Computers & Geosciences Vol. 16, No. 5, pp. 645–667, 1990
Printed in Great Britain.

0098-3004/90 $3.00 + 0.00

FE2DY: A FINITE ELEMENT FORTRAN PROGRAM FOR THE SOLUTION OF THE SHALLOW-WATER EQUATIONS WITH ENERGY CONSERVATION

J. STEPPELER
National Meteorological Center, Camp Springs MD, Washington, DC 20233, U.S.A.

(*Received* 8 *February* 1989; *accepted* 17 *October* 1989)

Abstract—A FORTRAN program is presented for the solution of the shallow-water equations. An efficient solution of the Galerkin equation is achieved by factorizing in the X- and Y-directions. The direct method used to solve the mass matrix equation is efficient as only one side diagonal occurs. The model is applied to the computation of a Rossby wave. A special combination of weights and internal projections makes the scheme energy conserving.

This feature increases the nonlinear stability of the model, as compared to nonconserving schemes.

Key Words: Shallow-water equations, Finite elements, Direct method, Initial value problem, FORTRAN 77.

INTRODUCTION

The conservation of integral constraints of the continuous equations by discretizations is considered important in numerical models of the atmosphere using finite difference schemes (Arakawa, 1966; Sadourny, 1975).

Galerkin approximations are becoming increasingly important for atmospheric models. The spectral method (Machenhauer, 1979; Orszag, 1970) is used widely. An alternative Galerkin method is the finite element method. Reviews of the application of finite element schemes in meteorology were given by Navon (1977) and Staniforth (1982).

The finite element scheme has some advantage over simple finite difference methods, because of its improved accuracy, as discussed by Cullen (1973). Finite element models for meteorological models were proposed by Cullen (1973), and an atmospheric model based on the finite element method is investigated by Carson and Cullen (1977). Energy conserving finite element schemes for vertical discretizations were introduced by Steppeler (1987). A comparison with a nonconserving scheme was performed by Steppeler (1986b), and the impact of using a conserving scheme was determined to be substantial.

For horizontal discretization by finite element schemes, energy conservation was discussed by Fix (1975), Jesperson (1974), and Cliffe (1981). It turned out that standard finite element schemes lead to energy conservation for the nondivergent equations. For the divergent equations, which are discussed here, special finite element schemes have to be used in order to obtain conservation properties.

It has become customary to test horizontal discretization schemes on the shallow-water equations. These equations describe a single homogenous layer of fluid. They are applied to a variety of problems, such as tidal flow, earthquake waves in oceans, etc. In meteorology they are used to forecast the geopotential. Even as meteorological centers adopted the more expensive three-dimensional models, the shallow-water equations remain popular as a method of testing numerical schemes.

A triangular finite element scheme for the shallow-water equations was proposed by Navon (1979). Conservation of integral constraints is not intrinsic for this model. Navon (1981, 1983) gives a method to achieve conservation of energy and potential enstrophy by restoring these constraints by a variational method. A full description of the method and the code used to implement is given in Navon (1987).

This paper uses an intrinsic method to obtain energy conservation, as proposed by Steppeler (1986) for the shallow-water equation. It uses rectangular elements as proposed by Cullen (1974) and Staniforth and Mitchell (1977).

A high numerical efficiency of rectangular finite element models can be achieved when composing the Galerkin operators out of one-dimensional ones (see Staniforth and Beaudoin, 1986). The computation of the Galerkin integral is done using a collocation grid of a higher resolution than the main nodepoint grid, as proposed by Staniforth and Beaudoin (1986) and applied in a numerical model by Steppeler (1988). The program confirms to FORTRAN 77. It has been run on Cyber 205 and Cray XMP 48. The CPU time used on the Cray was

3×10^{-4} sec per gridpoint and timestep and the program needed 1000 words of memory for the program using an 8×8 grid.

The performance of the model on supercomputers will depend on the degree of vectorization achieved. This normally will require the addition of some machine dependent features. This program demands large grids in order to obtain a good vector length. With an 80×80 grid it reached a performance of 3×10^{-5} sec per timestep and gridpoint. Different performance on different grids reflects only the efficiency of the code because of vectorization. The number of operations per gridpoint and timestep does not depend on the grid, as opposed to other highly accurate schemes, such as the spectral method.

GRIDS AND BASIC SOLUTION PROCEDURES

The divergent shallow-water equations to be discretized are:

$$\begin{aligned} \dot{U} &= fV - UU_X - VU_Y - gH_X \\ \dot{V} &= -fU - UV_Y - VV_Y - gH_Y \\ \dot{H} &= -(UH)_X - (VH)_Y \end{aligned} \tag{1}$$

where U, V are the velocity components, H is the height field, g the constant of gravity, and f the Coriolis parameters.

The finite element Galerkin scheme used for the solution of Equation (1) is given by Steppeler (1986a), and a detailed description of the finite difference formula can be obtained in Steppeler (1989), where also the results of test integrations and comparisons with nonconserving finite element schemes are reported.

For the convenience of the reader a short description of the method and special implementation features are given in the Appendix. Here a description of the three basic operations for the implementation of the program, interpolation, integration, and Gauss elimination, will be given.

The Galerkin scheme uses linear basis functions on a regular rectangular grid. The representation of the fields U, V, H is by nodepoint values $U_{\nu,\mu}, V_{\nu,\mu}, H_{\nu,\mu}$ on the regular grid $X_\nu Y_\mu$. This grid is termed the nodepoint grid and is shown in Figure 1A. If n indicates the time level, the representation of the basic fields is $U^n_{\nu,\mu}, V^n_{\nu,\mu}, H^n_{\nu,\mu}$, the square X_ν, $Y_\mu, X_{\nu+1}$, $Y_\mu, X_{\nu+1}, Y_{\mu+1}, X_\nu, Y_{\mu+1}$, which is shown in Figure 1A, is termed an element. Inside each element we assume bilinear interpolation for the fields U, V, H. This is used to generate more gridpoint values on finer meshes, the so-called collocation grids.

Using the variable ξ to denote either X or Y, the linear interpolation formula for an interval (a, b) can be written as

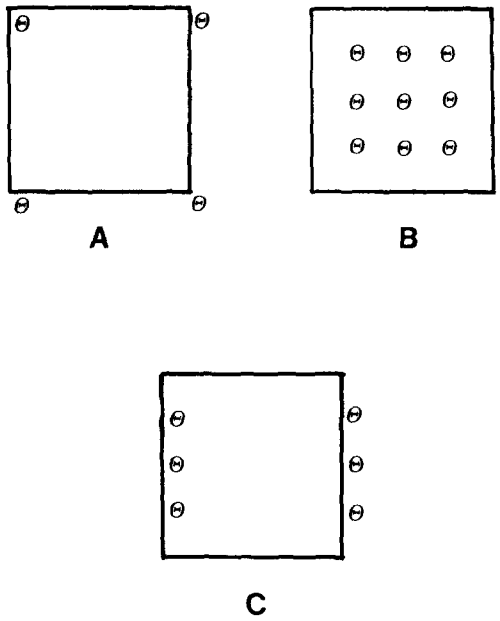

Figure 1. Grids used for field representation of field: A, main nodepoint grid; B, collocation grid; C, Y-collocation grid.

$$\psi(\xi) = -\frac{\psi(a)(b-\xi) + \psi(b)(\xi - a)}{b-a} \tag{2}$$

where ψ stands for U, V, or T.

Other variables, on which ψ may depend, are kept fixed in Equation (2). We always will interpolate to the three Gauss-points (see Abramowitz and Stegun, 1970) belonging to the interval (a, b). These are the three values $\xi_{\rho 1}$, $\rho = 1, 2, 3$.

$$\xi_\rho = \frac{b-a}{2}\xi'_\rho + \frac{b+a}{2} \tag{3}$$

with $\xi'_\rho = 0, \pm 0.774596669241483$.

Using Equation (2) for interpolation in the Y-direction ($\xi = Y$) and taking $(Y_\mu, Y_{\mu+1})$ as the interpolation interval for all X, the interpolation will lead from the nodepoint grid (Fig. 1A) to the Y-collocation grid shown in Figure 1C. Using now interpolation in the X-direction, Equation (2) with $\xi = X$ will give interpolated fields in the collocation grid, shown in Figure 1B.

In a similar way derivatives in the collocation and Y-collocation grids can be computed.

$$\psi_\xi(\xi) = \frac{\psi(b) - \psi(b)}{b-a}. \tag{4}$$

Using the interpolated fields and derivatives, the right-hand side of Equation (1) can be computed on the collocation grid (Fig. 1B). The integration and Gaussian elimination step are used to transform this back to the Y-collocation, and then to the nodepoint grid.

