SCANNED DOCUMENT

SCANNED DATE: 2/1/02

FILE DIRECTORY:

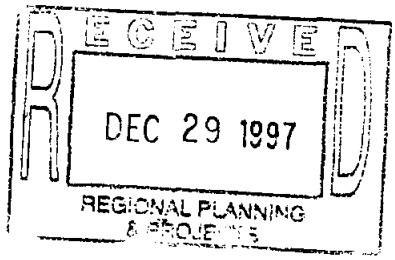
K:\Plan Share\R&PF Scanned Reports\ 95-423-097

Tidal Predictions For Galveston Bay, Texas Using Model Adcirc-2DDI

Colored Pictures (Maps)

TIDAL PREDICTIONS FOR GALVESTON BAY, TEXAS

USING MODEL ADCIRC -2DDI



J.J. Westerink
Department of Civil Engineering and Geological Sciences
University of Notre Dame
Notre Dame, IN 46556

R.A. Luettich, Jr.
Institute of Marine Sciences
University of North Carolina at Chapel Hill
Morehead City, NC 28557

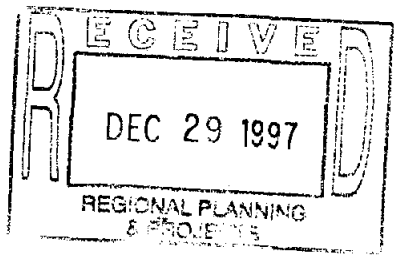
December 26, 1997

Prepared for the TEXAS WATER DEVELOPMENT BOARD
The State of Texas
Austin, TX 78711

Under Contract No. 95-483-097

TIDAL PREDICTIONS FOR GALVESTON BAY, TEXAS

USING MODEL ADCIRC -2DDI



J.J. Westerink
Department of Civil Engineering and Geological Sciences
University of Notre Dame
Notre Dame, IN 46556

R.A. Luettich, Jr.
Institute of Marine Sciences
University of North Carolina at Chapel Hill
Morehead City, NC 28557

December 26, 1997

Prepared for the TEXAS WATER DEVELOPMENT BOARD
The State of Texas
Austin, TX 78711

Under Contract No. 95-483-097

CONTENTS

SUMMARY

This report describes the application of model ADCIRC-2DDI, a two dimensional depth integrated finite element based hydrodynamic circulation code, to study tidal elevation and currents in the vicinity of Galveston Bay, Texas. Issues that are emphasized include the definition of hydrodynamically accurate and simple open ocean boundaries; the use of large domains; the importance of a high degree of grid resolution in coastal regions; and the use of finite element meshes with highly varying nodal densities in order to minimize the size of the discrete problem.

Three finite element meshes are examined in this report. The first is a regional grid which encompasses Galveston Bay and vicinity only. The second grid encompasses the entire Gulf of Mexico and incorporates the first grid. The third grid includes a significant portion of the western North Atlantic in addition to the entire Gulf of Mexico and Caribbean Sea. This third grid, referred to as the Eastcoast grid, incorporates both the second and therefore first grids. All three models are forced with surface elevation forcing function on the seaward boundaries. In addition, tidal potential functions force the interior domain for the Gulf of Mexico and Eastcoast models. The tides are verified at nineteen tidal elevation stations throughout the Gulf of Mexico. Current patterns are in particular examined in the vicinity of Galveston Bay, Texas.

TIDAL PREDICTIONS FOR GALVESTON BAY, TEXAS USING MODEL ADCIRC-2DDI

PART 1: INTRODUCTION

1. Numerical models are widely used to compute sea surface elevations and currents due to mesoscale processes such as tides and hurricane storm surge (Lynch, 1983; Westerink and Gray, 1991). However, a truly predictive capability for flow in coastal regions requires that all important scales of motion be sufficiently resolved in the numerically discrete form of the governing equations. Typically an increasingly greater degree of grid refinement is required as the landward boundary is approached, in order to resolve important processes and to prevent energy from aliasing. This latter requirement stems from factors such as shallow water waves being inherently slower and shorter in shallow water as well as the importance of near coastal bathymetry and geometry which dominate both elevation and current patterns. In order to provide sufficient resolution in the near shore region without excessively increasing the size of the discrete problem, a numerical method must be used which permits a very high degree of grid flexibility.

2. The finite element (FE) method inherently offers a degree of grid flexibility which is ideally suited for coastal modeling. However, early shallow water equation models using the FE method were plagued with severe spurious mode problems and typically required the heavy handed use of non-physical dissipation, limiting their usefulness as predictive tools (Gray, 1982). The introduction of wave-continuity equation (WCE) formulations by Lynch and Gray (1979) led to the development of robust FE depth integrated coastal circulation codes. Thorough testing (Lynch and Gray, 1979; Lynch, 1981; Walters, 1983, 1984; Drolet, 1989), extensive analysis (Platzman, 1981; Foreman, 1983; Drolet and Gray, 1988) and detailed field applications (Werner and Lynch, 1987; Walters, 1987; Walters, 1988, Werner and Lynch, 1989; Walters and Werner, 1989; Gray et al., 1987; Gray, 1989; Lynch and Werner, 1990; Lynch et al., 1988; Lynch et al., 1990; Foreman, 1988; Westerink et al., 1992a) carried out during the past decade have led to a broad based fundamental understanding of WCE formulations. These studies have demonstrated

the unique advantages of these formulations for FE applications in terms of achieving a concurrent high level of computational accuracy and efficiency.

3. In this report, we examine the application of ADCIRC-2DDI, a WCE based FE shallow water equation model (Luettich et al., 1992; Westerink et al., 1992b), to study surface elevation and flow due to tides in the vicinity of Galveston Bay, Texas. It is shown that WCE based FE methods lead to highly accurate and efficient predictions with an unprecedented degree of grid flexibility.

PART II: GOVERNING EQUATIONS AND NUMERICAL DISCRETIZATION

4. The computations described in this report were performed using ADCIRC-2DDI, the depth integrated option of a system of two and three dimensional hydrodynamic codes named ADCIRC (Luettich et al., 1992). ADCIRC-2DDI uses the depth integrated equations of mass and momentum conservation, subject to the incompressibility, Boussinesq and hydrostatic pressure approximations. Using the standard quadratic parameterization for bottom stress and neglecting baroclinic terms and lateral diffusion/dispersion effects leads to the following set of conservation statements in primitive non-conservative form expressed in a spherical coordinate system (Flather, 1988; Kolar et al., 1992):

$$\frac{\partial \zeta}{\partial t} + \frac{1}{R\cos\phi}\left(\frac{\partial UH}{\partial \lambda} + \frac{\partial (VH\cos\phi)}{\partial \phi}\right) = 0 \tag{1}$$

$$\frac{\partial U}{\partial t} + \frac{1}{R\cos\phi}U\frac{\partial U}{\partial \lambda} + \frac{1}{R}V\frac{\partial U}{\partial \phi} - \left(\frac{\tan\phi}{R}U + f\right)V = -\frac{1}{R\cos\phi}\frac{\partial}{\partial \lambda}\left[\frac{p_s}{\rho_0} + g(\zeta - \eta)\right] + \frac{\tau_{s\lambda}}{\rho_0 H} + -\tau_* U \tag{2}$$

$$\frac{\partial V}{\partial t} + \frac{1}{R\cos\phi}U\frac{\partial V}{\partial \lambda} + \frac{1}{R}V\frac{\partial V}{\partial \phi} + \left(\frac{\tan\phi}{R}U + f\right)U = -\frac{1}{R}\frac{\partial}{\partial \phi}\left[\frac{p_s}{\rho_0} + g(\zeta - \eta)\right] + \frac{\tau_{s\phi}}{\rho_0 H} - \tau_* V \tag{3}$$

where

t = time

λ, ϕ = degrees longitude (east of Greenwich positive) and degrees lattitude (north of the equator positive)

ζ = free surface elevation relative to the geoid

U,V = depth averaged horizontal velocities

R = radius of the Earth

H = $\zeta + h$ = total water column

h = bathymetric depth relative to the geoid

f = $2\Omega\sin\phi$ = Coriolis parameter

Ω = angular speed of the Earth

p_s = atmospheric pressure at the free surface

g = acceleration due to gravity

η = effective Newtonian equilibrium tide potential

ρ_0 = reference density of water

τ_{sx}, τ_{sy} = applied free surface stress

τ_* = $C_f \dfrac{(U^2 + V^2)^{1/2}}{H}$

C_f = bottom friction coefficient

5. A practical expression for the effective Newtonian equilibrium tide potential is given by Reid (1990) as:

$$\eta(\lambda, \phi, t) = \sum_{n,\, j} \alpha_{jn} C_{jn} f_{jn}(t_0) L_j(\phi) \cos[2\pi(t - t_0)/T_{jn} + j\lambda + \nu_{jn}(t_0)] \qquad (4)$$

where

C_{jn} = constant characterizing the amplitude of tidal constituent n of species j

α_{jn} = effective earth elasticity factor for tidal constituent n of species j

f_{jn} = time dependent nodal factor

ν_{jn} = time dependent astronomical argument

$j = 0, 1, 2$ = tidal species ($j = 0$, declinational; $j = 1$, diurnal; $j = 2$, semi-diurnal)

L_0 = $3\sin^2\phi - 1$

L_1 = $\sin(2\phi)$

L_2 = $\cos^2(\phi)$

λ, ϕ = degrees longitude and latitude respectively

t_0 = reference time

T_{jn} = period of constituent n of species j

Values for C_{jn} are presented by Reid (1990). We note that the value for the effective earth elastic-

ity factor is typically taken as 0.69 for all tidal constituents (Schwiderski, 1980; Hendershott, 1981) although its value has been shown to be slightly constituent dependent (Wahr, 1981; Woodworth, 1990).

6. To facilitate a FE solution to equations, (1) - (3), these equations are mapped from spherical form into a rectilinear coordinate system using a Carte Parallelogrammatique (CP) projection (Pearson, 1990):

$$x' = R(\lambda - \lambda_o)\cos\phi_o \tag{5}$$

$$y' = R\phi \tag{6}$$

where

λ_0, ϕ_0 = center point of the projection

Applying the CP projection to Equations (1) - (3) gives the shallow water equations in primitive non-conservative form expressed in the CP coordinate system:

$$\frac{\partial \zeta}{\partial t} + \frac{\cos\phi_0}{\cos\phi}\frac{\partial (UH)}{\partial x'} + \frac{1}{\cos\phi}\frac{\partial (VH\cos\phi)}{\partial y'} = 0 \tag{7}$$

$$\begin{aligned} &\frac{\partial U}{\partial t} + \frac{\cos\phi_0}{\cos\phi}U\frac{\partial U}{\partial x'} + V\frac{\partial U}{\partial y'} - \left(\frac{\tan\phi}{R}U + f\right)V \\ &= -\frac{\cos\phi_0}{\cos\phi}\frac{\partial}{\partial x'}\left[\frac{p_s}{\rho_o} + g(\zeta - \eta)\right] + \frac{\tau_{s\lambda}}{\rho_o H} - \tau_* U \end{aligned} \tag{8}$$

$$\begin{aligned} &\frac{\partial V}{\partial t} + \frac{\cos\phi_o}{\cos\phi}U\frac{\partial V}{\partial x'} + V\frac{\partial V}{\partial y'} + \left(\frac{\tan\phi}{R}U + f\right)U \\ &= -\frac{\partial}{\partial y'}\left[\frac{p_s}{\rho_o} + g(\zeta - \eta)\right] + \frac{\tau_{s\phi}}{\rho_0 H} - \tau_* V \end{aligned} \tag{9}$$

7. Utilizing the FE method to resolve the spatial dependence in the shallow water equations in their primitive form gives inaccurate solutions with severe artificial near $2 \cdot \Delta x$ modes (Gray, 1982). However, reformulating the primitive equations into a Generalized Wave Continuity Equation (GWCE) form gives highly accurate, noise free, FE based solutions to the shallow water

equations (Lynch and Gray, 1979; Kinnmark, 1984). The GWCE is derived by combining a time differentiated form of the primitive continuity equation and a spatially differentiated form of the primitive momentum equations recast into conservative form, reformulating the convective terms into non-conservative form and adding the primitive form of the continuity equation multiplied by a constant in time and space, τ_o (Lynch and Gray, 1979; Kinnmark, 1984; Luettich et al., 1992). The GWCE in the CP coordinate system is:

$$\frac{\partial^2 \zeta}{\partial t^2} + \tau_0 \frac{\partial \zeta}{\partial t} + \frac{\cos\phi_0}{\cos\phi} \frac{\partial}{\partial x'} \frac{\partial \zeta}{\partial t} [U - \frac{\cos\phi_0}{\cos\phi} UH \frac{\partial U}{\partial x'}$$

$$-VH \frac{\partial U}{\partial y'} + \left(\frac{\tan\phi}{R} U + f \right) VH - H \frac{\cos\phi_0}{\cos\phi} \frac{\partial}{\partial x'} \left(\frac{p_s}{\rho_0} + g(\zeta - \eta) \right)$$

$$- (\tau_* - \tau_0) UH + \frac{\tau_{s\lambda}}{\rho_0} \Big]$$

$$+ \frac{\partial}{\partial y'} [V \frac{\partial \zeta}{\partial t} - \frac{\cos\phi_0}{\cos\phi} UH \frac{\partial V}{\partial x'} - VH \frac{\partial V}{\partial y'} - \left(\frac{\tan\phi}{R} U + f \right) UH$$

$$-H \frac{\partial}{\partial y'} \left(\frac{p_s}{\rho_0} + g(\zeta - \eta) \right) - (\tau_* - \tau_0) VH + \frac{\tau_{s\phi}}{\rho_0} \Big]$$

$$-\frac{\partial}{\partial t} \left[\frac{\tan\phi}{R} VH \right] + -\tau_0 \left[\frac{\tan\phi}{R} VH \right] = 0 \qquad (10)$$

The GWCE, (10), is solved in conjunction with the primitive momentum equations in non-conservative form, (8) and (9).

8. The high accuracy of GWCE based FE solutions is a result of their excellent numerical amplitude and phase propagation characteristics. In fact, Fourier analysis indicates that in constant depth water and using linear interpolation, a linear tidal wave resolved with 25 nodes per wavelength is more than adequately resolved over the range of Courant numbers, $C = \sqrt{gh}\Delta t/\Delta x \leq 1.0$ (Luettich et al., 1992). Furthermore, the monotonic dispersion behavior of GWCE based FE solutions avoids generating artificial near $2 \cdot \Delta x$ modes which plague primitive based FE solutions (Platzman, 1981; Foreman, 1983). We note that the monotonic dispersion

behavior of GWCE based FE solutions is very similar to that associated with staggered finite difference solutions to the primitive shallow water equations (Westerink and Gray, 1991). GWCE based FE solutions to the shallow water equations allow for extremely flexible spatial discretizations which result in a highly effective minimization of the discrete size of any problem (Le Provost and Vincent, 1986; Foreman, 1988; Vincent and Le Provost, 1988; Westerink et al., 1992a).

9. The details of ADCIRC, our implementation of the GWCE based solution to the shallow water equations, are described by Luettich et al. (1992). As most GWCE based FE codes, ADCIRC applies three noded linear triangles for surface elevation, velocity and depth. Furthermore, the decoupling of the time and space discrete form of the GWCE and momentum equations, time independent and/or tri-diagonal system matrices, elimination of spatial integration procedures during time stepping, full vectorization of all major loops and pre-conditioned conjugate gradient matrix solvers results in a highly efficient code. A user manual further describes the details of using model ADCIRC-2DDI (Westerink et al., 1994b).

PART III: MODELING STRATEGY AND COMPUTATIONAL DOMAINS

10. The region of general interest consists of the vicinity of Galveston Bay, Texas. Due to the complex nature of tides and hurricanes in the Gulf of Mexico, it is very difficult to define surface elevations and/or currents which drive the boundaries of a regional model which encompasses only the immediate area of interest. The fundamental character of the tides in addition to flow features such as resonant shelf edge waves, hurricane forerunner and/or the complex wind patterns associated with a hurricane driving the flow onto the shelf, make it desirable to define larger computational domains which encompass regions well beyond the continental shelf adjacent to the area of interest. The modeling strategy has been to define basin wide or larger computational domains and to refine the region of interest to the degree required through the significant grid flexibility offered by GWCE based FE formulations. In this study three domains are examined in order to compute tidal elevations and flow in the vicinity of Galveston Bay. The first domain encompasses only Galveston Bay and the immediate vicinity as is shown in Figure 1a. A constant 1 component semi-diurnal forcing was applied on the shore parallel part of the open ocean boundary while no normal flow conditions were applied to the cross shelf portions of the open ocean boundary. The second domain encompasses the entire Gulf of Mexico as is shown in Figure 1b. This domain has the advantage that shelf to basin interactions are much better modeled than in a small regional domain. Two well defined open ocean boundaries of limited extent are used to specify the boundary forcing functions which define the interaction between the Atlantic Ocean and Caribbean Sea with the Gulf. The port boundary across the Strait of Florida runs from Cape Sable in Florida to Havana, Cuba. The second port boundary stretches across the Yucatan Channel from the vicinity of Cancun, Mexico to Cabo San Antonio, Cuba. These boundaries are forced with 5 astronomical tidal constituents (M_2, S_2, N_2, O_1 and K_1) from the *EASTCOAST_95* tidal constituent data base of Luettich and Westerink (1995). It is noted that several amphidromes or degenerate amphidromes exist in the vicinity of these ports, particularly for the diurnal constituents. Therefore the performance of the Gulf of Mexico domain with these boundaries should be carefully examined. The third domain encompasses a large portion of the

western North Atlantic ocean as well as the entire Gulf of Mexico and Caribbean Sea and is shown in Figure 1c. This domain, referred to as the Eastcoast domain, has a further advantage over the Gulf of Mexico domain in that it allows the basin to basin interactions between the Gulf of Mexico and the Atlantic ocean and Caribbean Sea to develop naturally, not requiring the details of these interactions to be specified in the ports where the response functions can be quite complex. The open ocean boundary in the Eastcoast model is driven with 5 constituents (M_2, S_2, N_2, O_1 and K_1) from LeProvost's (1995) FES95.2 data base. This data base was developed using a global tidal model and has been found to perform very well in deep ocean waters.

11. The finite element grids used for our simulations are shown in Figure 2. The Galveston and vicinity domain grid, designated as *GAL_G03_R4* grid and shown in Figure 2a, was obtained from Dr. Matsumoto at the Texas Water Development Board (1997). This grid was transformed to spherical coordinates and bathymetry was transformed to metric units. This grid contains 2213 nodes and 3397 elements. The Gulf of Mexico grid, designated as *GOMGAL_G05_R4* grid and shown in Figure 2b, was obtained from part of the *EASTCOAST_95* grid used to develop the corresponding data base. Extensive grid refinement was performed throughout the Gulf. Furthermore the *GAL_G03_R4* grid was incorporated exactly into the *GOMGAL_G05_R4* grid. It is noted that all the grid editing was done with XMGREDIT, a flexible interactive grid generation facility developed by Turner and Baptista (1991). The Gulf of Mexico grid, *GOMGAL_G05_R4* grid contains 11713 nodes and 21216 finite elements. The Eastcoast grid is designated as *EASTGAL_G03_R1* and is shown in Figure 2c. This grid is also based on the *EASTCOAST_95* grid, again with extensive grid refinement throughout the entire domain. This new eastcoast grid, *EASTGAL_G03_R1* identically incorporates the Gulf of Mexico grid, *GOMGAL_G05_R4* (and therefore also the Galveston grid, *GAL_G03_R4*). This grid contains 39812 nodes and 74246 finite elements. The level of grid refinement in this grid is based on numerous grid convergence studies as well as on the response functions obtained from previous computations. In general, the deepest waters in the Atlantic are relatively coarsely discretized

while the continental shelf waters and regions of detailed interest, in this case Galveston Bay and vicinity, are very finely resolved. The largest finite elements in this grid have a size of O(40) km while the finest elements are sized O(150) m.

12. The bathymetry in the three grids was obtained from a variety of sources. For the Galveston grid the bathymetry provided with the original grid by Dr. Matsumoto (1997) was used. For the Gulf of Mexico and Eastcoast grids, the bathymetry was obtained from the ETOPO5 data base from the National Center for Atmospheric Research and was supplemented by the National Ocean and Atmospheric Administration Digital U.S. Coastal Hydrography sounding data base (distributed by NOAA National Geophysical and Solar-Terrestrial Data Center in Boulder, CO) in all U.S. continental shelf waters with the exception of in the Galveston Bay domain, where the original bathymetry from the Galveston grid was applied. For actual simulations a minimum bathymetry of 0.5 meters was specified.

PART IV: TIDAL COMPUTATIONS

13. Tidal simulations for all three domains/grids were performed with the ADCIRC code run in depth integrated mode (2DDI option). The Gulf Mexico and Eastcoast computations were compared to long term field data obtained from the International Hydrographic Office (IHO), NOAA and Reid and Whitaker (1981) at nineteen elevation stations throughout the Gulf shown in Figure 3 and listed in Table 1. An additional 7 stations were defined to obtain detailed elevation and current data within the immediate vicinity of Galveston Bay. All simulations are entirely predictive and no calibration procedures were performed.

14. The open ocean boundary for the *GAL_G03_R4* simulation was forced with the M_2 constituent only with an amplitude of 0.50 m on the shore parallel portion of the open ocean boundary while weak no normal flow conditions were specified on the cross shelf portions of the open ocean boundary. No tidal potential forcing function was specified for this simulation. The *GOMGAL_G05_R4* simulation was forced with on the Strait of Florida and the Yucantan Channel using the K_1, O_1, M_2, N_2 and S_2 astronomical tidal constituents from the *EASTCOAST_95* tidal constituent data base of Luettich and Westerink (1995). The open ocean boundary in the *EASTGAL_G03_R1* model is driven using the K_1, O_1, M_2, N_2 and S_2 astronomical tidal constituents from LeProvost's (1995) FES95.2 data base. Both the *GOMGAL_G05_R4* and *EASTGAL_G03_R1* models applied tidal potential forcing functions due to the size of each domain as well as the resonant features of the Gulf. An effective tidal potential forcing within the interior domain was applied for the same five constituents as are forced on the boundaries. Amplitudes for the interior domain forcing function are listed in Table 2. The values for the effective earth elasticity factor, α, for the various constituents were obtained from Wahr (1981) and are also listed in Table 2. No nodal factors or astronomical arguments were applied to either the boundary or the interior domain forcing functions.

15. A constant value for the bottom friction coefficient, equal to c_f = 0.003, and a constant value for lateral eddy diffusion/dispersion equal to 10 m^2/sec were used in the simulations and were applied throughout all three domains. The bottom friction value is based on natural

river channels which are plain, clean and straight (Chow, 1959). The lateral eddy viscosity value is a physically realistic value in ocean and estuarine applications. It is noted that it is not necessary to apply any eddy viscosity for numerical reasons but it is physically necessitated in order to develop eddying with estuarine inlets. All nonlinearities including finite amplitude effects, nonlinear friction and convective acceleration were taken into consideration in the computations. The flooding/drying feature of the model was also activated.

16. The model was spun up from homogeneous initial conditions and a time ramp was used in order to avoid problems with short period gravity modes and vortex modes in the subinertial frequency range in addition to a free Helmholtz mode (Reid and Whitaker, 1981). A very smooth hyperbolic tangent time ramp function which acts over essentially fifteen days was applied to both boundary conditions and direct forcing functions. We tested both significantly underdamped and correctly damped Gulf of Mexico systems in order to establish spin up requirements. Although a free Helmholtz mode was excited, its amplitude relative to the forced modes was always small due to the smoothness of the hyperbolic tangent time ramp function. In fact, our estimates indicate that even for the underdamped systems, the free mode was less than 0.1% of the representative forced mode after about thirty days. Therefore, a fifteen day spin up is more than adequate for all signals of interest. The actual simulations were run for 120 days of which the first 30 days were discarded. A time step of 15 seconds was used such that the maximum Courant number based on wave celerity is approximately equal to unity. The requirement on Courant number is related to the explicit treatment of the nonlinear terms. Time weighting factors of 0.35, 0.30 and 0.35 were used for the future, current and past time levels in the GWCE equation and a Crank-Nicholson scheme was used for the momentum equations. Finally, the parameter τ_0 was set equal to 0.005 which represents a balance between the primitive continuity and wave equation portions of the GWCE equation and results in excellent Fourier propagation properties as well as excellent mass balance characteristics (Kolar et al, 1994a).

17. The results of the three simulation are qualitatively compared for elevation and currents in a number of movie files for the three simulations: For the *GAL_G03_R4* simulation,

Table 1: Data Stations for elevation and currents used in the Galveston Bay Simulation

Station No.	Name	λ (degree)	ϕ (degree)	type	measured data
1	Key West, FL	-81.1800003	24.716000	elevation	yes
2	Naples, FL	-81.1800003	26.129997	elevation	yes
3	Cedar Key, FL	-83.1031602	29.130004	elevation	yes
4	St. Marks Light, FL	-84.1182998	30.065997	elevation	yes
5	Alligator Bayou, FL	-85.1750000	30.132773	elevation	yes
6	Bay St. Louis, MS	-89.1319555	30.270683	elevation	yes
7	Cat Island, MS	-89.1166000	30.233002	elevation	yes
8	Southwest Pass, LA	-89.1428001	28.929999	elevation	yes
9	Point au Fer, LA	-91.1449997	29.286004	elevation	yes
10	Galveston, TX	-94.1780646	29.296207	elevation	yes
11	Port Aransas, TX	-97.1057999	27.825004	elevation	yes
12	South Padre Island, TX	-97.1150001	26.066002	elevation	yes
13	Madero, Mexico	-97.1711313	22.216007	elevation	yes
14	Coatzacoalcos, Mexico	-94.1412003	18.233697	elevation	yes
15	Campeche, Mexico	-90.1642026	19.850312	elevation	yes
16	Progreso, Mexico	-89.1650002	21.317480	elevation	yes
17	IAPSO #30-1.2	-89.1650002	24.760004	elevation	yes
18	IAPSO #30-1.2.13	-84.1249999	26.700001	elevation	yes
19	Havana, Cuba	-82.1366600	23.165764	elevation	yes
20	Galveston Bay - E01	-95.088247	29.068228	current	no
21	Galveston Bay - E02	-94.670157	29.331112	current	no
22	Galveston Bay - E03	-94.50731	29.494837	current	no
23	Galveston Bay - E04	-95.001626	29.229648	current	no
24	Galveston Bay - E05	-94.835314	29.441799	current	no
25	Galveston Bay - E06	-94.652833	29.525968	current	no
26	Galveston Bay - E07	-94.809905	29.665481	current	no

Table 2: C_{jn}, the Tidal Potential Constants (in meters), and α_{jn}, the Effective Earth Elasticity Factor, for Tidal Potential Forcing Functions (from Reid, 1990 and Wahr, 1981).

Constituent	C_{jn}	α_{jn}
K_1	0.141565	0.736
O_1	0.100514	0.695
N_2	0.046398	0.693
M_2	0.242334	0.693
S_2	0.112841	0.693

gal_entire.flc and gal_z1.flc; for the *GOMGAL_G05_R4* simulation, gomgal_entire.flc, gomgal_z1.flc and gomgal_z2.flc; for *EASTGAL_G03_R1* simulation, eastgal_entire.flc, eastgal_z1.flc and eastgal_z2.flc. These comparisons indicate that the wave propagation patterns and the associated currents are entirely in the cross shelf direction and are clearly not correct in the *GAL_G03_R4* simulation. The results from the *GOMGAL_G05_R4* and *EASTGAL_G03_R1* simulations are in fact very similar. These results indicate that the tides within the Gulf are predominantly diurnal or mixed with the exception of the tides on the west Florida Shelf, where they are strongly semi-diurnal. Reid and Whitaker (1981) note that the semi-diurnal tides off of Florida are associated with the near resonant response to the direct tidal potential forcing and they hypothesize that this response is related to the excitation of shelf edge waves.

18. Tidal elevations amplitudes and phases for the 5 forcing constituents (K_1, O_1, M_2, N_2 and S_2) are compared to data at the 19 measurement stations for the *GOMGAL_G05_R4* and *EASTGAL_G05_R4* simulations in Figure 4. Overall the quality of the comparisons is good, in particular when considering that no tuning was performed in our simulation. It is noted that the diurnal constituents, which in fact dominate the tidal signal in most of the Gulf with the exception of on the Florida shelf, are somewhat over-predicted throughout the Gulf. Furthermore the *GOMGAL_G05_R4* and *EASTGAL_G03_R1* simulations result in very similar response functions throughout the domain. It is clear that for the current discretizations, bathymetric definition and parameter specifications (including bottom friction and tidal potential reduction factors), there is

no justification for including the additional regions defined in the *EASTGAL_G03_R1* domain in order to study tides in the vicinity of Galveston Bay. Therefore for purposes of tidal calculations, the *GOMGAL_G05_R4* domain should be more than sufficient to define the character of tidal wave propagation and the associated current patterns in the vicinity of Galveston Bay. A fourteen day time history of tidal predicted tidal elevations (based on both a 5 constituent resynthesis of ADCIRC harmonically decomposed data and the raw time history produced by ADCIRC) at all nineteen data stations are shown in Figure 5.

19. It is noted that the model performance was excellent. The *GAL_G03_R4* grid ran at 0.045 CPU seconds per time step, the *GOMGAL_G05_R4* grid ran at 0.62 CPU seconds per time step and the *EASTGAL_G03_R1* ran at 2.21 CPU seconds on a SUN Ultra 140e.

PART V: DISCUSSION AND CONCLUSIONS

20. Large domains are extremely important in developing a truly predictive flow computation capability. Large domains reduce or even eliminate the tedious and very difficult task of boundary condition calibration and allow for a much more accurate interaction between the coastal region of immediate interest and the adjacent waters. Application of the *GOMGAL_G05_R4* domain allowed for the tides to propagate along the shelf in the vicinity of Galveston Bay and allowed for the development of realistic current patterns in the area. It was noted that the *EASTCOAST_95* tidal constituent data base did a realistic job of forcing the *GOMGAL_G05_R4* domain and that application of the *EASTGAL_G04_R1* domain was not necessary to obtain good tidal predictions in the Gulf and Mexico and in the vicinity of Galveston Bay. However it is noted that for hurricane storm surge predictions, the larger Eastcoast domain may be necessary for certain applications (Blaine et al, 1995). While the Gulf domain does an excellent job at representing peak hurricane storm surge, the Eastcoast domain in necessary to represent hurricane storm surge forerunner.

21. It is critical to resolve coastal areas for both tide and storm surge simulations. In order to correctly predict tidal elevations at open water locations, we must resolve the tidal waves of interest with a sufficient number of grid points. Our experience indicates that about thirty km resolution in deep waters such as in the Gulf of Mexico and Atlantic ocean is generally sufficient to accurately represent tidal flows with model ADCIRC. However, correct predictions of tidal velocities in a near coastal region may require a much greater level of grid detail. The tides propagating onto the shelf requires significantly increased level of grid resolution. Furthermore, near coastal bathymetry and geometry force the wavenumber content of the flow to be much higher than in the deep ocean. In fact, the flow adjacent to the land boundary may have the same or a higher wavenumber content than the land boundary itself. Not providing sufficient coastal resolution can drastically alter the flow results such that the entire character of the flow is misrepresented. In order to capture the details of the flow in the vicinity of Galveston Bay, required that we increase the grid resolution to a O(100m) near the breakwater tips. The numerical discretization

can be localized to account for the regional discretization requirements such that there is a high density of nodes in coastal regions and a much lower nodal density in more remote deep sea regions. In this way, accuracy can be achieved at a minimum expense in terms of the total number of grid points.

22. The FE method can be realistically implemented within the framework of GWCE formulations and applied to grids with highly variable nodal densities. The computations presented are unprecedented in their scope, level of localized detail and degree of grid size variability. They demonstrate that the accuracy performance of GWCE based FE formulations is excellent over a very wide range of grid and flow conditions. Furthermore, the flow results are inherently smooth without the use of numerical or nonphysical damping. Finally, the algorithms applied lead the very good CPU performance.

23. Currently, we are developing an updated data base for the Eastcoast domain which through increased resolution and parameter calibration strives to optimally match measured tidal elevation and flow data within the entire domain.

REFERENCES

Blain, C.A., J.J. Westerink and R.A. Luettich, "The Influence of Domain Size on the Response Characteristics of a Hurricane Storm Surge Model," *Journal of Geophysical Research*, **99, C9**, 18467-18479, 1994.

Blain, C.A., J.J. Westerink and R.A. Luettich, "Grid Convergence Studies for the Prediction of Hurricane Storm Surge," *International Journal for Numerical Methods in Fluids*, In Press, 1997.

Chow, V.T., *Open Channel Hydraulics*, McGraw-Hill, N.Y. N.Y., 1959.

Cialone, M.A., *The Coastal Modeling System: User's Manual*, CERC Miscellaneous Publication, U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS., 1991.

Drolet, J., and W.G. Gray, "On the Well Posedness of Some Wave Formulations of the Shallow Water Equations", *Advances in Water Resources*, 11, 84-91, 1988.

Drolet, J., "Application and Analysis of Wave Formulations of the Shallow Water Equations", Ph.D. thesis, Department of Civil Engineering, Princeton University, 1989.

Flather, R.A., Estimates of Extreme Conditions of Tide and Surge using a Numerical Model of the North-west European Continental Shelf, *Estuarine, Coastal and Shelf Science*, 24, 69-73, 1987.

Flather, R.A., A Numerical Model Investigation of Tides and Diurnal-Period Continental Shelf Waves along Vancouver Island", *J. of Physical Oceanography*, 18, 115-139, 1988.

Foreman, M.G.G., An Analysis of the Wave Equation Model for Finite Element Tidal Comparisons, *Journal of Computational Physics*, 52, 290-312, 1983.

Foreman, M.G.G., A Comparison of Tidal Models for the Southwest Coast of Vancouver Island, *Proceedings of the VII International Conference on Computational Methods in Water Resources*, held in Cambridge, MA, Elsevier, 1988.

Foreman, M.G.G., *Manual for Tidal Heights Analysis and Prediction*, Pacific Marine Science Report 77-10, Institute of Ocean Sciences, Patricia Bay, Victoria, B.C., 1977.

Foreman, M.G.G., "An Analysis of the 'Wave Equation' Model for Finite Element Tidal Computations", *Journal of Computational Physics*, 52, 290-312, 1983.

Foreman, M.G.G., "A Comparison of Tidal Models for the Southwest Coast of Vancouver Island", *Proceedings of the VII International Conference on Computational Methods in Water Resources*, held in Cambridge, MA, Elsevier, 1988.

Gray, W.G., Some Inadequacies of Finite Element Models as Simulators of Two-Dimensional Circulation, *Advances in Water Resources*, 5, 171-177, 1982.

Gray, W.G., A Finite Element Study of Tidal Flow Data for the North Sea and English Channel, *Advances in Water Resources*, 12, 143-154, 1989.

Gray, W.G., J. Drolet, and I.P.E. Kinnmark. "A Simulation of Tidal Flow in the Southern Part of the North Sea and the English Channel", *Advances in Water Resources*, 10, 131-137, 1987.

Gray, W.G., "A Finite Element Study of Tidal Flow Data for the North Sea and English Channel", *Advances in Water Resources*, 12, 143-154, 1989.

Grenier, R.R., R.A. Luettich and J.J. Westerink, "A Comparison of the Nonlinear Frictional Characteristics of Two-Dimensional and Three-Dimensional Models of a Shallow Tidal Embayment," *Journal of Geophysical Research*, **100, C7**, 13719-13735, 1995.

Hendershott, M.C., Long Waves and Ocean Tides, in *Evolution of Physical Oceanography*, 292-341, B.A. Warren and C. Wunsch, Eds., MIT Press, Cambridge, MA, 1981.

Kinnmark, I.P.E., The Shallow Water Wave Equations: Formulation, Analysis and Application, Ph.D. Dissertation, Department of Civil Engineering, Princeton University, 1984.

Kolar, R.L., and W.G. Gray., "Shallow Water Modeling in Small Water Bodies", *Proceedings of the Eighth International Conference on Computational Methods in Water Resources*, 149-155, 1990.

Kolar, R.L., J.J. Westerink, M.E. Cantekin and C.A. Blain, "Aspects of Nonlinear Simulations Using Shallow Water Models Based on the Wave Continuity Equation," *Computers and Fluids*, **23**, 3, 523-538, 1994a.

Kolar, R.L., W.G. Gray, J.J. Westerink and R.A. Luettich, "Shallow Water Modeling in Spherical Coordinates: Equation Formulation, Numerical Implementation and Application," *Journal of Hydraulic Research*, **32**, 1, 3-24, 1994b.

Kolar, R.L., W.G. Gray and J.J. Westerink, "Boundary Conditions in Shallow Water Models - An Alternative Implementation for Finite Element Codes," *International Journal for Numerical Methods in Fluids*, **22**, 603-618, 1996.

Le Provost, C. and P. Vincent, "Some Tests of Precision for a Finite Element Model of Ocean Tides," *Journal of Computational Physics*, 65, 273-291, 1986.

Le Provost, C., Personal communication, 1995.

Luettich, R.A. and J.J. Westerink, "A Solution for the Vertical Variation of Stress, Rather than Velocity, in a Three-Dimensional Circulation Model," *International Journal for Numerical Methods in Fluids*, **12**, 911-928, 1991.

Luettich, R.A., J.J. Westerink, and N.W. Scheffner, ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts and Estuaries, Report 1: Theory and Methodology of ADCIRC-2DDI and ADCIRC-3DL, *Coastal Engineering Research Center*, U.S. Army Engineers, 1992.

Luettich, R.A., S. Hu and J.J. Westerink, "Development of the Direct Stress Solution Technique for Three Dimensional Hydrodynamic Models Using Finite Elements," *International Journal for Numerical Methods in Fluids*, **19**, 295-319, 1994.

Lynch, D.R., Progress in Hydrodynamic Modeling, Review of U.S. Contributions, 1979-1982, *Reviews of Geophysics and Space Physics*, 21(3), 741-754, 1983.

Lynch, D.R., and W.G. Gray, A Wave Equation Model for Finite Element Tidal Computations, *Computers and Fluids*, 7, 207-228, 1979.

Lynch, D.R., "Comparison of Spectral and Time-Stepping Approaches for Finite Element Modeling of Tidal Circulation", *Oceans 81 Conference Record*, Vol. 2; Boston MA., September 16-18, IEEE Publ. No. 81CH1685-7, 1981.

Lynch, D.R., and F.E. Werner, "Three-Dimensional Hydrodynamics on Finite Elements, Part II: Nonlinear Time-Stepping Model", *International Journal on Numerical Methods in Fluids*, 1990.

Lynch, D.R., F.E. Werner, A. Cantos-Figuerola, and G. Parilla, "Finite Element Modeling of Reduced-Gravity Flow in the Alboran Sea: Sensitivity Studies", in *Seminario Sobre Oceanografia Fisica del Estrecho de Gibraltar*, Madrid, 283-295, 1988.

Lynch, D.R., F.E. Werner, J.M. Molines, and M. Fornerino, "Tidal Dynamics in a Coupled Ocean/Lake System", *Estuarine, Coastal and Shelf Science*, 31, 319-343, 1990.

Matsumoto, J., Personal communication, 1997.

Pearson F., Map Projections: Theory and Applications, CRC Press, Inc., Boca Raton, Florida, 1990.

Platzman, G.W., Some Response Characteristics of Finite Element Tidal Models, *Journal of Computational Physics*, 40, 36-63, 1981.

Reid, R.O., and R.E. Whitaker, Numerical Model for Astronomical Tides in the Gulf of Mexico, Technical Report for the U.S. Army Engineers, Department of Oceanography, Texas A&M University, 1981.

Reid, R.O., Waterlevel Changes, *Handbook of Coastal and Ocean Engineering*, J. Herbich, Ed., Gulf Publishing, Houston, Texas, 1990.

Schureman, P. (1940). *Manual of Harmonic Analysis and Prediction of Tides*, Special Publication No. 98, U.S. Department of Commerce.

Schwiderski, E.W., On Charting Global Ocean Tides, *Reviews in Geophysics and Space Physics*, 18, 243-268, 1980.

Turner, P.J. and A.M. Baptista, ACE/Gredit Users Manual: Software for Semi-automatic Generation of Two-Dimensional Finite Element Grids", CCALMR Software Report SDS2(91-2), Oregon Graduate Institute, Beaverton, OR, 1991.

U.S. Geological Survey, Atlas of Tidal Elevation and Current Observations on the Northeast American Continental Shelf and Slope, U.S. Geological Survey Bulletin #1611, 1984.

Vincent, P. and C. Le Provost, Semidiurnal Tides in the Northeast Atlantic From a Finite Element Numerical Model, *Journal of Geophysical Research*, 93, No. C1, 543-555, 1988.

Wahr, J.M., Body Tides on an Elliptical, Rotating, Elastic and Oceanless Earth, *Geophys. J.R. astr. Soc.* 64, 677-703, 1981.

Walters, R.A., A Finite Element Model For Tides and Currents with Field Applications, *Comm. Applied Numerical Methods*, 4, 401-411, 1988.

Walters, R.A. and F.E. Werner, A Comparison of Two Finite Element Models of Tidal Hydrodynamics Using a North Sea Data Set, *Advances in Water Resources*, 12(4), 184-193, 1989.

Walters, R.A., "Numerically Induced Oscillations in Finite Element Approximations to the Shallow Water Equations", *International Journal for Numerical Methods in Fluids*, 3, 591-604, 1983.

Walters, R.A., "Finite Element Solution Methods for Circulation in Estuaries", *Finite Elements in Water Resources; Proceedings of the 5th International Conference*, J.P. Laible et al. (eds.) Burlington, VT., 1984.

Walters, R.A., "A Model for Tides and Currents in the English Channel and Southern North Sea", *Advances in Water Resources*, 10, 138-148, 1987

Werner, F.E., and D.R. Lynch, Harmonic Structure of English Channel/Southern Bight Tides from a Wave Equation in Simulation, *Advances in Water Resources*, 12, 121-142., 1989.

Westerink, J.J., K.D. Stolzenbach and J.J. Connor, General Spectral Computations of the Nonlinear Shallow Water Tidal Interactions within the Bight of Abaco, *Journal of Physical Oceanography*, 19, 1348-1371, 1989.

Westerink, J.J., and W.G. Gray, Progress in Surface Water Modeling, *Reviews of Geophysics*, April Supplement, 210-217, 1991.

Westerink, J.J., R.A. Luettich, A.M. Baptista, N.W. Scheffner and P. Farrar, Tide and Storm Surge Predictions Using a Finite Element Model, *Journal of Hydraulic Engineering*, 118, 1373-1390, 1992a.

Westerink, J.J., J.C. Muccino and R.A. Luettich, Resolution Requirements for a Tidal Model of the Western North Atlantic and Gulf of Mexico, in Proceedings of the IX International Conference on Computational Methods in Water Resources, Denver, CO, T.F. Russell et al. eds., Computational Mechanics Publications, Southampton, UK, 1992b.

Westerink, J.J., R.A. Luettich and J.C. Muccino, "Modeling Tides in the Western North Atlantic Using Unstructured Graded Grids," *Tellus*, **46A**, 178-199, 1994a.

Westerink, J.J., R.A. Luettich, C.A. Blain and N.W. Scheffner, ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts and Estuaries, Report 2: Users Manual for ADCIRC-2DDI, *Coastal Engineering Research Center*, U.S. Army Engineers, 1994b.

Westerink, J.J., R.A. Luettich, J.K. Wu and R.L. Kolar, "The Influence of Normal Flow Boundary Conditions on Spurious Modes in Finite Element Solutions to the Shallow Water Equations," *International Journal for Numerical Methods in Fluids*, **18**, 1021-1060, 1994c.

