OUT PRINTS THE HEIGHT~,U-  AND  V - FIELDS.                             C
C     OUT CALLS THE SUBROUTINES  HAPPA AND LOOK                               C
C                                                                             C
C-------------------------------------------------------------------------------C
C                                                                             C
C     PARAMETERS:                                                             C
C                                                                             C
C     U,V,PSI     - REAL ARRAYS OF DIMENSION (NODE), FIELD VARIABLES          C
C                   TO BE PRINTED                                             C
C     JC,JR       - INTEGERS, NUMBER OF NODES IN X- RESP. IN Y DIRECTIONC
C     NODE        - INTEGER, TOTAL NUMBER OF NODES                            C
C     NOUTU,NOUTV- INTEGER , PRINT OPTIONS                                    C
C     TIME        - REAL, REAL TIME                                           C
C     NTIME       - INTEGER, TIMESTEP                                         C
C                                                                             C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE OUT(U,V,PSI,JC,JR,NODE,NOUTU,NOUTV,TIME,NTIME,F)
      COMMON /CONST/ H0,H1,H2,G,FST,BETA,XL,D
      DIMENSION PSI(NODE),U(NODE),V(NODE),F(NODE)
      PRINT 2000
      CALL MAPPA (PSI,2.E03,JC,JR,G)
      PRINT 2000
      PRINT 2108,  (PSI(K)/G,K=1,NODE),(PSI(K)/G,K=1,JR)
 2000 FORMAT (1H1)
```

```
      IF (NOUTU .NE. 0) PRINT 2105, (U(K),K=1,NODE),
     +                (U(K),K=1,JR)
      IF (NOUTV .NE. 0) PRINT 2106, (V(K),K=1,NODE),
     +         (V(K),K=1,JR)
 2105 FORMAT (1H1,"   U - VALUES", / , (1X,12G10.4))
 2106 FORMAT (10(/),1X,"   V - VALUES",/,(1X,12G10.4))
 2108 FORMAT (10(/),"   HEIGHT VALUES" ,/, (12F10.5))
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C     SOLVER SOLVES A SYSTEM OF LINEAR EQUATION  AX=R BY THE         C
C     GAUSS SEIDEL METHOD                                            C
C                                                                    C
C--------------------------------------------------------------------C
C                                                                    C
C     PARAMETERS:                                                    C
C     COMA        - REAL ARRAY OF DIMENSION (7,NODE)                 C
C                   ON ENTRY  COMA CONTAINS THE NON-ZERO ENTRIES     C
C                   OF MATRIX A.  UNCHANGED ON  EXIT                 C
C                                                                    C
C     RIGHT       - REAL ARRAY OF DIMENSION  (NODE)                  C
C                   ON ENTRY RIGHT CONTAINS THE ELEMENTS OF THE      C
C                   RIGHT HANDSIDE VECTOR  R. UNCHANGED ON EXIT.     C
C                                                                    C
C     NODE        - NUMBER OF NODES  (= NUMBER OF EQUATIONS)         C
C                                                                    C
C     XSOLV       - REAL ARRAY OF DIMENSION  (NODE)                  C
C                   ON ENTRY XSOLV CONTAINS A FIRST GUESS FOR THE    C
C                   SOLUTION VECTOR. ON EXIT XSOLV CONTAINS THE      C
C                   SOLUTION VECTOR                                  C
C                                                                    C
C     EPS         - RELATIVE ERROR                                   C
C                                                                    C
C     ITERMAX     - MAXIMAL NUMBER OF ITERATIONS                     C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SOLVER(COMA,RIGHT,XSOLV,NODE ,EPS,ITERMAX)
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      DIMENSION COMA(7,NODE),RIGHT(NODE),XSOLV(NODE)
      DO 90 L = 1 , ITERMAX
      XMAX = 0.
      DIFMAX = 0.
      DO 100 K = 1 , NODE
      SUM = 0.
      DO 50 KR= 1 , 6
      NLOC = LOCAT (KR,K)
      IF (NLOC .EQ. 0) GOTO 50
      VAL = COMA(KR,K)
      SUM = SUM + VAL * XSOLV(NLOC)
   50 CONTINUE
      XX = (RIGHT(K) - SUM) / COMA(7,K)
      XMAX = AMAX1(XMAX,ABS(XX))
      DIFF = XX - XSOLV(K)
      DIFMAX = AMAX1 (ABS(DIFF),DIFMAX)
      XSOLV(K) = XX
  100 CONTINUE
      IF (DIFMAX/XMAX .LT. EPS) RETURN
   90 CONTINUE
C
C     THE PROGRAM STOPS IF CONVERGENCE IS NOT REACHED AFTER ITERMAX TI
C
      PRINT 2001
 2001 FORMAT (1X,"NO CONVERGENCE")
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                    C
C     SOLVER SOLVES A SYSTEM OF LINEAR EQUATION  AX=R BY THE         C
C     GAUSS SEIDEL METHOD                                            C
C                                                                    C
C--------------------------------------------------------------------C
C                                                                    C
C     PARAMETERS:                                                    C
C     COMA        - REAL ARRAY OF DIMENSION (7,NODE)                 C
C                   ON ENTRY  COMA CONTAINS THE NON-ZERO ENTRIES     C
C                   OF MATRIX A.  UNCHANGED ON  EXIT                 C
C                                                                    C
C     RIGHT       - REAL ARRAY OF DIMENSION  (NODE)                  C
C                   ON ENTRY RIGHT CONTAINS THE ELEMENTS OF THE      C
C                   RIGHT HANDSIDE VECTOR  R. UNCHANGED ON EXIT.     C
C                                                                    C
C     NODE        - NUMBER OF NODES  (= NUMBER OF EQUATIONS)         C
C                                                                    C
C     XSOLV       - REAL ARRAY OF DIMENSION  (NODE)                  C
C                   ON ENTRY XSOLV CONTAINS A FIRST GUESS FOR THE    C
C                   SOLUTION VECTOR. ON EXIT XSOLV CONTAINS THE      C
C                   SOLUTION VECTOR                                  C
C                                                                    C
C     EPS         - RELATIVE ERROR                                   C
C                                                                    C
C     ITERMAX     - MAXIMAL NUMBER OF ITERATIONS                     C
C                                                                    C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SOLVER(COMA,RIGHT,XSOLV,NODE ,EPS,ITERMAX,SNARK,
     *NFIELD)
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      DIMENSION COMA(7,NODE),RIGHT(NODE),XSOLV(NODE)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 90 L = 1 , ITERMAX
      XMAX = 0.