Let again ξ denote either the X or the Y variable, and let $\psi(\xi)$ be a function obtained from a two-dimensional function $\psi(X, Y)$ by keeping the other variable fixed. When considering $\psi(\xi)$ on an interval (a, b), the definition of $\psi(\xi)$ is obtained from $\psi(X, Y)$ by specifying a line in the X–Y plane, which is either parallel to X or Y. For the purpose of integration we will use only lines contained entirely in one of the elements, that is the squares in Figure 1. We assume that the length of the interval (a, b) is $b - a = \Delta X = X_{\nu+1} - X_\nu = Y_{\nu+1} - Y_\nu$, and the integral of any function $\psi(\xi)$ can be computed as:

$$\int_a^b \psi(\xi)\,d\xi = \sum_{\rho=1}^{3} \psi(\xi_\rho) g_\rho \cdot (b-a) \tag{5}$$

with ξ_ρ being the Gaussian points of the interval (a, b) according to Equation (3) and the g_ν are the corresponding Gaussian integration weights.

$$g_1 = g_3 = 0.55555\ 55555\ 55555$$
$$g_2 = 0.88888\ 88888\ 88888. \tag{6}$$

In order to define the Galerkin integrals we need to define the one-dimensional basis functions $e_\nu(\xi) \cdot e_\nu(\xi)$ is defined as the piecewise linear function defined by the properties

$$e_\nu(\xi_\nu) = 1$$
$$e_\nu(\xi_\mu) = 0 \quad \text{for } \mu \neq \nu. \tag{7}$$

In Equation (7) ξ is either X or Y and ξ_ν is either X_ν or Y_ν.

If $RS(\xi)$ is the function occurring in the right-hand side of (1), restricted to a line parallel to either the X- or the Y-axis the Galerkin integral associated with the index is defined as

$$(G^i RS)_\nu = \int e_\nu(\xi) RS(\xi) w(\xi)\,d\xi. \tag{8}$$

In Equation (8) $w(\xi)$ is a weight function which will be selected to be either 1 or H.

Because according to Equation (7), $e_\nu(\xi)$ is different for 0 in the two intervals $(\xi_{\nu-1}, \xi_\nu)$ and $(\xi_\nu, \xi_{\nu+1})$ the integral in Equation (8) will have two parts of interval length ΔX, which are computed according to Equation (5) as a sum over three collocation points.

If $\xi = X$, and RS is given in the collocation grid (Fig. 1B), with Y being kept fixed, one can see from Figure 1 that the formation of the Galerkin integrals transforms from the collocation grid to the Y-collocation grid.

If $\xi = Y$, and RS is given in the Y-collocation grid (Fig. 1C), with X being kept fixed, Figure 1 shows that the formation of the Galerkin integrals transform from the Y-collocation grid to the nodepoint grid.

The Gaussian elimination procedure will transform the Y-collocation grid as well as the nodepoint grid into itself. It works on Galerkin integrals $(G^i RS)_\nu$ as defined in Equation (8). The index ν can be either in the X- or Y-direction, indicating X_ν or Y_ν. The corresponding gridlength is always ΔX.

$(G^i RS)_\nu$ will depend on another position Y or X, which is kept fixed and this dependency is dropped for simplicity. The Gaussian elimination will lead to values $(G^i RS)_\nu$ and is defined by the solution of the linear equation system

$$\alpha_\nu (G\,RS)_{\nu-1} + \beta_\nu (G\,RS)_\nu + \gamma_\nu (G\,RS)_{\nu-1} = (G^i RS)_\nu, \tag{9}$$

with

$$\alpha_\nu = \int d\xi e_{\nu-1}(\xi) e_\nu(\xi) w(\xi)$$
$$\beta_\nu = \int d\xi e_\nu(\xi) e_\nu(\xi) w(\xi)$$
$$\gamma_\nu = \int d\xi e_{\nu+1}(\xi) e_\nu(\xi) w(\xi). \tag{10}$$

w in Equation (10) is the same weight function as used in the computation (8) of the Galerkin integrals. The computation of the integrals in Equation (10) is done using Equation (5). The solution of the sparse matrix Equation (9) can be done efficiently using Gaussian elimination, as described by Ahlberg, Nilson, and Walsh (1967).

The operations described here, interpolation, integration, and Gaussian elimination are used in two versions, representing X- and Y-directions. Furthermore, the subroutines representing integration and Gaussian elimination are represented using $W = 1$ and $w = H$ in Equations (8) and (10). Some of these operations will transform field representations between the different grids shown in Figure 1. Representations of a field ψ in different grids correspond to a different number of indices in the program. Because an interval $(X_\nu, X_{\nu+1})$ or $(Y_\nu, Y_{\nu+1})$ can carry three collocation points, some of the indices will have the range 3. A field ψ will be represented in the nodepoint grid by $\psi_{\nu,\mu}$, in the Y-collocation grid by $\psi_{\nu,\mu,\rho}$, $\rho = 1, 2, 3$ and in the collocation grid as $\psi_{\nu,\chi,\mu,\rho}$, $\chi, \rho = 1, 2, 3$. The basic representation of the fields U, V, H is in the nodepoint grid. If RS is any function of the fields U, V, T and their derivatives, such as the right-hand side of Equation (1), or any part of it, the operations described can be used to compute a representation of RS in the nodepoint grid, $(G\,RS)_{\nu,\mu}$ which is termed the Galerkin projection of RS. In the course of this computation intermediate results will be transformed between the different grids shown in Figure 1. The following operations are done in order to compute $(G\,RS)_{\nu,\mu}$.

(a) Interpolation in the Y-direction of the fields U, V, T (transforms from nodepoint grid to the Y-collocation grid).

(b) Interpolation in the X-direction of the fields U, V, T (transforms from the Y-collocation grid to the collocation grid).
(c) Compute RS in the collocation grid (transforms from the collocation grid on to itself).
(d) Form Galerkin-integral in the X-direction (transforms from the collocation grid to the Y-collocation grid).
(e) Gaussian elimination in the X-direction (transforms from the Y-collocation grid on to itself).
(f) Form Galerkin integrals in the Y-direction (transforms from the Y-collocation grid to the nodepoint grid).
(g) Gaussian elimination in the Y-direction (transforms the nodepoint grid on to itself).

The Galerkin projection, as computed according to steps (a)–(g), is the basis of the discretization of Equation (1). A short description of the numerical procedure is given in the Appendix.

DESCRIPTION OF THE PROGRAM FE2DY

Initial values

The program is realized on a rectangular grid. The number of gridpoints is given by the variable L in a parameter statement. LCOL, given also in a parameter statement, is the number of collocation points. LCOL = 3 is used here, a choice which will make the computation of the Galerkin integrals exact. L can easily be changed in order to adjust the resolution of the model. The parameter implemented here is $L = 8$. Solid wall boundary conditions are implemented in the Y-direction, and periodic boundaries in the X-direction. This represents physically a channel around the Earth at midlatitudes. This is a system used frequently to test atmospheric models (see for example Navon, 1981).

The grid parameters selected are

$$\Delta X = \frac{4 \times 11}{7} 10^5 \text{ m}$$
$$f = 10^{-4} \text{ sec}^{-1}$$
$$\Delta t = 900 \text{ sec}. \tag{11}$$

The Coriolis parameter is considered as constant in space.

The initial values were the same as used by Grammeltvedt (1969)

$$H(X, Y) = H_0 + H_1 \tanh \frac{9(Y - Y_0)}{2D} + H_2 \frac{1}{\cosh^2\left(\frac{9Y - Y_0)}{D}\right)} \sin 2\pi x . \tag{12}$$

These initial values since have been used by a number of researchers (see for example Gustafsson, 1971).

The parameters used in Equation (12) are:

$$gH_0 = 20{,}000 \text{ m}^2 \text{ sec}^{-2}$$
$$gH_1 = 4400 \text{ m}^2 \text{ sec}^{-2}$$
$$gH_2 = 2660 \text{ m}^2 \text{ sec}^{-2}$$
$$L = D = 4400 \text{ km}. \tag{13}$$

They differ slightly from those used by Grammeltvedt (1969) and produce a stronger perturbation of the basic flow. The initial velocity field is defined by the geostrophic relations

$$U = -\frac{1}{f}\frac{\partial}{\partial Y} H$$
$$V = \frac{1}{f}\frac{\partial}{\partial X} H. \tag{14}$$

Description of the common blocks

Block COMGAU contains all variables describing the Gaussian collocation grid. XCOL and WCOL are Gaussian points and weights, as given by Abramowitz and Stegun (1970). EMU and EMUP1 contain the values of the one-dimensional basis functions $e_\mu(X)$ and $e_{\mu+1}(X)$ defined in Equation (2) at the collocation points belonging to the interval $(X_{\mu+1}, X_{\mu+1})$.

Block DYNVAR contains storage for the dynamical variables and their time derivatives in the main nodepoint grid (see Fig. 1A). Fields U, V, H belong to the time level $n\Delta t$. $U1$, $V1$, $H1$ are the corresponding fields at $(n-1)\Delta t$, and UT, VT, HT are the time derivatives of these fields at time $n\,\Delta t$. The parameters f(Coriolis parameter) and DX (grid length) also are stored in this block. X stores the grid values of the main nodeprint grid.

Block DY1 stores field values and field derivatives in the collocation grid (see Fig. 1B). UX, VX, HX, UY, VY, HY are X- and Y-derivatives of U, V, H. UTC, VTC, HTC are time derivatives of U, V, H. UC, VC, HC are the field values of U, V, H in the collocation grid.

Block Y1PO contains field values and derivatives in the Y-collocation grid. $UCCY$, $VCCY$, $HCCY$ are Y-derivatives of U, V, H. $UTCC$, $VTCC$, $HTCC$ are time derivatives of U, V, H and UCC, VCC, HCC are field values of U, V, H in the Y-collocation grid.