Woodworth, P.L., Summary of Recommendations to the UK Earth Observation Data Centre (UK-EODC) by the Proudman Oceanographic Laboratory (POL) for Tide Model Corrections on ERS-1 Geophysical Data Records, Proudman Oceanographic Laboratory Comm., 1990.

LIST OF FIGURES

ADDITIONAL PRODUCTS DELIVERED WITH THIS REPORT

1. Grid file for Galveston Bay and vicinity: gal_g03_r4.grd
2. Input file for Galveston Bay and vicinity: gal_g03_r4.inp
3. Grid file for the Gulf of Mexico including Galveston Bay and vicinity: gomgal_g05_r4.grd
4. Input file for the Gulf of Mexico including Galveston Bay and vicinity: gomgal_g05_r4.inp
5. Grid file for the Eastcoast including Galveston Bay and vicinity: eastgal_g03_r1.grd
6. Input file for the Eastcoast including Galveston Bay and vicinity: eastgal_g03_r1.inp
7. ADCIRC source codes: adc31_06.f and srcv2d.f
8. ADCIRC setup program: adcsetup31_06.f
9. Movie files with results of gal_g03_r4 simulation
   (a) gal_entire.flc
   (b) gal_z1.flc
10. Movie files with results of gomgal_g05_r4 simulation
    (a) gomgal_entire1.flc
    (b) gomgal_z1.flc
    (c) gomgal_z2.flc
11. Movie files with results of eastgal_g03_r1 simulation
    (a) eastgal_entire1.flc
    (b) eastgal_z1.flc
    (c) eastgal_z2.flc

Figure 1. Maps of pertinent domains indicating bathymetry in meters.
(a) Galveston Bay, Texas and vicinity

Figure 1. Maps of pertinent domains indicating bathymetry in meters.
(b) Gulf of Mexico

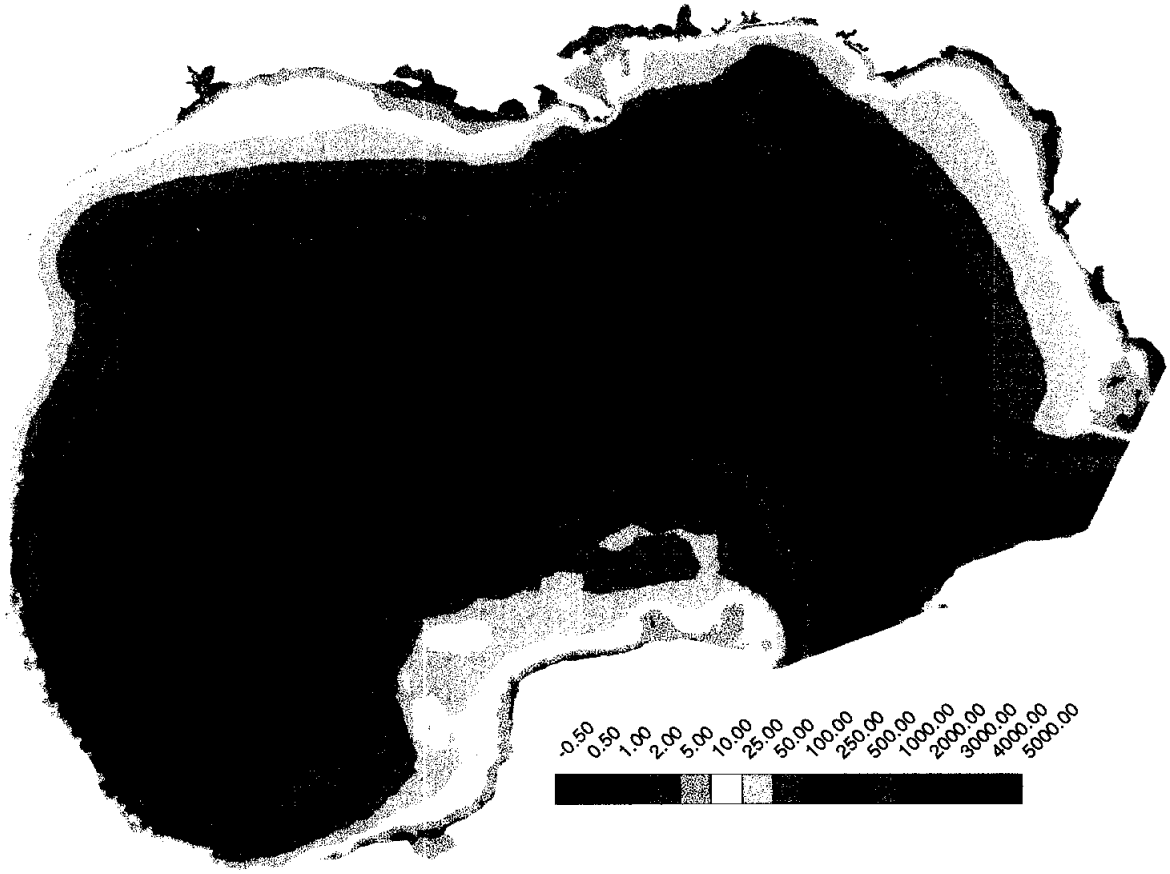


Figure 1. Maps of pertinent domains indicating bathymetry in meters.
(c) Eastcoast including the Gulf of Mexico and Caribbean Sea

Figure 2. Finite element discretizations
(a) Galveston Bay, Texas and vicinity: *GAL_G03_R4*

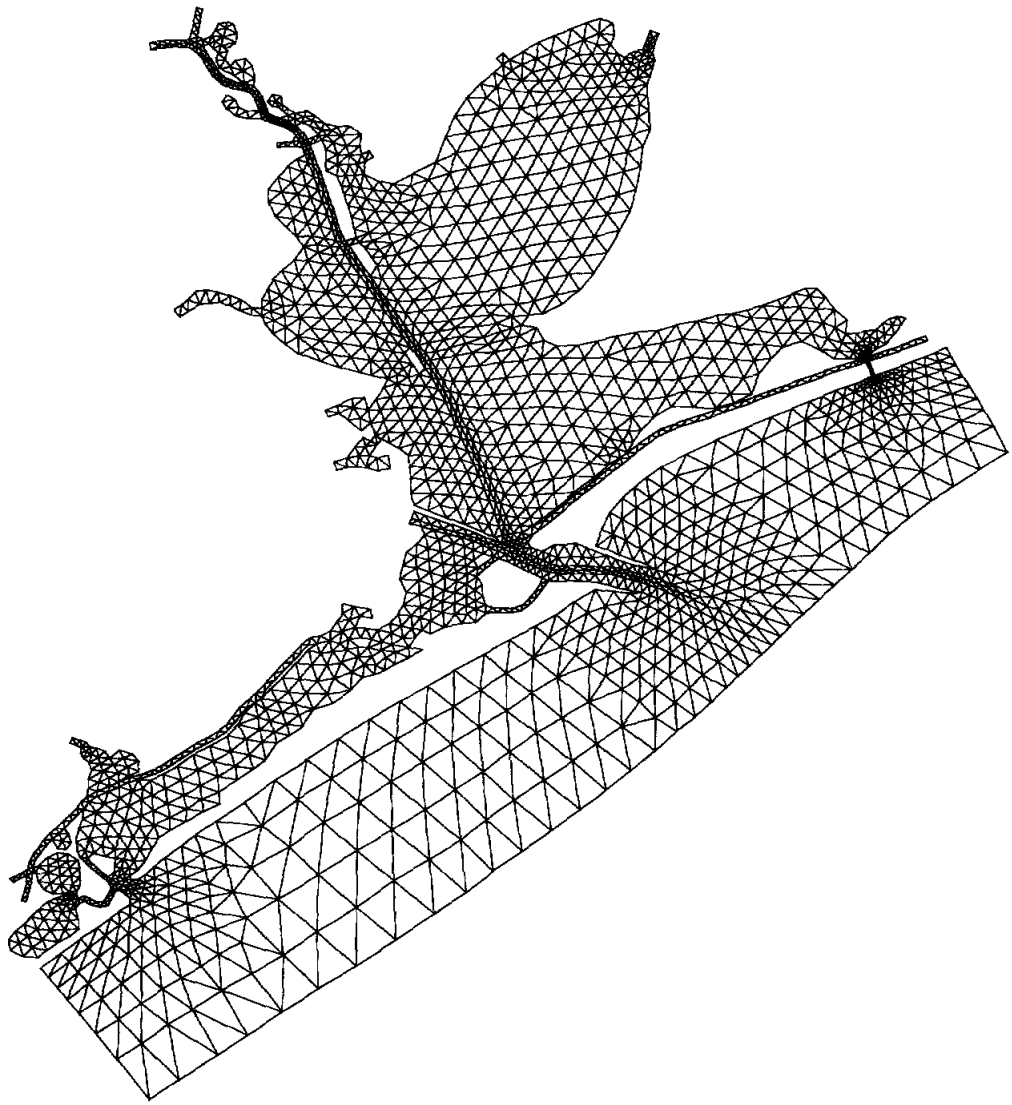


Figure 2. Finite element discretizations
(b) Gulf of Mexico: *GOMGAL_G05_R4*

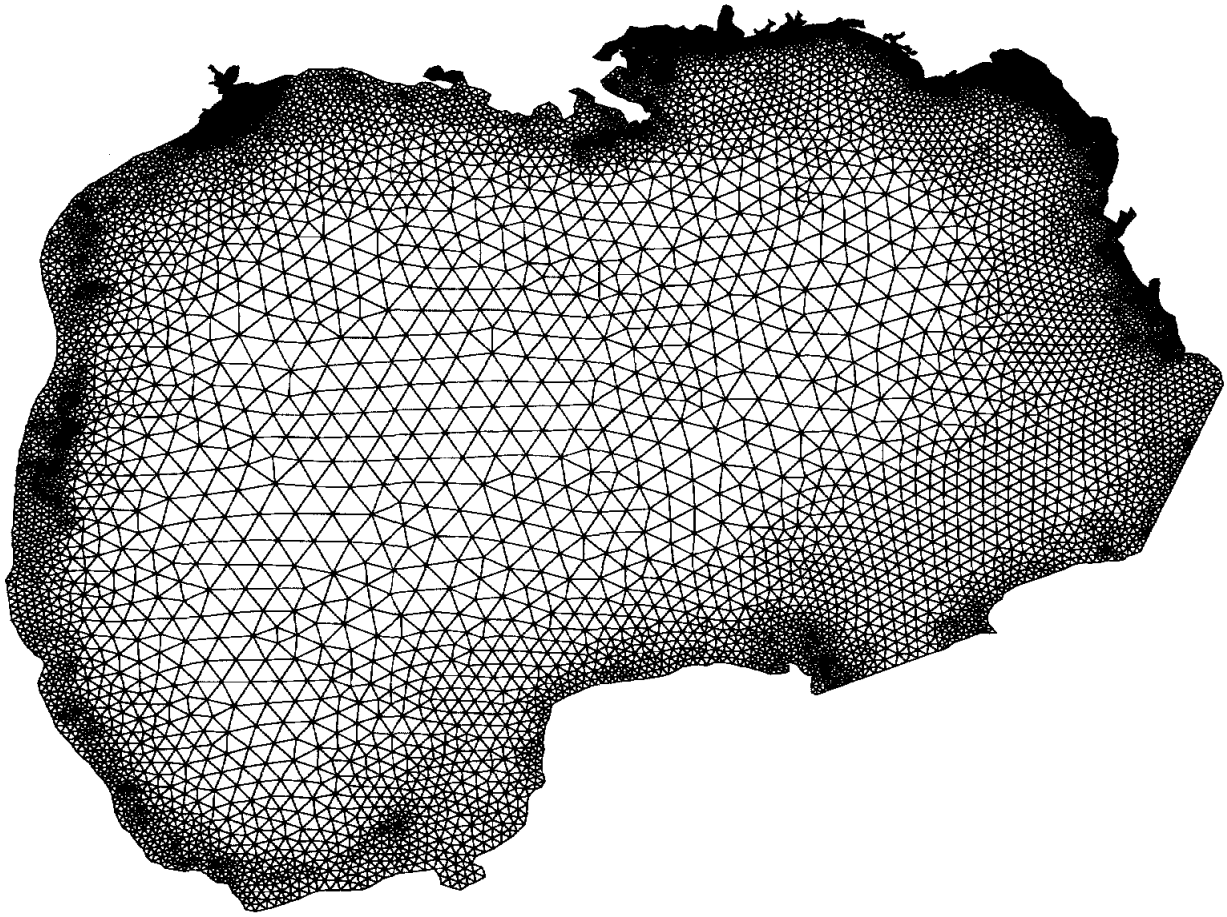


Figure 2. Finite element discretizations
(c) Eastcoast: *EASTGAL_G03_R1*

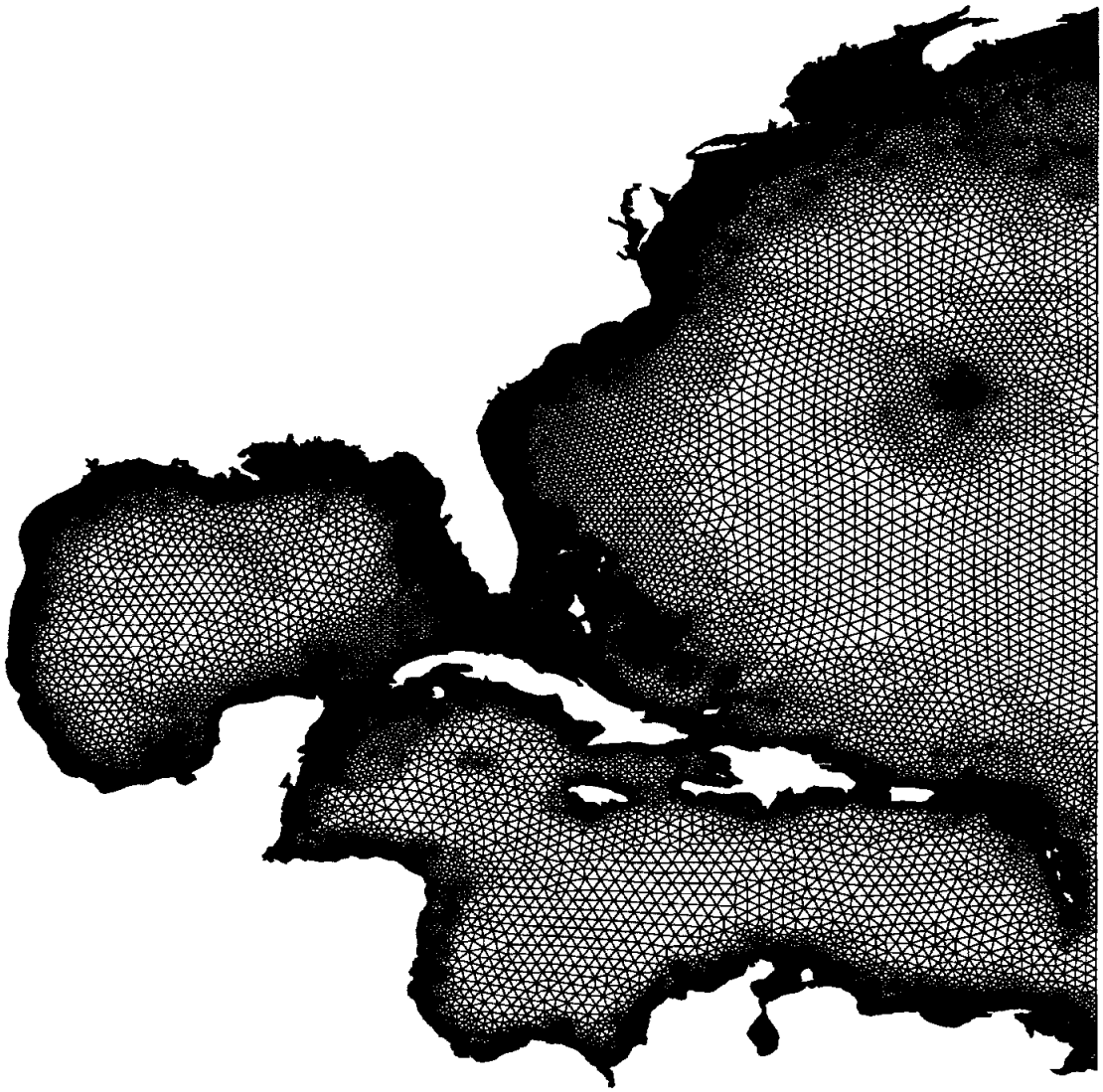


Figure 3. Locations of tidal elevation data (stations 1-19) and additional data (stations 20-26) recording stations.
(a) Gulf of Mexico

Figure 3. Locations of tidal elevation data (stations 1-19) and additional data (stations 20-26) recording stations.
(b) Detail of Galveston Bay, Texas