      DIFMAX = 0.
      DO 100 K = 1 , NODE
      SUM = 0.
      DO 50 KR= 1 , 6
```

```
      NLOC = LOCAT (KR,K)
      IF (NLOC .EQ. 0) GOTO 50
      VAL = COMA(KR,K)
      SUM = SUM + VAL*XSOLV(NLOC)
   50 CONTINUE
      TELL1=RIGHT(K)*SNARK-SUM*SNARK
      TELL2=COMA(7,K)*XSOLV(K)
      TELL3=COMA(7,K)*SNARK*XSOLV(K)
      XX=(TELL1+TELL2-TELL3)/(COMA(7,K))
      DIFF = XX - XSOLV(K)
      XMAX=AMAX1(XMAX,ABS(XX))
      DIFMAX = AMAX1 (ABS(DIFF),DIFMAX)
      XSOLV(K) = XX
  100 CONTINUE
      IF (DIFMAX/XMAX .GE. EPS) GOTO 90
      IF(NFIELD-0) 974,975,976
  974 PRINT 977
  977 FORMAT(2X,"H-FIELD SOLUTION:")
      GOTO 101
  975 PRINT 978
  978 FORMAT(2X,"U-FIELD SOLUTION:")
      GOTO 101
  976 PRINT 979
  979 FORMAT(2X,"V-FIELD SOLUTION:")
  101 PRINT 107,L
  107 FORMAT(2X,"NUMBER OF ITERATIONS FOR SOR-METHOD=",I4)
      PRINT 797,SNARK
  797 FORMAT(2X,"USING RELAXATION FACTOR=",F6.3)
      RETURN
   90 CONTINUE
C
C     THE PROGRAM STOPS IF CONVERGENCE IS NOT REACHED AFTER ITERMAX TI
C
      PRINT 2001
 2001 FORMAT (1X,"NO CONVERGENCE")
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     MAMULT  MULTIPLIES  A MATRIX STORED IN COMPACT FORM BY A         C
C     VECTOR  (M)V = R                                                 C
C                                                                      C
C----------------------------------------------------------------------C
C                                                                      C
C     PARAMETERS:                                                      C
C                                                                      C
C     COMA        - REAL ARRAY OF DIMENSION  (7,NODE)                  C
C                   ON  ENTRY CONTAINS  MATRIX  M. UNCHANGED ON EXIT   C
C                                                                      C
C     VECTOR      - REAL ARRAY OF DIMENSION (NODE)                     C
C                   ON ENTRY CONTAINS VECTOR V,  UNCHANGED  ON EXIT    C
C                                                                      C
C     RIGHT       - REAL ARRAY  OF DIMENSION (NODE)                    C
C                   ON EXIT CONTAINS  PRODUCT  R                       C
C                                                                      C
C     NODE        - NUMBER OF NODES                                    C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE MAMULT (COMA,VECTOR,RIGHT,NODE)
      DIMENSION COMA(7,NODE),VECTOR(NODE),RIGHT(NODE)
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      DO 10 K=1,NODE
   10 RIGHT(K)=0.
      DO 100 K=1,NODE
      DO 80 KR=1,6
      NLOC = LOCAT(KR,K)
      IF (NLOC .EQ. 0 ) GOTO 80
      RIGHT(K) = RIGHT(K) + COMA(KR,K) * VECTOR(NLOC)
   80 CONTINUE
      RIGHT(K) = RIGHT(K) + COMA(7,K) * VECTOR (K)
  100 CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     ASSEM GENERATES THE LOCAL  3X3 MATRICES FOR EACH ELEMENT AND     C
C     STORES THE COEFFICIENTS IN COMPACT FORM IN THE GLOBAL MATRIX     C
C     COMA.                                                            C
C                                                                      C
C----------------------------------------------------------------------C
C                                                                      C
C     PARAMETERS:                                                      C
C                                                                      C
C     COMA        - REAL ARRAY  OF DIMENSION (7,NODE)                  C
C                   ON EXIT  CONTAINS THE GLOBAL MATRIX                C
C                                                                      C
C     STI         - REAL ARRAY  OF DIMENSION  (3,3)                    C
C                   WORKING SPACE USED FOR THE ELEMENTMATRICES         C
C                                                                      C
C     NODE        - INTEGER                                            C
C                   NUMBER OF NODES                                    C
C                                                                      C
C     NELE        - INTEGER                                            C
C                   NUMBER OF ELEMENTS                                 C
C                                                                      C
C     NSWITCH     - INTEGER                                            C
C                   DETERMINES WHICH TYPE OF LOCAL ELEMENT MATRIX      C
C                   MUST BE USED                                       C
C     PSIUV       - REAL ARRAY OF DIMENSION (NODE)                     C
C                   ON ENTRY CONTAINS U,V OR PSI DEPENDENT WHAT TYPE   C
C                   OF MATRIX MUST BE CALCULATED.UNCHANGED ON          C
C                   EXIT.                                              C
C     CODI        - REAL ARRAY OF DIMENSION (3,NELE)                   C
C                   ON ENTRY, CONTAINS  DERIVATIVE OF  SHAPEFUNCTION   C
C                   (EITHER  X- OR Y-DIRECTON). UNCHANGED ON EXIT      C
C     AREA        - REAL                                               C
C                   AREA OF ONE ELEMENT                                C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE ASSEM  (COMA,STI,NODE,NELE,NSWITCH,PSIUV,CODI,AREA)
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      DIMENSION COMA(7,NODE),STI(3,3),CODI(3,NELE),B(3),A(3),PSIUV(NODE)
      NNOD = 3
      NN = 7*NODE
      DO 10 K = 1,NN
   10 COMA(K) = 0.
C
C     DECIDE WHICH ELEMENT MATRIX MUST BE CALCULATED
            GOTO (100,200,500,600), NSWITCH
C
C     GENERATE ELEMENT MATRIX
  100         DO 160 NE = 1,NELE
      DO 110 K = 1,NNOD
      STI(K,1) = CODI(1,NE)/ 6.