The main program FE2DY

The main program initializes parameters and fields and performs the time stepping loop. Subroutine PRF is used to print the H-field. More advanced system dependent plotting routines can be included in this routine. The time variable is N, and the label of the time loop is 100. The main numerical work is carried out in subroutine TEND which computes the time derivatives of the fields U, V, H. These then are used to perform the leapfrog timestep. The fields for time-level $N-1$($U1, V1, H1$) are overwritten with the values belonging to $N+1$. The time levels N and $N+1$ then are interchanged.

The variable *NE* can be used to control the length of the time-loop. The parameter EPS can be used to control the amount of spatial smoothing. NPLOCR is the time difference (in steps) between prints of the field *H*.

At every timestep integral invariants and the kinetic energy EKIN are printed. The integrals invariants printed are total energy *E*, defined by

$$E = \tfrac{1}{2}\mathrm{d}X\,\mathrm{d}Y H(U^2 + V^2 + H) \tag{15}$$

the potential estrophy *Z*, defined by

$$Z = \int \mathrm{d}X\,\mathrm{d}Y H\Theta^2$$

$$\Theta = \frac{V_X - U_Y + f}{H}, \tag{16}$$

where *H* is absolute vorticity, and the total mass *HM*, defined by

$$HM = \int H\,\mathrm{d}X\,\mathrm{d}Y. \tag{17}$$

The invariants are normalized by their values at the initial time. This is stored at timestep $N = 1$ on the variables *EE*, *ZZ*, and *HMM*.

The computation of the integral constraints is done in subroutine ENER. The spatial smoothing is carried out in the subroutine DIFF.

Initial values are produced by the FORTRAN loops 40 and 41. This is the only place in the program to introduce input data. More general read statements can be introduced here, if necessary.

Description of the subroutines

SUBROUTINE DIFF (EPS) provides a second-order smoothing operation of fields *U*, *V*, *H*. The operator is defined in Equation (A4).

It is designed to damp also the computational mode of the leapfrog scheme. The parameter EPS defines the amplitude of the diffusion.

Input are the arrays U, V, H, U1, V1, H1 all dimensioned by (L, L) from common block DYNVAR. Subroutine PRF prints the field *H*. More refined mesh dependent plotting routines can be inserted here. Input is the array H, dimensioned by (L, L) from common block DYNVAR.

SUBROUTINE ENER (E, Z, HM, EK) computes some internal invariants and the kinetic energy, as defined by Equations (15)–(17). The method used is the integration in the collocation grid. Input values are the field values *UC*, *VC*, *HC* in the collocation grid dimensioned C, LCOL (L, LCOL, L, LCOL) from block DY1. Output are the total energy *E*, the potential enstrophy *Z*, total mass *HM* and kinetic energy *EK*.

SUBROUTINE TEND does the main numerical work by computing the time derivatives of the fields *U*, *V*, *H*.

Output is on storage arrays UT, VT, HT dimensioned (L, L) in block DYNVAR, representing $\dot{U}$, $\dot{V}$, $\dot{H}$ in the main nodepoint grid (Fig. 1A). Input is on storage arrays U, V, H dimensioned (L, L) from block DYNVAR, representing the fields at time level *n* in the main nodepoint grid. Storage arrays in blocks DY1 and YIPO are used as intermediate storage. These values describe the fields *U*, *V*, *H* on the collocation grid (Fig. 1B) and the *Y*-collocation grid (Fig. 1C).

A number of subroutines are used to interpolate between the grids.

IPOLY is called upon to interpolate a field in the *Y*-collocation grid. IPOLY transforms from the grid A in Figure 1 to grid C.

IPOLX is called upon to interpolate a field in the *X*-direction. IPOLX transforms from grid C in Figure 1 to grid B.

DIFX computes the *X*-derivative ϕ_X of a field ϕ. It transforms from grid C in Figure 1 to grid B.

The dynamic equations are implemented according to Equation (A1).

The time derivatives of the fields are then Galerkin projected to give main nodepoint values of the field time derivatives.

SCALHX computes the Galerkin integrals of ϕ_i in *X*-direction. It transforms from the grid B in Figure 1 to the grid C. The weight $w = 1$ is used.

SCALUX is the same routine using the weight $w = H$.

GAUHX performs the Gaussian elimination in the *X*-direction. It operates on the grid B of Figure 1. The weight $w = 1$ is used.

GAUUX is the same routine using $w = H$.

SCALHY is called to compute the Galerkin integrals in *Y*-direction. It transforms from grid B in Figure 1 to grid A. The weight $w = 1$ is used.

SCALUY and GAUUY are the same routines as SCALHY and GAUHY, but using the weight $w = H$.

SUBROUTINE GAUO3IN (W, WI, NLEV, NBAND) initializes the Gaussian elimination procedure. The Gaussian elimination solves equations of the form

$$\mathbb{A}\,\mathbf{X} = \mathbf{B} \tag{18}$$

where $\mathbb{A}$ is a banded matrix.

Input are the array **W**, which contains the diagonals of the matrix $\mathbb{A}$, NLEV, containing the dimensional of the vector **X**, and NBAND, containing the number of side diagonals different from *O* above the diagonal of *A*. Output is the array W, and WI. The dimension of *WI* is (NLEV), and that of *W* is (2*NBAND + 1, NLEV).

GAUO3IN operates only on the matrix *A*, and not on the right-hand side of Equation (9). It has to be called only once if several equations of the form Equation (9) are to be solved involving the same matrix *A*.

SUBROUTINE GAUO3 (W, WI, NBAND, B, NLEV) performs a Gaussian elimination. The routine GAUO3IN has to be called before. W, W1, NBAND, NLEV have the same meaning as in

GAUO3IN. The array B dimensioned (NLEV) contains the right-hand side $\mathbf{B} = \{(G^{\dagger}RS)_v\}$ before the call GAUO3, and the solution vector **X** after the call.

SUBROUTINE IPOLY (UQ, UCQ, UXQ) interpolates a field in the Y-direction. Input is the array UQ dimensioned L, L, UQ, representing a field in grid A of Figure 1. Output are the arrays UCQ and UXQ dimensioned (L, L, LCOL). They represent the field and its X-derivative in grid C of Figure 1.

SUBROUTINE IPOLX (UQQ, UQ) interpolates a field in the X-direction. Input is the array UQQ dimensioned (L, L, LCOL). It represents a field in grid C of Figure 1. Output is the array UQ, dimensioned (L, LCOL, L, LCOL). It represents the field in grid B.

SUBROUTINE DIFX (UQQ, UQX) computes the X-derivative of a field. Input is the array UQQ, dimensioned (L, L, LCOL). It represents the field in grid C of Figure 1. Output is UQX, dimensioned (L, LCOL, L, LCOL) It represents the X-derivative of the field in grid B.

SUBROUTINE GAUUY (QT) performs a Gauss elimination in the Y-direction using the weight $w = H$. Input and output are the array QT, dimensioned (L, L). It represents a field in grid A of Figure 1. The matrix $\mathbb{A}$ of Equation (18) is computed using HCC from common block YIPO. Subroutines GAUO3IN and GAUO3 are called to prepare the elimination and to perform an elimination step on a column of data.

SUBROUTINE GAUHY (QT) is the same as GAUUY except using the weight $w = 1$.

SUBROUTINE GAUUX (QTQQ) performs a Gaussian elimination in the X-direction using the weight $w = H$. Input and output is the array QTQQ dimensioned (L, L, LCOL). It represents a field in grid C of Figure 1. GAUO3IN is called to initialize a Gaussian elimination, and GAUO3 is called to perform an elimination on a row. Periodic boundary conditions are accounted for by an adjustment procedure.

SUBROUTINE GAUHX (QTQQ) is the same as GAUUX, but using the weight $w = 1$.

SUBROUTINE SCALUX (HTQ, HTQQ) computes the scalar product in the X-direction with weight $w = H$. Input is the array HTQ, dimensioned (L, LCOL, L, LLCOL). It represents a field in grid B of Figure 1. Output is the array HTQQ, dimensioned (L, L, LCOL). It represents the field in grid C.

```
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1
 0  0  0  0 -1 -1  0  0
 2  2  1  0  0  0  0  2
 3  3  2  2  2  2  2  3
 3  3  3  3  3  3  3  3
```

Figure 2. Printout of initial H-field.

SUBROUTINE SCALHX (HYQ, HTQQ) is the same as SCALUX, but using the weight $w = 1$.

SUBROUTINE SCALUX (HTQQ, QT) computes the scalar product in the Y-direction, using the weight $w = H$. Input is the array QTQQ, dimensioned (L, L, LCOL). It represents a field in grid C of Figure 1. Output is the array QT, dimensioned (L, L). It represents the field in grid A.

The output of the model

The output routines used are rather simple and should be supplemented by system dependent plotting routines. The print of the H field is controlled by the variable NPLOCO. Figure 2 shows as an example the print of the initial H-field. The integral invariants are printed every timestep. A sample of this output is shown in Figure 3. The invariants are normalized by their initial value. Therefore a value of 1 would indicate exact conservation. Figure 4 shows plots of the initial H field and of the forecasted H field after 121 timesteps. The solution is similar to those obtained with the schemes investigated by Grammeltved (1969), but uses a much coarser resolution.