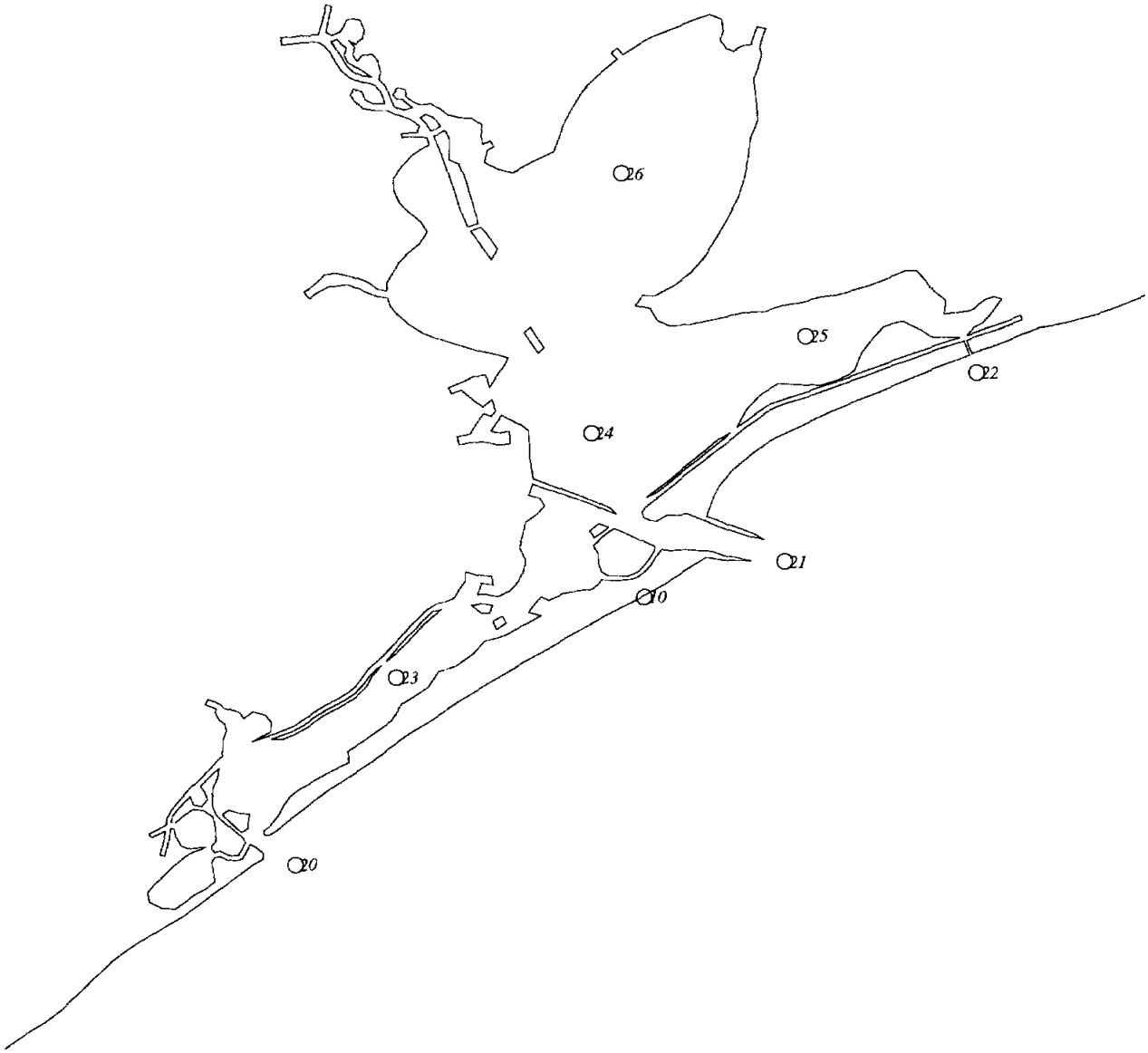


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (a) K_1 amplitude

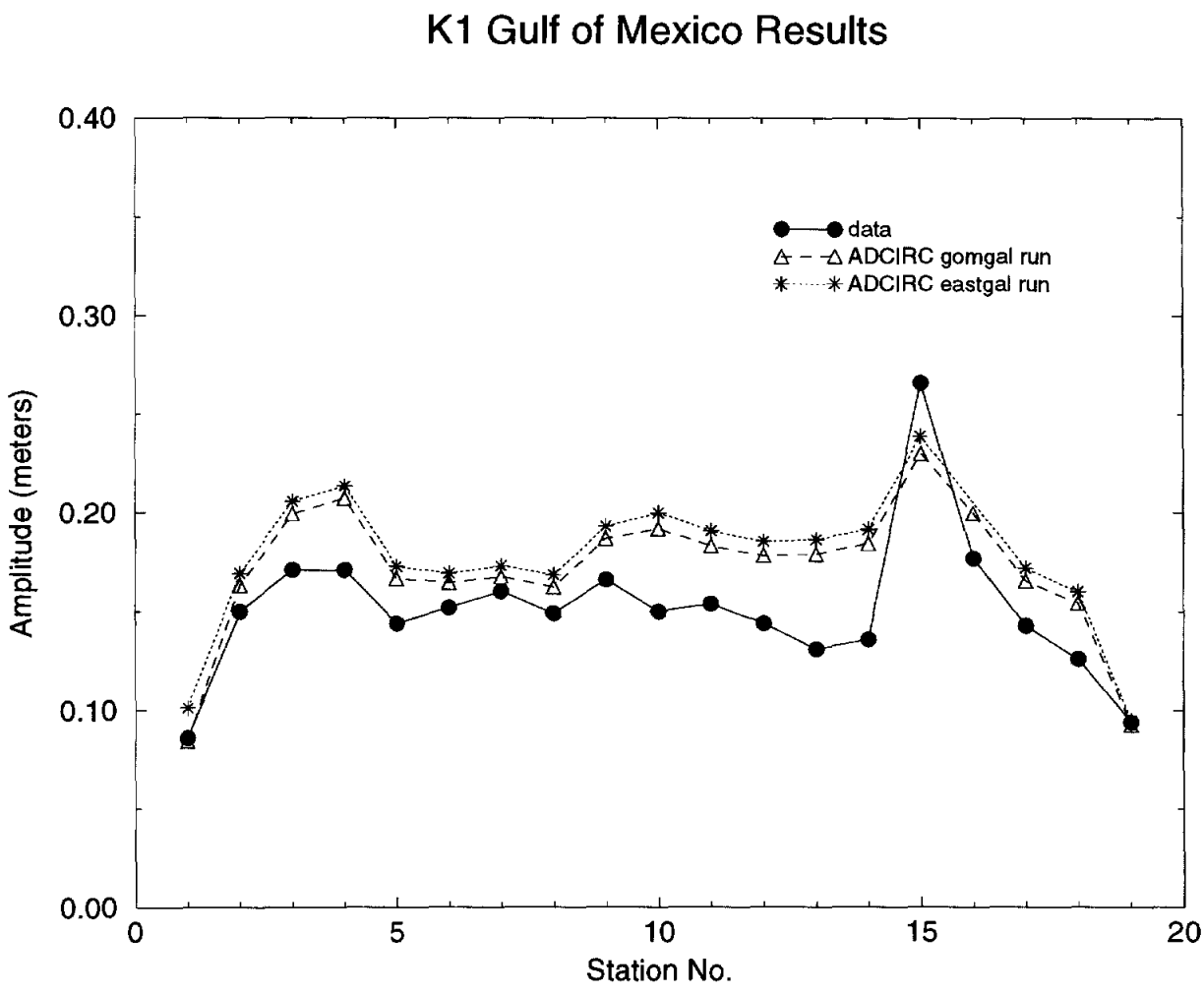


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (b) K_1 phase

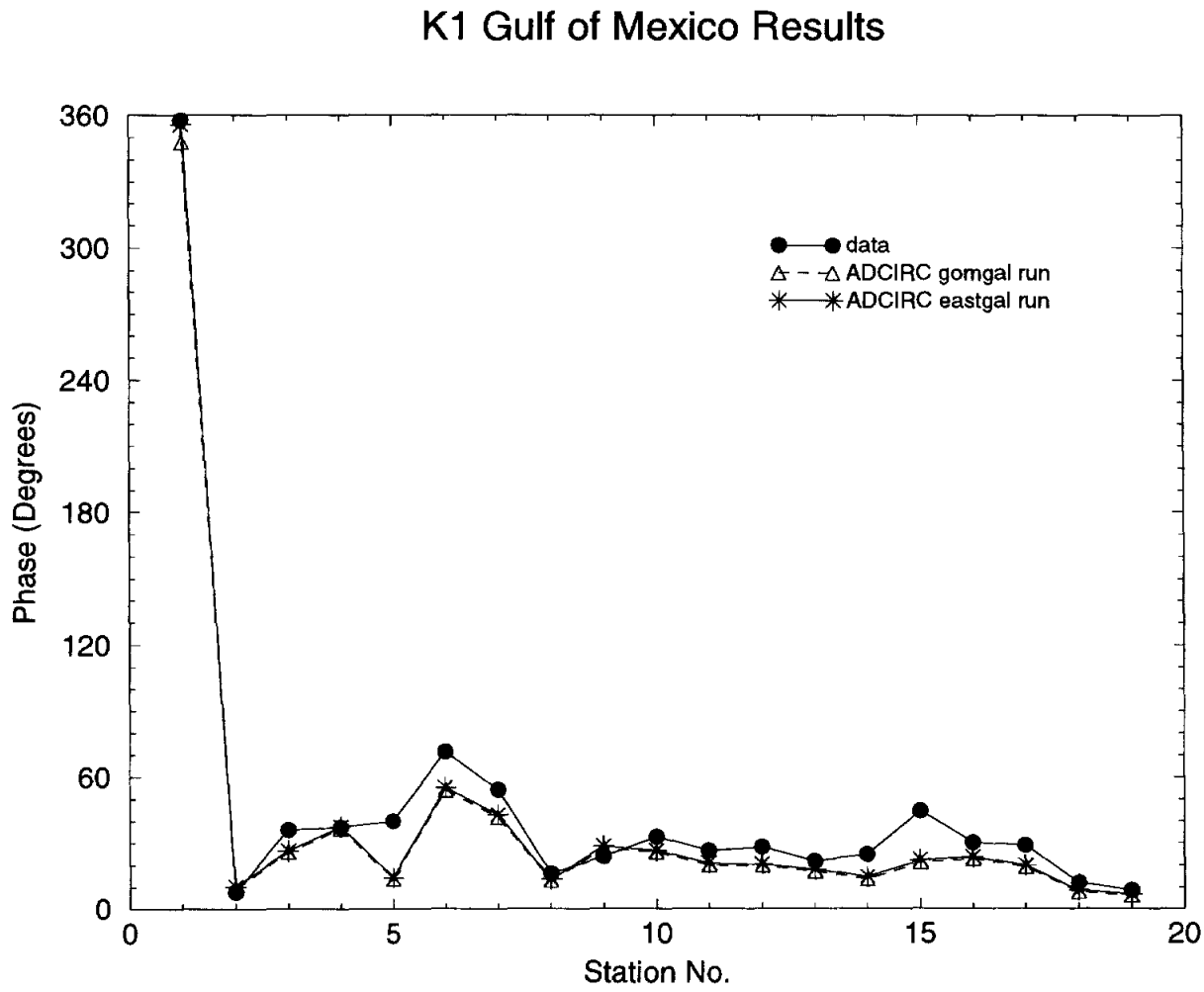


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(c) M_2 amplitude

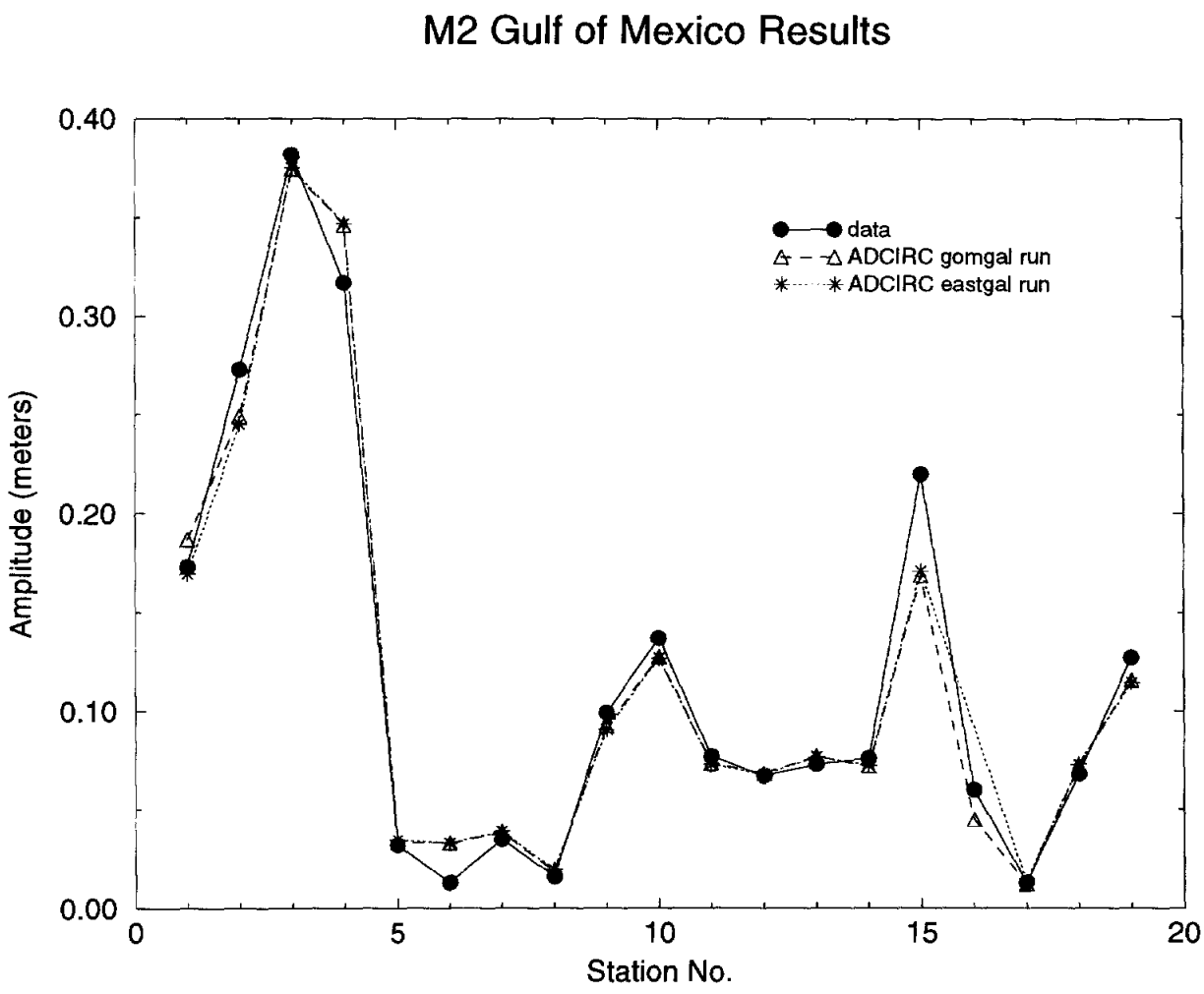


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (d) M_2 phase

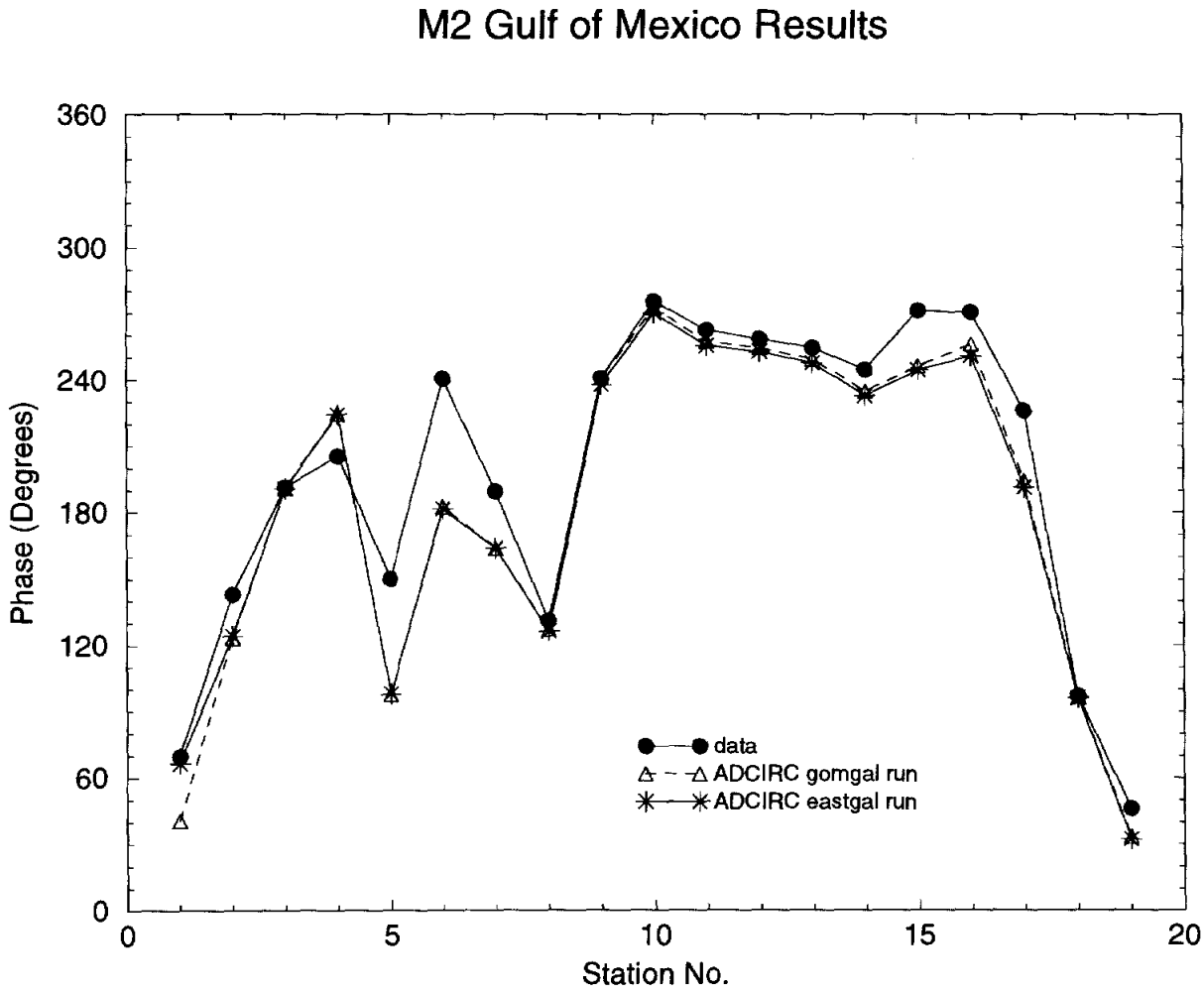


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(e) N_2 amplitude

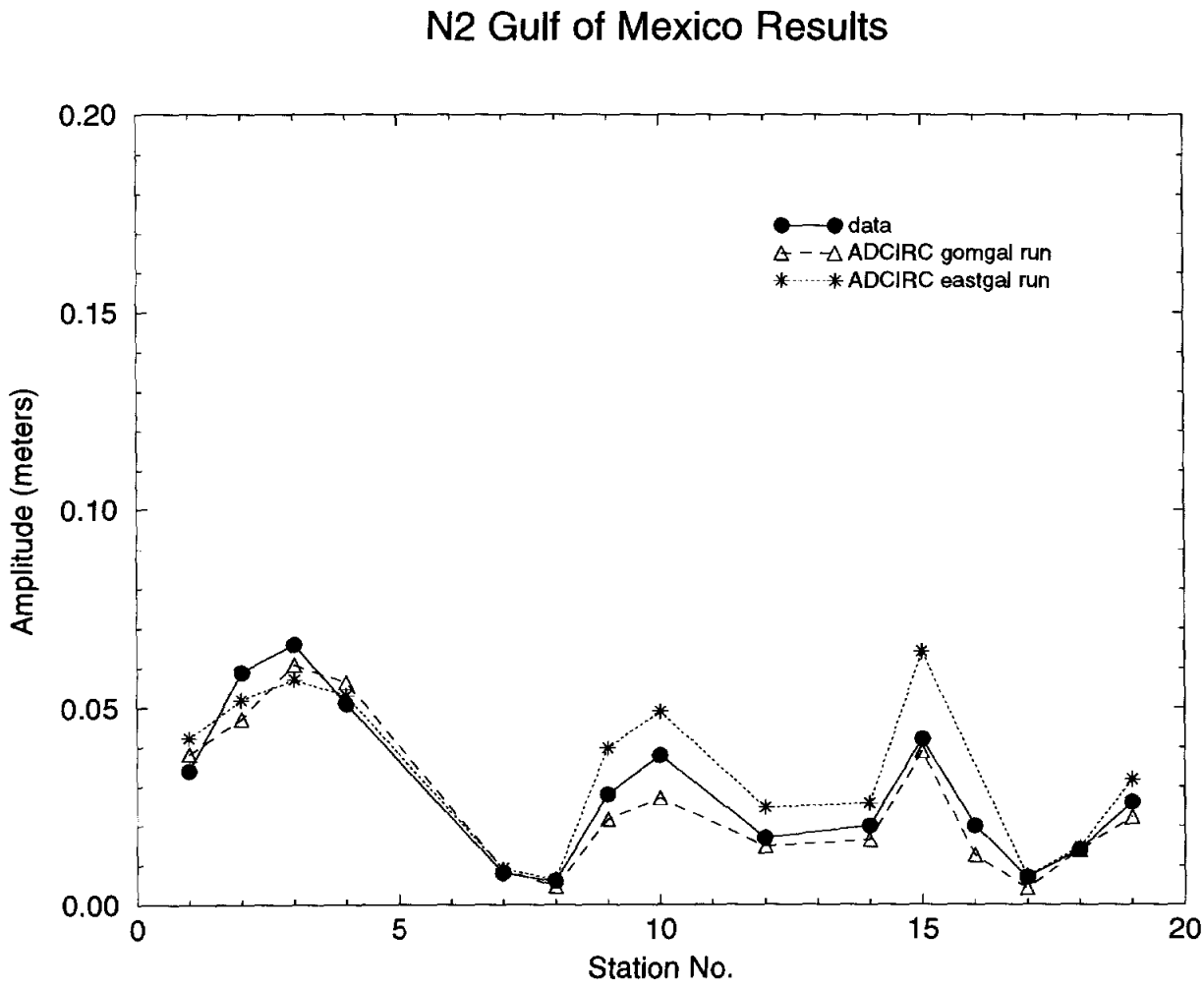


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(g) O_1 amplitude

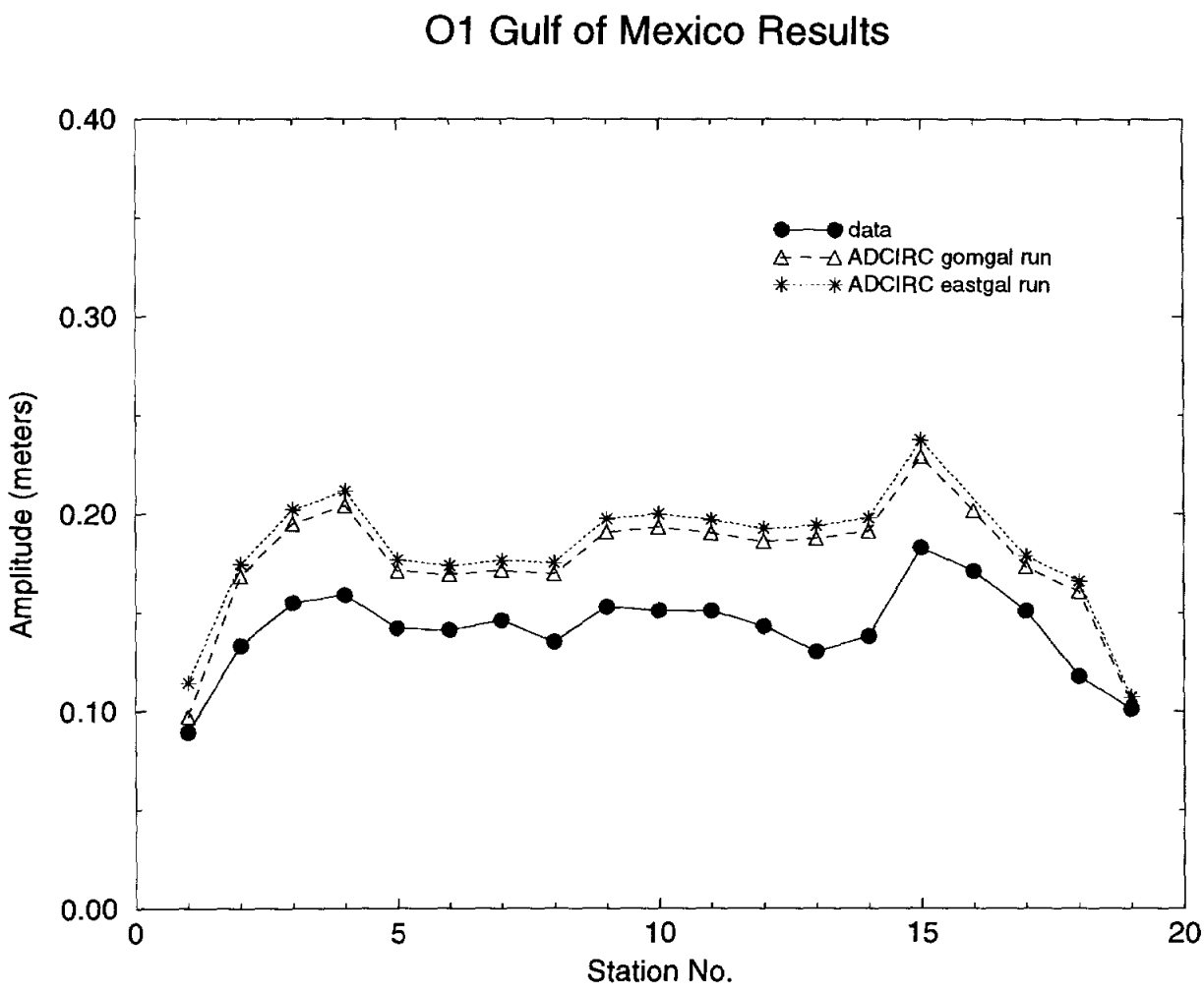


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (h) O_1 phase

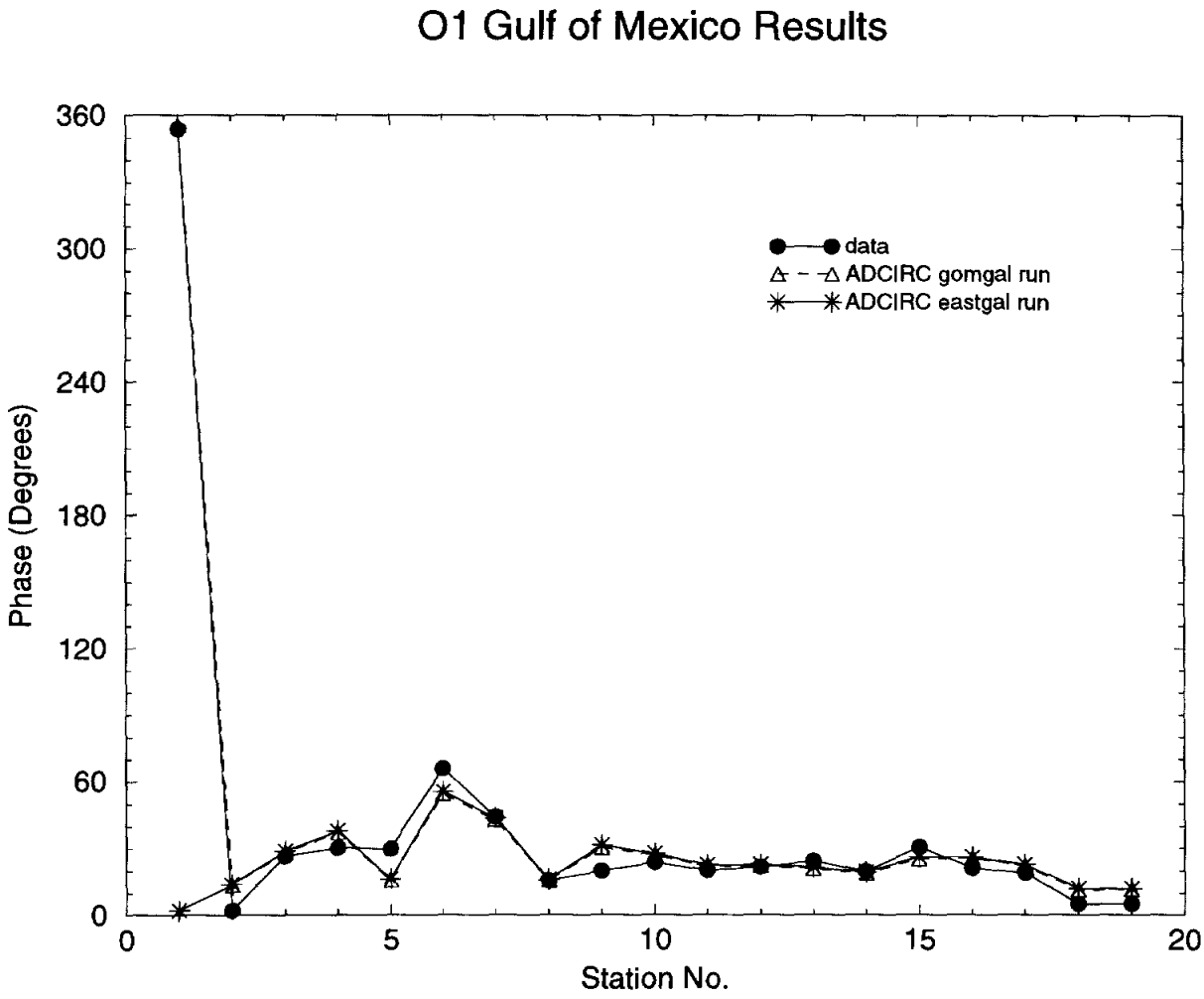


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(i) S_2 amplitude

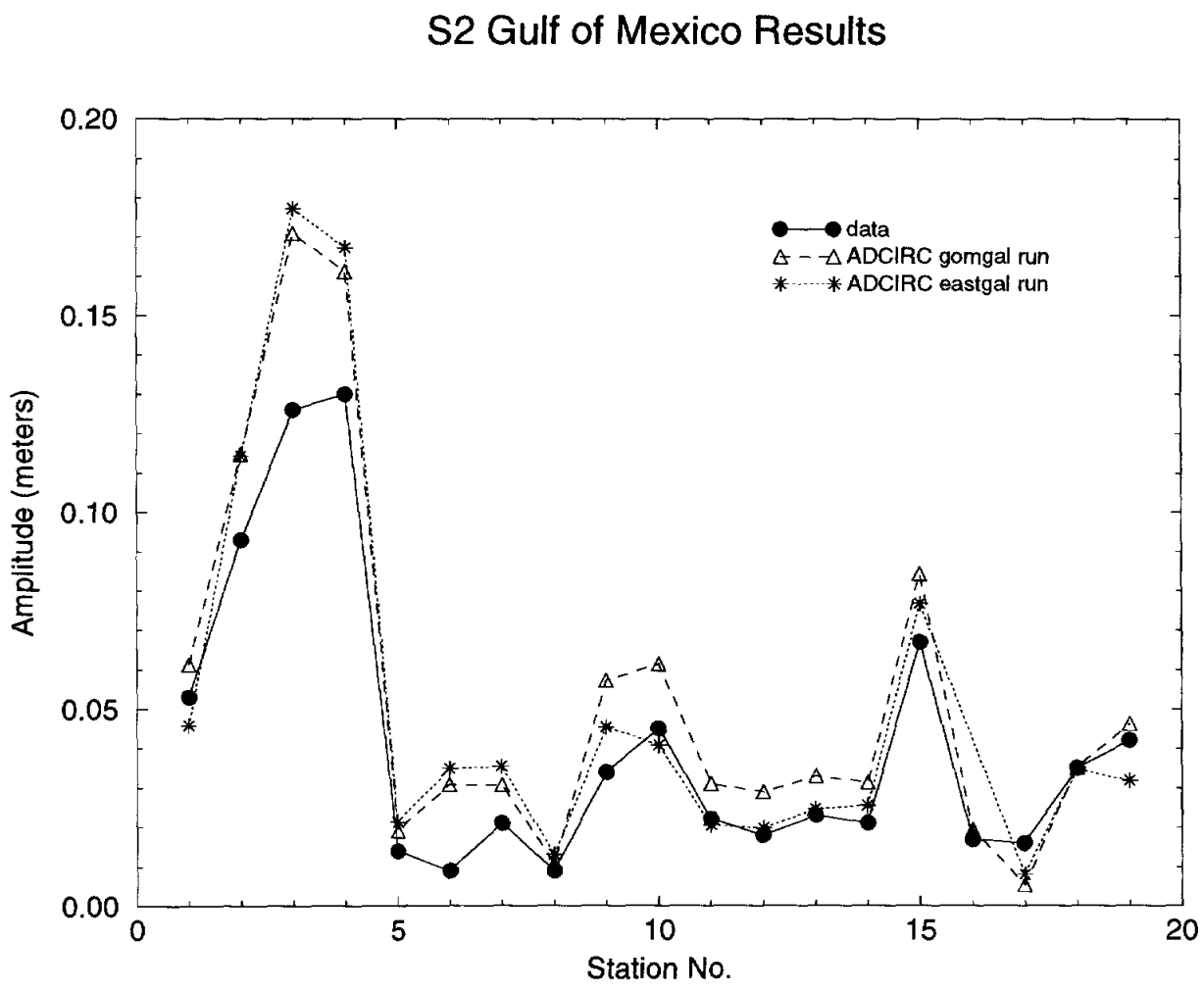


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(j) S_2 phase

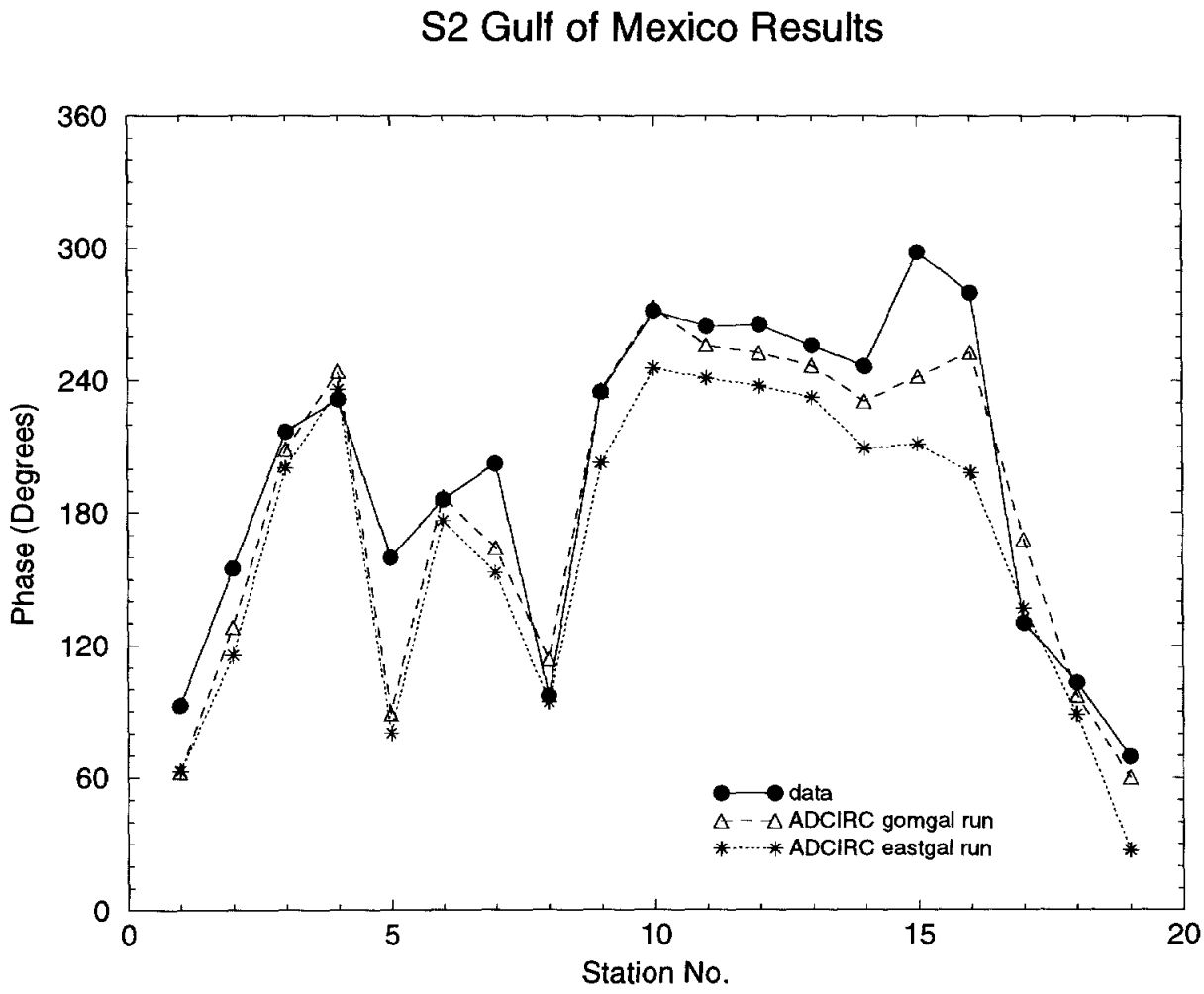


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(a) Key West, FL

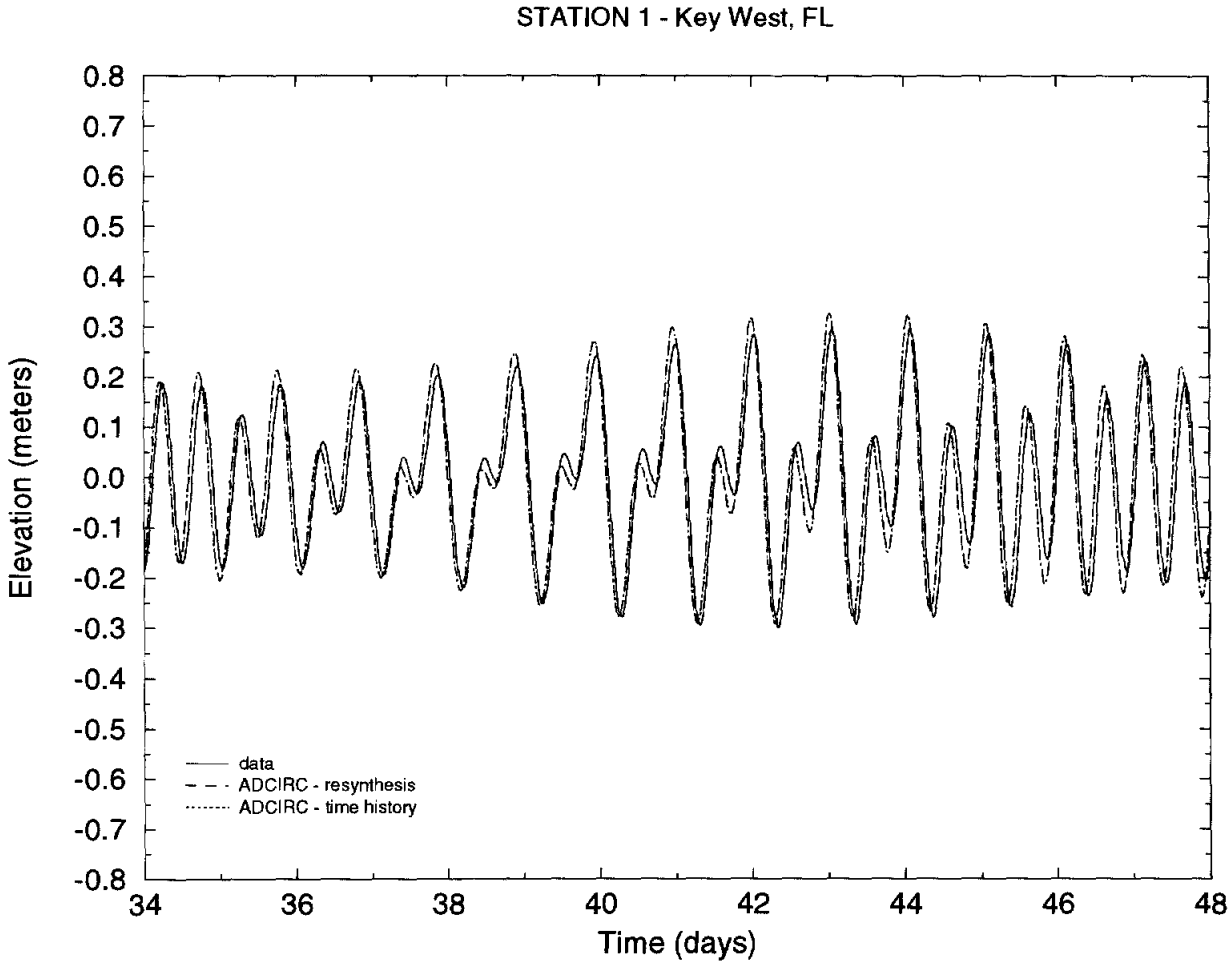


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(b) Naples, FL

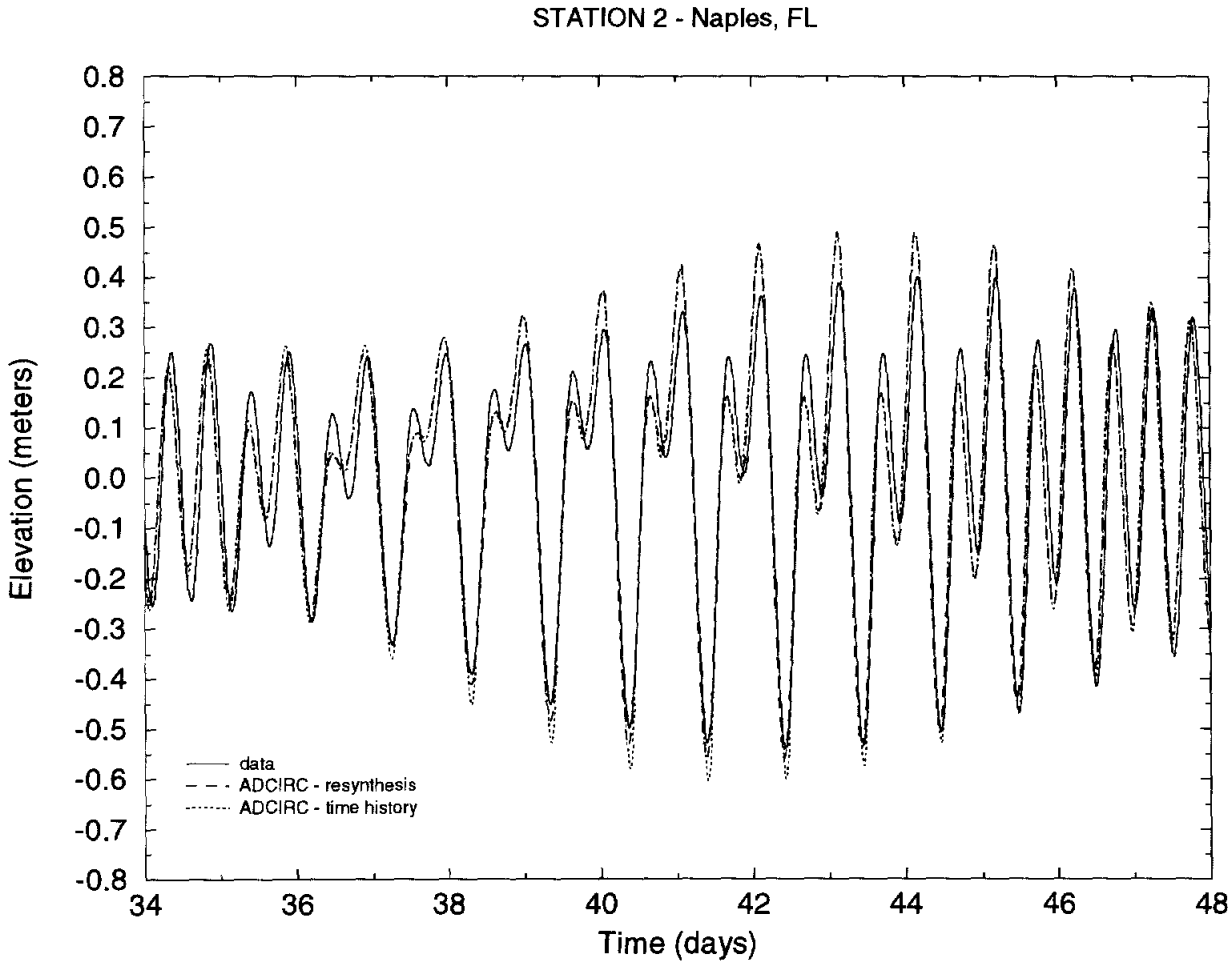


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(c) Cedar Key, FL

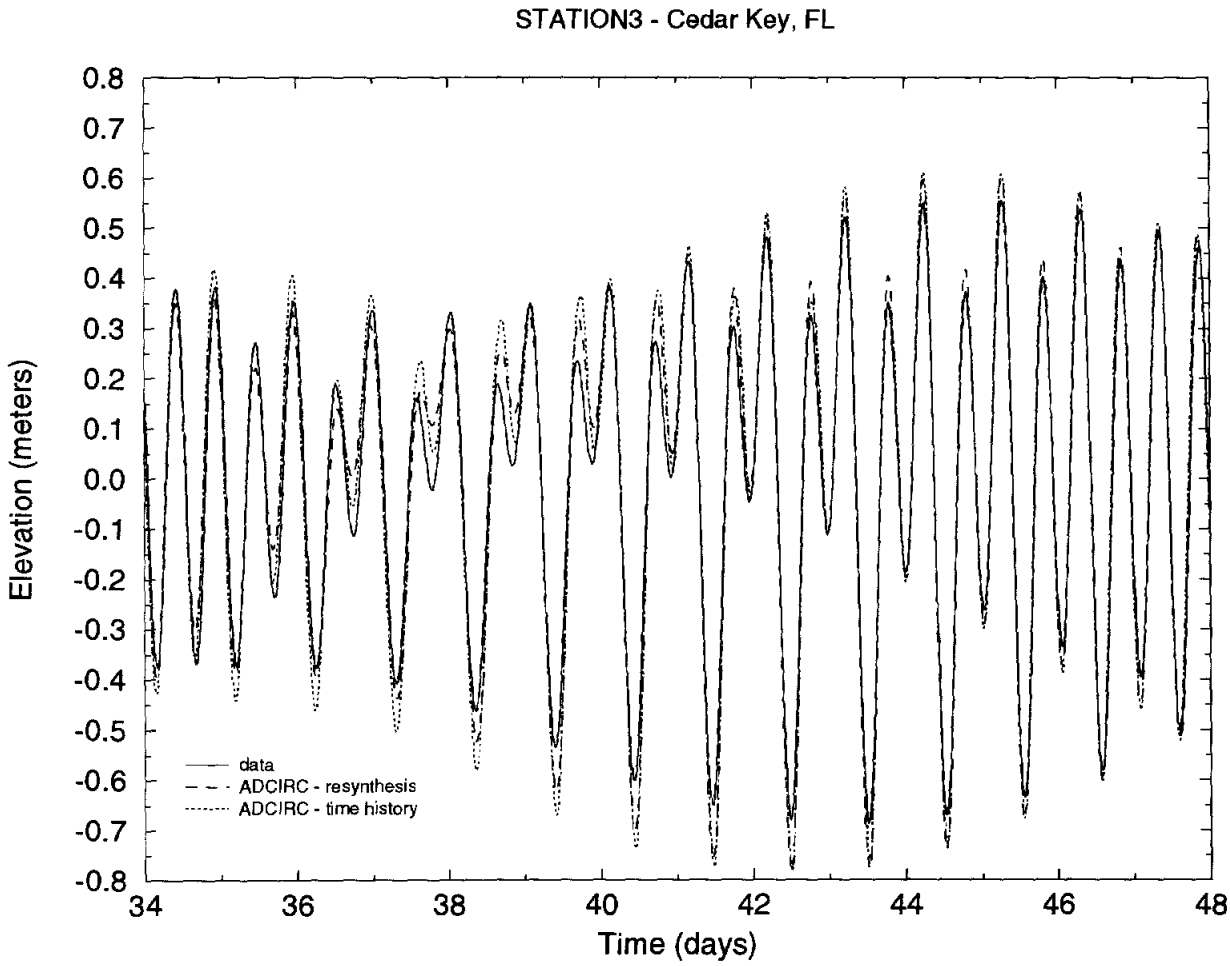


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(d) St. Marks Light, FL

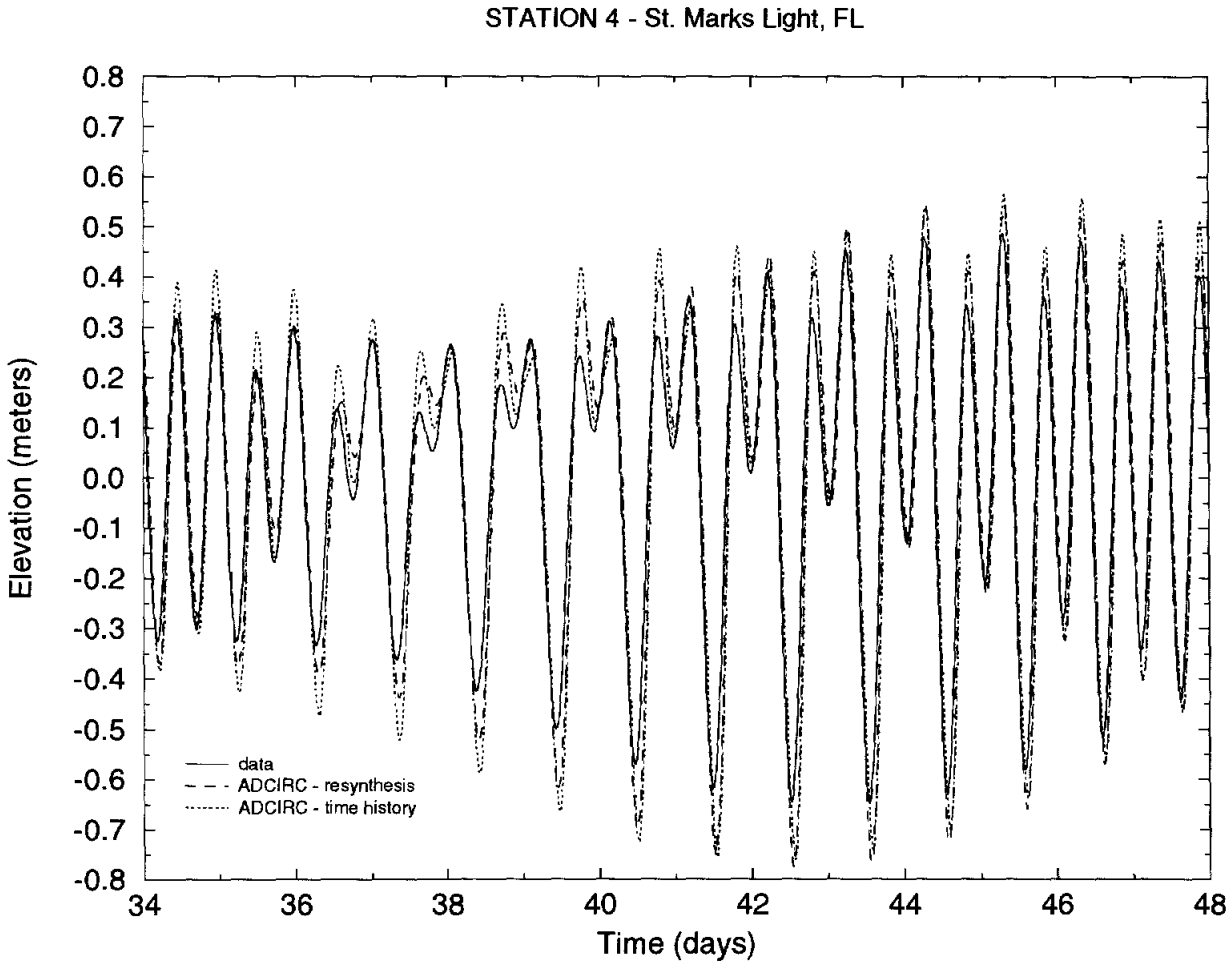


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(e)Alligator Bayou, FL

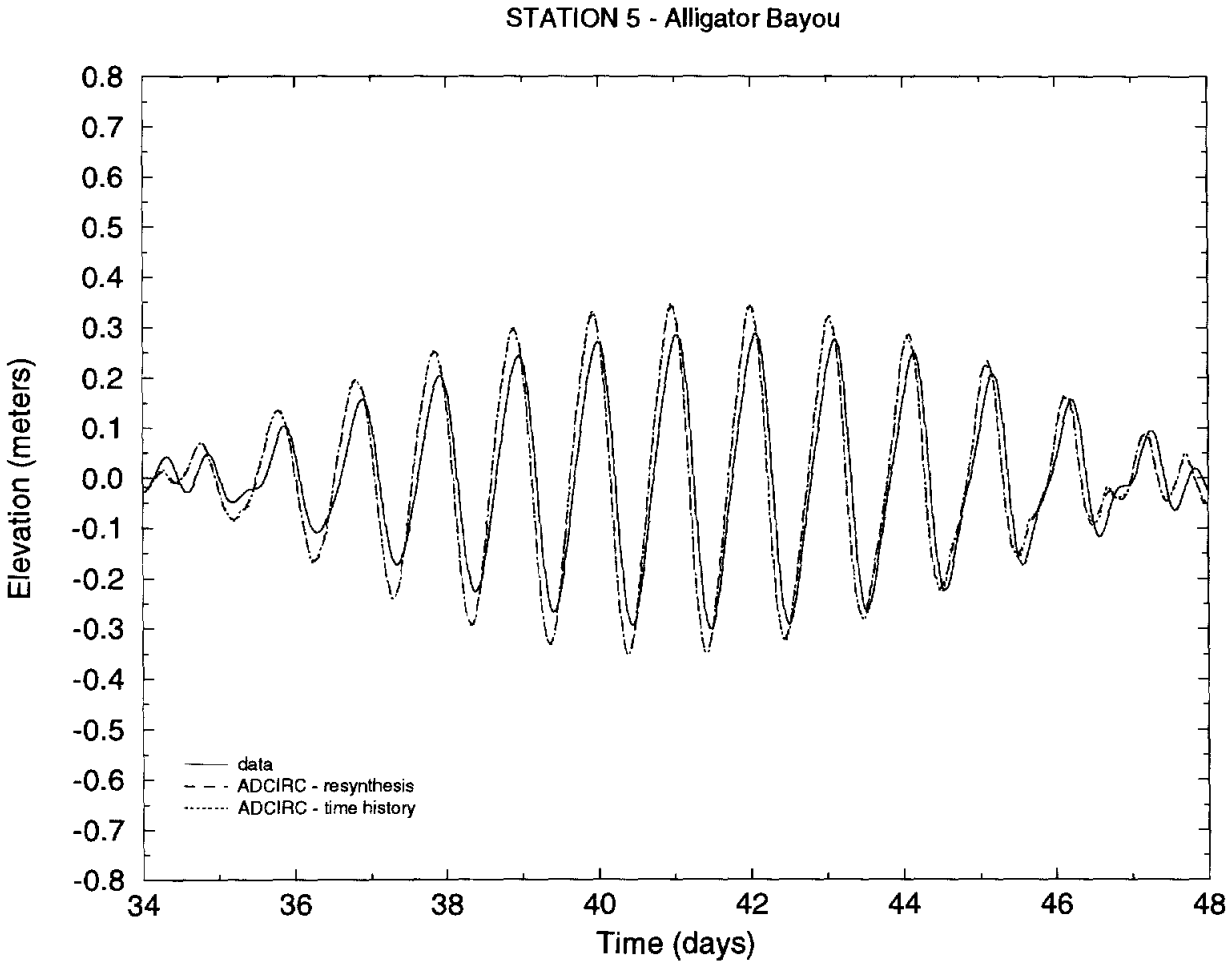


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(f) Bay St. Louis, MS

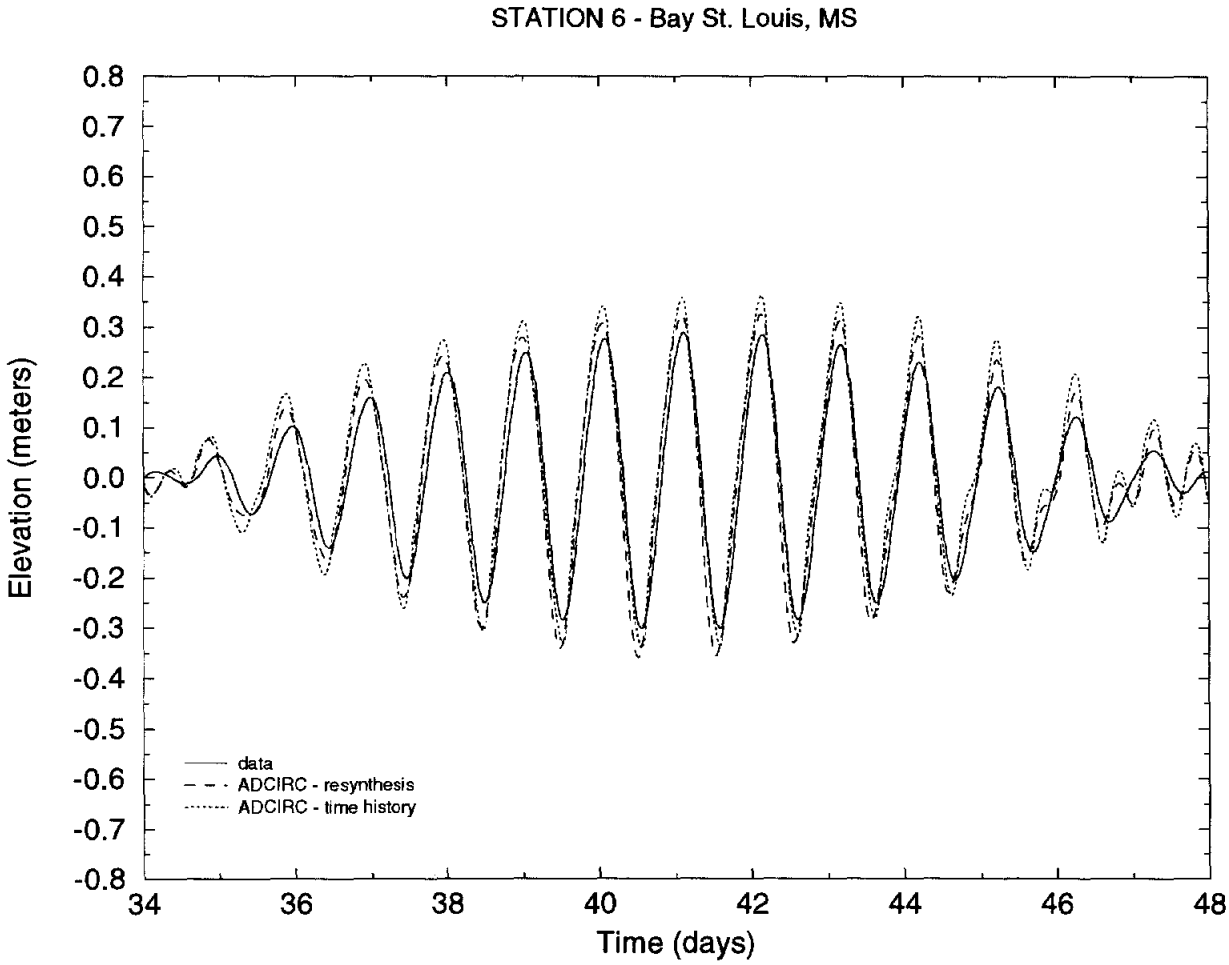


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(g) Cat Island, MS

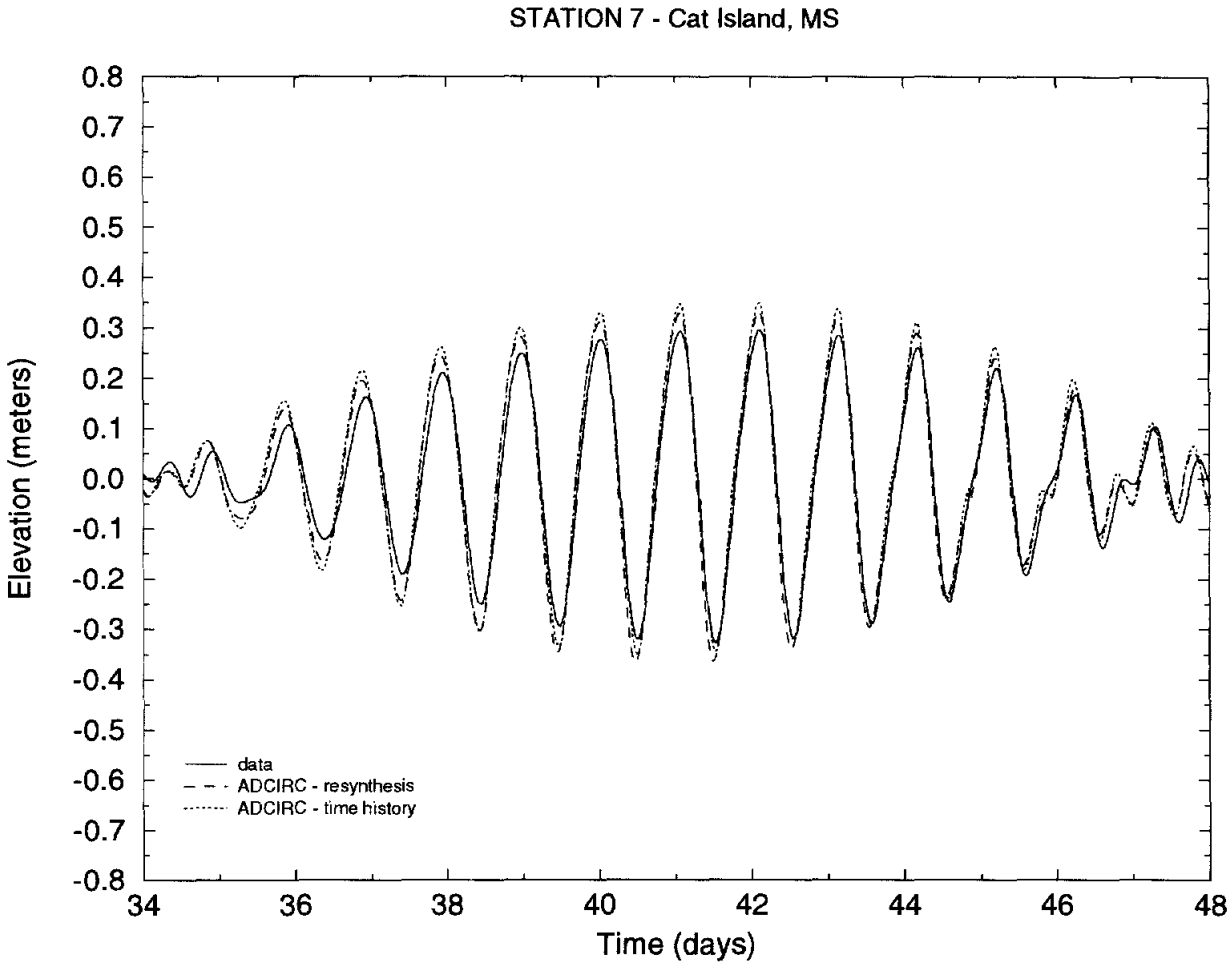


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(h) Southwest Pass, LA