      STI(K,2) = CODI(2,NE)/ 6.
      STI(K,3) = CODI(3,NE)/ 6.
  110 CONTINUE
C
C     ASSEMBLE GLOBAL MATRIX AND STORE IT IN COMPACT FORM
            DO 160 K = 1,NNOD
      IROW = NOD(K,NE)
      DO 160 J = 1,NNOD
      ICOL = NOD(J,NE)
      L = 7
      IF (ICOL .EQ. IROW) GOTO 150
      DO 140 L = 1,6
      IF (LOCAT(L,IROW) .EQ. ICOL) GOTO 150
  140 CONTINUE
  150 COMA(L,IROW) = COMA(L,IROW) + STI(K,J)
  160 CONTINUE
      RETURN
C
C     GENERATE ELEMENT MATRIX
  200         DO 260 NE = 1,NELE
      AA = 0.
      DO 210 K = 1,NNOD
      B(K) = CODI(K,NE)
      A(K) = PSIUV(NOD(K,NE))
  210 AA = AA + A(K)
      DO 220 K = 1,NNOD
      DO 220 L = 1,NNOD
  220 STI (K,L) = (AA + A(K)) * B(L) / 24.
C
C     ASSEMBLE GLOBAL MATRIX AND STORE IT IN COMPACT FORM
            DO 260 K = 1,NNOD
      IROW = NOD(K,NE)
      DO 260 J = 1,NNOD
      ICOL = NOD(J,NE)
      L = 7
      IF (ICOL .EQ. IROW) GOTO 250
      DO 240 L = 1,6
      IF (LOCAT(L,IROW) .EQ. ICOL) GOTO 250
  240 CONTINUE
  250 COMA(L,IROW) = COMA(L,IROW) + STI(K,J)
  260 CONTINUE
      RETURN
C
C     GENERATE ELEMENT MATRIX
  500 CO = AREA / 12.
      DO 510 K = 1,9
  510 STI(K) = CO * 1.0
      STI (1,1) = STI(2,2) = STI(3,3) = CO * 2.0
C
C     ASSEMBLE GLOBAL MATRIX AND STORE IT IN COMPACT FORM
       DO 560 NE = 1,NELE
      DO 560 K = 1,NNOD
      IROW = NOD(K,NE)
      DO 560 J = 1,NNOD
      ICOL = NOD(J,NE)
      L = 7
      IF (ICOL .EQ. IROW) GOTO 550
      DO 540 L = 1,6
      IF (LOCAT(L,IROW) .EQ. ICOL) GOTO 550
  540 CONTINUE
  550 COMA(L,IROW) = COMA(L,IROW) + STI(K,J)
  560 CONTINUE
      RETURN
C
C     GENERATE ELEMENT MATRIX
  600 DO 660 NE = 1,NELE
      AA = 0.
      DO 610 K = 1,NNOD
      A(K) = PSIUV(NOD(K,NE))
  610 AA = A(K) + AA
      DO 620 K = 1,NNOD
      S = A(K) + AA
      DO 620 L = 1,NNOD
  620 STI(L,K) = CODI(L,NE)*S/ 24.
C
C     ASSEMBLE GLOBAL MATRIX AND STORE IT IN COMPACT FORM
       DO 660 K = 1,NNOD
      IROW = NOD(K,NE)
      DO 660 J = 1,NNOD
      ICOL = NOD(J,NE)
      L = 7
      IF (ICOL .EQ. IROW) GOTO 650
      DO 640 L = 1,6
      IF (LOCAT(L,IROW) .EQ. ICOL) GOTO 650
  640 CONTINUE
  650 COMA(L,IROW) = COMA(L,IROW) + STI(K,J)
  660 CONTINUE
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     CORRES FINDS ALL THE NON-ZERO LOCATIONS OF THE GLOBAL MATRIX     C
C     AND STORES THEM IN  THE ARRAY LOCAT                              C
C                                                                      C
C----------------------------------------------------------------------C
C                                                                      C
C     PARAMETERS:                                                      C
C                                                                      C
C     NPRINT     - = 0 : NO PRINTOUT                                   C
C                  \/ 0 : THE ARRAY LOCAT  IS PRINTED  WHICH CONTAINS  C
C                         ALL INDICES OF THE NON-ZERO ENTRIES OF THE   C
C                         GLOBAL  MATRIX                               C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CORRES (NPRINT)
      COMMON /ELEMENT/ LOCAT(6,180),NOD(3,330)
      COMMON /DIM/ NODE , NELE , NCOL ,NROW
      DATA LOCAT /1080*0/
      NNOD = 3
      DO 40 NE = 1 , NELE