The main difference in performance between energy conserving and nonconserving finite element schemes is the increased nonlinear stability of the former scheme. Without diffusion (EPS = 0 in subroutine DIFF) the model as described here remains stable for 35 days. An extensive comparison of the model with a nonconserving model was given by Steppeler, Navon and Lu (1989). With EPS = 0 the nonconserving model remained stable for only 25 days. The minimum EPS-parameter necessary to obtain longterm stable model integration is EPS = 1/4000 for the nonconserving model and 1/10,000 for the conserving model as described in this paper.

Acknowledgment—The work was performed while the author was a visitor at the Supercomputer Computations Research Institute at Florida State University.

```
N,E=  1   1.00000000   0.03306268   1.00000000   1.00000000
N,E=  2   1.00004059   0.03296607   1.00006892   1.00000095
N,E=  3   1.00002968   0.03297248   1.00060751   1.00000090
N,E=  4   1.00003819   0.03300946   1.00129143   1.00000184
N,E=  5   1.00005473   0.03316058   1.00229862   1.00000176
```

Figure 3. Printout of integral constraints. Printed quantities are timestep, energy, kinetic energy, potential enstrophy, and total mass. Initial values of energy and potential enstrophy are used for normalization.

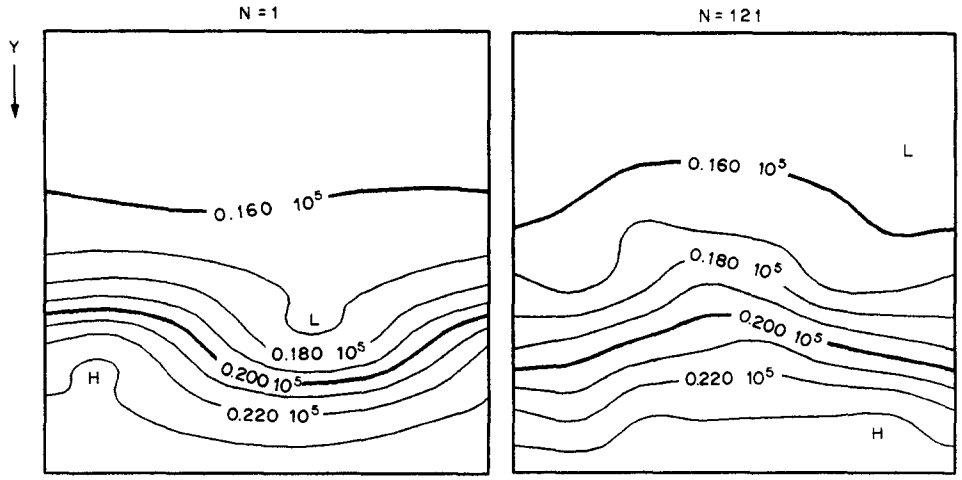

Figure 4. Initial H-field and forecast after 121 time steps. Y-coordinate is plotted downward, and X-coordinate to right.

REFERENCES

Abramowitz, M., and Stegun, I. A., 1970, Handbook of mathematical functions: Dover Publ., New York, 1046 p.

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

Arakawa, A., 1966, Computational design for long term numerical integration of the equation of fluid motion, 1. Two-dimensional incompressible flow: Jour. Comput. Phys., v.1, no. 1, p. 119–143.

Carson, D. J., and Cullen, M. J. P., 1977, Intercomparison of short-range numerical forecasts using finite-difference and finite-element models from the UK Meteorological Office: Beitr. Phys. Atmosph., v. 50, no. 1, p. 1–15.

Cliffe, K. A., 1981, On conservative finite-element formulations of the inviscid Boussinesq equations: Intern. Jour. Num. Meth. Fluid, v. 1, no. 1, p. 117–127.

Cullen, M. J. P., 1973, A simple finite-element method for meteorological problem: Jour. Inst. Math. Applics., v. 11, no. 1, p. 15–31.

Cullen, M. J., 1974, A finite-element method for a non-linear initial value problem: Jour. Math. Applics., v. 13, no. 2, p. 233–247.

Fix, G., 1975, Finite-element models for ocean circulation problems: SIAM Jour. Appl. Math., v. 29, no. 3, p. 371–387.

Grammeltvedt, A., 1969, A survey of finite-difference schemes for the primitive equations for a barotropic fluid: Mon. Wea. Rev., v. 97, no. 5, p. 384–404.

Gustafsson, B., 1971, An alternative direction implicit method for solving the shallow-water equations: Jour. Comput. Phys., v. 7, no. 2, p. 239–251.

Jesperson, D. C., 1974, Arakawa's method is a finite-element method: Jour. Comput. Phys., v. 16, no. 1, p. 383–390.

Machenhauer, B., 1979, The spectral method: GARP Publ., Series, v. 2, no. 17, p. 124–275.

Navon, I. M., 1977, A survey of finite-element methods in quasi-linear fluid flow problems: WISK Rept., no. 140, Nat. Research Inst. for Math. Sciences, Pretoria, South Africa, 44 p.

Navon, I. M., 1979, Finite-element simulation of the shallow-water equations model on a limited area domain: Appl. Math. Modell., v. 3, no. 1, p. 337–348.

Navon, I. M., 1981, Implementation of *a posteriori* method for enforcing conservation of potential enstrophy and mass in discretized shallow-water equation models: Mon. Wea. Rev., v. 109, p. 946–959.

Navon, I. M., 1983, Combined penalty multiplier optimization methods to enforce integral invariants conservation: Mon. Wea. Rev., v. 111, no. 6, p. 1228–1243.

Navon, I. M., 1987, FEUDX: A two-stage, high-accuracy, finite-element FORTRAN program for solving shallow-water equations: Computers & Geosciences, v. 13, no. 3, p. 255–285.

Orszag, S. A., 1970, Transform method for calculation of vector coupled sums, application to the spectral form of the vorticity equation: Jour. Atm. Sci., v. 27, p. 890–895.

Sadourny, R., 1975, The dynamics of finite-difference models of the shallow-water equations: Jour. Atm. Sci., v. 32, no. 4, p. 680–689.

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

Staniforth, A. N., and Mitchell, H. L., 1977, A semi-implicit finite-element barotropic model: Mon. Wea. Rev., v. 105, no. 1, p 154–169.

Staniforth, A. N., and Beaudoin, C., 1986, On the efficient evaluation of certain integrals in the Galerkin F. E. Method: Intern. Jour. Num. Meth. Fluid, v. 6, no. 2, p. 317–324.

Steppeler, J., 1986a, Energy conserving Galerkin finite-element schemes, for the primitive equations of numerical weather prediction: Jour. Comput. Phys., v. 69, no. 1, p. 258–264.

Steppeler, J., 1986b, Energy conserving finite schemes for numerical weather prediction, finite-element methods in flow problems: INRIA, Antibes, p. 359–364.

Steppeler, J., 1987, Quadratic finite-element schemes for the vertical discretization forecast models: Mon. Wea. Rev., v. 115, no. 8, p. 1575–1588.

Steppeler, J., 1988, A Galerkin finite-element-spectral weather forecast model in hybrid coordinates: Comput. Math. Applics., v. 16, no. 1/2, p. 23–30.

Steppeler, J., Navon, I. M., and Lu H. I., 1990, Finite-element schemes for extended integrations of atmospheric models, Jour. Comp. Phys., in press.

APPENDIX

Short Description of the Numerical Procedure

A theory of the method is described by Steppeler (1986a), and a more detailed account of the finite difference equations is given by Steppeler, Navon, and Lu (1989). The time derivatives of U, V, H in the nodepoint grid (Fig. 1A) are given by

$$\begin{aligned}\dot{U}_{\nu,\mu} &= G_1\langle \eta V - \{G_2[\tfrac{1}{2}(U^2+V^2)+gH]\}_X\rangle_{\nu,\mu}\\ \dot{V}_{\nu,\mu} &= G_1\langle -\eta U - \{G_2[\tfrac{1}{2}(U^2+V^2)+gH]\}_Y\rangle_{\nu,\mu}\\ \dot{H}_{\nu,\mu} &= -G_2[(UH)_X+(VH)_Y]_{\nu,\mu}\\ &= V_X - U_Y + f. \end{aligned} \tag{A1}$$

Here η represents the absolute vorticity.

Standard Galerkin schemes use only one Galerkin operator G. The combination of two operators G_1 and G_2 with different weights for their Galerkin integrals in Equation (A1) will achieve energy conservation.

The computation of the nodepoint values of the Galerkin projected right-hand sides in Equation (A1) is given by steps (a)–(g) in "Grids and Basic Solution Procedures". For the computation of the projection G_1, Equations (8), (10) are used with weight $w = H(X, Y)$ and G_2 is computed using $w = 1$.

The computation of $\dot{U}_{\nu,\mu}$, $\dot{V}_{\nu,\mu}$, and $\dot{H}_{\nu,\mu}$ will involve the following two steps:

(1) For $e = \frac{1}{2}(U^2+V^2)+gH$, compute $(G_2 e)_{\nu,\mu}$ using steps (a)–(g) as described in "Grids and Basic Solution Procedures".
(2) Interpolate also $G_2 e$ to the collocation grids and use steps (a)–(g) of "Grids and Basic Solution Procedures" to compute $\dot{U}_{\nu,\mu}$, $\dot{V}_{\nu,\mu}$, and $\dot{H}_{\nu,\mu}$.