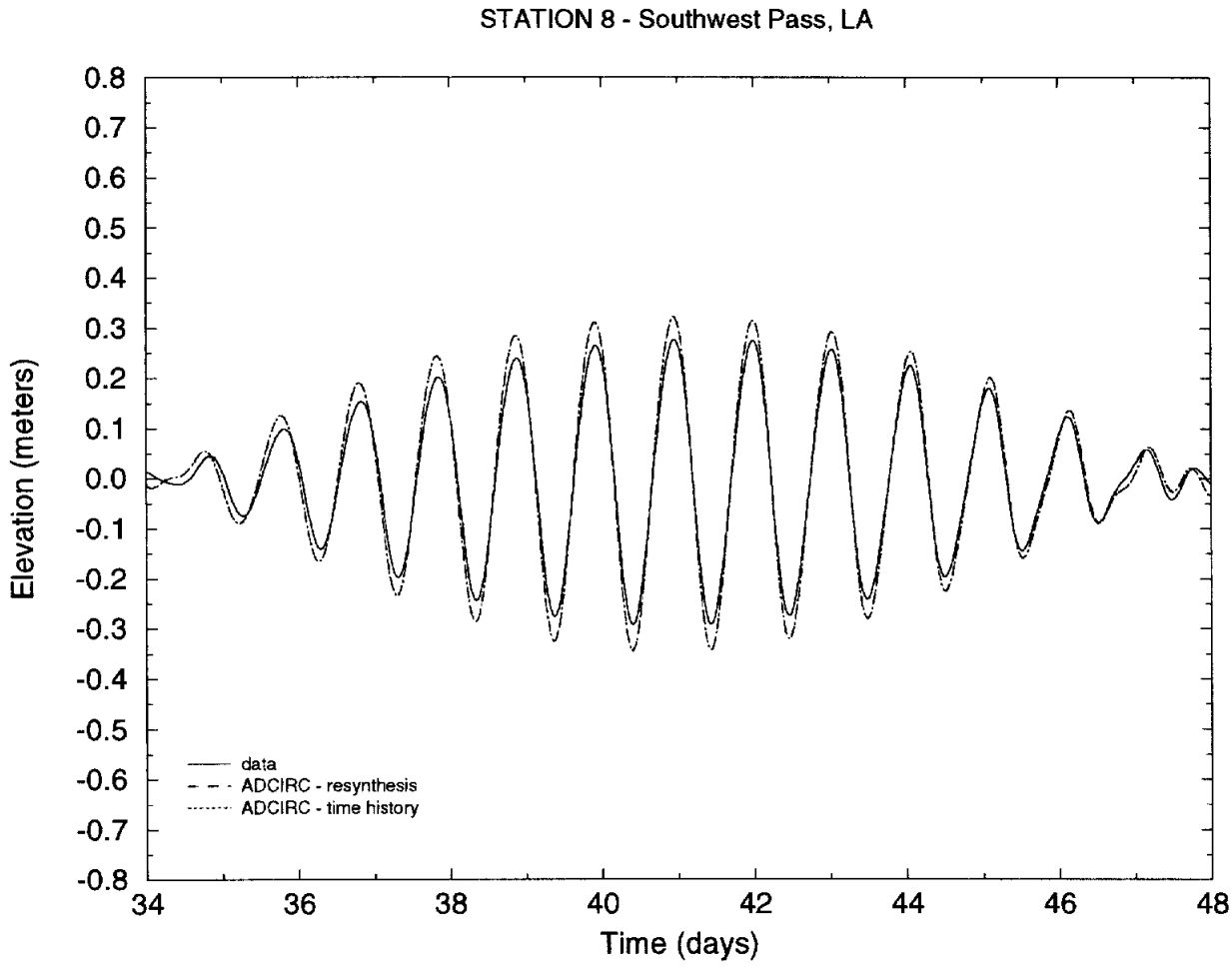


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(i) Point au Fer, LA

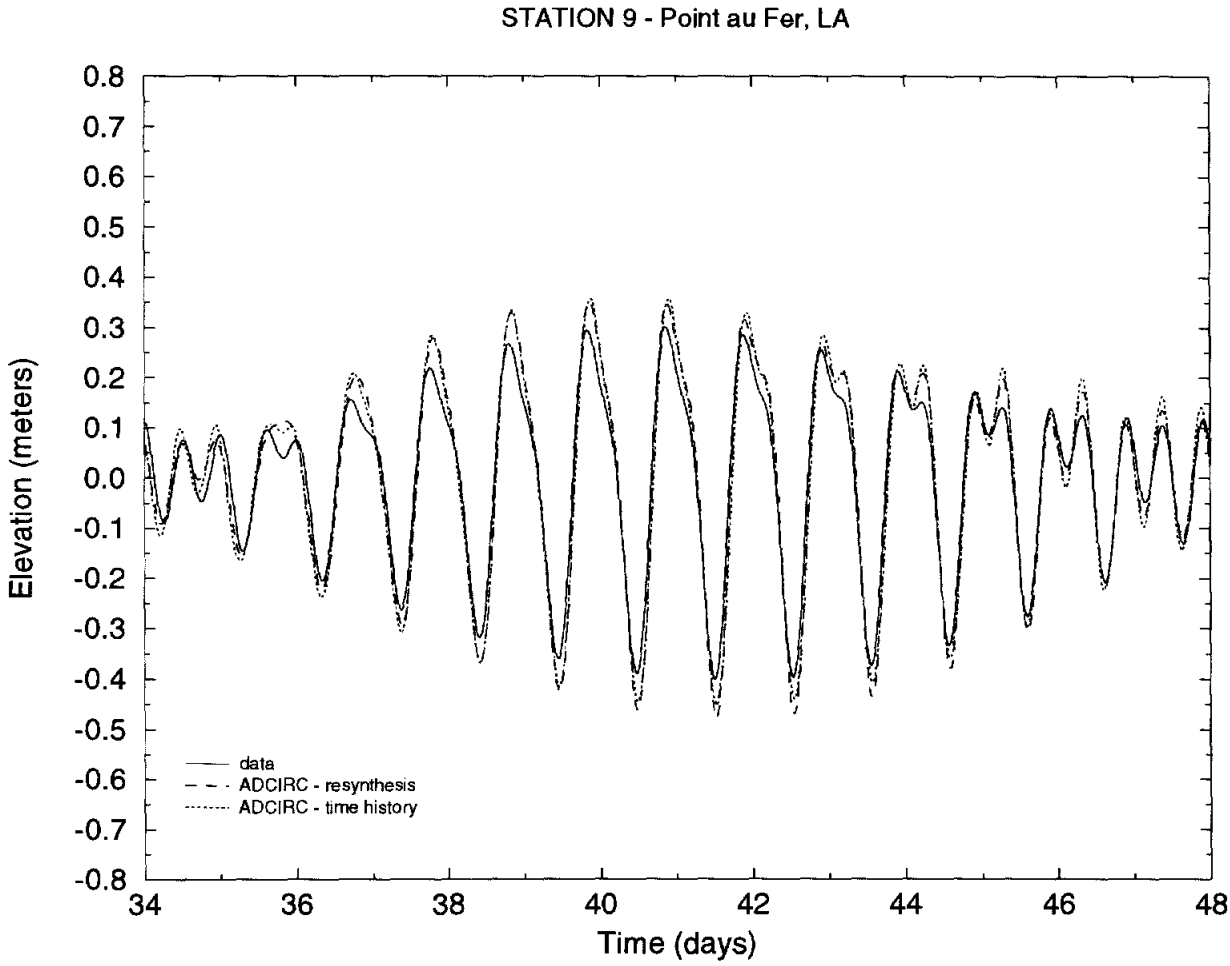


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(j) Galveston, TX

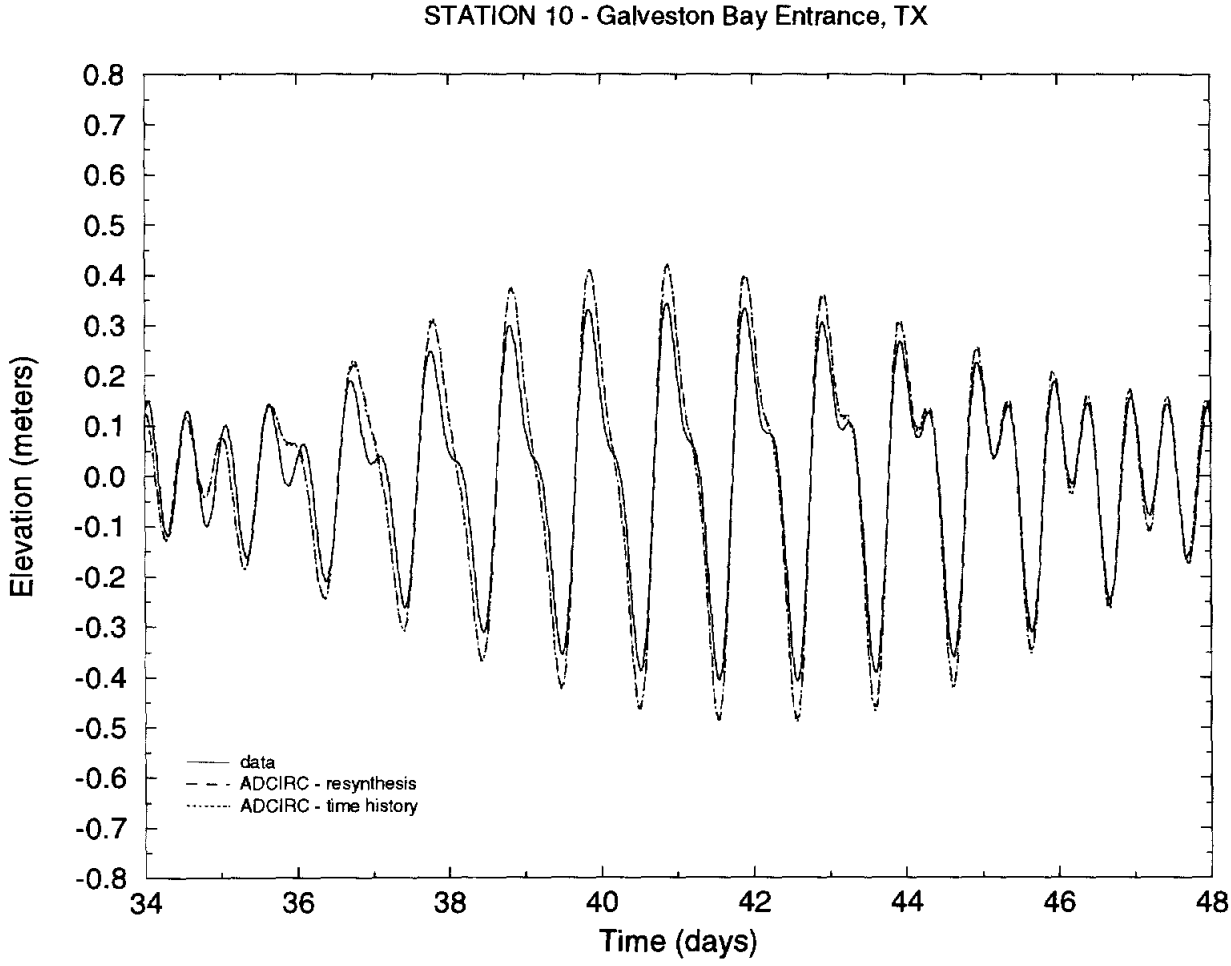


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(k) Port Arkansas, TX

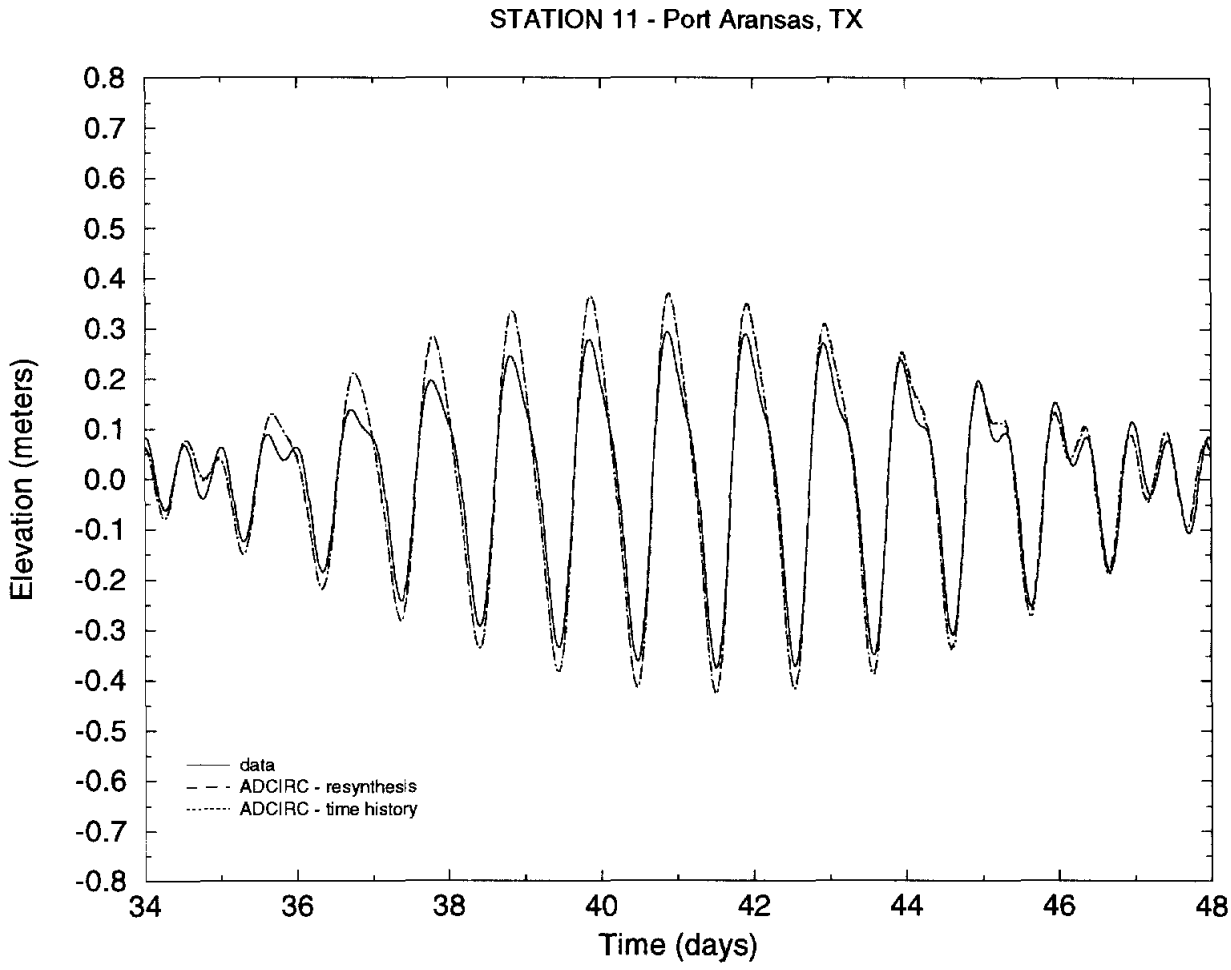


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(l) South Padre Island, TX

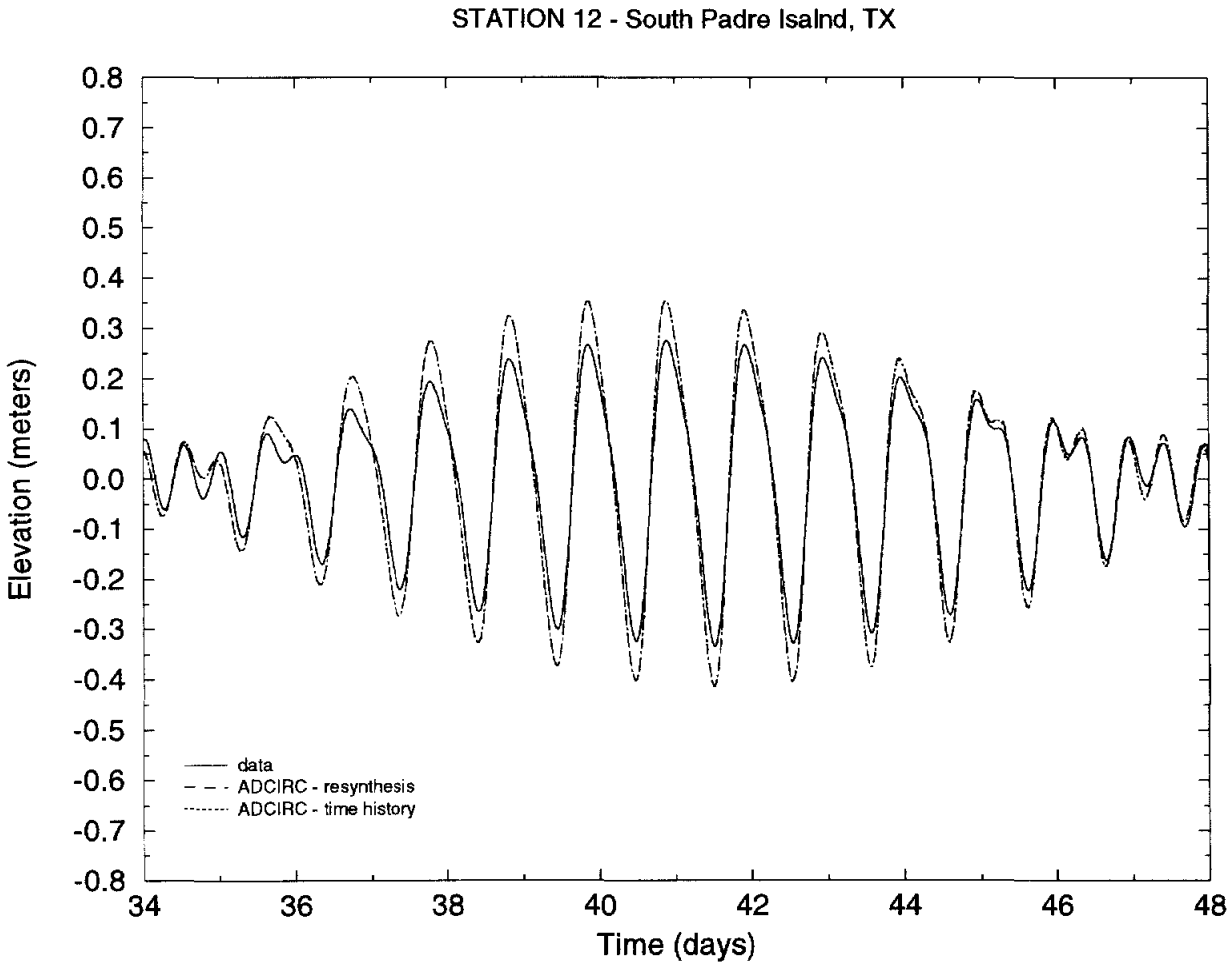


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(m) Madero, Mexico

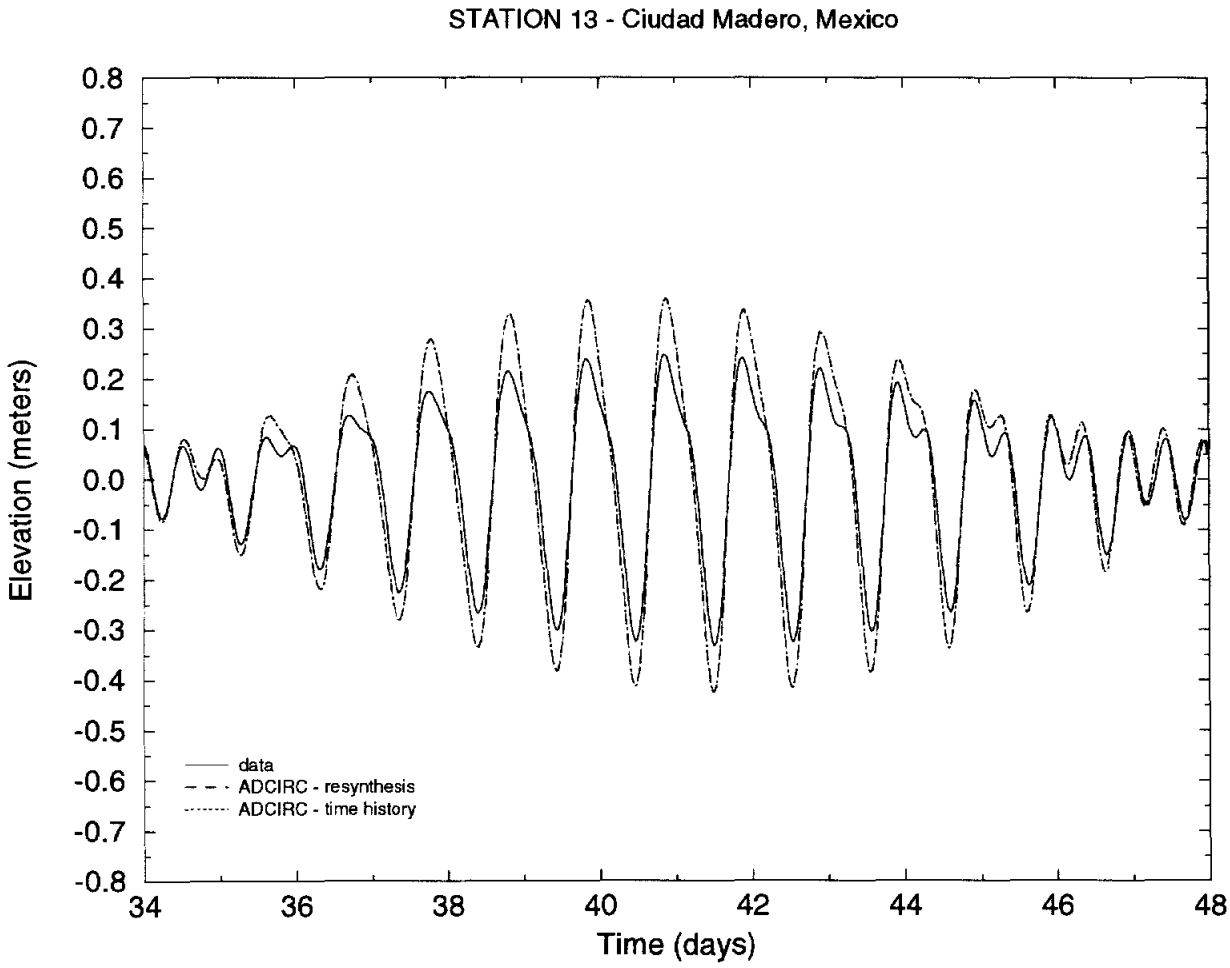


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(n) Coatzacoalcos, Mexico

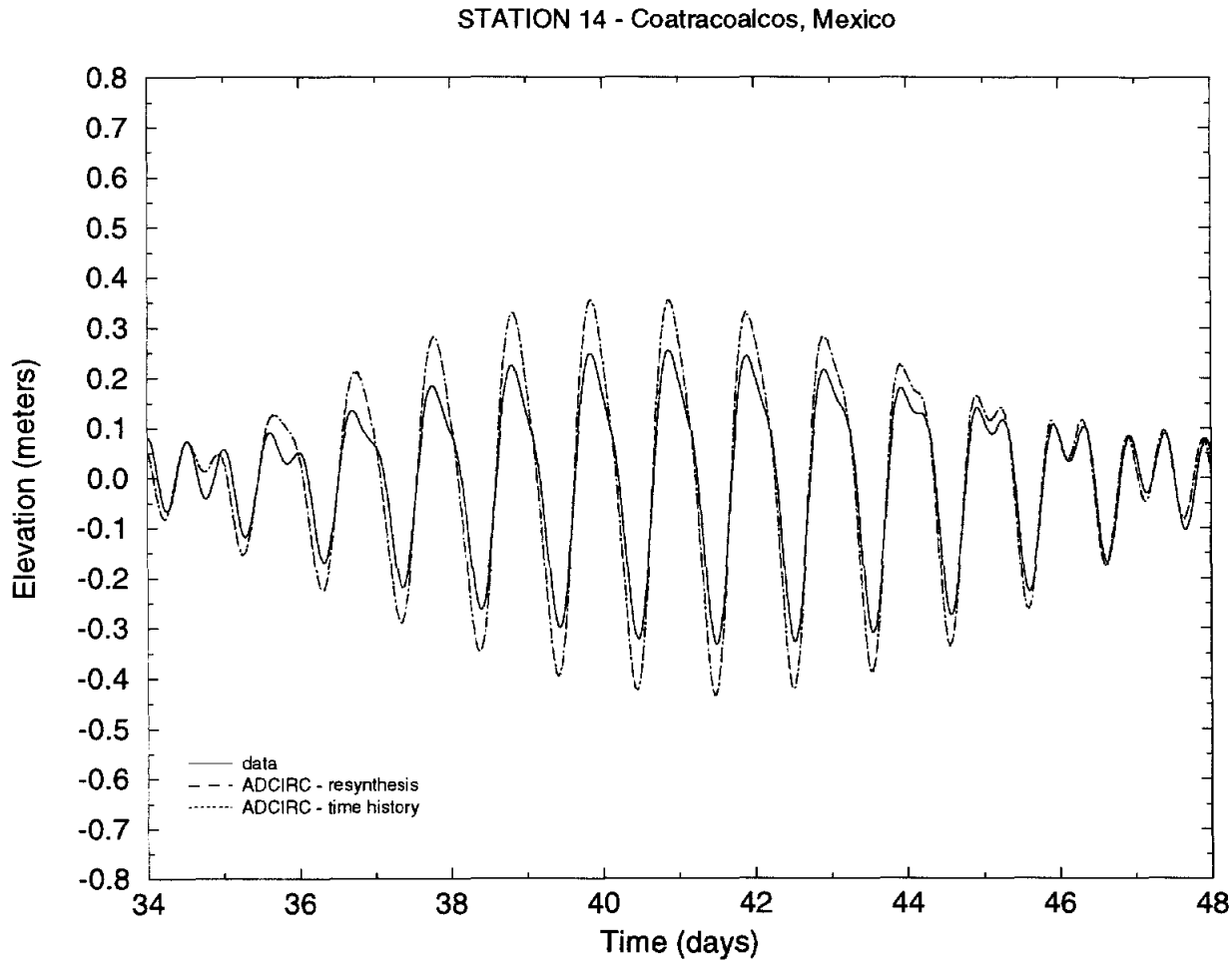


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(o) Campeche, Mexico

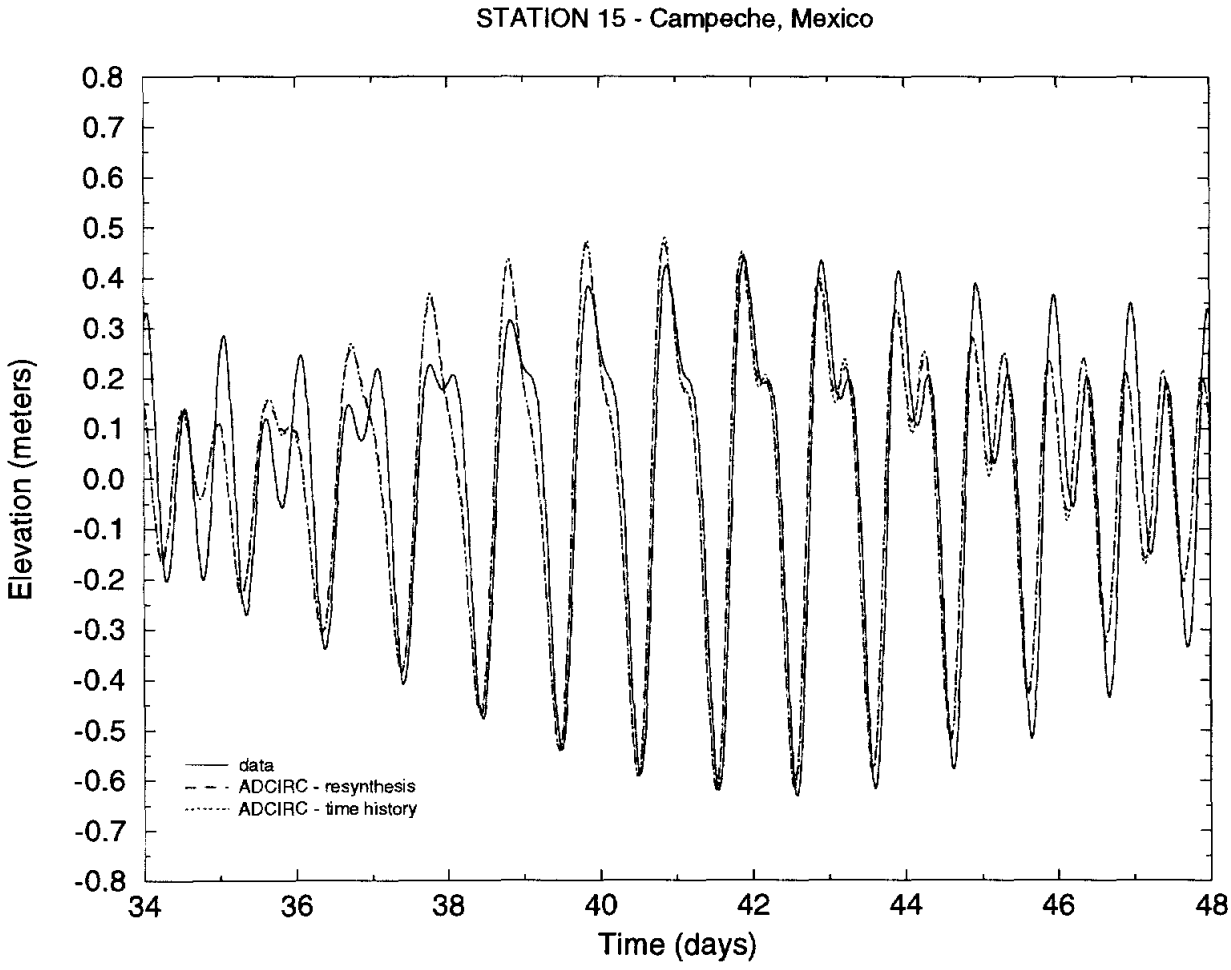


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(p) Progreso, Mexico

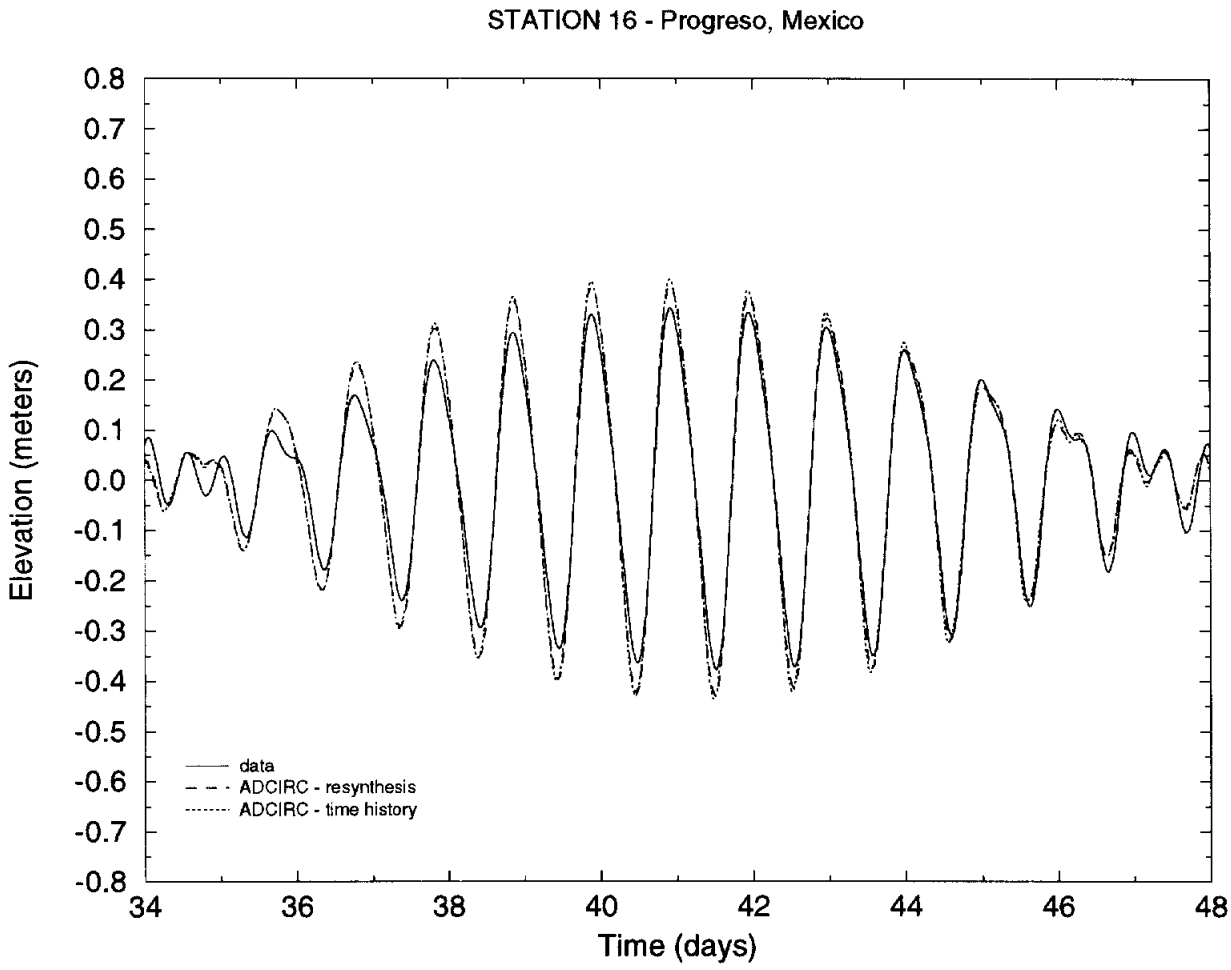


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(q) GOM Pelagic - IAPSO #30-1.2

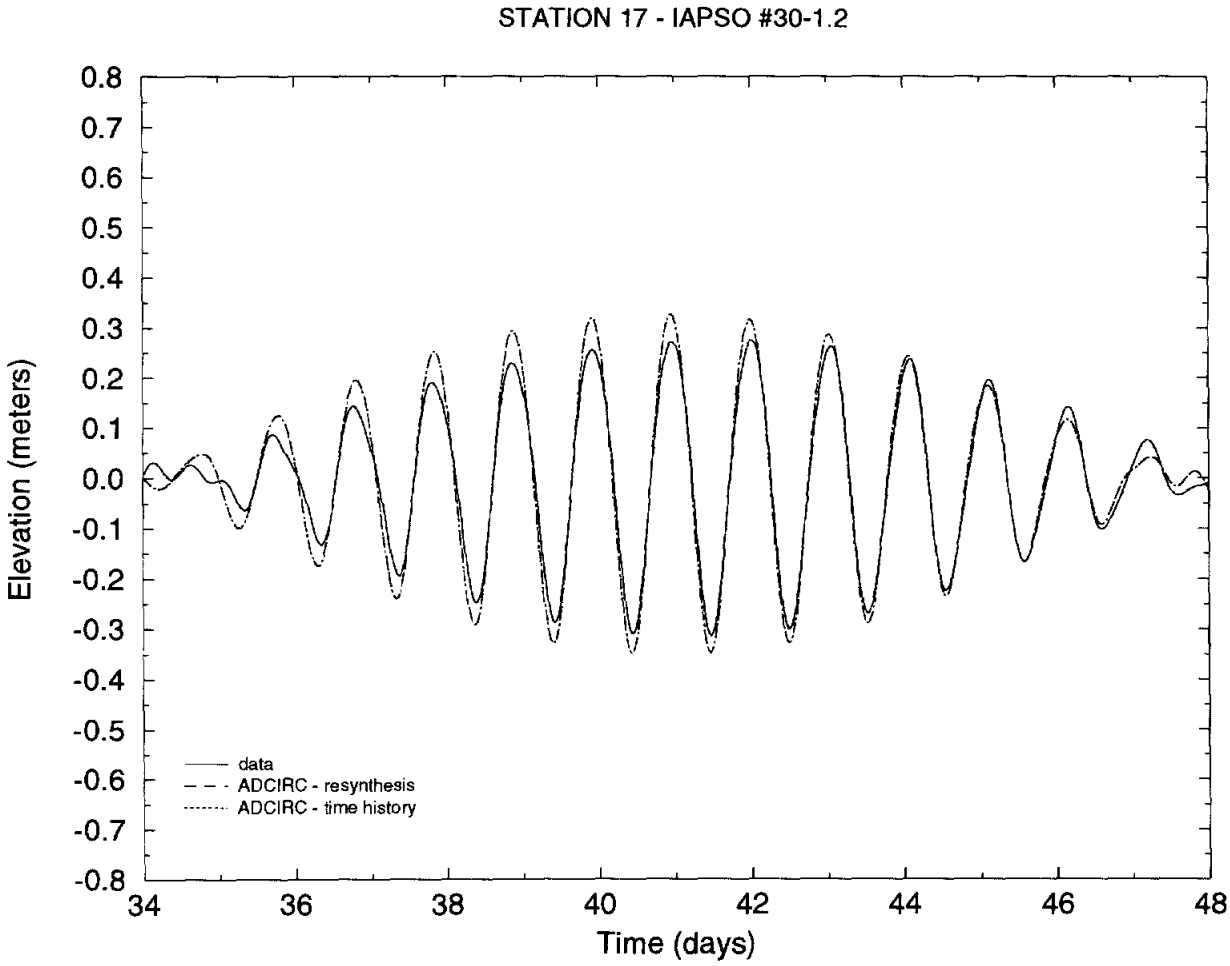


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(r) Florida Shelf - IAPSO #30-1.2.13

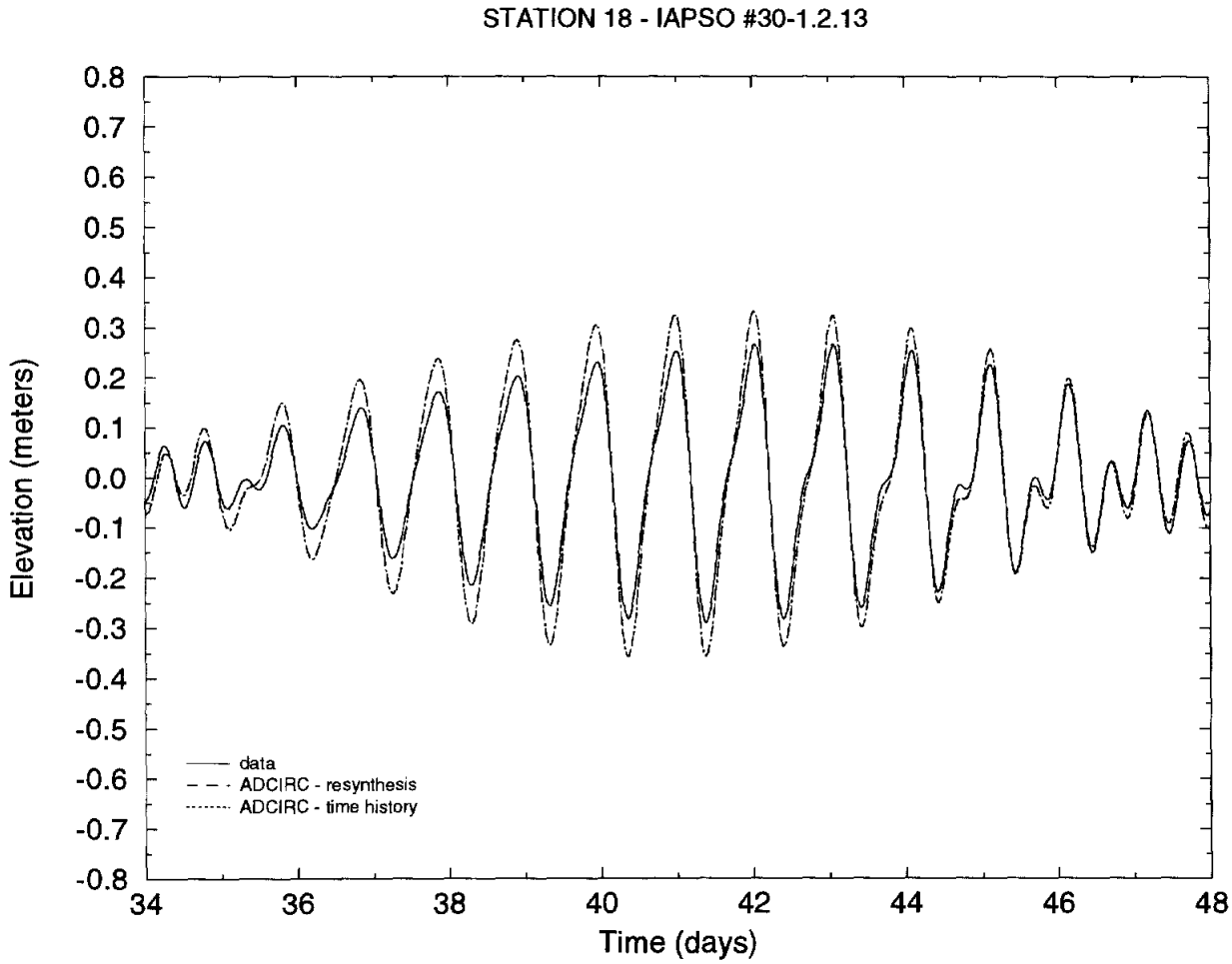
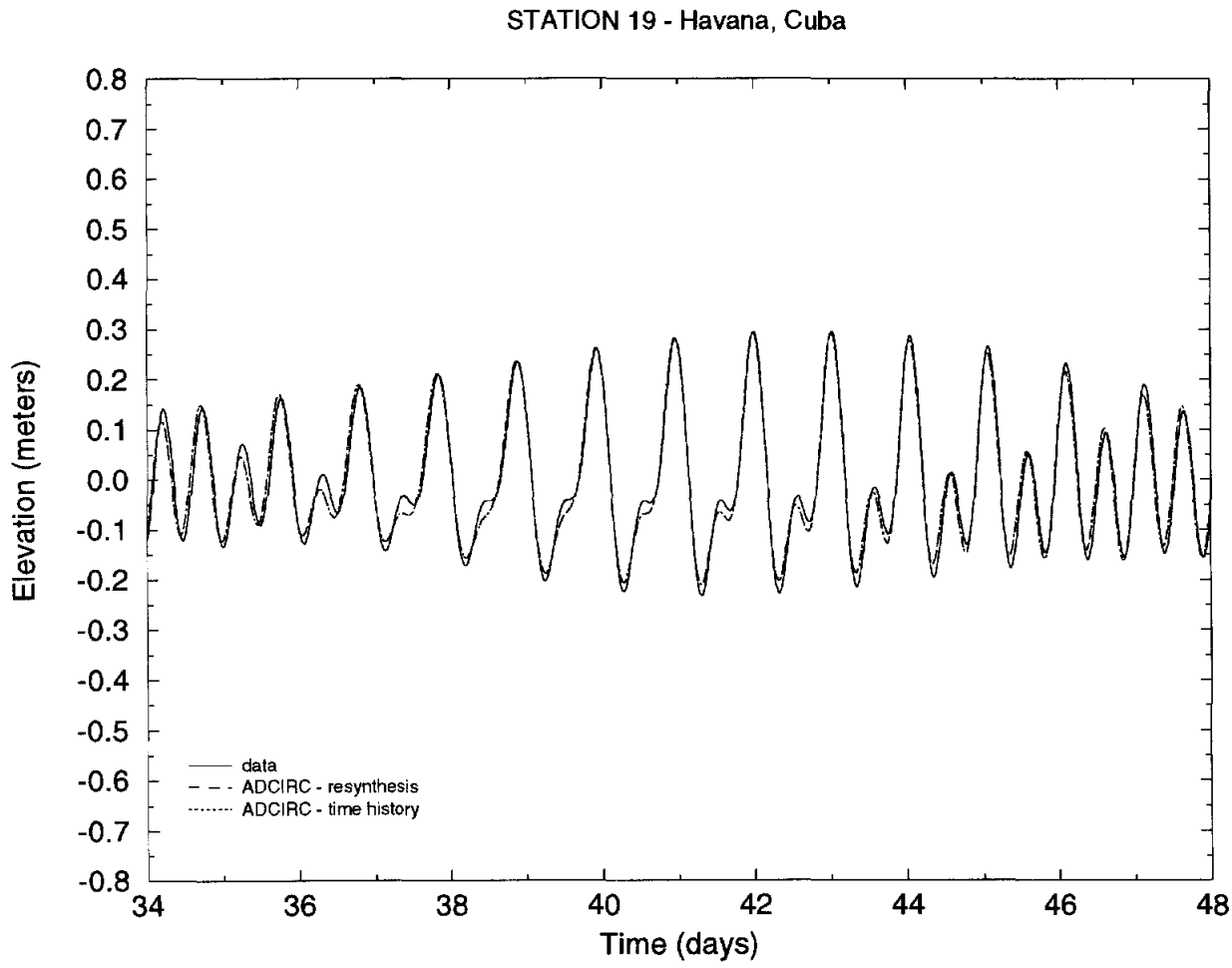


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(s) Havana, Cuba



Computer Model ADCIRC for TWDB Contract No. 95-483-097

ADCIRC
An Advanced Circulation Model for Shelves, Coastal Seas and Estuaries

J.J. Westerink
Department of Civil Engineering and Geological Sciences
University of Notre Dame
Notre Dame, IN 46556

and

R.A. Luettich, Jr
University of North Carolina at Chapel Hill
Institute of Marine Sciences
3431 Arendell St., Norehead City, NC 28557

Submitted to the Texas Water Development Board in partial fulfillment of
the requirements for Contract No. 95-483-097