      DO 40 K = 1 , NNOD
      IROW = IABS (NOD(K,NE))
      DO 40 J = 1 , NNOD
      ICOL = IABS (NOD(J,NE))
      IF (IROW .EQ. ICOL) GOTO 40
      DO 30 L = 1 , 6
      IF (LOCAT (L,IROW) .EQ. ICOL ) GOTO 40
      IF (LOCAT(L,IROW) .NE.0)  GOTO 30
      LOCAT(L,IROW) = ICOL
      GOTO 40
   30 CONTINUE
   40 CONTINUE
C
C     PRINT OUT (IF REQUIRED)
          IF (NPRINT .EQ. 0) RETURN
      PRINT 2002
      DO 50 K=1,NODE
   50 PRINT 2001,    K,(LOCAT(J,K),J=1,6)
 2001 FORMAT ((1X,I4,5X,6I6))
 2002 FORMAT (1H1,2X,4HNODE,3X,18HNEIGHBOURING NODES)
      RETURN
      END
      SUBROUTINE MAPPA (PSI,C,NX,NZ,G)
      DIMENSION FUN(30,30),ANS(4,116),IANS(116),NUM(10),PSI(1)
      DATA NUM/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,1H0/
    1 FORMAT(///4X,2315//)
    2 FORMAT(1H ,13)
    3 FORMAT(1H ,7X,116A1)
    4 FORMAT(1H+,7X,116A1)
      KK = 0
      DO 5 K = 1,NX
      DO 5 L = 1,NZ
      KK = KK + 1
    5 FUN(K,L) = PSI(KK)/G
      K=3
      N=5
      FK=K
      FN=N
      I=0
      NY=NZ-1
      LEND=K
      PRINT 1, (J,J=1,NZ)
      JB=1
   10 I=I+1
      PRINT 2,I
      IP1=I+1
      IF(IP1.GT.NX) IP1=1
      DO 15 J=1,NZ
      XDIF=(FUN(IP1,J)-FUN(I,J))/FK
      JX=1+N*(J-JB)
      ANS(1,JX)=FUN(I,J)
      DO 15 L=2,LEND
   15 ANS(L,JX)=ANS(L-1,JX)+XDIF
   18 DO 20 J=1,NY
      JX=1+N*(J-JB)
      DO 20 L=1,LEND
      YDIF=(ANS(L,JX+N)-ANS(L,JX))/FN
      M1=JX+1
      M3=JX+N-1
      DO 20 M=M1,M3
   20 ANS(L,M)=ANS(L,M-1)+YDIF
      MEND=M3
      DO 50 L=1,LEND
      DO 40 M=1,MEND
      IF(ANS(L,M).GE.0.) GO TO 30
      AANS=-ANS(L,M)
      KANS=C*AANS
      KKANS=2*(KANS/2)
      IF(KANS.EQ.KKANS) GO TO 35
   25 KANS=KANS/2
      KANS=MOD(KANS,10)
      IF(KANS.EQ.0) KANS=10
      IANS(M)=NUM(KANS)
      GO TO 40
   30 KANS=C*ANS(L,M)
      KKANS=2*(KANS/2)
      IF(KANS.EQ.KKANS) GO TO 25
   35 IANS(M)=1H
   40 CONTINUE
      IF(L.GT.1) GO TO 45
      PRINT 4,(IANS(M),M=1,MEND)
      GO TO 50
   45 PRINT 3,(IANS(M),M=1,MEND)
   50 CONTINUE
      IF(I-NX) 10,55,65
```

```
 55 LEND=1
    I=I+1
    PRINT 2,I
    DO 60 J=1,NZ
    JX=1+N*(J-JB)
60      ANS(1,JX)=FUN(1,J)
    GO TO 10
 65 PRINT 1,(J,J=1,NZ)
    RETURN
    END
    SUBROUTINE LOOK (UU,VV,PSIPSI,NX,NY,TIME,NTIME,G,NODE,F)
    COMMON /COR/ X(192),Y(192),DISTX,DISTY
    COMMON/INVRNT/ HMEAN,ZMEAN,ENERGY
    DIMENSION UU(NODE),VV(NODE),PSIPSI(NODE),F(NODE)
    DIMENSION U(30,30),V(30,30),PHI(30,30),H(30,30)
    DATA IND/0/,NSTEP/0/,TIMEA/0./
    IPR = NTIME
    DX = DISTX
    DY = DISTY
    KK = 0
    NXM = NX -1
    NYM = NY - 1
    DO 4 K = 1,NXM
    DO 4 L = 1,NY
    KK = KK + 1
    PHI(K,L) = PSIPSI(KK)
    U(K,L) = UU(KK)
    V(K,L) = VV(KK)
  4 CONTINUE
    TIMEB=SECOND(CPU)
    DPTIME=TIMEB-TIMEA
    IF(IND.GT.0) GO TO 5
    G4INV=1./(4.*G)
    AREA=NXM*(NY-1)
    ECNST=DX*DY/(G+G)
    E2=0.5*DX*DY
  5 SUMENG=0.
    HMEAN=0.
    FAC=0.5
    DO 40 K=1,NY
    IF(K.EQ.NY) FAC=0.5
    HEL=0.
    ENEREL=0.
    DO 10 J=1,NXM
    PHSQ = PHI(J,K)
    ENEREL=PHSQ*(PHSQ+U(J,K)*U(J,K)+V(J,K)*V(J,K))+ENEREL
 10 CONTINUE
    DO 15 J=1,NXM
    H(J,K) = PHI(J,K) / G
 15 HEL=HEL+H(J,K)
    IF(FAC.EQ.1) GO TO 35
    HEL=HEL*FAC
 35 HMEAN=HMEAN+HEL
    SUMENG=SUMENG+ENEREL
 40 FAC=1.0
    HMEAN=HMEAN/AREA
    ENERGY=SUMENG*ECNST
    ZMEAN=0.