Once the time derivatives have been computed, the solution is advanced in time using a leapfrog time-differencing scheme. We also introduce an optional second-order smoothing operator with coefficient ϵ, which reduces both spatially small scales and the computational mode of the leapfrog scheme, acting as a low-pass filter. The index n denotes the time level.

$$\begin{aligned} U^{n+1}_{\nu,\mu} &= U^{n-1}_{\nu,\mu} + 2\Delta t U_{\nu,\mu} + \epsilon\nabla^2 U\\ V^{n+1}_{\nu,\mu} &= V^{n-1}_{\nu,\mu} + 2\Delta t V_{\nu,\mu} + \epsilon\nabla^2 H\\ H^{n+1}_{\nu,\mu} &= H^{n-1}_{\nu,\mu} + 2\Delta t H_{\nu,\mu} + \epsilon\nabla^2 H \end{aligned} \tag{A2}$$

with

$$\begin{aligned} U^1_{\nu,\mu} &= U^0_{\nu,\mu} + \Delta t U^0_{\nu,}\\ V^1_{\nu,\mu} &= V^0_{\nu,\mu} + \Delta t V^0_{\nu,\mu}\\ H^1_{\nu,\mu} &= H^0_{\nu,\mu} + \Delta t H^0_{\nu,\mu}. \end{aligned} \tag{A3}$$

The smoothing operator ∇^2 is defined as

$$(\nabla^2 U)^n_\nu = (-4U^{n-1}_{\nu,\mu} + U^n_{\nu-1,\mu} + U^n_{\nu+1,\mu} U^n_{\nu,\mu-1} + U^n_{\nu,\mu+1}) \tag{A4}$$

where ∇^2 is the discrete Laplacian operator with the difference that the central star grid point is using the previous time level $(n-1)$. ϵ is a coefficient controlling the amount of smoothing.