December, 1997

```
C*****************************************************************************
C                                                                             *
C                                   ADCIRC                                    *
C                                                                             *
C    AN ADVANCED CIRCULATION MODEL FOR SHELVES, COASTAL SEAS AND ESTUARIES    *
C                                                                             *
C                                                                             *
C                            DEVELOPED BY:                                    *
C                                                                             *
C                         DR. R.A. LUETTICH, JR                               *
C                                                                             *
C                 UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL                 *
C                       INSTITUTE OF MARINE SCIENCES                          *
C                           3431 ARENDELL ST.                                 *
C                         MOREHEAD CITY, NC, 28557                            *
C                               919-726-6841 EXT. 137                         *
C                         EMAIL  RICK_LUETTICH@UNC.EDU                        *
C                                                                             *
C                                                                             *
C                           DR. J.J. WESTERINK                                *
C                                                                             *
C              DEPARTMENT OF CIVIL ENGINEERING AND GEOLOGICAL SCIENCES        *
C                        UNIVERSITY OF NOTRE DAME                             *
C                          NOTRE DAME, IN 46556                               *
C                            219-631-6475                                     *
C                        EMAIL JJW@PHOTIUS.CE.ND.EDU                          *
C                                                                             *
C                                                                             *
C           MAJOR FUNDING FOR THE DEVELOPMENT OF ADCIRC WAS PROVIDED BY       *
C                                                                             *
C                           DEPARTMENT OF THE ARMY                            *
C                        WATERWAYS EXPERIMENT STATION                         *
C                     COASTAL ENGINEERING RESEARCH CENTER                     *
C                           3909 HALLS FERRY RD                               *
C                         VICKSBURG, MI 39180-6199                            *
C                                                                             *
C*****************************************************************************
C                                                                             *
C                                                                             *
C                                                                             *
C              THE ADCIRC SOURCE CODE IS COPYRIGHTED, 1994-96 BY:             *
C                                                                             *
C                     R.A. LUETTICH, JR AND J.J. WESTERINK                    *
C                                                                             *
C             NO PART OF THIS CODE MAY BE REPRODUCED OR REDISTRIBUTED         *
C                  WITHOUT THE WRITTEN PERMISSION OF THE AUTHORS              *
C                                                                             *
C                                                                             *
C                                                                             *
C*****************************************************************************
C                                                                             *
C    ADCIRC IS A HIGHLY DEVELOPED COMPUTER PROGRAM FOR SOLVING THE EQUATIONS  *
C    OF MOTION FOR A BAROTROPIC FLUID, MAKING THE HYDROSTATIC PRESSURE        *
C    APPROXIMATION, AND USING THE FINITE ELEMENT METHOD FOR THE SPATIAL       *
C    DISCRETIZATION AND THE FINITE DIFFERENCE METHOD FOR THE TIME             *
C    DISCRETIZATION.                                                          *
C                                                                             *
C    ADCIRC CAN BE RUN IN EITHER A TWO-DIMENSIONAL DEPTH INTEGRATED (2DDI)    *
C    FORM OR IN 2 ALTERNATIVE THREE-DIMENSIONAL FORMS.                        *
C                                                                             *
C    THE HORIZONTAL SOLUTION USES THE GENERALIZED SHALLOW WATER EQUATIONS      *
C    (IN THEIR FULLY NONLINEAR FORM) ON EITHER A CARTESIAN OR SPHERICAL        *
C    COORDINATE SYSTEM.                                                       *
C                                                                             *
C    ELEVATION IS OBTAINED FROM THE SOLUTION OF A GENERALIZED WAVE-CONTINUITY *
C    EQUATION (GWCE).  THIS EQUATION CAN BE SOLVED USING EITHER A CONSISTENT  *
C    OR A LUMPED MASS MATRIX AND A PARTIALLY IMPLICIT OR A FULLY EXPLICIT TIME*
C    STEPPING PROCEDURE.                                                      *
C                                                                             *
C    VELOCITY IS OBTAINED FROM THE SOLUTION OF THE MOMENTUM EQUATIONS WHOSE   *
```

```
C     SPECIFIC FORM DEPENDS ON WHETHER THE MODEL IS RUN IN 2DDI OR 3D.
C                                                                       *
C     ADCIRC CAN BE FORCED BY:                                          *
C       SPECIFIED ELEVATION BOUNDARY CONDITIONS (E.G. TIDAL CIRCULATION)*
C       SPECIFIED (INCLUDING ZERO) NORMAL FLOW ACROSS BOUNDARIES        *
C       EXTERNAL AND INTERNAL BARRIER FLOW BOUNDARIES                   *
C       VARIABLE SPATIAL AND TEMPORAL SURFACE STRESS (WIND)             *
C       VARIABLE SPATIAL AND TEMPORAL ATMOSPHERIC PRESSURE FORCING      *
C       VARIABLE SPATIAL CORIOLIS                                       *
C       VARIABLE SPATIAL AND TEMPORAL TIDAL POTENTIAL FORCING           *
C                                                                       *
C     IF THE GWCE IS SET UP USING A LUMPED, FULLY EXPLICIT FORMULATION, NO *
C     MATRIX SOLVER IS NECESSARY.  HOWEVER, IN ALL OTHER CASES EITHER A DIRECT *
C     BANDED MATRIX SOLVER (PROGRAMS SGBCO AND SGBSL FROM LINPACK) OR ONE OF *
C     7 DIFFERENT ITERATIVE SOLVERS (FROM ITPACKV 2D) CAN BE USED TO SOLVE THE *
C     THE GWCE.  ONCE THE APPROPRIATE SOLVER IS OBTAINED, IT MUST BE COMPILED *
C     AND LINKED TO ADCIRC.                                             *
C     NOTE: BOTH THE DIRECT AND ITERATIVE SOLVERS ARE AVAILABLE OVER INTERNET *
C           FROM NETLIB.                                                *
C           NETLIB CAN BE REACHED AT: WWW.NELTIB.ORG                    *
C           IF YOU DON'T HAVE ACCESS TO A WEB BROWSER YOU CAN GET INFO BY *
C           SENDING EMAIL TO: NETLIB@ORNL.GOV                           *
C           FOR AN INDEX OF ALL NETLIB PROGRAMS SEND THE MESSAGE: SEND INDEX *
C           FOR AN INDEX OF LINPACK SEND THE MESSAGE: SEND INDEX FROM LINPACK *
C           FOR AN INDEX OF ITPACK SEND THE MESSAGE: SEND INDEX FROM ITPACK *
C           THE INDEXES FROM LINPACK AND ITPACK CONTAIN INSTRUCTIONS ON HOW TO *
C           DOWNLOAD THE SPECIFIC SOLVER SOURCE CODES.                  *
C     NOTE: ITPACKV 2D REQUIRES THE USER MAKE A COUPLE OF MACHINE DEPENDENT *
C           MODIFICATIONS BEFORE IT CAN BE COMPILED AND RUN.            *
C     NOTE: THE CHOICE OF SOLVER HAS LITTLE EFFECT ON THE MODEL RUN TIME. *
C           HOWEVER, THE ITERATIVE SOLVER USES MUCH LESS MEMORY THAN THE DIRECT*
C           SOLVER.  (ON SMALL PROBLEMS, E.G. 1000 NODES, THE ITERATIVE SOLVER *
C           REQUIRES ABOUT 1/2 THE MEMORY OF THE DIRECT SOLVER.  FOR LARGER *
C           PROBLEMS, E.G. >20,000 NODES, THE ITERATIVE SOLVER REQUIRES *
C            ABOUT 1/5 THE MEMORY OF THE DIRECT SOLVER.)                *
C                                                                       *
C     ADCIRC INCLUDES LEAST SQUARE ANALYSIS ROUTINES THAT WILL COMPUTE TIDAL *
C     HARMONIC CONSTITUENTS DURING THE COURSE OF THE RUN THEREBY AVOIDING THE *
C     NEED TO WRITE OUT LONG TIME SERIES FOR POST PROCESSING.           *
C                                                                       *
C     THE ADCIRC SOURCE CODE CAN BE OPTIMIZED FOR DIFFERENT COMPUTER    *
C     ARCHITECTURES (I.E., VECTOR AND SCALAR), AND FOR THE MATRIX SOLVER THAT *
C     WILL BE USED.  THIS OPTIMIZATION IS PERFORMED BY THE PROGRAM ADCSETUP.F *
C                                                                       *
C     PRIOR TO COMPILING AND RUNNING ADCIRC, THE SOURCE CODE PREPROCESSOR *
C     ADCSETUP.F MUST BE RUN.  THIS READS THE UNIT 14, 15, 17 ADCIRC INPUT *
C     FILES AND PROMPTS THE USER FOR A FEW ADDITIONAL PIECES OF INFORMATION. *
C     BASED ON THIS INPUT, THE PREPROCESSOR SETS ALL PARAMETER VALUES (USED *
C     IN DIMENSION STATEMENTS) AND OPTIMIZES THE CODE FOR THE SPECIFIC PROBLEM *
C     AND COMPUTER TO BE USED.                                          *
C                                                                       *
C***********************************************************************
C                                                                       *
C      -  THE ADCIRC SOURCE HAS BEEN CODED BY:                          *
C                                                                       *
C         2DDI VERSIONS                                                 *
C           J.K. WU           VERSIONS  1.00 -  2.36                  *
C           J.J. WESTERINK    VERSIONS  3.00 - 18.03                  *
C           R.A. LUETTICH     VERSIONS 19.00 - 19.07                  *
C           J.J. WESTERINK    VERSIONS 19.08 - 19.12                  *
C           R.A. LUETTICH     VERSIONS 20.00 - 23.15                  *
C                                                                       *
C         COMBINED 2DDI/3D VERSIONS                                     *
C           R.A. LUETTICH     VERSIONS 24.01 - 24.07                  *
C           R.R. GRENIER      VERSION  24.08                          *
C           R.A. LUETTICH     VERSIONS 24.09 - 29.10                  *
C           J.J. WESTERINK    VERSIONS 30.01 - 30.02                  *
C           R.A. LUETTICH     VERSIONS 30.03 - 31.06                  *
C                                                                       *
C***********************************************************************
```

```
C                                                                      *
C     - VERSION HISTORY                                                *
C                                                                      *
C     - 14.01/15.01 MODIFIED FROM VERSION 12.01                        *
C        WORKS IN EITHER CPP OR LAT/LONG COORDINATE SYSTEMS            *
C        THE STRATEGY IS BASED ON TRANSFORMING LAT/LONG SYSTEM INTO CPP *
C        THE LATERAL VISCOSITY TERM HAS NOT BEEN UPDATED               *
C                                                                      *
C     - 16.01/17.02 MODIFIED FROM VERSION 15.01                        *
C        ALL ASSEMBLY LOOPS HAVE BEEN UNROLLED FOR VECTORIZATION       *
C        ALSO MINOR I/O CHANGES FOR STATION AND GLOBAL OUTPUT          *
C                                                                      *
C     - 18.01/18.03 MODIFIED FROM VERSION 17.02                        *
C        OUTPUT STRUCTURE MODIFIED                                     *
C        NORMAL FLOW BOUNDARY CONDITION HANDLING MODIFIED SUCH THAT ACUTE *
C          ANGLES CAN NOW BE ZEROED IN BOTH DIRECTIONS                 *
C        PORTION OF CONVECTIVE TERMS NOW BUNDLED FOR FINITE AMPLITUDE  *
C          TERMS TO ENSURE CONSISTENCY                                 *
C                                                                      *
C     - 19.00/19.12 MODIFIED FROM VERSION 18.03                        *
C        GLOBAL ELEVATION AND VELOCITY OUTPUT SPLIT UP                 *
C        BINARY OUTPUT ADDED, SEVERAL BUGS FIXED, NONLINEAR OPTIONS REDONE *
C                                                                      *
C     - 20.00/20.10 MODIFIED FROM VERSION 19.12                        *
C        HOT START CAPABILITY ADDED                                    *
C        BUILT IN HARMONIC ANALYSIS OF MODEL RESULTS ADDED             *
C                                                                      *
C     - 21.00/21.07 MODIFIED FROM VERSION 20.10                        *
C        GWCE MATRIX MADE SYMMETRIC                                    *
C        ITERATIVE SOLVER INTRODUCED                                   *
C        DXXYY REMOVED                                                 *
C                                                                      *
C     - 22.00/22.10 MODIFIED FROM VERSION 21.07                        *
C        CALCULATION OF ELEMENTAL MATRICIES OPTIMIZED                  *
C        ELEMENTAL MATRICIES COMPUTED AS NEEDED (TO SAVE ON MEMORY)    *
C                                                                      *
C     - 23.00/23.15 MODIFIED FROM VERSION 22.10                        *
C        ABILITY TO OPTIMIZE THE VECTORIZABLE PARTS OF THE CODE FOR EITHER *
C        A SCALAR (I.E. WORKSTATION) OR VECTOR (I.E. CRAY) COMPUTER.   *
C        ABILITY TO OPTIMIZE NONVECTORIZABLE PARTS OF THE CODE FOR MAXIMUM *
C        SPEED (AT THE EXPENSE OF INCREASED MEMORY) OR FOR MINIMUM MEMORY *
C        AT THE EXPENSE OF REDUCED SPEED. NOTE THIS OPTION WAS REMOVED V27).*
C                                                                      *
C     - 24.01/24.20 MODIFIED FROM VERSION 23.15                        *
C        QUADRATURE REPLACED BY EXACT INTEGRATION OF ELEMENT MATRICIES *
C        DSS OVER VERTICAL ADDED ACTIVATING THE 3D DSS OPTION          *
C        PROBLEMS WITH EDDY VISCOSITY FIXED                            *
C        ITERATION ON EDDY VISCOSITY ELIMINATED                        *
C        VARIETY OF BUG FIXES                                          *
C        BETTER MEMORY OPTIMIZATION                                    *
C        WIND, TIDAL POTENTIAL INFO ELIMINATED FROM HOTSTART FILES     *
C                                                                      *
C     - 25.01/25.06 MODIFIED FROM VERSION 24.19                        *
C        ABILITY TO READ FLEET NUMERIC WIND FILES FOR WIND INPUT       *
C        ABILITY TO READ PBL/JAG WIND FILES FOR WIND/PRESSURE INPUT    *
C        PRECISION OF ELEMENT AREA CALC & STATION INTERPOLATION IMPROVED *
C                                                                      *
C     - 26.01/26.11 MODIFIED FROM VERSION 25.06                        *
C        BETTER MEMORY OPTIMIZATION AND RUN SPEED                      *
C        ALTERNATIVE FLOW BOUNDARY FLAGGING IMPLEMENTED                *
C        VS OVER VERTICAL ADDED ACTIVATING THE 3D VS OPTION            *
C        GWCE TIME DISCRETIZATION ALTERED TO COMPUTE ELEVATION DIFFERENCE *
C        EV TERM IN GWCE MADE FULLY EXPLICIT                           *
C        GWCE LHS MATRIX LUMPING OPTION ADDED                          *
C        GWCE DIAGONAL MATRIX SOLVER OPTION ADDED                      *
C                                                                      *
C     - 27.01/27.35 MODIFIED FROM VERSION 26.11                        *
C        ELEMENTAL BASED WETTING AND DRYING IMPLEMENTED                *
C        BUG HOTSTARTING WHEN WRITING BINARY TIME SERIES OUTPUT FILES FIXED *
C        NNODECODE(I) ADDED TO HOT START FILE                          *
```

```
C           BASIC INFORMATION ON TIME SERIES OUTPUT FILES IS INCLUDED IN THE   *
C             HOT START FILES WHETHER OR NOT THE TIME SERIES OUTPUT FILES ARE  *
C             BEING WRITTEN.                                                  *
C           PROBLEM ORDERING NEIGHBOR TABLE ON CRAY FIXED                     *
C           SPEED OPTIMIZATION REMOVED, ALL ELE MATRICIES COMPUTED ON THE FLY *
C           REVISED LATERAL STRESSES IMPLEMENTED                              *
C           DOUBLE PRECISION TIME IMPLEMENTED                                 *
C           SPHERICAL CORRECTION APPLIED TO ELE MATRICIES NOT NODAL VARIABLES  *
C           BUG IN EQUILIB ARG CORRECTION IN HARMONIC ANALYSIS ROUTINE FIXED   *
C           HARMONIC ANALYSIS SUBROUTINES REDONE                              *
C                                                                             *
C     - 28.01/28.06  MODIFIED FROM VERSION 27.35                              *
C           NORMAL FLOW BOUNDARY CONDITIONS ADDED                             *
C           OPTION TO USE ONLY THE NATURAL BOUNDARY CONDITION ON ZERO OR      *
C               SPECIFIED FLOW BOUNDARIES ADDED                               *
C           SEVERAL BUGS IN WIND FIELD HOT START FIXED                        *
C                                                                             *
C     - 29.01/29.10  MODIFIED FROM VERSION 28.06                              *
C           FULLY EXPLICIT, GALERKIN, 2D TRANSPORT ADDED                      *
C           OPTION ADDED TO START NEW OUTPUT FILES AFTER HOTSTART             *
C           ABILITY TO READ BINARY NWS FILES THAT HAVE BEEN UNPACKED WITH     *
C               UNPKGRB1 AND ARE SET UP ON GAUSS GRID FOR WIND/PRESSURE INPUT  *
C           BUG FIXED IN HURRICANE PRESSURE FIELDS.                           *
C           NORMAL FLOW B.C. SECTION CLEANED UP                               *
C           BUG FIXES FOR WIND INPUT OPTIONS.                                 *
C                                                                             *
C     - 30.01/30.03  MODIFIED FROM VERSION 29.06                              *
C           ALLOWS EXTERNAL BARRIER BOUNDARY OVERTOPPING AND SUPERCRITICAL     *
C               OUTFLOW FROM THE DOMAIN OVER THESE BOUNDARIES                 *
C           ALSO ALLOWS FOR INTERNAL BARRIER BOUNDARY OVERTOPPING AND         *
C               SUBCRITICAL AND/OR SUPERCRITICAL FLOW ACROSS THE INTERNAL     *
C               BOUNDARY                                                      *
C           BUG FIXES FOR WIND INPUT OPTIONS.                                 *
C                                                                             *
C     - 31.01/31.06 MODIFIED FROM VERSION 30.03                               *
C           ALLOWS SELF ATTRACTION/LOAD TIDE TO BE READ IN AND INCLUDED WITH  *
C               TIDAL POTENTIAL TERMS.                                        *
C           INCLUDES RADIATING BOUNDARY CONDITIONS.                           *
C           ADDITIONAL BUG FIX FOR PBL MODEL INPUT AND HARMONIC MEANS&VAR CALC *
C                                                                             *
C                                                                             *
C*****************************************************************************
C                                                                             *
C                                                                             *
C                                                                             *
C     -  STANDARD INPUT FILES ARE AS FOLLOWS:                                 *
C                                                                             *
C           UNIT 14 : FINITE ELEMENT GRID AND BOUNDARY INFORMATION FILE       *
C           UNIT 15 : INPUT FILE WHICH DEFINES THE MAJORITY OF INPUT          *
C                       PARAMETERS NECESSARY FOR RUNNING THE CODE             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  SUPPLEMENTAL INPUT FILES (ACTIVATED BY INPUT PARAMETERS              *
C          SPECIFIED IN UNIT 15 INPUT) ARE AS FOLLOWS:                        *
C                                                                             *
C           UNIT 20 : NON-PERIODIC TIME VARYING NORMAL FLOW/UNIT WIDTH VALUES  *
C                       AT "SPECIFIED NORMAL FLOW" BOUNDARY NODES.  THIS FILE  *
C                       IS ONLY USED WHEN A "SPECIFIED NORMAL FLOW" BOUNDARY   *
C                       CONDITION HAS BEEN SPECIFIED AND NFFR = 0.  (SEE       *
C                       DESCRIPTION OF UNIT 15 FILE FOR NFFR.)                 *
C           UNIT 21 : SPATIALLY VARIABLE FRICTION VALUES GIVEN AT THE NODES.   *
C                       THIS FILE IS ONLY READ IN WHEN NWP HAS BEEN SPECIFIED  *
C                       EQUAL TO 1 IN THE INPUT FILE UNIT 15                   *
C           UNIT 22 : WIND STRESS AND BAROMETRIC PRESSURE INFORMATION          *
C                       FILE: READ IN IF NWS = 1 THRU 4 IN THE UNIT 15 INPUT   *
C                       FILE.                                                  *
C           UNITS 200,206,212,218,224,230,236,242,248..... : WIND STRESS AND   *
C                       BAROMETRIC PRESSURE INFORMATION FILES AT 6 HOUR        *
C                       INTERVALS.  READ IN IF NWS = 10 IN THE UNIT 15 INPUT   *
```

```
C                    FILE.                    *
C        UNIT 67 OR 68: HOT START INPUT FILE                              *
C                                                                         *
C        NOTE: THE MORE CURRENT (IN TIME) OF EITHER fort.67 OR fort.68    *
C              SHOULD BE USED TO HOT START THE MODEL.  HOT START OUTPUT IS *
C              WRITTEN TO fort.67 AND fort.68 ON AN ALTERNATING BASIS SO   *
C              THAT IF THE COMPUTER CRASHES IN THE PROCESS OF WRITING ONE  *
C              OF THESE FILES, THE OTHER WILL BE UNAFFECTED AND CAN BE USED*
C              TO HOT START THE MODEL.                                    *
C              ON CERTAIN COMPUTERS, THE BUFFER MAY NOT BE EMPTIED ON A   *
C              CRASH. IN THIS CASE CERTAIN OUTPUT FILES MAY NOT INCLUDE   *
C              ALL THE INFORMATION THAT HAS BEEN PROCESSED AND OUTPUT     *
C              BY ADCIRC AND IT MAY BE BETTER TO USE THE EARLIEST         *
C              OF EITHER fort.67 OR fort.68.                              *
C                                                                         *
C                                                                         *
C                                                                         *
C     -  STANDARD OUTPUT FILES ARE AS FOLLOWS:                            *
C                                                                         *
C        UNIT 16 : OUTPUT FILE WHICH ECHO PRINTS INPUT PARAMETERS AND     *
C                    GRID, PROVIDES SOME PROCESSED INFORMATION AND        *
C                    PRINTS OUT ERROR MESSAGES REGARDING PROGRAM USE      *
C                                                                         *
C                                                                         *
C                                                                         *
C     -  SUPPLEMENTAL OUTPUT FILES (ACTIVATED BY INPUT PARAMETERS         *
C        SPECIFIED IN UNIT 15 INPUT) ARE AS FOLLOWS:                      *
C                                                                         *
C        UNIT 6 :  SCREEN OUTPUT : PROVIDES LIMITED RUN TIME INFORMATION  *
C                    AS WELL AS WARNINGS                                  *
C        UNIT 33 : DIAGNOSTIC OUTPUT FROM ITPACKV 2D IF AN ITERATIVE      *
C                  SOLVER IS SPECIFIED                                    *
C        UNIT 51 : HARMONIC CONSTITUENT ELEVATION VALUES AT SPECIFIED     *
C                    ELEVATION RECORDING STATION COORDINATES (ASCII)      *
C        UNIT 52 : HARMONIC CONSTITUENT VELOCITY VALUES AT SPECIFIED      *
C                    VELOCITY RECORDING STATION COORDINATES  (ASCII)      *
C        UNIT 53 : HARMONIC CONSTITUENT ELEVATIONS AT ALL NODES (ASCII)   *
C        UNIT 54 : HARMONIC CONSTITUENT VELOCITIES AT ALL NODES (ASCII)   *
C        UNIT 55 : COMPARISON BETWEEN THE MEAN AND VARIANCE OF THE TIME   *
C                  SERIES GENERATED BY THE MODEL AND THE MEAN AND VARIANCE *
C                  OF A TIME SERIES RESYNTHESIZED FROM THE COMPUTED HARMONIC*
C                  CONSTITUENTS.  THIS GIVES AN INDICATION OF HOW COMPLETE *
C                  THE HARMONIC ANALYSIS WAS.                             *
C        UNIT 61 : TIME SERIES ELEVATION VALUES AT SPECIFIED ELEVATION    *
C                    RECORDING STATION COORDINATES (ASCII OR BINARY)      *
C        UNIT 62 : TIME SERIES VELOCITY VALUES AT SPECIFIED VELOCITY      *
C                    RECORDING STATION COORDINATES (ASCII OR BINARY)      *
C        UNIT 63 : TIME SERIES ELEVATIONS AT ALL NODES (ASCII OR BINARY)  *
C        UNIT 64 : TIME SERIES VELOCITIES AT ALL NODES (ASCII OR BINARY)  *
C        UNIT 67 : HOT START OUTPUT FILE 1 = fort.67 (BINARY)             *
C        UNIT 68 : HOT START OUTPUT FILE 2 = fort.68 (BINARY)             *
C        UNIT 71 : TIME SERIES CONCENTRATION VALUES AT SPECIFIED          *
C                    CONCENTRATION RECORDING STATIONS.                    *
C        UNIT 73 : TIME SERIES CONCENTRATIONS AT ALL NODES (ASCII OR BINARY)*
C        UNIT 74 : TIME SERIES OF WIND STRESS AT ALL NODES (ASCII OR BINARY)*
C                                                                         *
C                                                                         *
C*************************************************************************
C                                                                         *
C                                                                         *
C                                                                         *
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 14              *
C                                                                         *
C        AGRID = ALPAHANUMERIC GRID IDENTIFICATION  (<=24 CHARACTERS)     *
C        NE,NP = NUMBER OF ELEMENTS AND NUMBER OF NODAL POINTS RESPECTIVELY *
C        JKI,X(JKI),Y(JKI),DP(JKI) , JKI=1,NP  = NODE NUMBER, X AND Y     *
C             COORDINATES, BATHYMETRIC VALUE ; NODES MUST BE INPUT IN AS- *
C             CENDING ORDER; IF ICS=1 IN UNIT 15 THEN X,Y REPRESENT STANDARD*
C             CARTESIAN COORDINATES SPECIFIED IN LENGTH UNITS CONSISTENT  *
C             WITH OTHER UNIT 15 INPUT (TYPICALLY METERS OR FEET);        *
```

```
C       IF ICS=2 IN UNIT 15, THEN X,Y REPRESENT DEGREES LONGITUDE      *
C       (DEGREES EAST OF GREENWICH IS POSITIVE AND DEGREES WEST OF     *
C       GREENWICH IS NEGATIVE) AND DEGREES LATITUDE (DEGREES           *
C       NORTH OF THE EQUATOR BEING POSITIVE AND DEGREES SOUTH OF THE   *
C       EQUATOR IS NEGATIVE) RESPECTIVELY.                             *
C       BATHYMETRIC VALUES ARE W.R.T. THE GEOID AND ARE POSITIVE       *
C       BELOW THE GEOID AND NEGATIVE ABOVE THE GEOID.                  *
C       BATHYMETRIC VALUES ABOVE THE GEOID OR ANY DEPTH SUFFICIENTLY   *
C       SMALL THAT NODES WILL DRY, REQUIRES THAT THE USER ENABLE THE   *
C       WETTING/DRYING FEATURE (NOLIFA=2) IN THE UNIT 15 INPUT FILE    *
C     JKI,NHY,NM(JKI,1),NM(JKI,2),NM(JKI,3) , JKI=1,NE  = ELEMENT      *
C       NUMBER, ELEMENT TYPE, AND ELEMENT CONNECTIVITY SPECIFIED       *
C       WITH A COUNTERCLOCKWISE ORIENTATION ; NOTE THAT THE ELEMENT    *
C       TYPE IS NOT AN ACTIVE VARIABLE AND THAT ONLY 3 NODE            *
C       LINEAR TRIANGLES ARE OPERATIONAL IN THIS VERSION OF THE CODE;  *
C       ELEMENTS MUST BE READ IN IN ASCENDING ORDER.                   *
C     NOPE = NUMBER OF ELEVATION BOUNDARY FORCING SEGMENTS             *
C     NETA = TOTAL NUMBER OF ELEVATION BOUNDARY NODES                  *
C     NVDLL(K),IBTYPE                                                  *
C     NBDV(K,I) K=1,NOPE , I=1,NVDLL(K) ; NOTE THAT NVDLL(K) AND IBTYPE*
C       ARE GIVEN FIRST FOR A SEGMENT AND ARE IMMEDIATELY FOLLOWED BY  *
C        THE NODES ON THAT SEGMENT.                                    *
C       NVDLL(K) = NUMBER OF NODES ON ELEVATION BOUNDARY SEGMENT K     *
C       IBTYPE =  BOUNDARY TYPE:                                       *
C              =  0 - ELEVATION SPECIFIED BOUNDARY                     *
C              =  1 - ELEVATION SPECIFIED & RADIATION BOUNDARY         *
C                     CONDITION (NOT IMPLEMENTED)                      *
C       NBDV(K,I) = NODE NUMBERS ON ELEVATION BOUNDARY SEGMENT K. THE  *
C                   DIRECTION IN WHICH INFORMATION IS GIVEN DOES NOT    *
C                   MATTER.                                            *
C     NBOU = NUMBER OF NORMAL FLOW (OR DISCHARGE) BOUNDARY SEGMENTS    *
C     NVEL = TOTAL NUMBER OF NORMAL FLOW BOUNDARY NODES.               *
C       NOTE THAT NVEL INCLUDES BOTH FRONT AND BACK NODES ON           *
C       INTERNAL BARRIER TYPE NORMAL FLOW BOUNDARIES.                  *
C     NVELL(K),IBTYPE  FOR EACH OF THE K=1,NBOU SEGMENTS; THESE VALUES *
C      ARE IMMEDIATLY FOLLOWED BY THE BOUNDARY NODE INFORMATION        *
C      FOR EACH SPECIFIC SEGMENT K                                     *
C       NVELL(K) = NUMBER OF NORMAL FLOW BOUNDARY NODES ON NORMAL      *
C        FLOW SEGMENT K                                                *
C       IBTYPE =  BOUNDARY TYPE:                                       *
C              =  0 - EXTERNAL BOUNDARY WITH NO NORMAL FLOW AS AN      *
C                     ESSENTIAL BOUNDARY CONDITION AND FREE TANGENTIAL*
C                     SLIP ALLOWED.                                    *
C                     THIS REPRESENTS A MAINLAND BOUNDARY WITH NO      *
C                     NORMAL FLOW.                                     *
C              =  1 - INTERNAL BOUNDARY WITH NO NORMAL FLOW AS AN      *
C                     ESSENTIAL BOUNDARY CONDITION AND FREE TANGENTIAL*
C                     SLIP ALLOWED.                                    *
C                     THIS REPRESENTS AN ISLAND BOUNDARY WITH NO       *
C                     NORMAL FLOW.                                     *
C              =  2 - EXTERNAL BOUNDARY WITH SPECIFIED (NON-ZERO)      *
C                     NORMAL FLOW AS AN ESSENTIAL BOUNDARY CONDITION   *
C                     AND FREE TANGENTIAL SLIP ALLOWED.                *
C                     THIS CAN REPRESENT A RIVER INFLOW, PLANT         *
C                     DISCHARGE OR OPEN OCEAN BOUNDARY IN WHICH NORMAL*
C                     FLOW IS SPECIFIED.                               *
C              =  3 - EXTERNAL BOUNDARY WITH A BARRIER WHICH ALLOWS    *
C                     FREE SURFACE SUPERCRITICAL OUTFLOW FROM THE      *
C                     DOMAIN ONCE IT HAS BEEN OVERTOPPED TREATED AS    *
C                     AN ESSENTIAL BOUNDARY CONDITION AND FREE         *
C                     TANGENTIAL SLIP ALLOWED.                         *
C                     THIS REPRESENTS MAINLAND (WEIR TYPE) BARRIERS    *
C                     WITH NORMAL OUTFLOW COMPUTED BASED ON ELEVATION *
C                     AT BOUNDARY NODE.                                *
C              =  4 - INTERNAL (ISLAND TYPE) BARRIER BOUNDARY WHICH    *
C                     ALLOWS FREE SURFACE SUBCRITICAL AND SUPER-        *
C                     CRITICAL NORMAL CROSS BARRIER INFLOW AND OUT-     *
C                     FLOW ONCE IT HAS BEEN OVERTOPPED.                *
C                     THE COMPUTED CROSS BARRIER FLOW IS TREATED AS    *
C                     AN ESSENTIAL BOUNDARY CONDITION.                 *
```

```
C                                TANGENTIAL SLIP IS ALLOWED.                    *
C                                THIS BOUNDARY CONDITION APPLIES TO (WEIR TYPE) *
C                                BARRIERS WHICH LIE WITHIN THE COMPUTATIONAL    *
C                                DOMAIN. NORMAL FLOW ACROSS THE BARRIER IS BASED*
C                                ON ELEVATIONS ON BOTH SIDES OF THE BARRIER.    *
C                           = 10 - EXTERNAL BOUNDARY WITH NO NORMAL FLOW AND NO *
C                                TANGENTIAL SLIP AS ESSENTIAL BOUNDARY CONDITIONS*
C                                THIS REPRESENTS A MAINLAND BOUNDARY WITH NO    *
C                                NORMAL FLOW AND NO SLIP.                       *
C                           = 11 - INTERNAL BOUNDARY WITH NO NORMAL FLOW AND NO *
C                                TANGENTIAL SLIP AS ESSENTIAL BOUNDARY CONDITIONS*
C                                THIS REPRESENTS AN ISLAND BOUNDARY WITH NO     *
C                                NORMAL FLOW AND NO SLIP                        *
C                           = 12 - EXTERNAL BOUNDARY WITH SPECIFIED (NON-ZERO)  *
C                                NORMAL FLOW AND NO TANGENTIAL SLIP ALLOWED.    *
C                                BOTH DIRECTIONS ARE TREATED AS ESSENTIAL       *
C                                BOUNDARY CONDITIONS.                           *
C                                THIS CAN REPRESENT A RIVER INFLOW, PLANT       *
C                                DISCHARGE OR OPEN OCEAN BOUNDARY IN WHICH NORMAL*
C                                FLOW IS SPECIFIED IN ADDITION TO NO SLIP.      *
C                           = 13 - EXTERNAL BOUNDARY WITH A BARRIER WHICH ALLOWS *
C                                FREE SURFACE SUPERCRITICAL OUTFLOW FROM THE    *
C                                DOMAIN ONCE IT HAS BEEN OVERTOPPED AND NO      *
C                                TANGENTIAL SLIP IS ALLOWED. BOTH DIRECTIONS ARE *
C                                TREATED AS ESSENTIAL BOUNDARY CONDITIONS.      *
C                                THIS REPRESENTS MAINLAND (WEIR TYPE) BARRIERS  *
C                                WITH NORMAL OUTFLOW COMPUTED BASED ON ELEVATION *
C                                AT BOUNDARY NODE AND NO SLIP.                  *
C                           = 20 - EXTERNAL BOUNDARY WITH NO NORMAL FLOW AS A   *
C                                NATURAL BOUNDARY CONDITION AND FREE TANGENTIAL *
C                                SLIP ALLOWED.                                  *
C                                THIS REPRESENTS A MAINLAND BOUNDARY WITH (WEAK) *
C                                NO NORMAL FLOW.                                *
C                           = 21 - INTERNAL BOUNDARY WITH NO NORMAL FLOW AS A   *
C                                NATURAL BOUNDARY CONDITION AND FREE TANGENTIAL *
C                                SLIP ALLOWED.                                  *
C                                THIS REPRESENTS AN ISLAND BOUNDARY WITH (WEAK) *
C                                NO NORMAL FLOW.                                *
C                           = 22 - EXTERNAL BOUNDARY WITH SPECIFIED (NON-ZERO)  *
C                                NORMAL FLOW AS A NATURAL BOUNDARY CONDITION    *
C                                AND FREE TANGENTIAL SLIP ALLOWED.              *
C                                THIS CAN REPRESENT A RIVER INFLOW, PLANT       *
C                                DISCHARGE OR OPEN OCEAN BOUNDARY IN WHICH (WEAK)*
C                                NORMAL FLOW IS SPECIFIED.                      *
C                           = 23 - EXTERNAL BOUNDARY WITH A BARRIER WHICH ALLOWS *
C                                FREE SURFACE SUPERCRITICAL OUTFLOW FROM THE    *
C                                DOMAIN ONCE IT HAS BEEN OVERTOPPED AS A NATURAL *
C                                BOUNDARY CONDITION AND FREE TANGENTIAL SLIP    *
C                                ALLOWED.                                       *
C                                THIS REPRESENTS MAINLAND (WEIR TYPE) BARRIERS  *
C                                WITH (WEAK) NORMAL OUTFLOW COMPUTED BASED ON   *
C                                ELEVATION AT BOUNDARY NODE.                    *
C                           = 24 - INTERNAL (ISLAND TYPE) BARRIER BOUNDARY WHICH *
C                                ALLOWS FREE SURFACE SUBCRITICAL AND SUPER-      *
C                                CRITICAL NORMAL CROSS BARRIER INFLOW AND OUT-   *
C                                FLOW ONCE IT HAS BEEN OVERTOPPED.              *
C                                THE COMPUTED CROSS BARRIER FLOW IS TREATED AS   *
C                                A NATURAL BOUNDARY CONDITION.                  *
C                                TANGENTIAL SLIP IS ALLOWED.                    *
C                                THIS BOUNDARY CONDITION APPLIES TO (WEIR TYPE) *
C                                BARRIERS WHICH LIE WITHIN THE COMPUTATIONAL    *
C                                DOMAIN. (WEAK) NORMAL FLOW ACROSS THE BARRIER  *
C                                IS BASED ON ELEVATIONS ON BOTH SIDES OF THE    *
C                                BARRIER.                                       *
C                           = 30 - IN GWCE, EXTERNAL BOUNDARY ALLOWS THE        *
C                                RADIATION OF WAVES NORMAL TO THE BOUNDARY VIA  *
C                                THE FLUX INTEGRAL.  NO CONSTRAINT IS PLACED ON *
C                                THE VELOCITY SOLUTION IN THE MOMENTUM EQN.     *
C <<<<<<<< IF IBTYPE = 0,1,2,10,11,12,20,21,22,30-> PROVIDE THE FOLLOWING      *
C                    INFORMATION  >>>>>>>>                                      *
```

```
C                 ...(K,I),I=1,NVELL(K) ; NOTE THAT NVELL(K) AND IBTYPE ARE GIVEN   *
C                 FIRST FOR A SEGMENT AND ARE IMMEDIATELY FOLLOWED BY THE NODE  *
C                 NUMBERS ON SEGMENT K.                                         *
C                 NBVV(K,I) = NODE NUMBERS ON NORMAL FLOW BOUNDARY SEGMENT K.   *
C                             NODES MUST BE SPECIFIED WITH LAND ALWAYS BEING ON *
C                               THE RIGHT, I.E., IN A COUNTERCLOCKWISE DIRECTION*
C                               FOR EXTERNAL (E.G. MAINLAND) BOUNDARIES AND A   *
C                               CLOCKWISE DIRECTION FOR INTERNAL (E.G. ISLAND)  *
C                               BOUNDARIES.                                     *
C                 NOTE: ALSO SEE <<<<<<<< GENERAL NOTES FOR NORMAL FLOW         *
C                         BOUNDARY CONDITION SPECIFICATION >>>>>>>>             *
C <<<<<<<< IF IBTYPE = 3,13,23 -> PROVIDE THE FOLLOWING INFORMATION FOR         *
C                   EXTERNAL BARRIER TYPE BOUNDARIES  >>>>>>>>                  *
C              NBVV(K,I),BARLANHT(I),BARLANCFSP(I) I=1,NVELL(K) ;               *
C                 NOTE THAT NVELL(K) AND IBTYPE ARE GIVEN FIRST FOR A SEGMENT   *
C                 AND ARE IMMEDIATELY FOLLOWED BY THE NODE NUMBERS AND NODAL    *
C                 BARRIER CHARACTERISTICS ON SEGMENT K.                         *
C                 NBVV(K,I) = NODE NUMBERS ON NORMAL FLOW BOUNDARY SEGMENT K.   *
C                             NODES MUST BE SPECIFIED WITH LAND ALWAYS BEING ON *
C                              THE RIGHT, I.E., IN A COUNTERCLOCKWISE DIRECTION *
C                              FOR EXTERNAL BARRIER BOUNDARIES                  *
C                 BARLANHT(I) = EXTERNAL BARRIER BOUNDARY NODE ELEVATION ABOVE  *
C                                 THE GEOID (POSITIVE ABOVE THE GEOID AND       *
C                                 NEGATIVE BELOW THE GEOID). BARLANHT(I) MUST   *
C                                 EXCEED THE INPUT BATYMETRIC VALUE SPECIFIED   *
C                                 AT THE ASSOCIATED GLOBAL NODE.                *
C                                 ACCOUNTING FOR SIGN CONVENTIONS FOR BATHYMETRY *
C                                 AND EXTERNAL BARRIER HEIGHT, WE MUST SATISFY  *
C                                 BARLANHT(I).GE.-DP(NBVV(K,I)                  *
C                                 IF WE DO NOT SATISFY THIS CONDITION, THE CODE *
C                                 WILL STOP.                                    *
C                 BARLANCFSP(I) = THE COEFFICIENT OF FREE SURFACE SUPERCRITICAL *
C                                   FLOW FOR SPECIFIED EXTERNAL BARRIER BOUNDARY *
C                                   NODE (TYPICAL VALUE = 1.0)                  *
C                 NOTES: OUTWARD NORMAL FLOW PER UNIT WIDTH AT AN EXTERNAL      *
C                         BARRIER BOUNDARY NODE I IS COMPUTED AS:               *
C                        IF(ETA2(NNBB).GT.BARLANHT(I))                          *
C                          QN2(I) = -(2./3.)*BARLANCFSP(I)                      *
C                                   *(ETA2(NNBB)-BARLANHT(I))*                  *
C                                   *((2./3.)*(ETA2(NNBB)-BARLANHT(I))*G)**0.5  *
C                        IF(ETA2(NNBB).LE.BARLANHT(I))                          *
C                          QN2(I) = 0                                           *
C                        WHERE                                                  *
C                         NNBB = NBVV(K,I) = GLOBAL NODE NUMBER FOR EXTERNAL    *
C                          BARRIER BOUNDARY NODE                                *
C                         ETA2(NNBB) IS THE KNOWN TIME LEVEL SURFACE ELEVATION  *
C                          SOLUTION AT THE EXTERNAL BARRIER BOUNDARY NODE AS    *
C                          COMPUTED IN THE CODE.                                *
C                        THIS FORMULA IS GIVEN BY LEENDERTSE (ASPECTS OF         *
C                         SIMSYS2D - A SYSTEM FOR TWO-DIMENSIONAL FLOW          *
C                         COMPUTATION, RAND/R-3572-USGS, 1987  AND IS SIMPLY    *
C                         THE FORMULA FOR A BROAD CRESTED WEIR (E.G. SEE        *
C                         HENDERSON, OPEN CHANNEL FLOW, SECTION 6.6)            *
C                        ALSO SEE <<<<<<<< GENERAL NOTES FOR NORMAL FLOW        *
C                         BOUNDARY CONDITION SPECIFICATION >>>>>>>>             *
C <<<<<<<< IF IBTYPE = 4,24 -> PROVIDE THE FOLLOWING INFORMATION FOR            *
C                   INTERNAL BARRIER TYPE BOUNDARIES  >>>>>>>>                  *
C              NBVV(K,I),IBCONN(I),BARINHT(I),BARINCFSB(I),BARINCFSP(I) FOR     *
C                 THE I=1,NVELL(K) NODES IN NORMAL FLOW BOUNDARY SEGMENT K      *
C                 NOTE THAT NVELL(K) AND IBTYPE ARE GIVEN FIRST FOR A SEGMENT   *
C                 AND ARE IMMEDIATELY FOLLOWED BY THE NODE NUMBERS AND INTERNAL *
C                 BARRIER CHARACTERISTICS ON SEGMENT K.                         *
C                 NOTE THAT NVELL(K) IS SET EQUAL TO THE NUMBER OF NODES ON THE *
C                 FRONT FACE OF THE BARRIER AND DOES NOT INCLUDE THE NODES ON   *
C                 THE BACK FACE WHICH ARE SPECIFIED IN IBCONN(I).               *
C                 NBVV(K,I) = NODE NUMBERS ON NORMAL FLOW BOUNDARY SEGMENT K    *
C                               WHICH IN THIS CASE IS AN INTERNAL BARRIER       *
C                               BOUNDARY SEGMENT. THESE NODES ONLY INCLUDE THE  *
C                               FRONT FACE NODES OF THE INTERNAL BARRIER        *