    DO 60 K=2,NYM
    DO 60 J=1,NXM
    JP1=J+1
    JM1=J-1
    IF(J.EQ.1) JM1=NXM
    IF(J.EQ.NXM) JP1=1
    VX=(V(JP1,K)-V(JM1,K))/(2.*DX)
    UY=(U(J,K+1)-U(J,K-1))/(2.*DY)
    A=VX-UY+F(K)
 60 ZMEAN=ZMEAN+A*A/H(J,K)
    ZMEAN=ZMEAN*E2
    IF(NTIME.LT.0) GO TO 45
    NSTEP=IPR
    TIMEA=SECOND(CPU)
    IF(IND.NE.0) GO TO 45
    EN2=ENERGY+ENERGY
    IND=1
    GO TO 50
 45 IF (ENERGY.GT.EN2) STOP 1
 50 RETURN
    END
    SUBROUTINE DX(R,S,A,NODE,NR,NC,DISTX)
    DIMENSION R(192),S(192),A(192),T(15),U(15),X(15),W(15),Z(15)
    NX=NC+1
    NY=NR+1
    NX1=NC
    NX2=NX-2
    COF=-84.*DISTX
    N0=-NY
    DO 1 J=1,NY
    N0=N0+1
    N=N0
    DO 2 I=1,NX
    N=N+NY
    T(I)=A(N)
  2 X(I)=S(N)
C     FIRST STAGE
    U(1)=(-5.*T(3)-32.*T(2)+32.*T(NX)+5.*T(NX1))/COF
    U(2)=(-5.*T(4)-32.*T(3)+32.*T(1)+5.*T(NX))/COF
    DO 3 I=3,NX2
  3 U(I)=(-5.*T(I+2)-32.*T(I+1)+32.*T(I-1)+5.*T(I-2))/COF
    U(NX1)=(-5.*T(1)-32.*T(NX)+32.*T(NX2)+5.*T(NX-3))/COF
    U(NX)=(-5.*T(2)-32.*T(1)+32.*T(NX1)+5.*T(NX2))/COF
    CALL CYCPNT(U,Z,NX)
C     SECONDSTAGE
    U(1)=(X(NX)*Z(NX)+X(NX)*Z(1)+X(1)*Z(NX)+X(1)*Z(2)+X(2)*Z(1)+X(2)*
   *Z(2))/12.+X(1)*Z(1)/2.
    DO 4 I=2,NX1
```

```
      IM=I-1
      IP=I+1
    4 U(I)=(X(IM)*Z(IM)+X(IM)*Z(I)+X(I)*Z(IM)+X(I)*Z(IP)+X(IP)*Z(I)+X(IP
     *)*Z(IP))/12.+X(I)*Z(I)/2.
      U(NX)=(X(NX1)*Z(NX1)+X(NX1)*Z(NX)+X(NX)*Z(NX1)+X(NX)*Z(1)+X(1)*Z(
     *NX)+X(1)*Z(1))/12.+X(NX)*Z(NX)/2.
      CALL CYCTRD(U,W,NX)
      N=N0
      DO 5 I=1,NX
      N=N+NY
    5 R(N)=W(I)
    1 CONTINUE
      N=NX*NY
      DO 6 J=1,NY
      N=N+1
    6 R(N)=R(J)
      RETURN
      END
      SUBROUTINE PENTDG(U,F,NX)
      DIMENSION U(NX),F(NX)
      REAL DEL(15),LAM(15),GAM(15),MU
C      SUBROUTINE PENTDG SOLVES THE EQUATIONS
C      A*U(I-2)+B*U(I-1)+C*(I)+D*(I+1)+E*(I+2)=F(I)
C      FOR 1.LE.I.LE.NX
C      WITH A=0 FOR I=1 AND I=2
C           B=0 FOR I=1
C           D=0 FOR I=NX
C           E=0 FOR I=(NX-1) AND I=NX
C
      A=1./70.
      B=16./70.
      C=36./70.
      D=B
      E=A
C
      NX1=NX-1
      NX2=NX-2
      NX3=NX-3
C      I=1
      DEL(1)=D/C
      LAM(1)=E/C
      GAM(1)=F(1)/C
C      I=2
      MU=C-B*DEL(1)
      DEL(2)=(D-B*LAM(1))/MU
      LAM(2)=E/MU
      GAM(2)=(F(2)-B*GAM(1))/MU
C      3.LE.I.LE.(NX-2)
      DO 1 I=3,NX2
      BETA=B-A*DEL(I-2)
      MU=C-BETA*DEL(I-1)-A*LAM(I-2)
      DEL(I)=(D-BETA*LAM(I-1))/MU
      LAM(I)=E/MU
      GAM(I)=(F(I)-BETA*GAM(I-1)-A*GAM(I-2))/MU
    1 CONTINUE
C      I=NX-1
      BETA=B-A*DEL(NX3)
      MU=C-BETA*DEL(NX2)-A*LAM(NX3)
      DEL(NX1)=(D-BETA*LAM(NX2))/MU
      GAM(NX1)=(F(NX1)-BETA*GAM(NX2)-A*GAM(NX3))/MU
C      I=NX
      BETA=B-A*DEL(NX2)
      MU=C-BETA*DEL(NX1)-A*LAM(NX2)
      GAM(NX)=(F(NX)-BETA*GAM(NX1)-A*GAM(NX2))/MU
C      BACK SOLUTION
      U(NX)=GAM(NX)
      U(NX1)=GAM(NX1)-DEL(NX1)*U(NX)
      DO 2 J=1,NX2
      I=NX1-J
      U(I)=GAM(I)-DEL(I)*U(I+1)-LAM(I)*U(I+2)
    2 CONTINUE
      RETURN
      END
      SUBROUTINE CYCPNT(D,Z,NX)
      DIMENSION D(15),Z(15),TMP(15),V(15),W(15,2),FN(15)
      A=1./70.
      B=16./70.
      C=36./70.