Program Listing

```
      PROGRAM FE2DY
C
      PARAMETER(L=8,LCOL=3)
      COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
      COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
C
C     GAUSSIAN WEIGHTS
```

```
C     POLYNOMIALS UP TO DEGREE 5
      XCOL(1)=-.77459 66692 41483
      XCOL(2)= 0.
      XCOL(3)= .77459 66692 41483
      WCOL(1)= .55555 55555 55555
      WCOL(2)= .88888 88888 88889
      WCOL(3)= .55555 55555 55555
C
C     SET UP INITIAL VALUES
C
      H0=20000.
      H11=4400.
      H2=2660.
      XL=4400000.
      XD=4400000.
      DH=1500.
      F=.0001
      DX=400000.*11./7.
       DT=900.
      EPS=1./10000.
      DO 40 IX=1,L
      DO 40 IY=1,L
      U(IX,IY)=0.
      V(IX,IY)=0.
      FAK=0.
      IIY=IY
        ARG=9.*(IIY*DX-6*DX)/(2.*XD)
      H(IX,IY)=H0+H11*TANH(ARG)
     I +H2*(1./COSH(9.*((IY-6)*DX/XD)))**2
     I *(SIN(2.*3.1415*IX*DX/XL))
      U1(IX,IY)=U(IX,IY)
      V1(IX,IY)=V(IX,IY)
      H1(IX,IY)=H(IX,IY)
40    CONTINUE
      DO 41 IX=1,L
      DO 41 IY=2,L-1
      IXP=IX+1
      IXM=IX-1
      IF(IXP.EQ.L+1)IXP=2
      IF(IXM.EQ.-1)IXM=L-1
      U(IX,IY)=-(H(IX,IY+1)-H(IX,IY-1))/(F*DX*2.)
      V(IX,IY)= (H(IXP,IY)-H(IXM,IY))/(F*DX*2.)
      U1(IX,IY)=U(IX,IY)
      V1(IX,IY)=V(IX,IY)
      H1(IX,IY)=H(IX,IY)
41    CONTINUE
C     TRANSFORM X-DIRECTION
C
      CALL PRF(H)
      DO 60 ICOL=1,LCOL
      EMU(ICOL)=(1.-XCOL(ICOL))/2.
      EMUP1(ICOL)=(XCOL(ICOL)+1.)/2.
60    CONTINUE
C
C     TIME STEPPING
C
C
      DT2=DT
      NE=25000
      DO 100 N=1,NE
      CALL TEND
      CALL ENER(E,Z,HM,EKIN)
C
C     STORE INTEGRAL CONSTRAINTS FOR INITIAL TIME
C
      IF(N.EQ.1)EE=E
      IF(N.EQ.1)ZZ=Z
      IF(N.EQ.1)HMM=HM
      E=E/EE
      Z=Z/ZZ
      HM=HM/HMM
      EKIN=EKIN/EE
      PRINT 89,N,E,EKIN,Z,HM
89    FORMAT(' N,E=',I10,4F15.8)
      DO 91 IX=1,L
      DO 91 IY=1,L
      U1(IX,IY)=U1(IX,IY)+DT2*UT(IX,IY)
      V1(IX,IY)=V1(IX,IY)+DT2*VT(IX,IY)
91    H1(IX,IY)=H1(IX,IY)+DT2*HT(IX,IY)
      DT2=2.*DT
C
C     EXCHANGE TIME LEVELS
C
```

```
      DO 94 IX=1,L
      DO 94 IY=1,L
      UU=U(IX,IY)
      U(IX,IY)=U1(IX,IY)
      U1(IX,IY)=UU
      VV=V(IX,IY)
      V(IX,IY)=V1(IX,IY)
      V1(IX,IY)=VV
      HH=H(IX,IY)
      H(IX,IY)=H1(IX,IY)
94    H1(IX,IY)=HH
      CALL DIFF(EPS)
100   CONTINUE
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
C     DIFF DOES A DIFFUSION STEP ON THE FIELDS                              C
C     U,V AND H                                                             C
C                                                                           C
C     ------------------------------------------------------------          C
C                                                                           C
C     PARAMETER                                                             C
C                                                                           C
C     EPS     -DIFFUSION PARAMETER                                          C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE DIFF(EPS)
      PARAMETER(L=8,LCOL=3)
      COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
      COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DO 10 IX=1,L
      DO 10 IY=2,L-1
      IXP=IX+1
      IXM=IX-1
      IF(IX.EQ.L)IXP=2
      IF(IX.EQ.1)IXM=L-1
      U(IX,IY)=U(IX,IY)+EPS*
     I (-4.*U(IX,IY)+U1(IXP,IY)
     I +U1(IXM,IY)+U1(IX,IY+1)+U1(IX,IY-1))
      V(IX,IY)=V(IX,IY)+EPS*
     I (-4.*V(IX,IY)+V1(IXP,IY)
     I +V1(IXM,IY)+V1(IX,IY+1)+V1(IX,IY-1))
      H(IX,IY)=H(IX,IY)+EPS*
     I (-4.*H(IX,IY)+H1(IXP,IY)
     I +H1(IXM,IY)+H1(IX,IY+1)+H1(IX,IY-1))
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                           C
C     PRF PRINTS TWO DIMENSIONAL FIELDS                                     C
C                                                                           C
C     ------------------------------------------------------------          C
C                                                                           C
C     PARAMETER                                                             C
C                                                                           C
C     H       -FIELD TO BE PRINTED                                          C
C                                                                           C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE PRF(H)
      PARAMETER(L=8,LCOL=3)
      DIMENSION H(L,L)
      DIMENSION IH(L)
      H0=20000.
      DH=1500.
      DO 10 IY=1,L
      DO 1 IX=1,L
      ZHEL=(H(IX,IY)+DH-H0)/DH
1     IH(IX)=ZHEL
      PRINT2,(IH(I),I=1,L)
2     FORMAT(' ',22(' ',1I2))
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     ENER COMPUTES DIAGNOSTIC QUANTITIES                               C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     E       -ENERGY                                                   C
C     Z       -POTENTIAL ENSTROPHY                                      C
C     HMS     -MASS                                                     C
C     EK      -KINETIC ENERGY                                           C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ENER(E,Z,HM,EK)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
       COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
       COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
       E=0.
       HM=0.
       Z=0.
       EK=0.
       DO 10 IX=1,L-1
       DO 10 ICOL=1,LCOL
       DO 10 IY=1,L-1
       DO 10 ICOL1=1,LCOL
       E=E+HC(IX,ICOL,IY,ICOL1)*(UC(IX,ICOL,IY,ICOL1)
     I *UC(IX,ICOL,IY,ICOL1)
     I +VC(IX,ICOL,IY,ICOL1)*VC(IX,ICOL,IY,ICOL1)
     I  +HC(IX,ICOL,IY,ICOL1))
     I  *DX*WCOL(ICOL1)*WCOL(ICOL)
     I  *.5*.5  *.5
       EK=EK+HC(IX,ICOL,IY,ICOL1)*(UC(IX,ICOL,IY,ICOL1)
     I *UC(IX,ICOL,IY,ICOL1)
     I +VC(IX,ICOL,IY,ICOL1)*VC(IX,ICOL,IY,ICOL1)
     I  )
     I  *DX*WCOL(ICOL1)*WCOL(ICOL)
     I  *.5*.5  *.5
       HM=HM+HC(IX,ICOL,IY,ICOL1)*DX*WCOL(ICOL1)*WCOL(ICOL)  *.5*.5
       XI=(VX(IX,ICOL,IY,ICOL1)-UY(IX,ICOL,IY,ICOL1)+F)
     I /HC(IX,ICOL,IY,ICOL1)
       Z=Z+HC(IX,ICOL,IY,ICOL1)*XI*XI
10    CONTINUE
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     TEND COMPUTES TIME DERIVATIVES FOR FIELDS                         C
C     U,V,H AND STORES THEM ON FIELDS UT,VT,HT                          C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE TEND
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
    * I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION
     I RH    (L,L) ,RHC(L,LCOL,L,LCOL),
     I RHCC(L,L,LCOL),RHY(L,LCOL,L,LCOL),
     I RHCCY(L,L,LCOL),RHX(L,LCOL,L,LCOL)
C     IPOLY INTERPOLATES IN Y.
C     IPOLX INTERPOLATES Y INTERPOLATED FIELDS IN X
      CALL IPOLY(U,UCC,UCCY)
      CALL IPOLX(UCC,UC)
      CALL IPOLX(UCCY,UY)
      CALL DIFX (UCC,UX)
      CALL IPOLY(V,VCC,VCCY)
      CALL IPOLX(VCC,VC)
      CALL IPOLX(VCCY,VY)
      CALL DIFX (VCC,VX)
      CALL IPOLY(H,HCC,HCCY)
      CALL IPOLX(HCC,HC)
      CALL IPOLX(HCCY,HY)
      CALL DIFX (HCC,HX)
      DO 5 ICOLX=1,LCOL
      DO 5 IX=1,L-1
      DO 5 ICOLY=1,LCOL
      DO 5 IY=1,L-1
5     RHC(IX,ICOLX,IY,ICOLY)=-.5*(UC(IX,ICOLX,IY,ICOLY)
     I *UC(IX,ICOLX,IY,ICOLY)
     I +VC(IX,ICOLX,IY,ICOLY)*VC(IX,ICOLX,IY,ICOLY))
     I -HC(IX,ICOLX,IY,ICOLY)
      CALL SCALHX(RHC,RHCC)
      CALL GAUHX(RHCC)
      CALL SCALHY(RHCC,RH)
      CALL GAUHY(RH)
      CALL IPOLY(RH,RHCC,RHCCY)
      CALL IPOLX(RHCC,RHC)
      CALL IPOLX(RHCCY,RHY)
      CALL DIFX (RHCC,RHX)
      DO 10 ICOLX=1,LCOL
      DO 10 IX=1,L-1
      DO 10 ICOLY=1,LCOL
      DO 10 IY=1,L-1
      XI=VX(IX,ICOLX,IY,ICOLY)-UY(IX,ICOLX,IY,ICOLY)+F
      UTC(IX,ICOLX,IY,ICOLY)=RHX(IX,ICOLX,IY,ICOLY)
     I +XI*VC(IX,ICOLX,IY,ICOLY)
      VTC(IX,ICOLX,IY,ICOLY)=RHY(IX,ICOLX,IY,ICOLY)
     I -XI*UC(IX,ICOLX,IY,ICOLY)
      HTC(IX,ICOLX,IY,ICOLY)=
     I -UC(IX,ICOLX,IY,ICOLY)*HX(IX,ICOLX,IY,ICOLY)
     I -UX(IX,ICOLX,IY,ICOLY)*HC(IX,ICOLX,IY,ICOLY)
```

```
     I -VC(IX,ICOLX,IY,ICOLY)*HY(IX,ICOLX,IY,ICOLY)
     I -VY(IX,ICOLX,IY,ICOLY)*HC(IX,ICOLX,IY,ICOLY)
10    CONTINUE
      CALL SCALHX(HTC,HTCC)
      CALL GAUHX(HTCC)
      CALL SCALUX(UTC,UTCC)
      CALL GAUUX(UTCC)
      CALL SCALUX(VTC,VTCC)
      CALL GAUUX(VTCC)
      CALL SCALHY(HTCC,HT)
      CALL GAUHY(HT)
      CALL SCALUY(UTCC,UT)
      CALL GAUUY(UT)
      CALL SCALUY(VTCC,VT)
      CALL GAUUY(VT)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     GAUO3IN INITIALISES A GAUSS ELIMINATION                           C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     W       -INPUT AND OUTPUT ARRAY, CONTAINING SIDE                  C
C              BANDS OF MATRIX A.                                       C
C     WI      -OUTPUT ARRAY                                             C
C     NLEV    -LENGTH OF DIAGONAL                                       C
C     NBAND   -NUMBER OF SIDE DIAGONALS                                 C
C                                                                       C
C     STORAGE LOCATION OF W:                                            C
C     W(MID+I,J)=A(J+I,J)                                               C
C     A(J,I)  -MATRIX TO BE INVERTED                                    C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GAUO3IN(W,WI,NLEV,NBAND)
C
      DIMENSION WI(NLEV)
      DIMENSION W(2*NBAND+1,NLEV)
      MID=NBAND+1
      NROW=NLEV
      NROWM1=NROW-1
      DO 50 I=1,NROWM1
C
      PIVOT=W(MID,I)
      JMAX=NBAND
      IF(NROW-I.LT.JMAX)JMAX=NROW-I
      DO 32 J=1,JMAX
32    W(MID+J,I)=W(MID+J,I)/PIVOT
      KMAX=JMAX
      DO 40K=1,KMAX
      IPK=I+K
      MIDMK=MID-K
      FAC=W(MIDMK,IPK)
      DO 40 J=1,JMAX
40    W(MIDMK+J,IPK)=W(MIDMK+J,IPK)-W(MID+J,I)*FAC
50    CONTINUE
      DO 60I=1,NLEV
60    WI(I)=1./W(MID,I)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     GAUO3  PERFORMS A GAUSSIAN ELIMINATION                            C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     W       -ARRAY CONTAINING COMPRESSED FORM                         C
C              OF MATRIX A.                                             C
C     WI      -INVERSE OF DIAGONAL ELEMENTS                             C
C     NLEV    -LENGTH OF DIAGONAL                                       C
C     NBAND   -NUMBER OF SIDE DIAGONALS                                 C
C     B       -RIGHT HAND SIDE OF EQUATIONS                             C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE GAUO3(W,WI,NBAND,B,NLEV)
C
C     FOR COMMENTS SEE SUBROUTINE GAU3IN
C
      DIMENSION B(NLEV)
      DIMENSION W(2*NBAND+1,NLEV)
      DIMENSION WI(NLEV)
C
      MID=NBAND+1
      NROWM1=NLEV-1
C     FOREWARD PASS
      DO 21 I=1,NROWM1
      JMAX=NBAND
      IF(NBAND.GT.NLEV-I)JMAX=NLEV-I
      DO 21 J=1,JMAX
21    B( I+J)=B( I+J)-B( I)*W(MID+J,I)
C
C     BACKWARD PASS
C
      DO 46 I=NLEV,2,-1
      B( I)=B( I)*WI(I)
      JMAX=NBAND
      IF(JMAX.GT.I-1)JMAX=I-1
      DO 46 J=1,JMAX
46    B( I-J)=B( I-J)-B( I)*W(MID-J,I)
      B( 1)=B( 1)*WI(1)
C
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     IPOLY INTERPOLATES IN Y-DIRECTION                                 C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     UQ      -INPUT FIELD                                              C
C     UCQ     -OUTPUT FIELD                                             C
C     UXQ     -OUTPUT FIELD FOR DERIVATIVE                              C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE IPOLY(UQ,UCQ,UXQ)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L)  ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION UQ(L,L),UCQ(L,L,LCOL),UXQ(L,L,LCOL)
      DO 10 IX=1,L
      DO 10 ICOLY=1,LCOL
      DO 10 IY=1,L-1
      UCQ(IX,IY,ICOLY)=(UQ(IX,IY)*(1.-XCOL(ICOLY))
     I +UQ(IX,IY+1)*(XCOL(ICOLY)+1.))*.5
      UXQ(IX,IY,ICOLY)=(UQ(IX,IY+1)-UQ(IX,IY))/DX
10    CONTINUE
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     IPOLX INTERPOLATES IN X-DIRECTION                                 C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     UQQ     -INPUT FIELD                                              C
C     UQ      -OUTPUT FIELD                                             C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE IPOLX(UQQ,UQ)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION UQQ(L,L,LCOL),UQ(L,LCOL,L,LCOL)
      DO 10 IX=1,L-1
      DO 10 ICOLX=1,LCOL
      DO 10 IY=1,L-1
      DO 10 ICOLY=1,LCOL
      UQ(IX,ICOLX,IY,ICOLY)=(UQQ(IX,IY,ICOLY)*(1.-XCOL(ICOLX))
     I +UQQ(IX+1,IY,ICOLY)*(XCOL(ICOLX)+1.))*.5
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     DIFX COMPUTES DERIVATIVE IN X-DIRECTION                           C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     UQQ     -INPUT FIELD                                              C
C     UQX     -OUTPUT FIELD                                             C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE DIFX (UQQ,UQX)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
```

```
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION UQQ(L,L,LCOL),UQX(L,LCOL,L,LCOL)
      DO 10 IX=1,L-1
      DO 10 ICOLX=1,LCOL
      DO 10 IY=1,L-1
      DO 10 ICOLY=1,LCOL
      UQX(IX,ICOLX,IY,ICOLY)=(UQQ(IX+1,IY,ICOLY)
     I -UQQ(IX,IY,ICOLY))/DX
10    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     GAUUY PERFORMS GAUSSIAN INVERSION IN Y-DIRECTION                  C
C     USING WEIGHT H                                                    C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     QT      -INPUT AND OUTPUT FIELD                                   C
C                                                                       C
C     SUBROUTINES GAUO3IN AND GAUO3 ARE USED TO                         C
C     PERFORM THE INVERSION ON A COLUMN                                 C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GAUUY(QT)
      PARAMETER(L=8,LCOL=3)
      DIMENSION QT(L,L),HEL(L)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      COMMON /A/ WU(3,L),WUI(L),WH(3,L),WHI(L)