C                              AN INTERNAL BARRIER BOUNDARY SEGMENT CONSISTS OF  *
```

```
C                     AN ISLAND WITH PARALLEL FRONT AND BACK FACES    *
C                     THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN    *
C                     THE NODES PLACED ON THE FRONT AND BACK FACES    *
C                   NODES ARE ONLY SPECIFIED FOR ONE OF THE TWO FACES *
C                     WITH THE PAIRED NODES BEING SPECIFIED IN        *
C                     IBCONN(I)                                       *
C                   NODES MUST BE SPECIFIED WITH THE INSIDE OF THE    *
C                     BARRIER ALWAYS BEING ON THE LEFT                *
C                     I.E., IN A CLOCKWISE DIRECTION FOR              *
C                     THIS INTERNAL TYPE BOUNDARY                     *
C              IBCONN(I) = PAIRED NODES FOR INTERNAL BARRIER BOUNDARY  *
C              BARINHT(I) = INTERNAL BARRIER BOUNDARY NODE ELEVATION ABOVE *
C                     THE GEOID (POSITIVE ABOVE THE GEOID AND         *
C                     NEGATIVE BELOW THE GEOID). BARINHT(I) MUST      *
C                     EXCEED THE INPUT BATYMETRIC VALUES SPECIFIED    *
C                     AT THE ASSOCIATED GLOBAL NODE AS WELL AS AT     *
C                     THE PAIRED NODE.                                *
C                     ACCOUNTING FOR SIGN CONVENTIONS FOR BATHYMETRY  *
C                     AND INTERNAL BARRIER HEIGHT, WE MUST SATISFY    *
C                     BARINHT(I).GE.-DP(NBVV(K,I)                     *
C                     BARINHT(I).GE.-DP(IBCONN(I))                    *
C                     IF WE DO NOT SATISFY THESE CONDITIONS, THE      *
C                     CODE WILL STOP.                                 *
C              BARINCFSB(I) = THE COEFFICIENT OF FREE SURFACE SUBCRITICAL *
C                        FLOW FOR SPECIFIED INTERNAL BARRIER BOUNDARY *
C                        NODE (TYPICAL VALUE = 1.0)                   *
C              BARINCFSP(I) = THE COEFFICIENT OF FREE SURFACE SUPERCRITICAL *
C                        FLOW FOR SPECIFIED INTERNAL BARRIER BOUNDARY *
C                        NODE (TYPICAL VALUE = 1.0)                   *
C              NOTES: CROSS BARRIER NORMAL FLOW PER UNIT WIDTH AT INTERNAL *
C                     BARRIER BOUNDARY NODE I IS COMPUTED AS:         *
C                    CASE 1 - WATER LEVEL BELOW BARRIER               *
C                      OCCURS IF((RBARWL1.LT.0.0).AND.(RBARWL2.LT.0.0)) *
C                      RESULTS IN NO FLOW ACROSS THE BARRIER          *
C                      QN2(I)=0.0                                     *
C                    CASE 2 - WATER LEVEL EQUAL ON BOTH SIDES OF BARRIER *
C                      OCCURS IF(RBARWL1.EQ.RBARWL2)                  *
C                      RESULTS IN NO FLOW ACROSS THE BARRIER          *
C                      QN2(I)=0.0                                     *
C                    CASE 3 - WATER LEVEL GREATER ON FRONT SIDE OF THE *
C                      BARRIER BUT ELEVATION DIFFERENCE SUCH THAT CROSS *
C                      BARRIER FLOW IS SUBCRITICAL. OCCURS            *
C                      IF((RBARWL1.GT.RBARWL2).AND.(RBARWL1.GT.0.0).AND. *
C                                              (RBARWL2.GT.RBARWL1F)) *
C                      RESULTS IN SUBCRITICAL NORMAL FLOW ACROSS THE  *
C                      BARRIER FROM FRONT TO BACK                     *
C                      QN2(I)=-RAMP*BARINCFSB(I)*RBARWL2*             *
C                         (2*G*(RBARWL1-RBARWL2))**0.5                *
C                    CASE 4 - WATER LEVEL GREATER ON FRONT SIDE OF THE *
C                      BARRIER BUT ELEVATION DIFFERENCE SUCH THAT CROSS *
C                      BARRIER FLOW IS SUPERCRITICAL. OCCURS          *
C                      IF((RBARWL1.GT.RBARWL2).AND.(RBARWL1.GT.0.0).AND. *
C                                              (RBARWL2.LE.RBARWL1F)) *
C                      RESULTS IN SUPERCRITICAL NORMAL FLOW ACROSS THE *
C                      BARRIER FROM FRONT TO BACK                     *
C                      QN2(I)=-RAMP*BARINCFSP(I)*RBARWL1F*(RBARWL1F*G)**0.5 *
C                    CASE 5 - WATER LEVEL LOWER ON FRONT SIDE OF THE  *
C                      BARRIER BUT ELEVATION DIFFERENCE SUCH THAT CROSS *
C                      BARRIER FLOW IS SUBCRITICAL. OCCURS            *
C                      IF((RBARWL2.GT.RBARWL1).AND.(RBARWL2.GT.0.0)   *
C                                        .AND.(RBARWL1.GT.RBARWL2F))  *
C                      RESULTS IN SUBCRITICAL NORMAL FLOW ACROSS THE  *
C                      BARRIER FROM BACK TO FRONT                     *
C                      QN2(I)=RAMP*BARINCFSB(I)*RBARWL1*              *
C                        (2*G*(RBARWL2-RBARWL1))**0.5                 *
C                    CASE 6 - WATER LEVEL LOWER ON FRONT SIDE OF THE  *
C                      BARRIER BUT ELEVATION DIFFERENCE SUCH THAT CROSS *
C                      BARRIER FLOW IS SUPERCRITICAL. OCCURS          *
C                      IF((RBARWL2.GT.RBARWL1).AND.(RBARWL2.GT.0.0)   *
C                                        .AND.(RBARWL1.LE.RBARWL2F))  *
```

```
C                       RESULTS IN SUPERCRITICAL NORMAL FLOW ACROSS THE
C                       BARRIER FROM BACK TO FRONT                      *
C                       QN2(I)=RAMP*BARINCFSP(I)*RBARWL2F*(RBARWL2F*G)**0.5  *
C                     WHERE                                             *
C                       ETA2(J) = THE KNOWN TIME LEVEL SURFACE ELEVATION  *
C                                 SOLUTION AT GLOBAL NODE J COMPUTED IN THE *
C                                 CODE                                  *
C                       NNBB1 = NBVV(K,I)                               *
C                             = GLOBAL NODE NUMBER ON FRONT SIDE OF BARRIER *
C                       NNBB2 = IBCONN(I)                               *
C                             = GLOBAL NODE NUMBER ON THE BACK SIDE OF THE *
C                               BARRIER                                 *
C                       RBARWL1 = ETA2(NNBB1)-BARINHT(I)                *
C                               = RELATIVE WATER HEIGHT ON FRONT SIDE OF  *
C                                 THE BARRIER                           *
C                       RBARWL2 = ETA2(NNBB2)-BARINHT(I)                *
C                               = RELATIVE WATER HEIGHT ON BACK SIDE OF   *
C                                 THE BARRIER                           *
C                       RBARWL1F=2.0*RBARWL1/3.0                        *
C                       RBARWL2F=2.0*RBARWL2/3.0                        *
C                     NOTE THAT THE FLOW ON THE BACK SIDE OF THE BARRIER *
C                      IS SET EQUAL AND OPPOSITE TO THE FLOW ON THE FRONT *
C                      SIDE OF THE BARRIER                              *
C                     THESE FORMULAE ARE GIVEN BY LEENDERTSE (ASPECTS OF *
C                      SIMSYS2D - A SYSTEM FOR TWO-DIMENSIONAL FLOW     *
C                      COMPUTATION, RAND/R-3572-USGS, 1987  AND ARE SIMPLY *
C                      THE FORMULAE FOR A BROAD CRESTED WEIR (E.G. SEE  *
C                      HENDERSON, OPEN CHANNEL FLOW)                    *
C                     ALSO SEE <<<<<<<< GENERAL NOTES FOR NORMAL FLOW   *
C                      BOUNDARY CONDITION SPECIFICATION >>>>>>>>        *
C <<<<<<<< GENERAL NOTES FOR NORMAL FLOW BOUNDARY CONDITION             *
C                                        SPECIFICATION >>>>>>>>         *
C                     ALL EXTERNAL (EXTERNAL NO NORMAL FLOW, EXTERNAL WITH *
C                      SPECIFIED NORMAL FLOW AND EXTERNAL BARRIER) BOUNDARY *
C                      SEGMENTS SHOULD BE LISTED IN CONSECUTIVE ORDER AROUND *
C                      THE OUTSIDE OF THE ENTIRE DOMAIN BEFORE ANY INTERNAL *
C                      (ISLAND WITH NO NORMAL FLOW OR INTERNAL BARRIER)  *
C                      BOUNDARY SEGMENTS ARE LISTED.                    *
C                     ALL NO NORMAL FLOW INTERNAL BOUNDARIES SHOULD BE CLOSED*
C                      BY REPEATING THE FIRST NODE AS THE LAST NODE.    *
C                     AN EXTERNAL BOUNDARY THAT COMPLETELY SURROUNDS THE *
C                      DOMAIN (E.G., A LAKE) SHOULD BE CLOSED BY REPEATING *
C                      THE FIRST NODE AS THE LAST NODE.                 *
C                     UNLESS THE BOUNDARY SEGMENT IS CLOSED, ALWAYS START *
C                      A BOUNDARY SEGMENT WHERE TWO DIFFERENT TYPES OF  *
C                      BOUNDARIES MEET.                                 *
C                     AT A NODE WHERE A EXTERNAL SPECIFIED NORMAL FLOW OR *
C                      EXTERNAL BARRIER BOUNDARY MEETS A EXTERNAL NO NORMAL *
C                      FLOW BOUNDARY, THE INITIAL LEG OF THE EXTERNAL   *
C                      SPECIFIED NORMAL FLOW BOUNDARY OR EXTERNAL BARRIER *
C                      BOUNDARY IS USED TO DETERMINE THE NORMAL AND     *
C                      TANGENTIAL DIRECTION.                            *
C                     EXTERNAL BOUNDARIES WITH SPECIFIED (NON-ZERO) NORMAL *
C                      FLOW BOUNDARY CONDITIONS AND EXTERNAL BARRIER    *
C                      BOUNDARIES MUST BE SEPERATED BY A EXTERNAL NO NORMAL *
C                      FLOW BOUNDARY SEGMENT OR AN ELEVATION SPECIFIED  *
C                      BOUNDARY SEGMENT (I.E. EXTERNAL BOUNDARIES WITH  *
C                      SPECIFIED (NON-ZERO) NORMAL FLOW BOUNDARY CONDITIONS *
C                      AND EXTERNAL BARRIER BOUNDARIES SEGMENTS CAN NOT *
C                      OVERLAP).                                        *
C                     INTERNAL BARRIER BOUNDARIES CAN OVERLAP EXTERNAL NO *
C                      NORMAL FLOW BOUNDARY SEGMENTS. IN THIS CASE THE  *
C                      EXTERNAL NO NORMAL FLOW NODES MAY BE TREATED IN THE *
C                      WEAK SENSE AND SET EQUAL TO ZERO IN THE GWCE FLOW *
C                      INTEGRAL ONLY, WHILE THE INTERNAL BARRIER OVERLAPPING *
C                      NODES ARE HANDLED AS INDICATED IN THE INPUT FILE. *
C                      SPECIFICALLY THE FOLLOWING CASES ARE CHECKED FOR: *
C                      - THE EXTERNAL NO FLOW BOUNDARY NODE IS SPECIFIED AS *
C                        ESSENTIAL WITH SLIP (IBTYPE=0) AND THE INTERNAL *
C                        BARRIER BOUNDARY NODE IS SPECIFIED AS ESSENTIAL *
```

```
C                             WITH SLIP (IBTYPE=4)                                *
C                             -> CODE RESETS THE EXTERNAL NO FLOW BOUNDARY NODE   *
C                             TO BE A NATURAL NO FLOW BOUNDARY NODE (IBTYPE=20)   *
C                           - THE EXTERNAL NO FLOW BOUNDARY NODE IS SPECIFIED AS  *
C                             ESSENTIAL WITH NO SLIP (IBTYPE=10) AND THE INTERNAL *
C                             BARRIER BOUNDARY NODE IS SPECIFIED AS ESSENTIAL     *
C                             WITH SLIP (IBTYPE=4)                                *
C                             -> CODE RESETS THE EXTERNAL NO FLOW BOUNDARY NODE   *
C                             TO BE A NATURAL NO FLOW BOUNDARY NODE (IBTYPE=20)   *
C                           - THE EXTERNAL NO FLOW BOUNDARY NODE IS SPECIFIED AS  *
C                             NATURAL WITH SLIP (IBTYPE=20) AND THE INTERNAL      *
C                             BARRIER BOUNDARY NODE IS SPECIFIED AS ESSENTIAL     *
C                             WITH SLIP (IBTYPE=4)                                *
C                             -> CODE TAKES NO ACTION TO CHANGE USER INPUT        *
C                           - THE EXTERNAL NO FLOW BOUNDARY NODE IS SPECIFIED AS  *
C                             ESSENTIAL WITH SLIP (IBTYPE=0) AND THE INTERNAL     *
C                             BARRIER BOUNDARY NODE IS SPECIFIED AS NATURAL       *
C                             WITH SLIP (IBTYPE=24)                               *
C                             -> CODE TAKES NO ACTION TO CHANGE USER INPUT        *
C                           - THE EXTERNAL NO FLOW BOUNDARY NODE IS SPECIFIED AS  *
C                             ESSENTIAL WITH NO SLIP (IBTYPE=10) AND THE INTERNAL *
C                             BARRIER BOUNDARY NODE IS SPECIFIED AS NATURAL       *
C                             WITH SLIP (IBTYPE=24)                               *
C                             -> CODE RESETS THE EXTERNAL NO FLOW BOUNDARY NODE   *
C                             TO BE AN ESSENTIAL NO FLOW WITH SLIP BOUNDARY NODE  *
C                             (IBTYPE=0)                                          *
C                           - THE EXTERNAL NO FLOW BOUNDARY NODE IS SPECIFIED AS  *
C                             NATURAL WITH SLIP (IBTYPE=20) AND THE INTERNAL      *
C                             BARRIER BOUNDARY NODE IS SPECIFIED AS NATURAL       *
C                             WITH SLIP (IBTYPE=24)                               *
C                             -> CODE TAKES NO ACTION TO CHANGE USER INPUT        *
C                          INTERNAL BARRIER BOUNDARIES CAN NOT OVERLAP EXTERNAL   *
C                           SPECIFIED FLOW BOUNDARY SEGMENTS, EXTERNAL BARRIER    *
C                           BOUNDARY SEGMENTS OR INTERNAL NO NORMAL FLOW          *
C                           BOUNDARIES.                                           *
C                          FOR ALL NORMAL FLOW BOUNDARIES (I.E. IBTYPE = 0,1,2,3, *
C                           4,10,11,12,13,20,21,22,23,24,30), THE FLOW BOUNDARY   *
C                           INTEGRAL IN THE GWCE IS EVALUATED WITH THE APPROPRIATE*
C                           (ZERO, SPECIFIED OR COMPUTED) FLOW  -> THIS IS A      *
C                           NATURAL B.C. IMPLEMENTATION                           *
C                          FOR SOME NORMAL FLOW BOUNDARIES (I.E. IBTYPE = 0,1,2,  *
C                           3,4,10,11,12,13), THE NORMAL DIRECTION VELOCITY       *
C                           COMPONENT IS SET EQUAL TO THE APPROPRIATE (ZERO,      *
C                           SPECIFIED, OR COMPUTED) FLOW VALUE BY ELIMINATING     *
C                           THE NORMAL DIRECTION MOMENTUM EQUATION (OBTAINED      *
C                           AFTER RE-ORIENTING THE X-Y MOMENTUM EQUATIONS INTO    *
C                           N-T DIRECTIONS) AND REPLACING THIS MOMENTUM EQUATION  *
C                           WITH THE NORMAL FLOW VALUE ON THE BOUNDARY EXPRESSED  *
C                           IN TERMS OF VELOCITY (DIVIDING THE NORMAL FLOW BY THE *
C                           ACTUAL TOTAL WATER COLUMN HEIGHT)  -> THIS IS AN      *
C                           ESSENTIAL B.C. IMPLEMENTATION WHICH ALSO INCLUDES THE *
C                           PREVIOUSLY DESCRIBED NATURAL B.C. IMPLEMENTATION      *
C                          FOR SOME NORMAL FLOW BOUNDARIES (I.E. IBTYPE = 10,11,  *
C                           12,13), THE TANGENTIAL DIRECTION VELOCITY COMPONENT   *
C                           IS SET EQUAL TO ZERO BY ELIMINATING THE TANGENTIAL    *
C                           DIRECTION MOMENTUM EQUATION AND REPLACING IT WITH     *
C                           A ZERO PRESCRIBED TANGENTIAL VELOCITY COMPONENT       *
C                           -> THIS IS AN ESSENTIAL B.C. IMPLEMENTATION WHICH     *
C                           ALSO INCLUDES THE PREVIOUSLY DESCRIBED NATURAL AND    *
C                           ESSENTIAL NORMAL FLOW B.C. IMPLEMENTATIONS.           *
C                           USE OF THIS BOUNDARY CONDITION IMPLEMENTATION         *
C                           REQUIRES CONSIDERABLE CARE SINCE STRICTLY SPEAKING    *
C                           THIS TYPE OF NO SLIP BOUNDARY CONDITION IS ONLY       *
C                           MATHEMATICALLY JUSTIFIABLE IF LATERAL VISCOUS TERMS   *
C                           ARE USED IN THE SIMULATION AND ONLY PHYSICALLY        *
C                           JUSTIFIABLE IF THE LATERAL BOUNDARY LAYERS ARE        *
C                           SUFFICIENTLY RESOLVED                                 *
C                                                                                 *
C                                                                                 *
C                                                                                 *
```

```
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 15
C                                                                       *
C       RUNDES = ALPHANUMERIC RUN DESCRIPTION  (<= 32 CHARACTERS)       *
C                                                                       *
C       RUNID = ALPHANUMERIC RUN IDENTIFICATION  (<= 24 CHARACTERS)     *
C                                                                       *
C       NFOVER = NON-FATAL ERROR OVERRIDE OPTION; WHEN NFOVER IS SET EQUAL *
C            TO 1, ALL CHECKED INPUT PARAMETERS WHICH ARE INCONSISTENT  *
C            WILL IN MOST CASES BE AUTOMATICALLY CORRECTED TO A DEFAULT  *
C            OR CORRECTED VALUE AND EXECUTION WILL CONTINUE.  BE SURE TO *
C            READ THE NONFATAL WARNING MESSAGES TO SEE WHAT THE PROGRAM  *
C            HAS DONE;  IF NFOVER IS SPECIFIED NOT EQUAL TO 1, ALL CHECKED *
C            INPUT PARAMETERS WHICH ARE INCONSISTENT WILL LEAD TO PROGRAM *
C            TERMINATION. NOTE THAT NFOVER DOES NOT EFFECT ALL FATAL    *
C            WARNING CHECKS WHICH WILL ALWAYS STOP EXECUTION;           *
C            FURTHERMORE ALL NON-FATAL RUNTIME ERROR CHECKS ARE TREATED  *
C            IN THE SAME MANNER AS INPUT PARAMETER CHECKS               *
C                                                                       *
C       NABOUT = PARAMETER FOR ABREVIATED OUTPUT ON UNIT 16 ; IF NABOUT=1 *
C            THEN THE OUTPUT TO UNIT 16 WILL BE ABREVIATED BY NOT       *
C            ECHO PRINTING MOST OF THE UNIT 14, 21, AND 22 INPUT        *
C            INFORMATION; IF NABOUT IS NOT SPECIFIED EQUAL TO 1, ALL THE *
C            INFORMATION FROM ALL INPUT FILES WILL BE ECHO PRINTED TO    *
C            UNIT 16                                                    *
C                                                                       *
C       NSCREEN = PARAMETER WHICH CONTROLS OUTPUT TO UNIT 6             *
C               = 0 NO OUTPUT IS DIRECTED TO UNIT 6                     *
C               = 1 SCREEN OUTPUT IS DIRECTED TO UNIT 6                 *
C                                                                       *
C       IHOT = PARAMETER WHICH CONTROLS WETHER THE MODEL IS HOT STARTED  *
C             NOTE: THE HOT START FACILITY IS ONLY AVAILABLE FOR 2DDI RUNS *
C               = 0 DO NOT HOT START THE MODEL (USE COLD START)         *
C               = 67 HOT START MODEL USING INPUT READ ON UNIT 67 (fort.67) *
C               = 68 HOT START MODEL USING INPUT READ ON UNIT 68 (fort.68) *
C                                                                       *
C       ICS = PARAMETER WHICH DETERMINES IN WHICH COORDINATE SYSTEM THE  *
C            CALCULATIONS ARE PERFORMED                                 *
C            ICS = 1 STANDARD CARTESIAN COORDINATE FROM OF THE GOVERNING *
C                    EQUATIONS IS USED IN THE MODEL. INPUT COORDIANTES ARE *
C                    IN STANDARD LENGTH UNITS (AS DETERMINED BY G)      *
C                    IF NTIP = 1 AND/OR NCOR = 1, THEN AN INVERSE CARTE *
C                    PARALLELOGRAMMATIQUE PROJECTION IS USED IN ORDER TO *
C                    OBTAIN THE COORDINATES FOR THE NODES IN DEGREES    *
C                    LONGITUDE AND LATITUDE SO THAT THE VARIABLE CORIOLIS *
C                    AND TIDAL POTENTIAL FORCING FUNCTION CAN BE COMPUTED *
C                    HOWEVER, WE NOTE THAT IF THE CORIOLIS PARAMETER    *
C                    VARIES SIGNIFICANTLY OVER THE DOMAIN AND/OR THE    *
C                    DOMAIN IS LARGE ENOUGH THAT TIDAL POTENTIAL FORCING *
C                    IS SIGNIFICANT, IT IS RECOMMENDED TO OPERATE THE CODE *
C                    IN WITH ICS =2 (WITH THE GOVERNING EQUATIONS BASED *
C                    ON SPERICAL EQUATIONS)                             *
C            ICS = 2 GOVERNING EQUATIONS ARE BASED ON THE SPHERICAL     *
C                    EQUATIONS TRANSFORMED INTO USING A CARTE PARALLEL0- *
C                    GRAMMATIQUE PROJECTION (CPP). THE INPUT COORDINATES *
C                    ARE IN DEGREES LONGITUDE AND LATITUDE WHICH ARE THEN *
C                    TRANSFORMED USING THE CPP PROJECTION               *
C                                                                       *
C       IM  = MODEL TYPE                                                *
C            = 0  2DDI MODEL, HYDRODYNAMICS ONLY                        *
C            = 1  3D VS MODEL, HYDRODYNAMICS ONLY                       *
C            = 2  3D DSS MODEL, HYDRODYNAMICS ONLY                      *
C            = 10 2DDI MODEL, HYDRODYNAMICS AND TRANSPORT               *
C                                                                       *
C       NOLIBF = A MODEL OPTION PARAMETER WHICH DETERMINES WHETHER BOTTOM *
C            STRESS IS APPLIED AS A LINEAR RELATIONSHIP OR USING A      *
C            NONLINEAR PARAMETERIZATION                                 *
C            NOLIBF = 0 LINEAR BOTTOM FRICTION.  FRICTION COEFFICIENT   *
C                       SPECIFIED BELOW                                 *
C            NOLIBF = 1 NONLINEAR BOTTOM FRICTION - STANDARD QUADRATIC  *
C                       BOTTOM FRICTION LAW.  FRICTION COEFFICIENT       *
```

```
C                     SPECIFIED BELOW.                                  *
C          NOLIBF = 2 NONLINEAR BOTTOM FRICTION - HYBRID BETWEEN FRICTION*
C                     LAW IN WHICH THE FRICTION COEFFICIENT INCREASES    *
C                     AS THE DEPTH DECREASES (E.G. A MANNING FRICTION    *
C                     LAW) AT SHALLOW DEPTHS AND A STANDARD QUADRATIC    *
C                     FRICTION LAW AT DEEP DEPTHS.                      *
C                     CF = CFMIN*[1+(HBREAK/H)**FTHETA]**(FGAMMA/FTHETA) *
C                     THE REQUIRED PARAMETERS ARE SPECIFIED BELOW.      *
C                                                                       *
C          NOLIFA = A MODEL OPTION PARAMETER WHICH DETERMINES HOW THE FINITE  *
C               AMPLITUDE COMPONENT OF THE TOTAL DEPTH IS CONSIDERED     *
C               NOLIFA = 0 FINITE AMPLITUDE TERMS IN THE EQUATIONS ARE NOT *
C                          TURNED ON, WETTING AND DRYING OF ELEMENTS IS   *
C                          DISABLED.                                      *
C               NOLIFA = 1 FINITE AMPLITUDE TERMS IN THE EQUATIONS ARE TURNED *
C                          ON, WETTING AND DRYING OF ELEMENTS IS DISABLED.*
C               NOLIFA = 2 FINITE AMPLITUDE TERMS IN THE EQUATIONS ARE TURNED *
C                          ON, WETTING AND DRYING OF ELEMENTS IS ENABLED  *
C                        NOTE, WHETHER WETTING AND DRYING IS ENABLED CHANGES *
C                          THE MEANING OF H0 AND THE NEED TO SPECIFY      *
C                          SEVERAL ADDITIONAL PARAMETERS ON THIS LINE.    *
C                          (SEE BELOW).                                   *
C                        NOTE, WHEN THE FINITE AMPLITUDE TERMS ARE TURNED ON, *
C                          THE TIME DERIVATIVE PORTION OF THE ADVECTIVE TERMS *
C                          SHOULD ALSO BE TURNED ON FOR PROPER MASS       *
C                          CONSERVATION AND CONSISTENCY (I.E. WHEN NOLIFA > 0,*
C                          NOLICAT SHOULD BE SET EQUAL TO 1)              *
C                                                                         *
C          NOLICA = A MODEL OPTION PARAMETER THAT DETERMINES WHETHER THE  *
C               ADVECTIVE TERMS (WITH THE EXCEPTION OF THE TIME DERIVATIVE *
C               PORTION THAT SHOWS UP IN THE GWCE) ARE TURNED ON          *
C               NOLICA = 0 ADVECTIVE TERMS CONTAINING SPATIAL DERIVATIVES  *
C                          ARE NOT INCLUDED IN THE COMPUTATIONS           *
C               NOLICA = 1 ADVECTIVE TERMS CONTAINING SPATIAL DERIVATIVES  *
C                          ARE INCLUDED IN THE COMPUTATIONS               *
C                        NOTE, WHEN THE SPATIAL DERIVATIVE PORTIONS OF THE *
C                          ADVECTIVE TERMS ARE INCLUDED, THE TIME DERIVATIVE *
C                          PORTION OF THE ADVECTIVE TERMS (IN THE GWCE) SHOULD*
C                          ALSO BE INCLUDED (I.E. WHEN NOLICA=1, NOLICAT=1)  *
C                                                                         *
C          NOLICAT = A MODEL OPTION PARAMETER THAT DETERMINES WHETHER THE *
C               TIME DERIVATIVE COMPONENT OF THE ADVECTIVE TERMS THAT SHOW UP *
C               IN THE GWCE ARE INCLUDED IN THE COMPUTATIONS              *
C               NOLICAT = 0 THE TIME DERIVATIVE COMPONENTS OF THE ADVECTIVE  *
C                          TERMS IN THE GWCE ARE NOT INCLUDED IN THE      *
C                          COMPUTATIONS                                   *
C               NOLICAT = 1 THE TIME DERIVATIVE COMPONENTS OF THE ADVECTIVE  *
C                          TERMS IN THE GWCE ARE INCLUDED IN THE COMPUTATIONS *
C                        NOTE, THESE TERMS SHOULD BE TURNED ON IF EITHER THE *
C                          FINITE AMPLITUDE OR THE SPATIAL DERIVATIVE     *
C                          COMPONENTS OF THE ADVECTIVE TERMS ARE TURNED ON FOR*
C                          PROPER MASS CONSERVATION AND SOLUTION CONSISTENCY  *
C                                                                         *
C          NWP = BOTTOM FRICTION OPTION PARAMETER                         *
C              = 0 FOR SPATIALLY CONSTANT LINEAR OR QUADRATIC BOTTOM FRICTION *
C                  COEFFICIENT                                            *
C              = 1 FOR SPATIALLY VARYING LINEAR OR QUADRATIC BOTTOM FRICTION  *
C                  UNIT 21 MUST BE OPENED:                                *
C                  WILL NOT WORK FOR NOLIBF = 2.                          *
C                                                                         *
C          NCOR = CORIOLIS OPTION PARAMETER                               *
C               = 0 FOR SPATIALLY CONSTANT CORIOLIS PARAMETER             *
C               = 1 FOR SPATIALLY VARIABLE CORIOLIS PARAMETER             *
C                                                                         *
C          NTIP = TIDAL POTENTIAL & SELF ATTRACTION/LOAD TIDE OPTION PARAMETER*
C               = 0 WHEN THE TIDAL POTENTIAL & SELF ATTRACTION/LOAD TIDE  *
C                   FUNCTIONS ARE NOT USED                                *
C               = 1 WHEN ONLY TIDAL POTENTIAL FUNCTION IS USED            *
C               = 2 WHEN BOTH TIDAL POTENTIAL & SELF ATTRACTION/LOAD TIDE *
C                   FUNCTIONS ARE USED.  IN THIS CASE THE SELF ATTRACTION/ *
```

```
C             LOAD TIDE INFORMATION IS READ IN FOR EACH CONSTITUENT AT  *
C             EACH NODE IN THE GRID ON UNIT 23.                         *
C                                                                       *
C          NWS = WIND STRESS AND SURFACE PRESSURE OPTION PARAMETER       *
C              = 0 NO WIND STRESS OR SURFACE PRESSURE IS APPLIED         *
C              > 0 SPATIALLY VARYING WIND STRESS AND SURFACE PRESSURE ARE *
C                APPLIED                                                *
C              = 1 WIND STRESS AND PRESSURE ARE READ IN AT ALL GRID NODES *
C                EVERY TIME STEP FROM UNIT 22.                          *
C              = 2 WIND STRESS AND PRESSURE ARE READ FROM THE UNIT 22 FILE AT *
C                ALL ADCIRC GRID NODES AT A WIND TIME INTERVAL THAT DOES NOT*
C                EQUAL THE MODEL TIME STEP.  INTERPOLATION IN TIME IS USED  *
C                TO SYNCRONIZE THE WIND AND PRESSURE INFORMATION WITH THE   *
C                MODEL TIME STEP.                                           *
C                THE WIND TIME INTERVAL MUST BE SPECIFIED LATER IN THE      *
C                UNIT 15 FILE.                                              *
C              = 3 WIND VELOCITY IS READ IN FROM A FLEET NUMERIC FORMAT WIND  *
C                FILE ON UNIT 22.  THIS DATA IS INTERPOLATED IN SPACE ONTO   *
C                THE ADCIRC GRID.  INTERPOLATION IN TIME IS USED TO          *
C                SYNCRONIZE THE WIND INFORMATION WITH THE MODEL TIME STEP.   *
C                GARRET'S FORMULA IS USED TO COMPUTE WIND STRESS FROM        *
C                VELOCITY.                                                  *
C                SEVERAL PARAMETERS DESCRIBING THE FLEET NUMERIC WIND FILE   *
C                MUST BE SPECIFIED LATER IN THE UNIT 15 FILE.               *
C              = 4 WIND VELOCITY AND PRESSURE ARE READ IN AT SELECTED ADCIRC  *
C                GRID NODES FROM A PBL/JAG FORMAT FILE ON UNIT 22.          *
C                INTERPOLATION IN TIME IS USED TO SYNCRONIZE THE WIND       *
C                INFORMATION WITH THE MODEL TIME STEP.  GARRET'S FORMULA IS *
C                USED TO COMPUTE WIND STRESS FROM VELOCITY.                 *
C                THE WIND TIME INTERVAL MUST BE SPECIFIED LATER IN THE UNIT *
C                15 FILE.                                                   *
C             =10 10 M HIGH WIND VELOCITY AND SURFACE PRESSURE ARE READ IN  *
C                FROM NWS WIND FILES.  THESE FILES ARE IN BINARY AND HAVE   *
C                BEEN CREATED FROM A LARGER GRIB FORMAT FILE USING UNPKGRB1.*
C                EACH FILE IS ASSUMED TO CONTAIN DATA ON A GAUSSIAN LON/LAT *
C                GRID FROM A SINGLE TIME.  CONSECUTIVE FILES ARE ASSUMED TO *
C                BE SEPARATED BY 6 HOURS IN TIME.  THE FILES ARE NAMED USING*
C                THE CONVENTION:                                            *
C                UNIT 200 -WIND & PRES AT RUN START OR HOT START            *
C                UNIT 206 -WIND & PRES 6 HOURS AFTER RUN START OR HOT START *
C                UNIT 212 -WIND & PRES 12 HOURS AFTER RUN START OR HOT START*
C                UNIT 218 -WIND & PRES 18 HOURS AFTER RUN START OR HOT START*
C                   ....                                                    *
C                AND SO ON UNTIL END OF RUN.                                *
C                NOTE: IF THE MODEL IS HOT STARTED, IT MUST BE DONE AT AN   *
C                  EVEN 6 HOUR INTERVAL SO THAT THE HOT START TIME          *
C                  CORRESPONDS TO THE TIME OF A WIND/PRESSURE FIELD FILE.   *
C                NOTE: IF THE MODEL IS HOT STARTED, THE WIND/PRESSURE FILES *
C                  MUST AGAIN BEGIN WITH UNIT 200 AND CONTINUE FROM THERE.  *
C                NOTE: ENOUGH WIND/PRESSURE FILES MUST BE PRESENT TO EXTEND *
C                  THROUGH THE ENDING TIME OF THE MODEL RUN.                *
C                SEVERAL PARAMETERS DESCRIBING THE MET FILE MUST BE         *
C                  SPECIFIED LATER IN THE UNIT 15 FILE.                     *
C                                                                           *
C          NRAMP = RAMP OPTION PARAMETER                                    *
C                = 0 NO RAMP FUNCTION IS USED FOR FORCING FUNCTIONS         *
C                = 1 A HYPERBOLIC TANGENT RAMP FUNCTION IS SPECIFIED AND    *
C                  APPLIED TO THE SURFACE ELEVATION SPECIFIED BOUNDARY      *
C                  CONDITIONS, THE TIDAL POTENTIAL FORCING FUNCTION AS WELL *
C                  AS THE WIND AND ATMOSPHERIC PRESSURE FORCING FUNCTIONS   *
C                                                                           *
C          G = GRAVITATIONAL CONSTANT : DETERMINES UNITS                    *
C                WE NOTE THAT THE CODE ALWAYS OPERATES IN SECONDS AND THAT *
C                THE SPECIFIED TIME UNITS FOR G MUST BE SECONDS             *
C                WHEN ICS IS SELECTED EQUAL TO 1 AND WHEN EITHER NTIP AND   *
C                OR NCOR HAVE BEEN SET EQUAL TO 1, G MUST BE SPECIFIED      *
C                EQUAL TO 9.81 M/SEC*SEC SINCE THE PROGRAM ONLY OPERATES    *
C                IN METRIC UNITS FOR THIS CASE                              *
C                WHEN ICS IS SELECTED EQUAL TO 2, G MUST BE                 *
C                SPECIFIED EQUAL TO 9.81 M/SEC*SEC SINCE THE PROGRAM ONLY   *
```

```
C                OPERATES IN METRIC UNITS FOR THIS CASE                  *
C                                                                        *
C          TAU0 = GENERALIZED WAVE-CONTINUITY EQUATION WEIGHTING FACTOR :  *
C                  TAU0 WEIGHS THE PRIMITIVE AND WAVE PORTIONS OF THE GWCE *
C                  TAU0 = 0 : PURE WAVE EQUATION                         *
C                  TAU0 >> 1 : PRIMITIVE CONTINUITY EQUATION             *
C                  SEE DETAILS IN USER MANUAL ON HOW TAU0 SHOULD BE SELECTED *
C                                                                        *
C          DT = TIME STEP ( IN SECONDS )                                 *
C                                                                        *
C          STATIM = STARTING SIMULATION TIME ( IN DAYS ) ; DEFINES THE   *
C                    STARTING TIME FOR THE SIMULATION ; THE FIRST TIME STEP *
C                    COMPUTES RESULTS FOR TIME=STATIM+DT                 *
C                                                                        *
C          REFTIM = REFERENCE TIME (IN DAYS) WITH RESPECT TO WHICH ALL NODAL *
C                    FACTORS AND EQUILIBRIUM ARGUMENTS ARE COMPUTED (I.E. *
C                    FFT(I),FACET(I),FF(I),FACE(I),FFHA(I),FACEHA(I) ).  *
C                                                                        *
C          IF(NWS.EQ.2) WTIMINC = TIME INTERVAL (IN SECONDS) BETWEEN WIND AND *
C                           PRESSURE DATA IN THE UNIT 22 FILE.           *
C                                                                        *
C          IF(NWS.EQ.3) IREFYR,IREFMO,IREFDAY,IREFHR,IREFMIN,REFSEC      *
C                           NOTE: THESE VALUES MUST BE CONSISTENT WITH THE *
C                                 TIMES IN THE FLEET NUMERIC WIND FILE.  *
C                           IREFYR = YEAR OF THE START OF THE SIMULATION *
C                           IREFMO = MONTH OF THE START OF THE SIMULATION *
C                           IREFDAY = DAY OF THE START OF THE SIMULATION *
C                           IREFHR = HOUR OF THE START OF THE SIMULATION *
C                           IREFMIN = MINUTE OF THE START OF THE SIMULATION *
C                           REFSEC = SECOND OF THE START OF THE SIMULATION *
C                           NOTE: THESE VALUES ARE COMBINED TOGETHER TO GIVE *
C                                 WREFTIM WHICH IS THE START TIME OF THE *
C                                 SIMULATION IN SECONDS SINCE THE BEGINNINIG *
C                                 OF THE CALANDER YEAR.                  *
C                           NOTE: THE CODE IS CURRENTLY CONFIGURED TO RUN WITH *
C                                 ONLY 1 CALANDER YEAR'S DATA.  IT IS NOT *
C                                 POSSIBLE TO COMBINE DATA FROM TWO DIFFERENT *
C                                 YEARS INTO A SINGLE FILE AND THEN RUN! *
C                                                                        *
C          IF(NWS.EQ.3) NWLAT,NWLON,WLATMAX,WLONMIN,WLATINC,WLONINC,WTIMINC *
C                           NWLAT = NUMBER OF LATITUDE VALUES IN WIND FILE *
C                           NWLON = NUMBER OF LONGITUDE VALUES IN WIND FILE *
C                           WLATMAX = MAXIMUM LATITUDE (DEG) OF DATA IN THE *
C                                     UNIT 22 FILE (< 0 SOUTH OF THE EQUATOR) *
C                           WLONMIN = MINIMUM LONGITUDE (DEG) OF DATA IN THE *
C                                     UNIT 22 FILE (< 0 WEST OF GRENICH  *
C                                     MERIDIAN)                          *
C                           WLATINC = LATITUDE INCREMENT (DEG) OF DATA IN THE *
C                                     UNIT 22 FILE (MUST BE > 0)         *
C                           WLONINC = LONGITUDE INCREMENT (DEG) OF DATA IN THE *
C                                     UNIT 22 FILE (MUST BE > 0)         *
C                           WTIMINC = TIME INTERVAL (IN SECONDS) BETWEEN WIND *
C                                     DATA IN THE UNIT 22 FILE.          *
C                                                                        *
C          IF(NWS.EQ.4) WTIMINC = TIME INTERVAL (IN SECONDS) BETWEEN WIND AND *
C                           PRESSURE DATA IN THE UNIT 22 FILE.           *
C                                                                        *
C          IF(NWS.EQ.10) NWLAT,NWLON,WTIMINC                             *
C                           NWLAT = NUMBER OF LATITUDE VALUES IN MET FILE *
C                                 = 190 FOR TYPE 126 GLOBAL GAUSSIAN GRID *
C                           NWLON = NUMBER OF LONGITUDE VALUES IN MET FILE *
C                                 = 384 FOR TYPE 126 GLOBAL GAUSSIAN GRID *
C                           WTIMINC = TIME INTERVAL (IN SEC) BETWEEN MET FILES *
C                                   = 6.*3600 = 21600 FOR AVN FILES.     *
C                                                                        *
C          RNDAY  = TOTAL NUMBER OF DAYS OF NUMERICAL SIMULATION         *
C                                                                        *
C          DRAMP = VALUE IN DAYS USED IN COMPUTING THE RAMP FUNCTION WHICH IS *
C                    EVALUATED AS  RAMP=TANH(2.0*IT*DT/(24.0*3600.0*DRAMP)) *
C                    WHERE IT EQUALS THE TIME STEP INDEX                 *
```

```
C              NOTE THAT DRAMP IS APPROX EQUAL TO THE NUMBER OF DAYS   *
C              AT WHICH RAMP=0.96                                      *
C                                                                      *
C     A00,B00,C00 = TIME WEIGHTING FACTORS (K+1,K,K-1) IN WAVE EQUATION *
C                                                                      *
C     IF((NOLIFA.EQ.0).OR.(NOLIFA.EQ.1)) H0                            *
C                     H0 = MINIMUM BATHYMETRIC DEPTH                   *
C                     ALL BATHYMETRIC DEPTHS IN THE INPUT GRID (UNIT 14) *
C                     ARE CHECKED AGAINST THIS DEPTH AND SET EQUAL TO H0 *
C                     IF THEY ARE INITIALLY LESS THAN H0               *
C     IF(NOLIFA.EQ.2) H0,NODEDRYMIN,NODEWETRMP,VELMIN                  *
C                     H0 = NOMINAL MINIMUM WATER DEPTH FOR DRYING.     *
C                     (TYPICAL VALUE = 0.01M)                          *
C                     NODEDRYMIN = MINIMUM NUMBER OF TIME STEPS A NODE *
C                     MUST REMAIN DRY AFTER DRYING BEFORE IT CAN WET.  *
C                     (TYPICAL VALUE 5 - 20)                           *
C                     NODEWETRMP = NUMBER OF TIME STEPS AFTER A NODE WETS*
C                     OVER WHICH THE MINIMUM WATER DEPTH FOR DRYING IS  *
C                     RAMPED UP FROM 0 TO H0.  THIS IS DONE SO THAT SMALL*
C                     OSCILLATIONS IN THE WATER DEPTH AFTER A NODE WETS *
C                     DO NOT CAUSE THE NODE TO DRY AGAIN.              *
C                     (TYPICAL VALUE 5 - 20)                           *
C                     VELMIN = MINIMUM VELOCITY FOR WETTING.           *
C                     A NODE ALONG THE WET/DRY INTERFACE WETS IF THE    *
C                     VELOCITY COMPONENTS AT THAT NODE THAT ARE DIRECTED *
C                     TOWARD ALL NEIGHBORING DRY NODES EXCEED VELMIN.  *
C                     (TYPICAL VALUE 0.05 M/S)                         *
C                                                                      *
C     SLAM0,SFEA0 = LONGITUDE AND LATITUDE ON WHICH CPP PROJECTION IS  *
C             CENTERED (IN DEGREES)                                    *
C                                                                      *
C     IF(NOLIBF.EQ.0) FFACTOR = BOTTOM FRICTION COEFFICIENT            *
C                     THE STANDARD LINEAR BOTTOM FRICTION COEFFICIENT  *
C                     TAU = FFACTOR; NOTE: IN THIS CASE FFACTOR SHOULD *
C                     BE EQUAL TO TAU0 WHICH WAS SPECIFIED ABOVE.      *
C                     FFACTOR IS ALWAYS READ IN, HOWEVER IT IS ONLY USED *
C                     IN THE COMPUTATIONS WHEN NWP=0                   *
C     IF(NOLIBF.EQ.1) FFACTOR = BOTTOM FRICTION COEFFICIENT            *
C                     THE STANDARD QUADRATIC BOTTOM FRICTION COEFFICIENT *
C                     CF = FFACTOR;                                    *
C                     FFACTOR IS ALWAYS READ IN, HOWEVER IT IS ONLY USED *
C                     IN THE COMPUTATIONS WHEN NWP=0                   *
C     IF(NOLIBF.EQ.2) FFACTOR,HBREAK,FTHETA,FGAMMA                     *
C                     FFACTOR = CFMIN THIS IS THE STANDARD QUADRATIC   *
C                               FRICTION COEFFICIENT THAT IS APPROACHED *
C                               IN DEEP WATER.                         *
C                     HBREAK = THE BREAK DEPTH, IN DEEPER WATER THE    *
C                              FRICTION RELATION LOOKS LIKE A STANDARD *
C                              QUADRATIC RELATION.  IN SHALLOWER WATER *
C                              THE FRICTION FACTOR INCREASES AS THE DEPTH*
C                              DECREASES (E.G. LIKE A MANNING TYPE     *
C                              FRICTION LAW).                          *
C                                    (HBREAK = 1 M)                    *
C                     FTHETA = A PARAMETER THAT DETERMINES HOW RAPIDLY *
C                              THE HYBRED RELATIONSHIP APPROACHES EACH *
C                              ASYMPTOTIC LIMIT.                       *
C                                   (FTHETA=10 IS RECOMMENDED)         *
C                     FGAMMA = A PARAMETER THAT DETERMINES HOW THE     *
C                              FRICTION FACTOR INCREASES AS THE WATER  *
C                              DEPTH DECREASES. SETTING THIS TO 1/3 GIVES*
C                              A MANNING FRICTION LAW TYPE OF BEHAVIOR *
C                                   (FGAMMA=1/3 IS RECOMMENDED)        *
C                                                                      *
C     EVM,EVC = SPATIALLY CONSTANT LATERAL VISCOSITY/DIFFUSIVITY (L^2/T) *
C             EVM = LATERAL VISCOSITY FOR MOMENTUM EQUATIONS           *
C             EVC = LATERAL DIFFUSIVITY FOR TRANSPORT EQUATION         *
C             NOTE: IT IS ONLY NECESSARY TO SPECIFY EVC IF IM=10,      *
C                   OTHERWISE, IT CAN BE OMITTED.                      *
C                                                                      *
C     CORI = CONSTANT CORIOLIS COEFFICIENT : THIS VALUE IS ALWAYS      *
```

```
C              READ IN, HOWEVER IT IS ONLY USED IN THE COMPUTATIONS        *
C              WHEN NCOR=0                                                 *
C                                                                          *
C          NTIF = NUMBER OF TIDAL POTENTIAL CONSTITUENTS                   *
C                                                                          *