      NX1=NX-1
      NX2=NX-2
      NX3=NX-3
      NX4=NX-4
C        CALCULATE W=E(INVERSE)FN
C        FIRST COLUMN
      DO 1 I=2,NX4
    1 FN(I)=0.
      FN(1)=A
      FN(NX3)=A
      FN(NX2)=B
      CALL PENTDG(TMP,FN,NX2)
      DO 2 I=1,NX2
      W(I,1)=TMP(I)
C        SECOND COLUMN
    2 W(I,2)=TMP(NX1-I)
C        CALCULATE V=E(INVERSE)D
      CALL PENTDG(V,D,NX2)
      GW11=A*W(1,1)+A*W(NX3,1)+B*W(NX2,1)
      GW12=A*W(1,2)+A*W(NX3,2)+B*W(NX2,2)
      GW21=B*W(1,1)+A*W(2,1)+A*W(NX2,1)
      GW22=B*W(1,2)+A*W(2,2)+A*W(NX2,2)
      GV1=A*V(1)+A*V(NX3)+B*V(NX2)
      GV2=B*V(1)+A*V(2)+A*V(NX2)
      DMGV1=D(NX1)-GV1
      DMGV2=D(NX)-GV2
      C11=C-GW11
      C12=B-GW12
```

```
      C21=B-GW21
      C22=C-GW22
      CDET=C11*C22-C12*C21
      C111=C22/CDET
      C112=(-C12)/CDET
      C121=(-C21)/CDET
      C122=C11/CDET
      Z(NX1)=C111*DMGV1+C112*DMGV2
      Z(NX)=C121*DMGV1+C122*DMGV2
      DO 5 I=1,NX2
      WZ=W(I,1)*Z(NX1)+W(I,2)*Z(NX)
    5 Z(I)=V(I)-WZ
      RETURN
      END
      SUBROUTINE NCTRD(U,D,NX)
      DIMENSION U(NX),D(NX),BET(15),GAM(15)
      A=1./6.
      B=2./3.
      C=A
      AC=1./36.
      NX1=NX-1
      NX2=NX-2
      NX3=NX-3
C       I=1
      BET(1)=1.
      GAM(1)=D(1)
C       I=2
      BET(2)=1.
      GAM(2)=D(2)
C       I=3
      BET(3)=1.
      GAM(3)=D(3)-A*GAM(2)
      DO 1 I=4,NX2
      BET(I)=B-AC/BET(I-1)
      GAM(I)=(D(I)-A*GAM(I-1))/BET(I)
    1 CONTINUE
C       I=NX-1
      BET(NX1)=1.
      GAM(NX1)=D(NX1)
C       I=NX
      GAM(NX)=D(NX)
      U(NX)=GAM(NX)
      U(NX1)=GAM(NX1)
      DO 2 I=2,NX3
      J=NX-I
      U(J)=GAM(J)-C*U(J+1)/BET(J)
    2 CONTINUE
      U(2)=GAM(2)
      U(1)=GAM(1)
      RETURN
      END
      SUBROUTINE CYCTRD(D,Z,Nx)
      DIMENSION D(15),Z(15),W(15),V(15),FN(15)
      A=1./6.
      B=2./3.
      NX1=NX-1
      NX2=NX-2
C       CALCULATE W=E(INVERSE)FN
      FN(1)=A
      FN(NX1)=A
      DO 1 I=2,NX2
    1 FN(I)=0.
      CALL TRIDG(W,FN,NX1)
C       CALCULATE V=E(INVERSE)D
      CALL TRIDG(V,D,NX1)
      GW=A*(W(1)+W(NX1))
      GV=A*(V(1)+V(NX1))
      DMGV=D(NX)-GV
      Z(NX)=DMGV/(B-GW)
      DO 2 I=1,NX1
    2 Z(I)=V(I)-Z(NX)*W(I)
      RETURN
      END
      SUBROUTINE TRIDG(U,D,NX)
      DIMENSION U(15),D(15),BET(15),GAM(15)
C
C       THE EQUATIONS ARE
C       A*U(I-1)+B*U(I)+C*U(I+1)=D(I)
C       WITH A=0 FOR I=1
C       AND  C=0 FOR I=NX
      A=1./6.
      B=2./3.
      C=A
      AC=1./36.
      BET(1)=B
      GAM(1)=D(1)/B
      DO 1 I=2,NX
      BET(I)=B-AC/BET(I-1)
      GAM(I)=(D(I)-A*GAM(I-1))/BET(I)
    1 CONTINUE
      U(NX)=GAM(NX)
      NX1=NX-1
      DO 2 I=1,NX1
      J=NX-I
      U(J)=GAM(J)-C*U(J+1)/BET(J)
    2 CONTINUE
      RETURN
      END
      SUBROUTINE DY(R,S,A,NODE,NR,NC,DISTY)
      DIMENSION R(192),S(192),A(192),T(12),U(12),X(12),W(12),Z(12)
      NX=NC+1
      NY=NR+1
      NY1=NY-1
      NY2=NY-2
      NY3=NY-3
      NY4=NY-4
```

```
      COF=-84.*DISTY
      COF2=12.*DISTY
      N0=-NY
      DO 1 I=1,NX
      N0=N0+NY
      N=N0
      DO 2 J=1,NY
      N=N+1
      T(J)=A(N)
    2 X(J)=S(N)
C       FIRST STAGE
      U(1)=0.
      U(2)=(-32.*T(3)+32.*T(1))/COF
      DO 3 J=3,NY2
    3 U(J)=(-5.*T(J+2)-32.*T(J+1)+32.*T(J-1)+5.*T(J-2))/COF
      U(NY1)=(-32.*T(NY)+32.*T(NY2))/COF
      U(NY)=0.
      CALL PENTDG(Z,U,NY)
      Z(1)=(-25.*T(1)+48.*T(2)-36.*T(3)+16.*T(4)-3.*T(5))/COF2
      Z(2)=(-3.*T(1)-10.*T(2)+18.*T(3)-6.*T(4)+T(5))/COF2
      Z(NY1)=(-T(NY4)+6.*T(NY3)-18.*T(NY2)+10.*T(NY1)+3.*T(NY))/COF2
      Z(NY)=(3.*T(NY4)-16.*T(NY3)+36.*T(NY2)-48.*T(NY1)+25.*T(NY))/COF2
C       SECOND STAGE
      T0=4.*T(1)-6.*T(2)+4.*T(3)-T(4)
      TNYP1=4.*T(NY)-6.*T(NY1)+4.*T(NY2)-T(NY3)
      X0=4.*X(1)-6.*X(2)+4.*X(3)-X(4)
      XNYP1=4.*X(NY)-6.*X(NY1)+4.*X(NY2)-X(NY3)
      Z0=(-25.*T0+48.*T(1)-36.*T(2)+16.*T(3)-3.*T(4))/COF2
      ZNYP1=(3.*T(NY3)-16.*T(NY2)+36.*T(NY1)-48.*T(NY)+25.*TNYP1)/COF2
      U(1)=(X0*Z0+X0*Z(1)+X(1)*Z0+X(1)*Z(2)+X(2)*Z(1)+X(2)*Z(2))/12.+
     *X(1)*Z(1)/2.