C     SET UP MASS MATRIX
      DO 900 IX=1,L
      DO 1   IY=1,L
      HEL(IY)=QT(IX,IY)
1     CONTINUE
      LL=L-1
      NBAND=1
      DO 10 J=1,2*NBAND+1
      DO 10 I=1,L
      WU(J,I)=0.
10    CONTINUE
      DO 100 ICOL=1,LCOL
      DO 50 J=2,LL
      WU(1,J)=WU(1,J)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
     I *HCC(IX,J-1,ICOL)
      WU(2,J)=WU(2,J)+.5*DX*EMUP1(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
     I *HCC(IX,J-1,ICOL)
      WU(2,J)=WU(2,J)+.5*DX*EMU(ICOL)*EMU(ICOL)*WCOL(ICOL)
     I *HCC(IX,J,ICOL)
      WU(3,J)=WU(3,J)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
     I *HCC(IX,J,ICOL)
50    CONTINUE
```

```
100   CONTINUE
C     PREPARE INVERSION
      DO 110 I=2,LL
      WU(1,I-1)=WU(1,I)
      WU(2,I-1)=WU(2,I)
      WU(3,I-1)=WU(3,I)
      WUI(I-1)=WUI(I)
      HEL(I-1)=HEL(I)
110   CONTINUE
      CALL GAUO3IN(WU,WUI,LL-1,1)
C     INVERSION
      CALL GAUO3(WU,WUI,1,HEL,LL-1)
      DO 120 I=L,2,-1
120   HEL(I)=HEL(I-1)
      HEL(1)=0.
      HEL(L)=0.
      DO 800 IY=1,L
      QT(IX,IY)=HEL(IY)
800   CONTINUE
900   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     GAUHY PERFORMS GAUSSIAN INVERSION IN Y-DIRECTION                 C
C     USING WEIGHT 1                                                   C
C                                                                      C
C     ---------------------------------------------------              C
C                                                                      C
C     PARAMETERS                                                       C
C                                                                      C
C     QT      -INPUT AND OUTPUT FIELD                                  C
C                                                                      C
C     SUBROUTINES GAUO3IN AND GAUO3 ARE USED TO                        C
C     PERFORM THE INVERSION ON A COLUMN                                C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      SUBROUTINE GAUHY(QT)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      COMMON /A/ WU(3,L),WUI(L),WH(3,L),WHI(L)
      DIMENSION QT(L,L),HEL(L)
C     SET UP MASS MATRIX
      DO 900 IX=1,L
      DO 1   IY=1,L
      HEL(IY)=QT(IX,IY)
1     CONTINUE
      LL=L-1
      NBAND=1
      DO 10 J=1,2*NBAND+1
      DO 10 I=1,L
      WH(J,I)=0.
10    CONTINUE
      DO 100 ICOL=1,LCOL
      DO 20 J=2,L
      WH(1,J)=WH(1,J)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
      WH(2,J)=WH(2,J)+.5*DX*EMUP1(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
```

```
20    CONTINUE
      DO 40 J=1,LL
      WH(2,J)=WH(2,J)+.5*DX*EMU(ICOL)*EMU(ICOL)*WCOL(ICOL)
      WH(3,J)=WH(3,J)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
40    CONTINUE
100   CONTINUE
      CALL GAUO3IN(WH,WHI,L,1)
C     INVERSION
      CALL GAUO3(WH,WHI,1,HEL,L)
      DO 800 IY=1,L
      QT(IX,IY)=HEL(IY)
800   CONTINUE
900   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     GAUUX PERFORMS GAUSSIAN INVERSION IN X-DIRECTION                  C
C     USING WEIGHT H                                                    C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     QTQQ    -INPUT AND OUTPUT FIELD                                   C
C                                                                       C
C     SUBROUTINES GAUO3IN AND GAUO3 ARE USED TO                         C
C     PERFORM THE INVERSION ON A COLUMN                                 C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GAUUX(QTQQ)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L)   ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      COMMON /A/ WU(3,L),WUI(L),WH(3,L),WHI(L)
      DIMENSION QTQQ(L,L,LCOL),HEL(L),HEL1(L),HEL2(L)
C     SET UP MASS MATRIX
      DO 900 IY=1,L-1
      DO 900 ICOLY=1,LCOL
      DO 1   IX=1,L-1
      HEL(IX)=QTQQ(IX,IY,ICOLY)
      HEL1(IX)=0.
      HEL2(IX)=0.
1     CONTINUE
      LL=L-1
      NBAND=1
      DO 10 J=1,2*NBAND+1
      DO 10 I=1,L-1
      WH(J,I)=0.
10    CONTINUE
      DO 100 ICOL=1,LCOL
      DO 20 J=2,L
      JP=J
      IF(JP.EQ.L)JP=1
      WH(1,JP)=WH(1,JP)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
     I *HC(J-1,ICOL,IY,ICOLY)
      WH(2,JP)=WH(2,JP)+.5*DX*EMUP1(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
     I *HC(J-1,ICOL,IY,ICOLY)
```

```
20     CONTINUE
       DO 40 J=1,LL
       WH(2,J)=WH(2,J)+.5*DX*EMU(ICOL)*EMU(ICOL)*WCOL(ICOL)
      I *HC(J,ICOL,IY,ICOLY)
       WH(3,J)=WH(3,J)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
      I *HC(J,ICOL,IY,ICOLY)
40     CONTINUE
100    CONTINUE
       HEL1(L-1)=WH(3,L-1)
       HEL2(1)=WH(1,1)
       CALL GAUO3IN(WH,WHI,L-1,1)
C      INVERSION
       CALL GAUO3(WH,WHI,1,HEL,L-1)
       CALL GAUO3(WH,WHI,1,HEL1,L-1)
       CALL GAUO3(WH,WHI,1,HEL2,L-1)
       A11=1.+HEL1(1)
       A12=HEL2(1)
       A21=HEL1(L-1)
       A22=1.+HEL2(L-1)
       XN= (HEL(1)*A22-HEL(L-1)*A12)/(A22*A11-A21*A12)
       X0=-(HEL(1)*A21-HEL(L-1)*A11)/(A22*A11-A21*A12)
       DO 2 I=1,L-1
       HEL(I)=HEL(I)-XN*HEL1(I)-X0*HEL2(I)
2      CONTINUE
       HEL(L)=XN
       DO 800 IX=1,L
       QTQQ(IX,IY,ICOLY)=HEL(IX)
800    CONTINUE
900    CONTINUE
       RETURN
       END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                          C
C      GAUHX PERFORMS GAUSSIAN INVERSION IN X-DIRECTION                    C
C      USING WEIGHT 1                                                      C
C                                                                          C
C      ------------------------------------------------------              C
C                                                                          C
C      PARAMETERS                                                          C
C                                                                          C
C      QTQQ    -INPUT AND OUTPUT FIELD                                     C
C                                                                          C
C      SUBROUTINES GAUO3IN AND GAUO3 ARE USED TO                           C
C      PERFORM THE INVERSION ON A COLUMN                                   C
C                                                                          C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE GAUHX(QTQQ)
       PARAMETER(L=8,LCOL=3)
        COMMON/COMGAU/
      I WCOL(LCOL),XCOL(LCOL)
      I,EMU(LCOL),EMUP1(LCOL)
C
       COMMON /DYNVAR/
      I U    (L,L) ,H    (L,L),V    (L,L),
      I UT   (L,L) ,HT   (L,L),VT   (L,L),
      I U1   (L,L) ,H1   (L,L),V1   (L,L),
      I X    (L) ,DX,F
       COMMON/DY1/
      I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
      I ,VX(L,LCOL,L,LCOL)
      I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
      I ,VY(L,LCOL,L,LCOL)
      I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
      I ,VC(L,LCOL,L,LCOL)
      I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
      I ,VTC(L,LCOL,L,LCOL)
        COMMON /YIPO/
      I HCC(L,L,LCOL),HCCY(L,L,LCOL)
      I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
      I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
      I,HTCC(L,L,LCOL)
      I,UTCC(L,L,LCOL)
      I,VTCC(L,L,LCOL)
       COMMON /A/ WU(3,L),WUI(L),WH(3,L),WHI(L)
       DIMENSION QTQQ(L,L,LCOL),HEL(L),HEL1(L),HEL2(L)
C      SET UP MASS MATRIX
       DO 900 IY=1,L-1
       DO 900 ICOLY=1,LCOL
       DO 1   IX=1,L-1
       HEL(IX)=QTQQ(IX,IY,ICOLY)
       HEL1(IX)=0.
       HEL2(IX)=0.
```

```
1     CONTINUE
      LL=L-1
      NBAND=1
      DO 10 J=1,2*NBAND+1
      DO 10 I=1,L-1
      WH(J,I)=0.
10    CONTINUE
      DO 100 ICOL=1,LCOL
      DO 20 J=2,L
      JP=J
      IF(JP.EQ.L)JP=1
      WH(1,JP)=WH(1,JP)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
      WH(2,JP)=WH(2,JP)+.5*DX*EMUP1(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
20    CONTINUE
      DO 40 J=1,LL
      WH(2,J)=WH(2,J)+.5*DX*EMU(ICOL)*EMU(ICOL)*WCOL(ICOL)
      WH(3,J)=WH(3,J)+.5*DX*EMU(ICOL)*EMUP1(ICOL)*WCOL(ICOL)
40    CONTINUE
100   CONTINUE
      HEL1(L-1)=WH(3,L-1)
      HEL2(1)=WH(1,1)
      CALL GAUO3IN(WH,WHI,L-1,1)
C     INVERSION
      CALL GAUO3(WH,WHI,1,HEL,L-1)
      CALL GAUO3(WH,WHI,1,HEL1,L-1)
      CALL GAUO3(WH,WHI,1,HEL2,L-1)
      A11=1.+HEL1(1)
      A12=HEL2(1)
      A21=HEL1(L-1)
      A22=1.+HEL2(L-1)
      XN= (HEL(1)*A22-HEL(L-1)*A12)/(A22*A11-A21*A12)
      X0=-(HEL(1)*A21-HEL(L-1)*A11)/(A22*A11-A21*A12)
      DO 2 I=1,L-1
      HEL(I)=HEL(I)-XN*HEL1(I)-X0*HEL2(I)
2     CONTINUE
      HEL(L)=XN
      DO 800 IX=1,L
      QTQQ(IX,IY,ICOLY)=HEL(IX)
800   CONTINUE
900   CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                      C
C     SCALUX COMPUTES THE SCALAR PRODUCT IN X-DIRECTION                C
C     USING WEIGHT H                                                   C
C                                                                      C
C     ---------------------------------------------------------        C
C                                                                      C
C     PARAMETERS                                                       C
C                                                                      C
C     HTQ     -INPUT FIELD                                             C
C     HTQQ    -OUTPUT FIELD                                            C
C                                                                      C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE SCALUX(HTQ,HTQQ)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
       COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
       COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION
```

```
     I HTQ(L,LCOL,L,LCOL)
     I,HTQQ(L,L,LCOL)
      DO 10 IX=1,L
      DO 10 ICOLY=1,LCOL
      DO 10 IY=1,L
      HTQQ(IX,IY,ICOLY)=0.
10    CONTINUE
      DO 20 IX=1,L-1
      ITOP=IX+1
      IF(ITOP.EQ.L)ITOP=1
      DO 20 ICOLX=1,LCOL
      WB=(1.-XCOL(ICOLX))*.5*WCOL(ICOLX)
      WT=(XCOL(ICOLX)+1.)*.5*WCOL(ICOLX)
      DO 15 ICOLY=1,LCOL
      DO 15 IY=1,L-1
      HTQQ(IX,IY,ICOLY)=
     I HTQQ(IX,IY,ICOLY)+HTQ(IX,ICOLX,IY,ICOLY)*WB*DX*.5
     I *HC(IX,ICOLX,IY,ICOLY)
      HTQQ(ITOP,IY,ICOLY)=
     I HTQQ(ITOP,IY,ICOLY)+HTQ(IX,ICOLX,IY,ICOLY)*WT*DX*.5
     I *HC(IX,ICOLX,IY,ICOLY)
15    CONTINUE
20    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                            C
C     SCALHX COMPUTES THE SCALAR PRODUCT IN X-DIRECTION                      C
C     USING WEIGHT 1                                                         C
C                                                                            C
C     ---------------------------------------------------------              C
C                                                                            C
C     PARAMETERS                                                             C
C                                                                            C
C     HTQ     -INPUT FIELD                                                   C
C     HTQQ    -OUTPUT FIELD                                                  C
C                                                                            C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE SCALHX(HTQ,HTQQ)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION
     I HTQ(L,LCOL,L,LCOL)
     I,HTQQ(L,L,LCOL)
      DO 10 IX=1,L
      DO 10 ICOLY=1,LCOL
      DO 10 IY=1,L
      HTQQ(IX,IY,ICOLY)=0.
10    CONTINUE
      DO 20IX=1,L-1
      ITOP=IX+1
      IF(ITOP.EQ.L)ITOP=1
      DO 20 ICOLX=1,LCOL
      WBH=(1.-XCOL(ICOLX))*.5*WCOL(ICOLX)
      WTH=(XCOL(ICOLX)+1.)*.5*WCOL(ICOLX)
      DO 15 ICOLY=1,LCOL
      DO 15 IY=1,L-1
      HTQQ(IX,IY,ICOLY)=
```

```
     I HTQQ(IX,IY,ICOLY)+HTQ(IX,ICOLX,IY,ICOLY)*WBH*DX*.5
      HTQQ(ITOP,IY,ICOLY)=
     I HTQQ(ITOP,IY,ICOLY)+HTQ(IX,ICOLX,IY,ICOLY)*WTH*DX*.5
15    CONTINUE
20    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     SCALUY COMPUTES THE SCALAR PRODUCT IN Y-DIRECTION                 C
C     USING WEIGHT H                                                    C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     HTQQ    -INPUT FIELD                                              C
C     QT      -OUTPUT FIELD                                             C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        SUBROUTINE SCALUY(HTQQ,QT)
      PARAMETER(L=8,LCOL=3)
       COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
      COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L)  ,DX,F
      COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
       COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
      DIMENSION
     I QT(L,L)
     I,HTQQ(L,L,LCOL)
      DO 10 IX=1,L
      DO 10 IY=1,L
      QT(IX,IY)=0.
10    CONTINUE
      DO 20 IY=1,L-1
      ITOP=IY+1
      DO 20 ICOLY=1,LCOL
      WBH=(1.-XCOL(ICOLY))*.5*WCOL(ICOLY)
      WTH=(XCOL(ICOLY)+1.)*.5*WCOL(ICOLY)
      DO 15IX=1,L
      QT(IX,IY)=
     I QT(IX,IY)+HTQQ(IX,IY,ICOLY)*WBH*DX*.5
     I *HCC(IX,IY,ICOLY)
      QT(IX,ITOP)=
     I QT(IX,ITOP)+HTQQ(IX,IY,ICOLY)*WTH*DX*.5
     I *HCC(IX,IY,ICOLY)
15    CONTINUE
20    CONTINUE
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                       C
C     SCALHY COMPUTES THE SCALAR PRODUCT IN Y-DIRECTION                 C
C     USING WEIGHT 1                                                    C
C                                                                       C
C     ------------------------------------------------------            C
C                                                                       C
C     PARAMETERS                                                        C
C                                                                       C
C     HTQQ    -INPUT FIELD                                              C
C     QT      -OUTPUT FIELD                                             C
C                                                                       C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