C          TIPOTAG(I) : SEE DESCRIPTION OF TPK(I),AMIGT(I),ETRF(I),FFT(I)  *
C              AND FACET(I)                                                *
C                                                                          *
C          TPK(I),AMIGT(I),ETRF(I),FFT(I),FACET(I), I=1,NTIF ; TIDAL       *
C              POTENTIAL AMPLITUDE, FREQUENCY, EARTH TIDE POTENTIAL        *
C              REDUCTION FACTOR (GENERALLY TAKEN TO BE 0.690 FOR ALL       *
C              CONSTITUENTS (HENDERSHOTT) BUT FOR MORE PRECISE             *
C              CALCULATIONS CAN TAKE ON SLIGHTLY DIFFERENT VALUES (E.G.    *
C              SEE WAHR, 1981)), NODAL FACTOR AND EQUILIBRIUM              *
C              ARGUMENT IN DEGREES : THESE VALUES ARE PRECEDED             *
C              BY TIPOTAG(I) WHICH IS AN ALPHANUMERIC DESCRIPTOR (I.E.     *
C              THE CONSTITUENT NAME)                                       *
C                                                                          *
C          NBFR = NUMBER OF TIDAL FORCING FREQUENCIES ON ELEVATION SPECIFIED *
C              BOUNDARIES                                                  *
C                                                                          *
C          BOUNTAG(I) : SEE DESCRIPTION OF AMIG(I),FF(I),FACE(I)           *
C                                                                          *
C          AMIG(I),FF(I),FACE(I) I=1,NBFR ; FORCING FREQUENCY,             *
C              NODAL FACTOR, EQUILIBRIUM ARGUMENT IN DEGREES FOR TIDAL     *
C              FORCING ON ELEVATION SPECIFIED BOUNDARIES: THESE VALUES ARE*
C              PRECEDED BY BOUNTAG(I), AN ALPHANUMERIC DESCRIPTOR (I.E.    *
C              THE CONSTITUENT NAME)                                       *
C                                                                          *
C          ALPHA : SEE DESCRIPTION OF EMO(I,J),EFA(I,J)  (<= 10 CHARACTERS) *
C                                                                          *
C          EMO(I,J),EFA(I,J)  I=1,NBFR , J=1,NETA ; AMPLITUDE AND PHASE (IN *
C              DEGREES) OF THE HARMONIC FORCING FUNCTION AT THE ELEVATION *
C              SPECIFIED BOUNDARIES FOR FREQUENCY I AND ELEVATION          *
C              SPECIFIED BOUNDARY FORCING NODE J.                          *
C              NOTE THAT THE PARAMETER NETA IS DEFINED AND READ IN FROM    *
C              UNIT 14 INPUT : THE FORCING VALUES ARE PRECEDED BY AN       *
C              ALPHANUMERIC DESCRIPTOR ALPHA TO FACILITATE VERIFYING THAT *
C              THE CORRECT DATA MATCHES A GIVEN FREQUENCY                  *
C                                                                          *
C          ANGINN : FLOW BOUNDARY NODES WHICH ARE SET UP TO HAVE A NORMAL FLOW*
C              ESSENTIAL BOUNDARY CONDITION AND HAVE AN INNER ANGLE LESS   *
C              THAN ANGINN (SPECIFIED IN DEGREES) WILL HAVE THE TANGENTIAL*
C              VELOCITY ZEROED. IN EITHER CASE, THE NORMAL VELOCITY WILL   *
C              BE DETERMINED FROM THE ESSENTIAL BOUNDARY CONDITION.        *
C                                                                          *
C          NFFR =  NUMBER OF FREQUENCIES IN THE SPECIFIED NORMAL FLOW EXTERNAL*
C              BOUNDARY CONDITION.  IF NFFR=0, THE NORMAL FLOW BOUNDARY    *
C              CONDITION IS ASSUMED TO BE NON-PERIODIC AND WILL BE READ IN*
C              FROM UNIT 20.                                               *
C              NFFR IS ONLY INCLUDED IN THE UNIT 15 FILE IF ONE OR MORE    *
C              SPECIFIED (NON-ZERO) NORMAL FLOW EXTERNAL BOUNDARIES WERE   *
C              DEFINED IN THE UNIT 14 INPUT, (IBTYPE=2,12 OR 22).          *
C                                                                          *
C          IF(NFFR.EQ.0) FTIMINC = TIME INCREMENT (SECS) OF NORMAL FLOW/UNIT *
C              WIDTH VALUES READ IN FROM UNIT 20 FILE.                     *
C              FTIMINC IS ONLY INCLUDED IN THE UNIT 15 FILE IF ONE OR      *
C              MORE SPECIFIED (NON-ZERO) NORMAL FLOW EXTERNAL BOUNDARIES   *
C              WERE DEFINED IN THE UNIT 14 INPUT, (IBTYPE=2,12 OR 22)      *
C              AND NFFR=0.                                                 *
C                                                                          *
C          FBOUNTAG(I) : SEE DESCRIPTION OF FAMIG(I),FFF(I),FFACE(I)       *
C                                                                          *
C          FAMIG(I),FFF(I),FFACE(I) I=1,NFFR ; FORCING FREQUENCY, NODAL    *
C              FACTOR, EQUILIBRIUM ARGUMENT IN DEGREES FOR PERIODIC NORMAL*
C              FLOW FORCING ON FLOW BOUNDARIES. THESE VALUES ARE PRECEEDED*
C              BY FBOUNTAG(I), AN ALPHANUMERIC DESCRIPTOR (I.E. THE        *
C              CONSTITUENT NAME)                                           *
C                                                                          *
C          FALPHA: SEE DESCRIPTION OF QNAM(I,J),QNPH(I,J)  (<=10 CHARACTERS) *
```

```
C
C       QNAM(I,J),QNPH(I,J) I=1,NFFR, J=1,NFLBN; AMPLITUDE AND PHASE (IN   *
C              DEGREES) OF THE PERIODIC NORMAL FLOW/UNIT WIDTH             *
C              (E.G. M^2/S) FOR FREQUENCY I AND "SPECIFIED NORMAL FLOW"    *
C              BOUNDARY NODE J.  A POSITIVE FLOW/UNIT WIDTH IS INTO THE    *
C              DOMAIN AND A NEGATIVE FLOW/UNIT WIDTH IS OUT OF THE DOMAIN. *
C              NOTE: NFLBN IS THE TOTAL NUMBER OF SPECIFIED NORMAL FLOW    *
C                    EXTERNAL BOUNDARY NODES THAT WERE DESIGNATED IN THE   *
C                    UNIT 14 FILE AS HAVING A NON ZERO NORMAL FLOW         *
C                    (IBTYPE=2,12,22).                                     *
C              NOTE: THE FORCING VALUES ARE PRECEDED BY AN ALPHANUMERIC    *
C                    DESCRIPTOR FALPHA TO FACILITATE VERIFYING THAT THE    *
C                    CORRECT DATA MATCHES A GIVEN FREQUENCY                *
C                                                                          *
C       NOUTE,TOUTSE,TOUTFE,NSPOOLE = OUTPUT PARAMETERS WHICH CONTROL THE  *
C              TIME SERIES OUTPUT PROVIDED FOR ELEVATIONS SOLUTIONS AT     *
C              SELECTED ELEVATION RECORDING STATIONS (UNIT 61 OUTPUT)      *
C              NOUTE =-2 OUTPUT IS PROVIDED AT THE SELECTED ELEVATION      *
C                        RECORDING STATIONS IN BINARY FORMAT               *
C                        FOLLOWING A HOT START, A NEW UNIT 61 FILE IS      *
C                        CREATED.                                          *
C              NOUTE =-1 OUTPUT IS PROVIDED AT THE SELECTED ELEVATION      *
C                        RECORDING STATIONS IN STANDARD ASCII FORMAT.       *
C                        FOLLOWING A HOT START, A NEW UNIT 61 FILE IS      *
C                        CREATED.                                          *
C              NOUTE = 0 NO OUTPUT IS PROVIDED AT THE SELECTED ELEVATION   *
C                        RECORDING STATIONS                                *
C              NOUTE = 1 OUTPUT IS PROVIDED AT THE SELECTED ELEVATION      *
C                        RECORDING STATIONS IN STANDARD ASCII FORMAT.       *
C                        FOLLOWING A HOT START, CONTINUED OUTPUT IS MERGED*
C                        INTO THE EXISTING UNIT 61 FILE.                   *
C              NOUTE = 2 OUTPUT IS PROVIDED AT THE SELECTED ELEVATION      *
C                        RECORDING STATIONS IN BINARY FORMAT               *
C                        FOLLOWING A HOT START, CONTINUED OUTPUT IS MERGED*
C                        INTO THE EXISTING UNIT 61 FILE.                   *
C              TOUTSE = THE NUMBER OF DAYS AFTER WHICH ELEVATION STATION   *
C                       DATA IS RECORDED TO UNIT 61 (TOUTSE IS RELATIVE    *
C                       TO STATIM)                                         *
C              TOUTFE = THE NUMBER OF DAYS AFTER WHICH ELEVATION STATION   *
C                       DATA CEASES TO BE RECORDED TO UNIT 61 (TOUTFE IS   *
C                       RELATIVE TO STATIM)                                *
C              NSPOOLE = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS  *
C                        SPOOLED TO UNIT 61; I.E. THE OUTPUT IS SPOOLED    *
C                        TO UNIT 61 EVERY NSPOOLE TIME STEPS AFTER TOUTSE  *
C                                                                          *
C       NSTAE = THE NUMBER OF ELEVATION RECORDING STATIONS (THIS IS        *
C              ALWAYS READ IN REGARDLESS OF THE VALUE OF NOUTE)            *
C                                                                          *
C       XEL(I),YEL(I)  I=1,NSTAE ; THE COORDINATES OF THE ELEVATION        *
C              RECORDING STATION I, FOR ALL NSTAE STATIONS.                *
C              IF ICS = 1, COORDINATES ARE INPUT AS STANDARD CARTESIAN     *
C              IF ICS = 2, COORDINATES ARE INPUT AS DEGREES LONGITUDE      *
C              AND LATITUDE                                                *
C              IF AN ELEVATION RECORDING STATION IS INPUT WHICH DOES NOT   *
C              LIE WITHIN THE COMPUTATIONAL DOMAIN, A NON-FATAL ERROR      *
C              MESSAGE WILL APPEAR.  IF NFOVER HAS BEEN SET EQUAL TO 1,    *
C              THE CODE WILL ESTIMATE THE NEAREST ELEMENT AND USE THAT     *
C              AS THE BASIS OF INTERPOLATION. A PROXIMITY INDEX IS ALSO     *
C              PRINTED OUT, WHICH INDICATES HOW CLOSE OR FAR THE STATION   *
C              COORDINATES ARE FROM THE NEAREST ELEMENT. THIS INDEX MAY    *
C              BE INTERPRETED AS THE NUMBER OF ELEMENTS THAT THE STATION   *
C              LIES FROM THE NEAREST ELEMENT                               *
C                                                                          *
C       NOUTV,TOUTSV,TOUTFV,NSPOOLV = OUTPUT PARAMETERS WHICH CONTROL THE  *
C              TIME SERIES OUTPUT PROVIDED FOR VELOCITY SOLUTIONS AT       *
C              SELECTED VELOCITY RECORDING STATIONS (UNIT 62 OUTPUT)       *
C              NOUTV =-2 OUTPUT IS PROVIDED AT THE SELECTED VELOCITY       *
C                        RECORDING STATIONS IN BINARY FORMAT               *
C                        FOLLOWING A HOT START, A NEW UNIT 62 FILE IS      *
C                        CREATED.                                          *
```

```
C                NOUTV =-1 OUTPUT IS PROVIDED AT THE SELECTED VELOCITY      *
C                          RECORDING STATIONS IN STANDARD ASCII FORMAT       *
C                          FOLLOWING A HOT START, A NEW UNIT 62 FILE IS      *
C                          CREATED.                                          *
C                NOUTV = 0 NO OUTPUT IS PROVIDED AT THE SELECTED VELOCITY    *
C                          RECORDING STATIONS                                *
C                NOUTV = 1 OUTPUT IS PROVIDED AT THE SELECTED VELOCITY       *
C                          RECORDING STATIONS IN STANDARD ASCII FORMAT       *
C                          FOLLOWING A HOT START, CONTINUED OUTPUT IS MERGED*
C                          INTO THE EXISTING UNIT 62 FILE.                   *
C                NOUTV = 2 OUTPUT IS PROVIDED AT THE SELECTED VELOCITY       *
C                          RECORDING STATIONS IN BINARY FORMAT               *
C                          FOLLOWING A HOT START, CONTINUED OUTPUT IS MERGED*
C                          INTO THE EXISTING UNIT 62 FILE.                   *
C                TOUTSV = THE NUMBER OF DAYS AFTER WHICH VELOCITY STATION    *
C                         DATA IS RECORDED TO UNIT 62 (TOUTSV IS RELATIVE    *
C                         TO STATIM)                                         *
C                TOUTFV = THE NUMBER OF DAYS AFTER WHICH VELOCITY STATION    *
C                         DATA CEASES TO BE RECORDED TO UNIT 62  (TOUTFV IS  *
C                         RELATIVE TO STATIM)                                *
C                NSPOOLV = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS  *
C                          SPOOLED TO UNIT 62; I.E. THE OUTPUT IS SPOOLED    *
C                          TO UNIT 62 EVERY NSPOOLV TIME STEPS AFTER TOUTSV  *
C                                                                            *
C           NSTAV = THE NUMBER OF VELOCITY RECORDING STATIONS (THIS IS       *
C                   ALWAYS READ IN REGARDLESS OF THE VALUE OF NOUTV)         *
C                                                                            *
C           XEV(I),YEV(I)  I=1,NSTAV ; THE COORDINATES OF THE VELOCITY       *
C                   RECORDING STATION I, FOR ALL NSTAV STATIONS              *
C                   IF ICS = 1, COORDINATES ARE INPUT AS STANDARD CARTESIAN  *
C                   IF ICS = 2, COORDINATES ARE INPUT AS DEGREES LONGITUDE   *
C                   AND LATITUDE                                             *
C                   IF A VELOCITY RECORDING STATION IS INPUT WHICH DOES NOT  *
C                   LIE WITHIN THE COMPUTATIONAL DOMAIN, A NON-FATAL ERROR   *
C                   MESSAGE WILL APPEAR.  IF NFOVER HAS BEEN SET EQUAL TO 1, *
C                   THE CODE WILL ESTIMATE THE NEAREST ELEMENT AND USE THAT  *
C                   AS THE BASIS OF INTERPOLATION. A PROXIMITY INDEX IS ALSO  *
C                   PRINTED OUT, WHICH INDICATES HOW CLOSE OR FAR THE STATION *
C                   COORDINATES ARE FROM THE NEAREST ELEMENT. THIS INDEX MAY  *
C                   BE INTERPRETED AS THE NUMBER OF ELEMENTS THAT THE STATION *
C                   LIES FROM THE NEAREST ELEMENT                            *
C                                                                            *
C           NOUTC,TOUTSC,TOUTFC,NSPOOLC = OUTPUT PARAMETERS WHICH CONTROL THE *
C                   TIME SERIES OUTPUT PROVIDED FOR CONCENTRATION SOLUTIONS AT *
C                   SELECTED CONCENTRATION RECORDING STATIONS (UNIT 71 OUTPUT) *
C                   NOUTC =-2 OUTPUT IS PROVIDED AT THE SELECTED CONCENTRATION *
C                             RECORDING STATIONS IN BINARY FORMAT            *
C                             FOLLOWING A HOT START, A NEW UNIT 71 FILE IS   *
C                             CREATED.                                       *
C                   NOUTC =-1 OUTPUT IS PROVIDED AT THE SELECTED CONCENTRATION *
C                             RECORDING STATIONS IN STANDARD ASCII FORMAT     *
C                             FOLLOWING A HOT START, A NEW UNIT 71 FILE IS   *
C                             CREATED.                                       *
C                   NOUTC = 0 NO OUTPUT IS PROVIDED AT THE SELECTED          *
C                             CONCENTRATION RECORDING STATIONS               *
C                   NOUTC = 1 OUTPUT IS PROVIDED AT THE SELECTED CONCENTRATION *
C                             RECORDING STATIONS IN STANDARD ASCII FORMAT     *
C                             FOLLOWING A HOT START, CONTINUED OUTPUT IS MERGED*
C                             INTO THE EXISTING UNIT 71 FILE.                *
C                   NOUTC = 2 OUTPUT IS PROVIDED AT THE SELECTED CONCENTRATION *
C                             RECORDING STATIONS IN BINARY FORMAT            *
C                             FOLLOWING A HOT START, CONTINUED OUTPUT IS MERGED*
C                             INTO THE EXISTING UNIT 71 FILE.                *
C                   TOUTSC = THE NUMBER OF DAYS AFTER WHICH CONCENTRATION    *
C                            STATION DATA IS RECORDED TO UNIT 71 (TOUTSC IS  *
C                            RELATIVE TO STATIM)                             *
C                   TOUTFC = THE NUMBER OF DAYS AFTER WHICH CONCENTRATION    *
C                            STATION DATA CEASES TO BE RECORDED TO UNIT 71   *
C                           (TOUTFC IS RELATIVE TO STATIM)                   *
C                   NSPOOLC = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS *
```

```
C                     SPOOLED TO UNIT 71; I.E. THE OUTPUT IS SPOOLED   *
C                     TO UNIT 71 EVERY NSPOOLC TIME STEPS AFTER TOUTSC *
C             NOTE: THIS LINE IS ONLY READ IN IF TRANSPORT IS INCLUDED  *
C                   IN THE MODEL RUN (I.E. IM=10)                       *
C                                                                       *
C       NSTAC = THE NUMBER OF CONCENTRATION RECORDING STATIONS          *
C             NOTE: THIS LINE IS ONLY READ IN IF TRANSPORT IS INCLUDED  *
C                   IN THE MODEL RUN (I.E. IM=10)                       *
C             NOTE: THIS IS READ IN EVEN IF NOUTC=0                     *
C                                                                       *
C       XEC(I),YEC(I)  I=1,NSTAC ; THE COORDINATES OF THE CONCENTRATION *
C             RECORDING STATION I, FOR ALL NSTAC STATIONS.              *
C             NOTE: THIS LINE IS ONLY READ IN IF TRANSPORT IS INCLUDED  *
C                   IN THE MODEL RUN (I.E. IM=10)                       *
C             NOTE: THE COORDINATES MUST BE CONSISTENT (I.E. CARTESIAN OR*
C                   SPHERICAL) WITH THE UNIT 14 FILE AND THE COORDINATE *
C                   DESIGNATION PARAMETER, ICS, IN THE UNIT 15 FILE.    *
C             NOTE: IF A CONCENTRATION RECORDING STATION IS INPUT WHICH *
C                   DOES NOT LIE WITHIN THE COMPUTATIONAL DOMAIN, A NON-*
C                   FATAL ERROR MESSAGE WILL APPEAR.  IF NFOVER HAS BEEN*
C                   SET EQUAL TO 1, THE CODE WILL ESTIMATE THE NEAREST  *
C                   ELEMENT AND USE THAT AS THE BASIS OF INTERPOLATION. *
C                   A PROXIMITY INDEX IS PRINTED OUT IN THE UNIT 16 FILE*
C                   THAT INDICATES HOW CLOSE OR FAR THE STATION         *
C                   COORDINATES ARE FROM THE NEAREST ELEMENT. THIS INDEX*
C                   MAY BE INTERPRETED AS THE NUMBER OF ELEMENTS THAT THE*
C                   STATION LIES FROM THE NEAREST ELEMENT               *
C                                                                       *
C       NOUTGE,TOUTSGE,TOUTFGE,NSPOOLGE=OUTPUT PARAMETERS WHICH CONTROL *
C             THE TIME SERIES OUTPUT PROVIDED FOR GLOBAL ELEVATION      *
C             SOLUTIONS AT ALL NODES WITHIN THE DOMAIN (UNIT 63 OUTPUT) *
C             NOUTGE =-2 GLOBAL ELEVATION OUTPUT IS PROVIDED IN         *
C                        BINARY FORMAT                                  *
C                        FOLLOWING A HOT START, A NEW UNIT 63 FILE IS   *
C                        CREATED.                                       *
C             NOUTGE =-1 GLOBAL ELEVATION OUTPUT IS PROVIDED IN         *
C                        STANDARD ASCII FORMAT                          *
C                        FOLLOWING A HOT START, A NEW UNIT 63 FILE IS   *
C                        CREATED.                                       *
C             NOUTGE = 0 NO GLOBAL ELEVATION OUTPUT IS PROVIDED         *
C             NOUTGE = 1 GLOBAL ELEVATION OUTPUT IS PROVIDED IN         *
C                        STANDARD ASCII FORMAT                          *
C                        FOLLOWING A HOT START, CONTINUED OUTPUT IS     *
C                        MERGED INTO THE EXISTING UNIT 63 FILE.         *
C             NOUTGE = 2 GLOBAL ELEVATION OUTPUT IS PROVIDED IN         *
C                        BINARY FORMAT                                  *
C                        FOLLOWING A HOT START, CONTINUED OUTPUT IS     *
C                        MERGED INTO THE EXISTING UNIT 63 FILE.         *
C             TOUTSGE = THE NUMBER OF DAYS AFTER WHICH GLOBAL ELEVATION *
C                       DATA IS RECORDED TO UNIT 63 (TOUTSGE IS RELATIVE*
C                       TO STATIM)                                      *
C             TOUTFGE = THE NUMBER OF DAYS AFTER WHICH GLOBAL ELEVATION *
C                       DATA CEASES TO BE RECORDED TO UNIT 63 (TOUTFGE  *
C                       IS RELATIVE TO STATIM)                          *
C             NSPOOLGE = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS*
C                        SPOOLED TO UNIT 63; I.E. THE OUTPUT IS SPOOLED *
C                       TO UNIT 63 EVERY NSPOOLGE TIME STEPS AFTER TOUTFGE*
C                                                                       *
C       NOUTGV,TOUTSGV,TOUTFGV,NSPOOLGV=OUTPUT PARAMETERS WHICH CONTROL *
C             THE TIME SERIES OUTPUT PROVIDED FOR GLOBAL VELOCITY       *
C             SOLUTIONS AT ALL NODES WITHIN THE DOMAIN (UNIT 64 OUTPUT) *
C             NOUTGV =-2 GLOBAL VELOCITY OUTPUT IS PROVIDED IN          *
C                        BINARY FORMAT                                  *
C                        FOLLOWING A HOT START, A NEW UNIT 64 FILE IS   *
C                        CREATED.                                       *
C             NOUTGV =-1 GLOBAL VELOCITY OUTPUT IS PROVIDED IN          *
C                        STANDARD ASCII FORMAT                          *
C                        FOLLOWING A HOT START, A NEW UNIT 64 FILE IS   *
C                        CREATED.                                       *
C             NOUTGV = 0 NO GLOBAL VELOCITY OUTPUT IS PROVIDED          *
```

```
C                NOUTGV = 1 GLOBAL VELOCITY OUTPUT IS PROVIDED IN          *
C                           STANDARD ASCII FORMAT                          *
C                           FOLLOWING A HOT START, CONTINUED OUTPUT IS      *
C                           MERGED INTO THE EXISTING UNIT 64 FILE.         *
C                NOUTGV = 2 GLOBAL VELOCITY OUTPUT IS PROVIDED IN          *
C                           BINARY FORMAT                                  *
C                           FOLLOWING A HOT START, CONTINUED OUTPUT IS      *
C                           MERGED INTO THE EXISTING UNIT 64 FILE.         *
C                TOUTSGV = THE NUMBER OF DAYS AFTER WHICH GLOBAL VELOCITY  *
C                          DATA IS RECORDED TO UNIT 64 (TOUTSGV IS RELATIVE*
C                          TO STATIM)                                      *
C                TOUTFGV = THE NUMBER OF DAYS AFTER WHICH GLOBAL VELOCITY  *
C                          DATA CEASES TO BE RECORDED TO UNIT 64 (TOUTFGV   *
C                          IS RELATIVE TO STATIM)                          *
C                NSPOOLGV = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS*
C                           SPOOLED TO UNIT 64; I.E. THE OUTPUT IS SPOOLED  *
C                           TO UNIT 64 EVERY NSPOOLGV TIME STEPS AFTER TOUTSGV*
C                                                                           *
C           NOUTGC,TOUTSGC,TOUTFGC,NSPOOLGC                                 *
C                OUTPUT PARAMETERS WHICH CONTROL THE TIME SERIES OUTPUT     *
C                PROVIDED FOR GLOBAL CONCENTRATION SOLUTIONS AT ALL NODES   *
C                WITHIN THE DOMAIN (UNIT 73 OUTPUT)                         *
C                NOUTGC =-2 GLOBAL CONCENTRATION OUTPUT IS PROVIDED IN      *
C                           BINARY FORMAT                                   *
C                           FOLLOWING A HOT START, A NEW UNIT 73 FILE IS    *
C                           CREATED.                                        *
C                NOUTGC =-1 GLOBAL CONCENTRATION OUTPUT IS PROVIDED IN      *
C                           STANDARD ASCII FORMAT                           *
C                           FOLLOWING A HOT START, A NEW UNIT 73 FILE IS    *
C                           CREATED.                                        *
C                NOUTGC = 0 NO GLOBAL CONCENTRATION OUTPUT IS PROVIDED      *
C                NOUTGC = 1 GLOBAL CONCENTRATION OUTPUT IS PROVIDED IN      *
C                           STANDARD ASCII FORMAT                           *
C                           FOLLOWING A HOT START, CONTINUED OUTPUT IS      *
C                           MERGED INTO THE EXISTING UNIT 73 FILE.          *
C                NOUTGC = 2 GLOBAL CONCENTRATION OUTPUT IS PROVIDED IN      *
C                           BINARY FORMAT                                   *
C                           FOLLOWING A HOT START, CONTINUED OUTPUT IS      *
C                           MERGED INTO THE EXISTING UNIT 73 FILE.          *
C                TOUTSGC = THE NUMBER OF DAYS AFTER WHICH GLOBAL           *
C                          CONCENTRATION DATA IS RECORDED TO UNIT 73        *
C                          (TOUTSGC IS RELATIVE TO STATIM)                  *
C                TOUTFGC = THE NUMBER OF DAYS AFTER WHICH GLOBAL           *
C                          CONCENTRATION DATA CEASES TO BE RECORDED TO      *
C                          UNIT 73 (TOUTFGC IS RELATIVE TO STATIM)          *
C                NSPOOLGC = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS*
C                          SPOOLED TO UNIT 73; I.E. THE OUTPUT IS SPOOLED   *
C                          TO UNIT 73 EVERY NSPOOLGC TIME STEPS AFTER       *
C                          TOUTFGC                                          *
C                NOTE: THIS LINE IS ONLY READ IN IF TRANSPORT IS INCLUDED   *
C                      IN THE MODEL RUN (I.E. IM=10)                        *
C                                                                           *
C           NOUTGW,TOUTSGW,TOUTFGW,NSPOOLGW                                 *
C                OUTPUT PARAMETERS WHICH CONTROL THE TIME SERIES OUTPUT     *
C                PROVIDED FOR WIND STRESS AT ALL NODES WITHIN THE DOMAIN    *
C                (UNIT 74 OUTPUT)                                           *
C                NOUTGW =-2 GLOBAL WIND STRESS OUTPUT IS PROVIDED IN        *
C                           BINARY FORMAT                                   *
C                           FOLLOWING A HOT START, A NEW UNIT 74 FILE IS    *
C                           CREATED.                                        *
C                NOUTGW =-1 GLOBAL WIND STRESS OUTPUT IS PROVIDED IN        *
C                           STANDARD ASCII FORMAT                           *
C                           FOLLOWING A HOT START, A NEW UNIT 74 FILE IS    *
C                           CREATED.                                        *
C                NOUTGW = 0 NO GLOBAL WIND STRESS OUTPUT IS PROVIDED        *
C                NOUTGW = 1 GLOBAL WIND STRESS OUTPUT IS PROVIDED IN        *
C                           STANDARD ASCII FORMAT                           *
C                           FOLLOWING A HOT START, CONTINUED OUTPUT IS      *
C                           MERGED INTO THE EXISTING UNIT 74 FILE.          *
C                NOUTGW = 2 GLOBAL WIND STRESS OUTPUT IS PROVIDED IN        *
```

```
C                         BINARY FORMAT                                 *
C                         FOLLOWING A HOT START, CONTINUED OUTPUT IS     *
C                         MERGED INTO THE EXISTING UNIT 74 FILE.         *
C               TOUTSGW = THE NUMBER OF DAYS AFTER WHICH GLOBAL WIND STRESS*
C                         DATA IS RECORDED TO UNIT 74 (TOUTSGW IS RELATIVE *
C                         TO STATIM)                                     *
C               TOUTFGW = THE NUMBER OF DAYS AFTER WHICH GLOBAL WIND STRESS*
C                         DATA CEASES TO BE RECORDED TO UNIT 74 (TOUTFGW  *
C                         IS RELATIVE TO STATIM)                         *
C               NSPOOLGW = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS*
C                          SPOOLED TO UNIT 74; I.E. THE OUTPUT IS SPOOLED *
C                         TO UNIT 74 EVERY NSPOOLGW TIME STEPS AFTER TOUTSGW*
C               NOTE: THIS LINE IS ONLY READ IN IF WIND FORCING IS INCLUDED*
C                     IN THE MODEL RUN (I.E. NWS>0)                      *
C                                                                        *
C          NHARFR = NUMBER OF FREQUENCIES TO INCLUDE IN HARMONIC ANALYSIS OF *
C                   MODEL RESULTS                                        *
C                   NOTE: HARMONIC OUTPUT IS ONLY AVAILABLE FOR 2DDI ELEVATION *
C                         AND VELOCITY                                   *
C                                                                        *
C          HAFNAM(I) : SEE DESCRIPTION OF HAFREQ(I),HAFF(I),HAFACE(I)    *
C                                                                        *
C          HAFREQ(I),HAFF(I),HAFACE(I) I=1,NHARFR                        *
C                  PARAMETERS DESCRIBING THE CONSTITUENTS TO BE INCLUDED IN *
C                  THE HARMONIC ANALYSIS OF MODEL RESULTS                *
C                  HAFREQ(I) = FREQUENCY (RAD/S)                         *
C                  HAFF(I) =   NODAL FACTOR                              *
C                  HAFACE(I) = EQUILIBRIUM ARGUMENT (DEGREES)            *
C                  NOTE: THESE VALUES ARE PRECEDED BY HAFNAM, AN ALPHANUMERIC *
C                        DESCRIPTOR (I.E. THE CONSTITUENT NAME) WHOSE LENGTH *
C                        MUST BE <= 10 CHARACTERS                        *
C                  NOTE: IF A STEADY COMPONENT WILL BE INCLUDED IN THE   *
C                        HARMONIC ANALYSIS, THIS MUST BE THE FIRST CONSTITUENT*
C                        LISTED (I.E., THE CONSTITUENT CORRESPONDING TO I=1). *
C                                                                        *
C          THAS,THAF,NHAINC,FMV                                          *
C                  PARAMETERS THAT CONTROL THE CALCULATION OF HARMONIC    *
C                  CONSTITUENTS BOTH AT STATIONS AND GLOBALLY            *
C                  THAS =    THE NUMBER OF DAYS AFTER WHICH DATA STARTS TO BE *
C                            HARMONICALLY ANALYSED (THAS IS RELATIVE TO STATIM)*
C                  THAF =    THE NUMBER OF DAYS AFTER WHICH DATA CEASES TO BE *
C                            HARMONICALLY ANALYSED (THAF IS RELATIVE TO STATIM)*
C                  NHAINC = THE NUMBER OF TIME STEPS AT WHICH INFORMATION IS *
C                            HARMONICALLY ANALYSED (INFORMATION EVERY NHAINC *
C                           TIME STEPS AFTER THAS IS USED IN HARMONIC ANALYSIS)*
C                  FMV =     FRACTION OF THE HARMONIC ANALYSIS PERIOD     *
C                            (EXTENDING BACK FROM THE END OF THE HARMONIC  *
C                            ANALYSIS PERIOD) TO USE FOR COMPARING THE     *
C                            WATER ELEVATION AND VELOCITY MEANS AND VARIANCES *
C                            FROM THE RAW MODEL TIME SERIES WITH CORRESPONDING *
C                            MEANS AND VARIANCES OF A TIME SERIES RESYNTHESIZED*
C                            FROM THE HARMONIC CONSTITUENTS.             *
C                            THIS COMPARISON IS HELPFUL FOR IDENTIFYING  *
C                            NUMERICAL INSTABILITIES AND FOR DETERMINING HOW *
C                            COMPLETE THE HARMONIC ANALYSIS WAS.         *
C                            EXAMPLES:                                   *
C                              FMV = 0. - DO NOT COMPUTE ANY MEANS AND VARS. *
C                              FMV = 0.1 - COMPUTE MEANS AND VARS. OVER FINAL *
C                                    10% OF PERIOD USED IN HARMONIC ANALYSIS *
C                              FMV = 1.0 - COMPUTE MEANS AND VARS. OVER ENTIRE *
C                                PERIOD USED IN HARMONIC ANALYSIS        *
C                            NOTE: THE MEANS AND VARIANCE CALCULATIONS ARE ONLY*
C                                  DONE IF GLOBAL HARMONIC CALCULATIONS ARE *
C                                  PERFORMED.  RESULTS ARE WRITTEN OUT TO UNIT *
C                                  55.  A SUMMARY OF THE POOREST COMPARISONS *
C                                  THROUGHOUT THE DOMAIN AND THE NODE NUMBERS *
C                                  WHERE THESE OCCURRED IS GIVEN AT THE END OF *
C                                  THE UNIT 16 OUTPUT FILE.              *
C                            NOTE: THE TIME SERIES RESYSTHESIS FROM THE  *
C                                  HARMONIC CONSTITUENTS CAN USE UP A LOT OF *
C
```

```
C                              CPU TIME SINCE THIS IS DONE FOR EVERY TIME  *
C                              STEP DURING THE SPECIFIED PART OF THE       *
C                              HARMONIC ANALYSIS PERIOD.  IF THE HARMONIC  *
C                              ANALYSIS PERIOD EXTENDS FOR ONLY A FEW DAYS,*
C                              IT IS PRACTICAL TO SET FMV=1.  OTHERWISE,   *
C                              IT BECOMES UNREASONABLY TIME CONSUMING TO   *
C                              COMPUTE MEANS AND VARIANCES FOR MORE THAN   *
C                              10-20 DAYS.  ULTIMATELY, THE PRACTICAL LIMIT*
C                              TO THESE CALCULATIONS DEPENDS ON THE NUMBER *
C                              OF NODES, THE NUMBER OF CONSTITUENTS IN THE *
C                              HARMONIC ANALYSIS, AND THE SIZE OF THE TIME *
C                              STEP.                                       *
C                                                                          *
C           NHASE,NHASV,NHAGE,NHAGV                                        *
C                   PARAMETERS THAT CONTROL THE SPATIAL LOCATIONS WHERE     *
C                   HARMONIC ANALYSIS IS PERFORMED                         *
C                   NHASE = 0 NO HARMONIC ANALYSIS IS PERFORMED AT THE SELECTED*
C                             ELEVATION RECORDING STATIONS                 *
C                   NHASE = 1 HARMONIC ANALYSIS IS PERFORMED AT THE SELECTED   *
C                             ELEVATION RECORDING STATIONS (OUTPUT ON UNIT 51) *
C                   NOTE: THE STATIONS ARE AS SPECIFIED IN THE SECTION ON TIME *
C                         SERIES STATION ELEVATION OUTPUT                  *
C                   NHASV = 0 NO HARMONIC ANALYSIS IS PERFORMED AT THE SELECTED*
C                             VELOCITY RECORDING STATIONS                  *
C                   NHASV = 1 HARMONIC ANALYSIS IS PERFORMED AT THE SELECTED   *
C                             VELOCITY RECORDING STATIONS (OUTPUT ON UNIT 52)  *
C                   NOTE: THE STATIONS ARE AS SPECIFIED IN THE SECTION ON TIME *
C                         SERIES STATION VELOCITY OUTPUT                   *
C                   NHAGE = 0 NO HARMONIC ANALYSIS IS PERFORMED FOR GLOBAL     *
C                             ELEVATIONS                                   *
C                   NHAGE = 1 HARMONIC ANALYSIS IS PERFORMED FOR GLOBAL        *
C                             ELEVATIONS (OUTPUT ON UNIT 53)               *
C                   NHAGV = 0 NO HARMONIC ANALYSIS IS PERFORMED FOR GLOBAL     *
C                             VELOCITIES                                   *
C                   NHAGV = 1 HARMONIC ANALYSIS IS PERFORMED FOR GLOBAL        *
C                             VELOCITIES (OUTPUT ON UNIT 54)               *
C                                                                          *
C           NHSTAR,NHSINC                                                  *
C                   PARAMETERS THAT CONTROL THE GENERATION OF HOT START OUTPUT *
C                   NHSTAR = 0 NO HOT START OUTPUT FILES GENERATED             *
C                   NHSTAR = 1 HOT START OUTPUT FILES GENERATED                *
C                   NHSINC = THE NUMBER OF TIME STEPS AT WHICH HOT START OUTPUT*
C                            FILE IS GENERATED (HOT START FILE IS GENERATED    *
C                            EVERY NHSINC TIME STEPS)                          *
C                                                                          *
C           ITITER,ISLDIA,CONVCR,ITMAX                                     *
C                   PARAMETERS THAT PROVIDE INFORMATION ABOUT THE SOLVER THAT  *
C                   WILL BE USED FOR THE GWCE.                             *
C                   ITITER = -1 ONLY FOR LUMPED, EXPLICIT GWCE, MATRIX IS      *
C                               DIAGONAL AND NO EXTERNAL SOLVER IS NEEDED      *
C                   ITITER = 0 USE DIRECT BANDED SOLVER                        *
C                   ITITER = 1 USE ITERATIVE JCG SOLVER (FROM ITPACKV 2D)      *
C                   ITITER = 2 USE ITERATIVE JSI SOLVER (FROM ITPACKV 2D)      *
C                   ITITER = 3 USE ITERATIVE SOR SOLVER (FROM ITPACKV 2D)      *
C                   ITITER = 4 USE ITERATIVE SSORCG SOLVER (FROM ITPACKV 2D)   *
C                   ITITER = 5 USE ITERATIVE SSORSI SOLVER (FROM ITPACKV 2D)   *
C                   ITITER = 6 USE ITERATIVE RSCG SOLVER (FROM ITPACKV 2D)     *
C                   ITITER = 7 USE ITERATIVE RSSI SOLVER (FROM ITPACKV 2D)     *
C                   ISLDIA = 0 FATAL ERROR MESSGS ONLY FROM ITPACKV 2D(UNIT 33)*
C                   ISLDIA = 1 WARNING MESSGS AND MINIMUM OUTPUT FROM          *
C                              ITPACKV 2D (UNIT 33)                        *
C                   ISLDIA = 2 REASONABLE SUMMARY OF ALGORITHM PROGRESS FROM   *
C                              ITPACKV 2D (UNIT 33)                        *
C                   ISLDIA = 3 PARAMETER VALUES AND INFORMATIVE COMMENTS FROM  *
C                              ITPACKV 2D (UNIT 33)                        *
C                   ISLDIA = 4 APPROXIMATE SOLUTION AFTER EACH ITERATION FROM  *
C                              ITPACKV 2D (UNIT 33)                        *
C                   ISLDIA = 5 ORIGINAL SYSTEM FROM ITPACKV 2D (UNIT 33)       *
C                   CONVCR = ABSOLUTE CONVERGENCE CRITERIA (SHOULD BE NO       *
C                              SMALLER THAN 500 TIMES THE MACHINE PRECISION)   *
```

```
C                ITMAX =  MAXIMUM NUMBER OF ITERATIONS EACH TIME STEP        *
C                NOTE: ALL OF THE PARAMETERS MUST BE INPUT REGARDLESS OF     *
C                      WHETHER A DIRECT OR ITERATIVE SOLVER IS SELECTED.     *
C                      HOWEVER, ISLDIA, CONVCR AND ITMAX ARE ONLY USED WITH *
C                      THE ITERATIVE SOLVERS)                               *
C                NOTE: WE HAVE HAD THE BEST LUCK USING THE JCG SOLVER (THE  *
C                      JSI ALSO WORKS OK)  THE OTHERS EITHER BLOW UP RAPIDLY*
C                      OR DON'T WORK BECAUSE THE GRID CAN NOT BE RED/BLACK  *
C                      ORDERED.  WE TYPICALLY USE CONVCR=1E-6 ON THE CRAY   *
C                      AND CONVCR=2.98E-5 ON A 32 BIT MACHINE.  AFTER THE   *
C                      FIRST FEW TIME STEPS, THE SOLUTIONS USUALLY CONVERGE *
C                      WITHIN 5-10 ITERATIONS.                              *
C                                                                           *
C                                                                           *
C                                                                           *
C                                                                           *
C                                                                           *
C     -  DESCRIPTION OF HOT START INPUT VARIABLES READ IN FROM UNIT 67 OR 68 *
C                                                                           *
C          IDENTICAL TO UNIT 67 AND UNIT 68 OUTPUT. (SEE BELOW)             *
C                                                                           *
C          NOTE: IT IS OFTEN CONVENIENT TO CHANGE INTPUT VALUES BEFORE      *
C                RESTARTING A MODEL RUN.  ABOUT THE ONLY VARIABLES THAT     *
C                CAN NOT BE CHANGED ARE:                                    *
C                                                                           *
C                DT - (TIME STEP)                                           *
C                ANYTHING HAVING TO DO WITH TIME SERIES OR HARMONIC OUTPUT  *
C                   AFTER THE MODEL HAS STARTED WRITING OR COMPUTING THESE  *
C                   QUANTITIES.  THE ONLY EXCEPTION IS THE ENDING TIME OF   *
C                   AN OUTPUT TIME SERIES CAN BE CHANGED (E.G.IF THE LENGTH *
C                   OF THE RUN IS INCREASED)                                *
C                                                                           *
C                                                                           *
C                                                                           *
C                                                                           *
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 20                *
C                                                                           *
C           DO I=1,NFLBN                                                    *
C              QNIN(I)                                                      *
C              END DO                                                       *
C           WHERE:                                                          *
C             NFLBN = THE TOTAL NUMBER OF FLOW BOUNDARY NODES THAT WERE     *
C                 DESIGNATED IN THE UNIT 14 FILE AS HAVING A NON ZERO NORMAL*
C                 FLOW.                                                     *
C             QNIN(I) = NORMAL FLOW/UNIT WIDTH (E.G., M^2/S) AT SPECIFIED   *
C                 NORMAL FLOW NODE I.  A POSITIVE FLOW/UNIT WIDTH IS INTO THE*
C                 DOMAIN AND A NEGATIVE FLOW/UNIT WIDTH IS OUT OF THE DOMAIN.*
C                                                                           *
C           NOTES:                                                          *
C           - THE FIRST DATA SET IS PROVIDED AT TIME=STATIM.  SUBSEQUENT    *
C             DATA SETS ARE PROVIDED EVERY FLOW TIME INTERVAL.              *
C           - THE FLOW TIME INTERVAL MUST BE SET IN THE UNIT 15 FILE        *
C           - ENOUGH SETS OF NORMAL FLOW/UNIT WIDTH VALUES MUST BE PROVIDED *
C             TO EXTEND FOR THE ENTIRE MODEL RUN, OTERWISE THE RUN WILL CRASH!*
C                                                                           *
C                                                                           *
C                                                                           *
C                                                                           *
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 21                *
C                                                                           *
C          AFRIC = ALPAHANUMERIC FRICTION FILE ID (<= 24 CHARACTERS)        *
C          JKI,FRIC(JKI) JKI=1,NP  = NODE NUMBER,  NODAL BOTTOM FRICTION    *
C               COEFFICIENT. NODAL VALUES MUST BE READ IN ASCENDING ORDER.  *
C                                                                           *
C                                                                           *
C                                                                           *
C                                                                           *
C                                                                           *
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 22                *
C                                                                           *
C          IF(NWS.EQ.1)  NHG,WSX2(NHG),WSY2(NHG),PR2(NHG), NHG=1,NP         *
```

```
C            WHERE:                                                          *
C              NHG = NODE NUMBER,                                            *
C              WSX2(NHG) = APPLIED HORIZONTAL FREE SURFACE STRESS IN THE     *
C                          X-DIRECTION DIVIDED BY THE REFERENCE DENSITY OF   *