      DO 4 J=2,NY1
      JM=J-1
      JP=J+1
    4 U(J)=(X(JM)*Z(JM)+X(JM)*Z(J)+X(J)*Z(JM)+X(J)*Z(JP)+X(JP)*Z(J)+X(
     *JP)*Z(JP))/12.+X(J)*Z(J)/2.
      U(NY)=(X(NY1)*Z(NY1)+X(NY1)*Z(NY)+X(NY)*Z(NY1)+X(NY)*ZNYP1+XNYP1*
     *Z(NY)+XNYP1*ZNYP1)/12.+X(NY)*Z(NY)/2.
      CALL NCTRD(W,U,NY)
      N=N0
      W(1)=0.
      W(NY)=0.
      DO 5 J=1,NY
      N=N+1
    5 R(N)=W(J)
    1 CONTINUE
      N=NX*NY
      DO 6 J=1,NY
      N=N+1
    6 R(N)=R(J)
      RETURN
      END
      SUBROUTINE FUNCT(N,XC,FC,GC)
      COMMON/OLD/ XO(540),H0,Z0,E0,ALPHA,BEETA,TG,F(180),NX,NY,DX,DY,
     1PNLTH,PNLTZ,PNLTE,UH,UZ,UE,ETA,EMINN
      COMMON/INVRNT/HMEAN,ZMEAN,ENERGY
      DIMENSION XC(540),GC(540),DZ(540),DH(180)
      M12=N/3
      M21=M12+1
      M22=M12+M12
      M31=M22+1
C       CALCULATE FUNCTION VALUE FC
      SUMSQ=0.
      DO 1 II=1,M22
    1 SUMSQ=SUMSQ+ALPHA*(XC(II)-XO(II))**2
      DO 2 II=M31,N
    2 SUMSQ=SUMSQ+BEETA*(XC(II)-XO(II))**2
C       CALCULATE E,Z AND H
      G=0.5*TG
      CALL LOOK(XC(1),XC(181),XC(361),NX+1,NY,0.,-1,G,M12,F)
      HDIF=HMEAN-H0
      EDIF=ENERGY-E0
      ZDIF=ZMEAN-Z0
C
      FC=SUMSQ+PNLTE*EDIF**2+PNLTZ*ZDIF**2+UZ*ZDIF+UE*EDIF
     *+PNLTH*HDIF**2+UH*HDIF
C
C       CALCULATE DZ/DU AND DZ/DV AND DZ/DH
      TDX=DX*2.
      TDY=DY*2.
      DY2=DY/2.
      DX2=DX/2.
      C1=G*(-DX*DY)/2.
      C2=G*DX2
      C3=G*DY2
      NYM1=NY-1
      AREA=1./(NX*(NY-1))
      AREA2=AREA/2.
      IU=0
      DO 3 I=1,NX
      DO 3 J=1,NY
      IU=IU+1
      IV=IU+M12
      IH=IU+M22
      DZ(IU)=0.
      DZ(IV)=0.
      DZ(IH)=0.
      DH(IU)=AREA2
      IF(J.EQ.1.OR.J.EQ.NY) GO TO 4
      DH(IU)=AREA
      IHMS=IH-NY
      IVMS=IV-NY
      IVNS=IVMS-NY
      IUMP=IU-NY+1
      IUMM=IU-NY-1
      IF(I.NE.1) GO TO 5
```

```
C       CYCLIC X BOUNDS
        IHMS=IHMS+M12
        IVMS=IVMS+M12
        IVNS=IVNS+M12
        IUMP=IUMP+M12
        IUMM=IUMM+M12
      5 IF(I.EQ.2) IVNS=IVNS+M12
        IHPS=IH+NY
        IVPS=IV+NY
        IVQS=IVPS+NY
        IUPP=IU+NY+1
        IUPM=IU+NY-1
        IF(I.NE.NX) GO TO 6
        IHPS=IHPS-M12
        IVPS=IVPS-M12
        IVQS=IVQS-M12
        IUPP=IUPP-M12
        IUPM=IUPM-M12
      6 IF(I.EQ.(NX-1)) IVQS=IVQS-M12
        DZ(IV)=((((XC(IV)-XC(IVNS))/TDX-(XC(IUMP)-XC(IUMM))/TDY+F(J))
       1/XC(IHMS))-(((XC(IVQS)-XC(IV))/TDX-(XC(IUPP)-XC(IUPM))/TDY+
       2F(J))/XC(IHPS)))*C3
        DZ(IH)=((XC(IVPS)-XC(IVMS))/TDX-(XC(IU+1)-XC(IU-1))/TDY+F(J))**2
       */(XC(IH)**2))*C1
      4 DZUL=0.
        DZUR=0.
        IF(J.GE.NYM1) GO TO 7
        IVPP=IV+NY+1
        IVMP=IV-NY+1
        IF(I.EQ.1) IVMP=IVMP+M12
        IF(I.EQ.NX) IVPP=IVPP-M12
        DZUL=C2*((XC(IVPP)-XC(IVMP))/TDX-(XC(IU+2)-XC(IU))/TDY+
       1F(J+1))/XC(IH+1)
        IF(J.LE.2) GO TO 8
      7 IVPM=IV+NY-1
        IVMM=IV-NY-1
        IF(I.EQ.1) IVMM=IVMM+M12
        IF(I.EQ.NX) IVPM=IVPM-M12
        DZUR=C2*((XC(IVPM)-XC(IVMM))/TDX-(XC(IU)-XC(IU-2))/TDY+
       1F(J-1))/XC(IH-1)
      8 DZ(IU)=DZUL-DZUR
      3 CONTINUE
C       CALCULATE DF/DU
        TALPHA=2.*ALPHA
        TBETA=2.*BEETA
        C2=DX*DY
        C3=C2*EDIF/G
        C1=2.*C3
        C4=UE*DX*DY/G
        C5=C4/2.