```
      SUBROUTINE SCALHY(HTQQ,QT)
     PARAMETER(L=8,LCOL=3)
      COMMON/COMGAU/
     I WCOL(LCOL),XCOL(LCOL)
     I,EMU(LCOL),EMUP1(LCOL)
C
     COMMON /DYNVAR/
     I U    (L,L) ,H    (L,L),V    (L,L),
     I UT   (L,L) ,HT   (L,L),VT   (L,L),
     I U1   (L,L) ,H1   (L,L),V1   (L,L),
     I X    (L) ,DX,F
     COMMON/DY1/
     I  UX(L,LCOL,L,LCOL),HX(L,LCOL,L,LCOL)
     I ,VX(L,LCOL,L,LCOL)
     I ,UY(L,LCOL,L,LCOL),HY(L,LCOL,L,LCOL)
     I ,VY(L,LCOL,L,LCOL)
     I ,UC(L,LCOL,L,LCOL),HC(L,LCOL,L,LCOL)
     I ,VC(L,LCOL,L,LCOL)
     I ,UTC(L,LCOL,L,LCOL),HTC(L,LCOL,L,LCOL)
     I ,VTC(L,LCOL,L,LCOL)
      COMMON /YIPO/
     I HCC(L,L,LCOL),HCCY(L,L,LCOL)
     I,UCC(L,L,LCOL),UCCY(L,L,LCOL)
     I,VCC(L,L,LCOL),VCCY(L,L,LCOL)
     I,HTCC(L,L,LCOL)
     I,UTCC(L,L,LCOL)
     I,VTCC(L,L,LCOL)
     DIMENSION
     I QT(L,L)
     I,HTQQ(L,L,LCOL)
     DO 10 IX=1,L
     DO 10 IY=1,L
     QT(IX,IY)=0.
10   CONTINUE
     DO 20 IY=1,L-1
     ITOP=IY+1
     DO 20 ICOLY=1,LCOL
     WBH=(1.-XCOL(ICOLY))*.5*WCOL(ICOLY)
     WTH=(XCOL(ICOLY)+1.)*.5*WCOL(ICOLY)
     DO 15IX=1,L
     QT(IX,IY)=
     I QT(IX,IY)+HTQQ(IX,IY,ICOLY)*WBH*DX*.5
     QT(IX,ITOP)=
     I QT(IX,ITOP)+HTQQ(IX,IY,ICOLY)*WTH*DX*.5
15   CONTINUE
20   CONTINUE
     RETURN
     END
```