        DO 10 IU=1,M12
        IH=IU+M22
     10 GC(IU)=TALPHA*(XC(IU)-XO(IU))+PNLTE*C1*XC(IU)*XC(IH)+2.*ZDIF*
       1DZ(IU)*PNLTZ+C4*XC(IU)*XC(IH)+UZ*DZ(IU)
C       CALCULATE DF/DV
        DO 11 IV=M21,M22
        IH=IV+M12
     11 GC(IV)=TALPHA*(XC(IV)-XO(IV))+PNLTE*C1*XC(IV)*XC(IH)+2.*ZDIF*
       1DZ(IV)*PNLTZ+C4*XC(IV)*XC(IH)+UZ*DZ(IV)
C       CALCULATE DF/DH
        DO 12 IU=1,M12
        IV=IU+M12
        IH=IU+M22
        UVH=XC(IU)**2+XC(IV)**2+TG*XC(IH)
     12 GC(IH)=TBETA*(XC(IH)-XO(IH))+PNLTH*2.*HDIF*DH(IU)+UH*DH(IU)
       *+2.*ZDIF*DZ(IH)*PNLTZ+UZ*DZ(IH)+PNLTE*UVH*C3+C5*UVH
        RETURN
        END
        SUBROUTINE SMOOTH(ZK,TEM,NX,NY)
        DIMENSION C(3,2),ZK(12,15),TEM(15)
        DATA ((C(I,J),I=1,3),J=1,2)/3.8798,-1.77097,0.331065,0.375,0.25,
       1 0.0625/
        NX1=NX-1
        NY1=NY-1
        DO 300 KK=1,2
        C1=C(1,KK)
        C2=C(2,KK)
        C3=C(3,KK)
C       SMOOTH IN Y-DIRECTION
        DO 301 I=1,NX
        DO 3005 J=2,NY1
        IF(J EQ.2) GO TO 25
        IF(J.EQ.NY1) GO TO 35
        GO TO 40
     25 TWO=ZK(I,J+2)+2.*ZK(I,J-1)-ZK(I,J)
        GO TO 50
     35 TWO=2.*ZK(I,J+1)-ZK(I,J)+ZK(I,J-2)
        GO TO 50
     40 TWO=ZK(I,J+2)+ZK(I,J-2)
     50 CONTINUE
        TEM(J)=C1*ZK(I,J)+C2*(ZK(I,J-1)+ZK(I,J+1))+TWO*C3
   3005 CONTINUE
        DO 3006 J=2,NY1
   3006 ZK(I,J)=TEM(J)
    301 CONTINUE
C       SMOOTH IN X-DIRECTION
        DO 303 J=1,NY
        DO 302 I=2,NX1
        IF(I.EQ.2) GO TO 170
        IF(I.EQ.NX1) GO TO 185
        GO TO 200
    170 TWO=2.*ZK(I-1,J)-ZK(I,J)+ZK(I+2,J)
        GO TO 230
    185 TWO=ZK(I-2,J)+2.*ZK(I+1,J)-ZK(I,J)
        GO TO 230
```

```
  200 TWO=ZK(I-2,J)+ZK(I+2,J)
  230 TEM(I)=C1*ZK(I,J)+C2*(ZK(I-1,J)+ZK(I+1,J))+C3*TWO
  302 CONTINUE
      DO 3025 I=2,NX1
 3025 ZK(I,J)=TEM(I)
  303 CONTINUE
  300 CONTINUE
      RETURN
      END
      SUBROUTINE MONIT(N,XC,FC,GC,NCALL,TEST)
      LOGICAL TEST
      COMMON/OLD/ XO(540),H0,Z0,E0,ALPHA,BETA,TG,F(180),NX,NY,DX,DY,
     1PNLTH,PNLTZ,PNLTE,UH,UZ,UE,ETA,EMINN
      COMMON/INVRNT/HMEAN,ZMEAN,ENERGY
      DIMENSION XC(540),GC(540)
      SUMSQ=0.
      NXNY=NX*NY
      GNORM=0.
      DO 1 IU=1,NXNY
      IV=IU+NXNY
      IH=IV+NXNY
      SUMSQ=SUMSQ+ALPHA*((XC(IU)-XO(IU))**2+(XC(IV)-XO(IV))**2)+BETA*
     1(XC(IH)-XO(IH))**2
      GNORM=GNORM+GC(IU)**2+GC(IV)**2+GC(IH)**2
    1 CONTINUE
      GNORM=SQRT(GNORM)
      IH=NXNY+NXNY
      G=TG/2.
      CALL LOOK(XC(1),XC(181),XC(361),NX+1,NY,0.,-1,G,NXNY,F)
      HDIF=HMEAN-H0
      EDIF=ENERGY-E0
      ZDIF=ZMEAN-Z0
      ENORM=SQRT(ZDIF**2+EDIF**2)
      EENRM=ETA*ENORM
      TEST=(GNORM.LE.AMAX1(EENRM,EMINN))
      PRINT 2,NCALL,SUMSQ,GNORM,HDIF,ZDIF,EDIF,ENORM,EENRM,EMINN
    2 FORMAT("0SUMSQ AT CALL ",I5," = ",E12.4," , GNORM = ",E12.4/
     *" ERRORS    , H, Z \& E : ",3E14.6/" NORM = ",E14.6,10X,
     *" ETA * NORM = ",E14.6,10X," MIN. VALUE = ",E14.6)
      IF (TEST) EMINN=AMAX1(EENRM,EMINN)
      RETURN
      END
```