C                          WATER AT THE NODE (SHOULD BE UNITS (LENGTH/TIME)^2) *
C              WSY2(NHG) = APPLIED HORIZONTAL FREE SURFACE STRESS IN THE     *
C                          Y-DIRECTION DIVIDED BY THE REFERENCE DENSITY OF   *
C                          WATER AT THE NODE (SHOULD BE UNITS (LENGTH/TIME)^2) *
C              PR2(NHG) =  APPLIED ATMOSPHERIC PRESSURE AT THE FREE SURFACE  *
C                          (N/M^2 = Pa) DIVIDED BY THE REFERENCE DENSITY OF  *
C                          WATER DIVIDED BY GRAVITY.                         *
C            NOTES:                                                          *
C            - THE FIRST DATA SET IS PROVIDED AT TIME=STATIM+DT.  SUBSEQUENT *
C              DATA SETS ARE PROVIDED AT EVERY TIME STEP                     *
C            - DATA MUST BE PROVIDED FOR THE ENTIRE MODEL RUN, OTERWISE THE RUN*
C              WILL CRASH!!!                                                 *
C                                                                            *
C          IF(NWS.EQ.2)  NHG,WSX2(NHG),WSY2(NHG),PR2(NHG), NHG=1,NP          *
C            WHERE:                                                          *
C              NHG = NODE NUMBER,                                            *
C              WSX2(NHG) = APPLIED HORIZONTAL FREE SURFACE STRESS IN THE     *
C                          X-DIRECTION DIVIDED BY THE REFERENCE DENSITY OF   *
C                          WATER AT THE NODE (SHOULD BE UNITS (LENGTH/TIME)^2) *
C              WSY2(NHG) = APPLIED HORIZONTAL FREE SURFACE STRESS IN THE     *
C                          Y-DIRECTION DIVIDED BY THE REFERENCE DENSITY OF   *
C                          WATER AT THE NODE (SHOULD BE UNITS (LENGTH/TIME)^2) *
C              PR2(NHG) =  APPLIED ATMOSPHERIC PRESSURE AT THE FREE SURFACE  *
C                          (N/M^2 = Pa) DIVIDED BY THE REFERENCE DENSITY OF  *
C                          WATER DIVIDED BY GRAVITY.                         *
C            NOTES:                                                          *
C            - THE FIRST DATA SET IS PROVIDED AT TIME=STATIM.  SUBSEQUENT    *
C              DATA SETS ARE PROVIDED EVERY WIND TIME INTERVAL.              *
C            - DATA MUST BE PROVIDED FOR THE ENTIRE MODEL RUN, OTERWISE THE RUN*
C              WILL CRASH!!!                                                 *
C            - THE WIND TIME INTERVAL MUST BE SET IN THE UNIT 15 FILE        *
C                                                                            *
C          IF(NWS.EQ.3)                                                      *
C            IWTIME                                                          *
C            DO I=1,NWLAT                                                    *
C               WSPEED(I,J) J=1,NWLON                                        *
C               END DO                                                       *
C            DO I=1,NWLAT                                                    *
C               WDIR(I,J) J=1,NWLON                                          *
C               END DO                                                       *
C            WHERE:                                                          *
C              IWTIME = TIME OF THE WIND FIELD IN THE FOLLOWING INTEGER FORMAT *
C                       YEAR*1000000 + MONTH*10000 + DAY*100 + HR            *
C              WSPEED(I,J) = WIND SPEED IN METER/SEC                         *
C              WDIR(I,J)   = DIRECTION WIND BLOWS FROM IN DEG CW FROM NORTH  *
C            NOTES:                                                          *
C            - THE FIRST DATA SET MUST BE AT OR BEFORE THE DATE AND TIME LISTED*
C              IN THE UNIT 15 FILE AS THE BEGINNING TIME OF THE SIMULATION.  *
C            - DATA SETS ARE PROVIDED EVERY WTIMINC, WHERE THIS PARAMETER IS *
C              THE WIND TIME INTERVAL AND IS SPECIFIED IN THE UNIT 15 FILE.  *
C            - DATA MUST BE PROVIDED FOR THE ENTIRE MODEL RUN, OTERWISE THE RUN*
C              WILL CRASH!!!                                                 *
C            - VALUES FOR NWLAT, NWLON, WTIMINC, THE BEGINNING TIME OF THE   *
C              AND SEVERAL OTHER PARAMETERS MUST BE SET IN THE UNIT 15 FILE. *
C            - THE FOLLOWING TRANSFORMATIONS ARE PREFORMED TO PUT THIS INFO   *
C              INTO USABLE FORM FOR THE MODEL CALCULATIONS                   *
C                WIND_STRESS = DRAG_COEFF*0.001293*WIND_VEL*WIND_SPEED       *
C                DRAG_COEFF = 0.001*(0.75+0.067*WIND_SPEED)                  *
C                IF(DRAG_COEFF.GT.0.003) DRAG_COEFF=0.003                    *
C                                                                            *
C          IF(NWS.EQ.4)                                                      *
C            NHG,WSX2(NHG),WSY2(NHG),PR2(NHG)                                *
C            WHERE:                                                          *
C              NHG = NODE NUMBER,                                            *
C              WSX2(NHG) = APPLIED HORIZONTAL WIND VELOCITY (KNOTS) BLOWING  *
C                          TOWARD THE + X-DIRECTION                          *
```

```
C             WSY2(NHG) = APPLIED HORIZONTAL WIND VELOCITY (KNOTS) BLOWING    *
C                         TOWARD THE + Y DIRECTION                            *
C             PR2(NHG) =  APPLIED ATMOSPHERIC PRESSURE AT THE FREE SURFACE    *
C                         (MILIBARS)                                          *
C          NOTES:                                                             *
C          - THIS LINE MUST HAVE THE FORMAT I8,3E13.5                         *
C          - THIS LINE IS REPEATED FOR AS MANY NODES AS DESIRED               *
C          - A LINE CONTAINING THE # SYMBOL IN COLUMN 2 INDICATES NEW WIND     *
C            AND PRESSURE FIELDS (I.E., VALUES AT THE NEXT TIME INCREMENT)    *
C            BEGIN ON THE FOLLOWING LINE.                                     *
C          - EACH NODE THAT IS NOT CONTAINED IN THE UNIT 22 FILE IS ASSUMED   *
C            TO HAVE ZERO WIND VELOCITY AND PRESSURE = 1013.0                 *
C          - THE FOLLOWING TRANSFORMATIONS ARE PREFORMED TO PUT THIS INFO     *
C            INTO USABLE FORM FOR THE MODEL CALCULATIONS                      *
C              WIND_VEL (M/S) = WSX2*1.04*0.5144, WSY2*1.04*0.5144            *
C              WIND_STRESS = DRAG_COEFF*0.001293*WIND_VEL*WIND_SPEED          *
C              DRAG_COEFF = 0.001*(0.75+0.067*WIND_SPEED)                     *
C              IF(DRAG_COEFF.GT.0.003) DRAG_COEFF=0.003                       *
C              PR2*100/GRAVITY/1000.                                          *
C          - THE FIRST DATA SET IS PROVIDED AT TIME=STATIM. SUBSEQUENT DATA   *
C            SETS ARE PROVIDED EVERY WIND TIME INTERVAL.                      *
C          - DATA MUST BE PROVIDED FOR THE ENTIRE MODEL RUN, OTHERWISE THE    *
C            RUN WILL CRASH!!!                                                *
C          - THE WIND TIME INTERVAL MUST BE SET IN THE UNIT 15 FILE           *
C                                                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 23                  *
C          THIS FILE IS ONLY USED IF NTIP=2.                                  *
C          THE FORMAT IS IDENTICAL TO THE "TEA" HARMONIC FORMAT WITH NO       *
C             VELOCITY INFORMATION INCLUDED.                                  *
C          INFORMATION IS GROUPED BY CONSTITUENTS THAT ARE ASSUMED TO BE IN   *
C             THE SAME ORDER AS THE TIDAL POTENTIAL TERMS LISTED IN THE       *
C             UNIT 15 FILE.                                                   *
C          PHASES MUST BE IN DEGREES.                                         *
C          AMPLITUDES MUST BE IN A UNIT COMPATABLE WITH THE UNITS OF GRAVITY  *
C          THESE VALUES ARE MODIFIED BY THE NODAL FACTOR AND EQUILIBRIUM      *
C             ARGUEMENT PROVIDED FOR THE TIDAL POTENTIAL TERMS.               *
C        DO I=1,NTIF                                                          *
C           TITLE LINE                    - NOTE: THE FIRST FOUR LINES FOR    *
C           FREQUENCY                             EACH CONSTITUENT MUST BE    *
C           1                                     PRESENT IN THE FILE BUT THEY*
C           CONSTITUENT NAME (E.G., M2) -         ARE ACTUALLY SKIPPED OVER   *
C                                                 DURING THE READ.            *
C           DO J=1,NP                                                         *
C              NODE #, AMP(I,NODE #) (DET BY GRAVITY), PHASE(I,NODE #) (DEG)  *
C              END DO                                                         *
C           END DO                                                            *
C                                                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF INPUT VARIABLES READ IN FROM UNIT 200, 206, 212, ...  *
C                                                                             *
C        FILE TYPE: ACCESS='sequential',FORM='unformatted'                    *
C                                                                             *
C        IF(NWS.EQ.10)                                                        *
C           DO N=1,3                                                          *
C              READ(10,END=1100) LENPDS,LENKGDS,NWORDS                        *
C              IF(LENPDS.GT.0) READ(10,END=1100) (PDS(J),J=1,LENPDS)          *
C              IF(LENKGDS.GT.0) READ(10,END=1100) (KGDS(J),J=1,LENKGDS)       *
C              IF(NWORDS.GT.0) READ(10,END=1100) (GRBDATA(N,J),J=1,NWORDS)    *
C              END DO                                                         *
C           WHERE:                                                            *
C             NWORDS=NUMBER OF NODES IN GAUSSIAN GRID                         *
C             GRBDATA(1,J) = U 10 M (CM/S)                                    *
C             GRBDATA(2,J) = V 10 M (CM/S)                                    *
C             GRBDATA(3,J) = SURF P (N/M^2)                                   *
C           NOTES:                                                            *
```

```
C           - THE FIRST DATA SET MUST BE AT THE BEGINNING OF THE COLD OR HOT  *
C             START,                                                          *
C           - DATA SETS ARE PROVIDED EVERY 6 HOURS                            *
C           - DATA MUST BE PROVIDED FOR THE ENTIRE MODEL RUN, OTERWISE THE RUN*
C             WILL CRASH!!!                                                   *
C           - THE FOLLOWING TRANSFORMATIONS ARE PREFORMED TO PUT THIS INFO    *
C             INTO USABLE FORM FOR THE MODEL CALCULATIONS                     *
C               WIND_STRESS = DRAG_COEFF*0.001293*WIND_VEL*WIND_SPEED         *
C               DRAG_COEFF = 0.001*(0.75+0.067*WIND_SPEED)                    *
C               IF(DRAG_COEFF.GT.0.003) DRAG_COEFF=0.003                      *
C               PR2/GRAVITY/1000.                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 51 (ASCII)           *
C                                                                             *
C         NHARFR = NUMBER OF FREQUENCIES HARMONICALLY ANALYZED                *
C         FREQUENCY, NODAL FACTOR, EQUILIBRIUM ARGUEMENT, CONSTITUENT NAME    *
C         NSTAE = NUMBER OF ELEVATION STATIONS INCLUDED IN THE ANALYSIS       *
C         (J,(AMP(I,J), PHASE(I,J) I=1,NHARFR) J=1,NSTAE)                     *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 52 (ASCII)           *
C                                                                             *
C         NHARFR = NUMBER OF FREQUENCIES HARMONICALLY ANALYZED                *
C         FREQUENCY, NODAL FACTOR, EQUILIBRIUM ARGUEMENT, CONSTITUENT NAME    *
C         NSTAV = NUMBER OF VELOCITY STATIONS INCLUDED IN THE ANALYSIS        *
C         (J,(UAMP(I,J),UPHASE(I,J),VAMP(I,J),VPHASE(I,J) I=1,NHARFR)        *
C                                                              J=1,NSTAV)     *
C                                                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 53 (ASCII)           *
C                                                                             *
C         NHARFR = NUMBER OF FREQUENCIES HARMONICALLY ANALYZED                *
C         FREQUENCY, NODAL FACTOR, EQUILIBRIUM ARGUEMENT, CONSTITUENT NAME    *
C         NP = NUMBER OF NODES IN GRID                                        *
C         (J,(AMP(I,J),PHASE(I,J) I=1,NHARFR) J=1,NP)                         *
C                                                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 54 (ASCII)           *
C                                                                             *
C         NHARFR = NUMBER OF FREQUENCIES HARMONICALLY ANALYZED                *
C         FREQUENCY, NODAL FACTOR, EQUILIBRIUM ARGUEMENT, CONSTITUENT NAME    *
C         NP = NUMBER OF NODES IN GRID                                        *
C         (J,(UAMP(I,J),UPHASE(I,J),VAMP(I,J),VPHASE(I,J) I=1,NHARFR) J=1,NP)*
C                                                                             *
C                                                                             *
C                                                                             *
C                                                                             *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 55 (ASCII)           *
C                                                                             *
C         NP = NUMBER OF NODES IN GRID                                        *
C         (J,                                                                 *
C         TSEA(J),HAEA(J),HAEA(J)/TSEA(J),TSEV(J),HAEV(J),HAEV(J)/TSEV(J)     *
C                                                                  J=1,NP)*
C         (J,                                                                 *
C         TSUA(J),HAUA(J),HAUA(J)/TSUA(J),TSUV(J),HAUV(J),HAUV(J)/TSUV(J)     *
C         TSVA(J),HAVA(J),HAVA(J)/TSVA(J),TSVV(J),HAVV(J),HAVV(J)/TSVV(J)     *
C                                                                  J=1,NP)*
C         TSEA - MEAN ELEVATION IN THE RAW MODEL TIME SERIES                  *
C         HAEA - MEAN ELEVATION IN THE RESYNTHESIZED TIME SERIES              *
C         TSEV - ELEVATION VARIANCE IN THE RAW MODEL TIME SERIES              *
C         HAEV - ELEVATION VARIANCE IN THE RESYNTHESIZED TIME SERIES          *
C         TSUA - MEAN X-VELOCITY IN THE RAW MODEL TIME SERIES                 *
```

```
C        HAUA - MEAN X-VELOCITY IN THE RESYNTHESIZED TIME SERIES          *
C        TSUV - X-VELOCITY VARIANCE IN THE RAW MODEL TIME SERIES          *
C        HAUV - X-VELOCITY VARIANCE IN THE RESYNTHESIZED TIME SERIES      *
C        TSVA - MEAN Y-VELOCITY IN THE RAW MODEL TIME SERIES              *
C        HAVA - MEAN Y-VELOCITY IN THE RESYNTHESIZED TIME SERIES          *
C        TSVV - Y-VELOCITY VARIANCE IN THE RAW MODEL TIME SERIES          *
C        HAVV - Y-VELOCITY VARIANCE IN THE RESYNTHESIZED TIME SERIES      *
C                                                                         *
C                                                                         *
C                                                                         *
C    -  GENERAL INFORMATION REGARDING OUTPUT TO UNITS 61,62,63,64,71,73,74*
C                                                                         *
C        THE STARTING REFERENCE TIME FOR THE COMPUTATIONS IS TIME = STATIM *
C             OUTPUT INFORMATION IS TAGGED WITH OUTPUT TIME=STATIM+IT*DT   *
C             (WHERE IT = TIME STEP) AND/OR IT=TIME STEP.                 *
C        OUTPUT TO UNITS 61,62,63,64,71,73, 74 MAY BE IN ASCII OR BINARY  *
C             FORMAT DEPENDING ON HOW NOUTE, NOUTV, NOUTC, NOUTGE, NOUTGV,*
C             NOUTGC, NOUTGW WERE SET IN UNIT 15 INPUT                    *
C                                                                         *
C                                                                         *
C                                                                         *
C                                                                         *
C    -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 61                *
C                                                                         *
C        RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT   *
C             DESCRIPTIONS                                                *
C        NTRSPE, NSTAE, DT*NSPOOLE, NSPOOLE, IRTYPE                       *
C             NTRSPE = THE NUMBER OF DATA SETS TO BE SPOOLED TO UNIT 61   *
C             NSTAE  = THE NUMBER OF STATIONS WITHIN EACH DATA SET        *
C             DT*NSPOOLE = THE TIME INCREMENT BETWEEN DATA SETS (SEC)     *
C             NSPOOLE = THE TIMESTEP INCREMENT BETWEEN DATA SETS          *
C             IRTYPE = 1  (THE RECORD TYPE)                               *
C        TIME,IT ;  TIME (IN SECONDS) AND TIME STEP AT WHICH RESULTS AT   *
C                   ELEVATION STATIONS ARE PROVIDED                       *
C        DO I=1,NSTAE                                                     *
C           I,ET00(I);    STATION NUMBER AND SURFACE ELEVATION AT NSTAE   *
C                         ELEVATION RECORDING STATIONS.                   *
C           END DO                                                        *
C       (NOTE: THE STATION NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)  *
C                                                                         *
C                                                                         *
C                                                                         *
C                                                                         *
C    -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 62                *
C                                                                         *
C        RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT   *
C             DESCRIPTIONS                                                *
C        NTRSPV, NSTAV, DT*NSPOOLV, NSPOOLV, IRTYPE                       *
C             NTRSPV = THE NUMBER OF DATA SETS TO BE SPOOLED TO UNIT 62   *
C             NSTAV  = THE NUMBER OF STATIONS WITHIN EACH DATA SET        *
C             DT*NSPOOLV = THE TIME INCREMENT BETWEEN DATA SETS (SEC)     *
C             NSPOOLV = THE TIMESTEP INCREMENT BETWEEN DATA SETS          *
C             IRTYPE = 2  (THE RECORD TYPE)                               *
C        TIME,IT ;  TIME (IN SECONDS) AND TIME STEP AT WHICH RESULTS AT   *
C                   VELOCITY STATIONS ARE PROVIDED                        *
C        DO I=1,NSTAV                                                     *
C           I,UU00(I),VV00(I);    STATION NUMBER, X VELOCITY AND Y VELOCITIY *
C                                 AT NSTAV VELOCITY RECORDING STATIONS    *
C           END DO                                                        *
C       (NOTE: THE STATION NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)  *
C                                                                         *
C                                                                         *
C                                                                         *
C                                                                         *
C    -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 63                *
C                                                                         *
C        RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT   *
C             DESCRIPTIONS                                                *
C        NDSETSE,NP,DT*NSPOOLGE,NSPOOLGE,IRTYPE                           *
C             NDSETSE = THE NUMBER OF DATA SETS TO BE SPOOLED TO UNIT 63  *
```

```
C              NP = THE NUMBER OF NODE POINTS WITHIN EACH DATA SET              *
C              DT*NSPOOLGE = THE TIME INCREMENT BETWEEN DATA SETS (SEC)         *
C              NSPOOLGE = THE TIMESTEP INCREMENT BETWEEN DATA SETS              *
C              IRTYPE = 1  (THE RECORD TYPE)                                    *
C         TIME,IT ; TIME (IN SECONDS) AND TIME STEP AT WHICH RESULTS AT ALL     *
C                   NODES ARE PROVIDED                                         *
C         DO I=1,NP                                                            *
C            I,ETA2(I);    NODE NUMBER AND SURFACE ELEVATION FOR ALL NODES    *
C                          IN THE DOMAIN.                                     *
C            END DO                                                           *
C         (NOTE: THE NODE NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)       *
C                                                                            *
C                                                                            *
C                                                                            *
C                                                                            *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 64                  *
C                                                                            *
C         RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT     *
C              DESCRIPTIONS                                                  *
C         NDSETSV,NP,DT*NSPOOLGV,NSPOOLGV,IRTYPE                             *
C              NDSETSV - THE # OF DATA SETS TO BE SPOOLED TO UNIT 64         *
C              NP      - THE # OF NODE POINTS WITHIN EACH DATA SET           *
C              DT*NSPOOLGV - THE TIME INCREMENT BETWEEN DATA SETS (SEC)      *
C              NSPOOLGV - THE TIMESTEP INCREMENT BETWEEN DATA SETS           *
C              IRTYPE = 2  (THE RECORD TYPE)                                 *
C         TIME,IT ; TIME(IN SECONDS) AND TIME STEP AT WHICH RESULTS AT ALL   *
C                   NODES ARE PROVIDED                                       *
C          DO I=1,NP                                                         *
C            I,UU2(I),VV2(I); NODE NUMBER, X VELOCITY AND Y VELOCITY         *
C                             FOR ALL NODES IN THE DOMAIN.                   *
C            END DO                                                          *
C         (NOTE: THE NODE NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)      *
C                                                                            *
C                                                                            *
C                                                                            *
C                                                                            *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 71                  *
C                                                                            *
C         RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT     *
C              DESCRIPTIONS                                                  *
C         NTRSPC, NSTAC, DT*NSPOOLC, NSPOOLC, IRTYPE                         *
C              NTRSPC = THE NUMBER OF DATA SETS TO BE SPOOLED TO UNIT 71     *
C              NSTAC  = THE NUMBER OF STATIONS WITHIN EACH DATA SET          *
C              DT*NSPOOLC = THE TIME INCREMENT BETWEEN DATA SETS (SEC)       *
C              NSPOOLC = THE TIMESTEP INCREMENT BETWEEN DATA SETS            *
C              IRTYPE = 1  (THE RECORD TYPE)                                 *
C         TIME,IT ;  TIME (IN SECONDS) AND TIME STEP AT WHICH RESULTS AT     *
C                    CONCENTRATION STATIONS ARE PROVIDED                     *
C         DO I=1,NSTAC                                                       *
C            I,CC00(I);    STATION NUMBER AND CONCENTRATION AT NSTAC        *
C                          CONCENTRATION RECORDING STATIONS.                 *
C            END DO                                                          *
C        (NOTE: THE STATION NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)    *
C                                                                            *
C                                                                            *
C                                                                            *
C                                                                            *
C     -  DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 73                  *
C                                                                            *
C         RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT     *
C              DESCRIPTIONS                                                  *
C         NDSETSC,NP,DT*NSPOOLGC,NSPOOLGC,IRTYPE                             *
C              NDSETSC = THE NUMBER OF DATA SETS TO BE SPOOLED TO UNIT 73    *
C              NP = THE NUMBER OF NODE POINTS WITHIN EACH DATA SET           *
C              DT*NSPOOLGC = THE TIME INCREMENT BETWEEN DATA SETS (SEC)      *
C              NSPOOLGC = THE TIMESTEP INCREMENT BETWEEN DATA SETS           *
C              IRTYPE = 1  (THE RECORD TYPE)                                 *
C         TIME,IT ; TIME (IN SECONDS) AND TIME STEP AT WHICH RESULTS AT ALL  *
C                   NODES ARE PROVIDED                                       *
C         DO I=1,NP                                                          *
```

```
C             I,C1(I);     NODE NUMBER AND CONCENTRATION FOR ALL NODES IN THE  *
C                          DOMAIN.                                             *
C               END DO                                                         *
C           (NOTE: THE NODE NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)      *
C                                                                              *
C                                                                              *
C                                                                              *
C                                                                              *
C                                                                              *
C     -   DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNIT 74                   *
C                                                                              *
C           RUNDES,RUNID,AGRID ; ALPHANUMERIC DESCRIPTORS : SEE INPUT UNIT     *
C                DESCRIPTIONS                                                  *
C           NDSETSW,NP,DT*NSPOOLGW,NSPOOLGW,IRTYPE                             *
C                NDSETSW - THE # OF DATA SETS TO BE SPOOLED TO UNIT 74         *
C                NP      - THE # OF NODE POINTS WITHIN EACH DATA SET           *
C                DT*NSPOOLGW - THE TIME INCREMENT BETWEEN DATA SETS (SEC)      *
C                NSPOOLGW - THE TIMESTEP INCREMENT BETWEEN DATA SETS           *
C                IRTYPE = 2  (THE RECORD TYPE)                                 *
C           TIME,IT ; TIME(IN SECONDS) AND TIME STEP AT WHICH RESULTS AT ALL   *
C                     NODES ARE PROVIDED                                       *
C            DO I=1,NP                                                         *
C              I,WSX2(I),WSY2(I); NODE NUMBER, X WIND STRESS AND Y WIND STRESS *
C                                 FOR ALL NODES IN THE DOMAIN.                 *
C               END DO                                                         *
C           (NOTE: THE NODE NUMBER IS NOT INCLUDED IF THE FILE IS BINARY)      *
C                                                                              *
C                                                                              *
C                                                                              *
C                                                                              *
C     -   DESCRIPTION OF OUTPUT VARIABLES WRITTEN TO UNITS 67 AND 68 (BINARY)  *
C                                                                              *
C           IM - TYPE OF MODEL RUN                                             *
C           TIME,IT - TIME (IN SECONDS) AND TIME STEP OF HOT START FILE        *
C           DO I=1,NP                                                          *
C             ETA1(I)                                                          *
C             ETA2(I)                                                          *
C             UU1(I)                                                           *
C             VV1(I)                                                           *
C             IF(IM.EQ.10) CH1(I)                                              *
C             NNODECODE(I)                                                     *
C             END DO                                                           *
C           IESTP,NSCOUE                                                       *
C           IVSTP,NSCOUV                                                       *
C           ICSTP,NSCOUC                                                       *
C           IGEP,NSCOUGE                                                       *
C           IGVP,NSCOUGV                                                       *
C           IGCP,NSCOUGC                                                       *
C           IGWP,NSCOUGW                                                       *
C           IF(IHARIND.EQ.1)                                                   *
C             ICHA                                                             *
C             NZ,NF,MM,NP,NSTAE,NSTAV,NHASE,NHASV,NHAGE,NHAGV,ICALL            *
C             NHARFR-NF                                                        *
C             (HAFNAM(I),HAFREQ(I),HAFF(I),HAFACE(I)  I=1,NHARFR)              *
C             TIMEUD                                                           *
C             (A(I,J) I=1,MM, J=1,MM)                                          *
C             IF(NHASE.EQ.1) (STAELV(I,N) I=1,MM N=1,NSTAE)                    *
C             IF(NHASV.EQ.1) (STAULV(I,N),STAVLV(I,N) I=1,MM N=1,NSTAV)        *
C             IF(NHAGE.EQ.1) (GLOELV(I,N) I=1,MM N=1,NGLOE)                    *
C             IF(NHAGV.EQ.1) (GLOULV(I,N),GLOVLV(I,N) I=1,MM N=1,NGLOV)        *
C             IF(FMV.GT.0.)                                                    *
C               NTSTEPS                                                        *
C               (ELAV(I),ELVA(I), I=1,NP)                                      *
C               (XVELAV(I),YVELAV(I),XVELVA(I),YVELVA(I) I=1,NP)               *
C               ENDIF                                                          *
C             ENDIF                                                            *
C                                                                              *
C                                                                              *
C*******************************************************************************
C                                                                              *
C                                                                              *
```

```
C     -  PARAMETERS WHICH MUST BE SET WITHIN THE MAIN CODE AND THE            *
C        SUBROUTINES TO CONTROL THE DIMENSIONING OF ARRAYS ARE AS FOLLOWS:   *
C                                                                            *
C        NOTE: THESE VALUES ARE SET AUTOMATICALLY BY THE PREPROCESSING       *
C              PROGRAM ADCSETUP                                              *
C                                                                            *
C        MNP = MAXIMUM NUMBER OF NODAL POINTS                                *
C        MNWP = 1 IF NO METEOROLOGIC FORCING, = MNP IF METEOROLOGIC FORCING  *
C        MNE = MAXIMUM NUMBER OF ELEMENTS                                    *
C        MNBW = MAXIMUM HALF BAND WIDTH OF WAVE EQUATION MATRIX              *
C               (IF AN ITERATIVE SOLVER IS USED, MNBW IS NOT USED FOR        *
C                ANYTHING, THEREFORE IT CAN BE SET TO ANY VALUE OR REMOVED   *
C                FROM THE PARAMETER LIST)                                    *
C        MNBOU = MAXIMUM NUMBER OF FLOW BOUNDARY SEGMENTS INCLUDING          *
C                 EXTERNAL AND INTERNAL SEGMENTS                             *
C        MNVEL = MAXIMUM TOTAL NUMBER OF FLOW BOUNDARY NODES +1              *
C        MNOPE = MAXIMUM NUMBER OF ELEVATION BOUNDARY SEGMENTS               *
C        MNETA = MAXIMUM TOTAL NUMBER OF SPECIFIED ELEVATION BOUNDARY NODES  *
C        MNBFR = MAXIMUM NUMBER OF PERIODIC SPECIFIED ELEVATION BOUNDARY     *
C                FORCING CONSTITUENTS                                        *
C        MNFFR = MAXIMUM NUMBER OF PERIODIC SPECIFIED NORMAL FLOW BOUNDARY   *
C                FORCING CONSTITUENTS                                        *
C        MNTIF = MAXIMUM NUMBER OF TIDAL POTENTIAL CONSTITUENTS              *
C        MNSTAE = MAXIMUM NUMBER OF ELEVATION RECORDING STATIONS             *
C        MNSTAV = MAXIMUM NUMBER OF VELOCITY RECORDING STATIONS              *
C        MNSTAC = MAXIMUM NUMBER OF CONCENTRATION RECORDING STATIONS         *
C        MNHARF = MAXIMUM NUMBER OF CONSTITUENTS TO INCLUDE IN HARMONIC      *
C                 ANALYSIS OF MODEL RESULTS (NOTE: TO RUN ON THE CRAY, IF    *
C                 4*(MNHARF/4) < MNHARF, MNHARF SHOULD BE ROUNDED UP TO THE  *
C                 NEXT LARGER INTEGER THAT IS EVENLY DIVISIBLE BY 4)         *
C        MNEI    = 1+MAXIMUM NUMBER OF NODES CONNECTED TO ANY ONE NODE IN THE*
C                 FINITE ELEMENT GRID                                        *
C                                                                            *
C                                                                            *
C****************************************************************************


CUSER...
CUSER...SET PARAMETER STATEMENTS
CUSER....THE PARAMETERS HAVE TO BE RE-SET PRIOR TO EXECUTION TO INSURE
CUSER....SUFFICIENT SPACE HAS BEEN ALLOCATED FOR A SPECIFIC PROBLEM
CUSER....PARAMETER STATEMENTS MUST ALSO BE ADJUSTED IN THE SUBROUTINES
CUSER....AND FUNCTION STATEMENTS
CUSER...
      PARAMETER(MNP=         1082)
      PARAMETER(MNWP=        1082)
      PARAMETER(MNE=         1894)
      PARAMETER(MNBW=            1)
      PARAMETER(MNBFR=           1)
      PARAMETER(MNFFR=           1)
      PARAMETER(MNTIF=           1)
      PARAMETER(MNBOU=           3)
      PARAMETER(MNVEL=         247)
      PARAMETER(MNOPE=           2)
      PARAMETER(MNETA=          28)
      PARAMETER(MNSTAE=            1)
      PARAMETER(MNSTAV=            1)
      PARAMETER(MNSTAC=            1)
      PARAMETER(MNHARF=            4)
      PARAMETER(MNEI=            9)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

C...
C...DIMENSION ALL ARRAYS AND DEFINE COMMON BLOCKS
C...
      DIMENSION SLAM(MNP),SFEA(MNP),X(MNP),Y(MNP)
      DIMENSION NM(MNE,3),DP(MNP),SFAC(MNP)
      DIMENSION ETAS(MNP),ETA1(MNP),ETA2(MNP)
```

```
      DIMENSION UU1(MNP),UU2(MNP),VV1(MNP),VV2(MNP)
      DIMENSION QU(MNP),QV(MNP),QW(MNP)
      DIMENSION FRIC(MNP),CORIF(MNP),EVM(MNP)
      DIMENSION NNODECODE(MNP),NODECODE(MNP),NODEREP(MNP)
      DIMENSION NEITAB(MNP,MNEI),NNEIGH(MNP),MJU(MNP)
      DIMENSION TPK(MNTIF),AMIGT(MNTIF),FFT(MNTIF)
      DIMENSION FACET(MNTIF),PERT(MNTIF),ETRF(MNTIF)
      DIMENSION AMIG(MNBFR),FF(MNBFR),FACE(MNBFR),PER(MNBFR)
      DIMENSION EMO(MNBFR,MNETA),EFA(MNBFR,MNETA)
      DIMENSION NVDLL(MNOPE),NBDV(MNOPE,MNETA),NBD(MNETA)
      DIMENSION NVELL(MNBOU),NBVV(MNBOU,0:MNVEL)
      DIMENSION NBV(MNVEL),LBCODEI(MNVEL)
      DIMENSION QN0(MNVEL),QN1(MNVEL),QN2(MNVEL)
      DIMENSION BNDLEN2O3(MNVEL)
      DIMENSION ME2GW(MNVEL),CSII(MNVEL),SIII(MNVEL)
      DIMENSION FAMIG(MNFFR),FFF(MNFFR),FFACE(MNFFR),FPER(MNFFR)
      DIMENSION BARLANHT(MNVEL),BARLANCFSP(MNVEL)
      DIMENSION BARLANHTR(MNVEL),BARLANCFSPR(MNVEL)
      DIMENSION BARINHT(MNVEL),BARINCFSB(MNVEL),BARINCFSP(MNVEL)
      DIMENSION IBCONN(MNVEL)
      DIMENSION BARINHTR(MNVEL),BARINCFSBR(MNVEL),BARINCFSPR(MNVEL)
      DIMENSION IBCONNR(MNVEL),NTRAN1(MNVEL),NTRAN2(MNVEL)
      DIMENSION BTRAN3(MNVEL),BTRAN4(MNVEL),BTRAN5(MNVEL)
      DIMENSION RBARWL1AVG(MNVEL),RBARWL2AVG(MNVEL)
      DIMENSION NIBNODECODE(MNP)
      DIMENSION QNAM(MNFFR,MNVEL),QNPH(MNFFR,MNVEL)
      DIMENSION QNIN1(MNVEL),QNIN2(MNVEL)
      DIMENSION LBCODE(MNP),CSI(MNP),SII(MNP)
      DIMENSION XEL(MNSTAE),YEL(MNSTAE),SLEL(MNSTAE),SFEL(MNSTAE)
      DIMENSION NNE(MNSTAE),ET00(MNSTAE)
      DIMENSION STAIE1(MNSTAE),STAIE2(MNSTAE),STAIE3(MNSTAE)
      DIMENSION XEV(MNSTAV),YEV(MNSTAV),SLEV(MNSTAV),SFEV(MNSTAV)
      DIMENSION NNV(MNSTAV),UU00(MNSTAV),VV00(MNSTAV)
      DIMENSION STAIV1(MNSTAV),STAIV2(MNSTAV),STAIV3(MNSTAV)
      DIMENSION XEC(MNSTAC),YEC(MNSTAC),SLEC(MNSTAC),SFEC(MNSTAC)
      DIMENSION NNC(MNSTAC),CC00(MNSTAC)
      DIMENSION STAIC1(MNSTAC),STAIC2(MNSTAC),STAIC3(MNSTAC)
      DIMENSION HAFREQ(MNHARF),HAFF(MNHARF),HAFACE(MNHARF)
      DIMENSION CH1(MNP),QB(MNP),QA(MNP),SOURSIN(MNP),EVC(MNP)
      DIMENSION WSX1(MNWP),WSY1(MNWP),PR1(MNWP)
      DIMENSION WSX2(MNWP),WSY2(MNWP),PR2(MNWP)
      DIMENSION WVNX1(MNWP),WVNY1(MNWP),PRN1(MNWP)
      DIMENSION WVNX2(MNWP),WVNY2(MNWP),PRN2(MNWP)

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
      DIMENSION AUV11(MNP),AUV12(MNP),TK(MNP)
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c     DIMENSION DUU1(MNP),DUV1(MNP),DVV1(MNP),BSX1(MNP),BSY1(MNP)
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C3DDSS.
C3DDSS.UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE
C3DDSS.COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c     DIMENSION AUV11(MNP),AUV12(MNP),AUV13(MNP),AUV14(MNP)
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.
```

```
CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
      DIMENSION TIP1(MNP),TIP2(MNP)
      DIMENSION SALTAMP(MNTIF,MNP),SALTPHA(MNTIF,MNP)
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...
      DIMENSION QTEMA(MNE,3),QTEMB(MNE,3)
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CVEC....

CSOLIT...
CSOLIT...UNCOMMENT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLIT...COMMENT OUT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLIT...OR THE DIAGONAL MATRIX SOLVER
CSOLIT...
      DIMENSION OBCCOEF(MNETA,MNEI-1),COEF(MNP,MNEI)
      DIMENSION IWKSP(3*MNP),WKSP(6*MNP+400),IPARM(12),RPARM(12)
CSOLIT...
CSOLIT...END OF ITERATIVE SOLVER SECTION (MORE FOLLOW BELOW)
CSOLIT...

CSOLDIR...
CSOLDIR...UNCOMMENT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLDIR...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIR...OR THE DIAGONAL MATRIX SOLVER
CSOLDIR...
c     DIMENSION OBCCOEF(MNETA,MNEI-1),COEF(MNP,MNEI)
c     DIMENSION ABD(3*MNBW+1,MNP),IPV(MNP),ZX(MNP)
CSOLDIR...
CSOLDIR...END OF DIRECT SOLVER STATEMENTS (MORE FOLLOW BELOW)
CSOLDIR...

CSOLDIA...
CSOLDIA...UNCOMMENT THE FOLLOWING LINES TO USE THE DIAGONAL SOLVER ON THE
CSOLDIA...GWCE MATRIX.  THIS IS ONLY FOR A FULLY EXPLICIT AND LUMPED GWCE
c     DIMENSION COEF(MNP)
CSOLDIA...
CSOLDIA...END OF DIAGONAL SOLVER STATEMENTS (MORE FOLLOW BELOW)
CSOLDIA...

C...
C...DECLARE REAL*8 AND CHAR VARIABLES, DEFINE COMMON BLOCKS AND EQUIVALENCES
C...
      REAL*8 STATIM,REFTIM,TIME,DTDP
      REAL*8 AVGXY,DIF1R,DIF2R,DIF3R
      REAL*8 AREAS,AEMIN,AE,AA,A1,A2,A3,X1,X2,X3,X4,Y1,Y2,Y3,Y4
      REAL*8 FDX1,FDX2,FDX3,FDY1,FDY2,FDY3
      REAL*8 FDX1OA,FDX2OA,FDX3OA,FDY1OA,FDY2OA,FDY3OA,AREAIE
      REAL*8 DDX1,DDX2,DDX3,DDY1,DDY2,DDY3
      REAL*8 DXX11,DXX12,DXX13,DXX21,DXX22,DXX23,DXX31,DXX32,DXX33
      REAL*8 DYY11,DYY12,DYY13,DYY21,DYY22,DYY23,DYY31,DYY32,DYY33
      REAL*8 DXY11,DXY12,DXY13,DXY21,DXY22,DXY23,DXY31,DXY32,DXY33
      REAL*8 PI,DEG2RAD,RAD2DEG
      REAL*8 AMIG,PER,AMIGT,PERT,FAMIG,FPER
      REAL*8 WREFTIM,WTIMED,WTIME2,WTIME1,WTIMINC,QTIME1,QTIME2,FTIMINC
      CHARACTER*32 RUNDES
      CHARACTER*24 RUNID,AGRID,AFRIC
      CHARACTER*4 RDES4(8),RID4(6),AID4(6)
      CHARACTER*8 RDES8(4),RID8(3),AID8(3)
      CHARACTER*10 ALPHA,HAFNAM(MNHARF)
```

```
      CHARACTER*5 TIPOTAG(MNTIF),BOUNTAG(MNBFR),FBOUNTAG(MNFFR)
      COMMON/LSQFREQS/ NHARFR,HAFREQ,HAFF,HAFACE,HAFNAM
      COMMON/HGRID/ X,Y,NM,DP,SFAC
      COMMON/ELEAREA/ AREAS(MNE)

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      COMMON/MEANSQ/ TIMEBEG,NTSTEPS,DT,FMV,ITMV
      COMMON/MEANSQE/ ELAV,ELVA
      COMMON/MEANSQV/ XVELAV,YVELAV,XVELVA,YVELVA
      REAL XVELAV(MNP),YVELAV(MNP),XVELVA(MNP),YVELVA(MNP)
      REAL ELAV(MNP),ELVA(MNP),DT,FMV
      REAL*8 TIMEBEG
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c     COMMON /EXTMODE1/ WSX1,WSY1,WSX2,WSY2,BSX1,BSY1
c     COMMON /EXTMODE2/ UU2,VV2,DUU1,DUV1,DVV1
c     COMMON /EXTMODE3/ ETA1,ETA2,CORIF,QU,QV,LBCODE,CSI,SII
c     COMMON /EXTMODE5/ NP,NOLICA,NOLIFA,NSCREEN,IHOT,ICS
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C3DDSS.
C3DDSS.UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE
C3DDSS.COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c     COMMON /EXTMODE4/ AUV11,AUV12,AUV13,AUV14
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

      EQUIVALENCE (RDES4(1),RDES8(1),RUNDES), (RID4(1),RID8(1),RUNID),
     &            (AID4(1),AID8(1),AGRID)

C...
C...THE NEXT SECTION RELATES TO MACHINE DEPENDENT PRECISION
C...

CMACHSUN...
CMACHSUN....UNCOMMENT THE ABRUPT UNDERFLOW LINE FOR SUN 4 AND SPARC COMPUTERS
CMACHSUN....COMMENTED OUT THIS LINE FOR OTHER COMPUTERS
CMACHSUN...
c     CALL ABRUPT_UNDERFLOW()
CMACHSUN...
CMACHSUN....END OF SUN SPECIFIC STATEMENTS
CMACHSUN...

CMACH32...
CMACH32...UNCOMMENT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH32...32 BIT MACHINE
CMACH32...COMMENT OUT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH32...64 BIT MACHINE OR DOUBLE PRECISION ON A 32 BIT MACHINE
CMACH32...NOTE: THESE LINES SET PRECISION LIMIT FOR PROCESSING INPUT DATA
CMACH32...       AND SET RECORD LENGTH FOR BINARY OUTPUT
CMACH32...
c     NBYTE=4
CMACH32...
CMACH32...END OF 32 BIT MACHINE SPECIFIC STATEMENTS (MORE FOLLOW BELOW)
CMACH32...

CMACH64...
```

```
CMACH64...UNCOMMENT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH64...64 BIT MACHINE OR DOUBLE PRECISION ON A 32 BIT MACHINE
CMACH64...COMMENT OUT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH64...32 BIT MACHINE
CMACH64...NOTE: THESE LINES SET PRECISION LIMIT FOR PROCESSING INPUT DATA
CMACH64...      AND SET RECORD LENGTH FOR BINARY OUTPUT
CMACH64...
      NBYTE=8
CMACH64...
CMACH64...END OF 64 BIT MACHINE SPECIFIC STATEMENTS (MORE FOLLOW BELOW)
CMACH64...

C...
C...SET THE MAXIMUM NUMBER OF DIGITS OF PRECISION THE GRID CAN BE EXPECTED TO HAVE
C...NOTE: IF THE GRID WAS BUILT ON A 32 BIT COMPUTER, IT SHOULD BE ACCURATE TO ABOUT
C...7 DIGITS.  THUS, IF THE NODAL SPACING REQUIRES MORE THAN 5 DIGITS OF PRECISION,
C...IT IS UNLIKELY THAT THE MODEL RESULTS WILL BE TRUSTWORTHY.
C...
      NPREC=5

C...
C...DEFINE PI AND DEG TO RAD CONVERSIONS
C...
      PI=3.141592653589793D0
      DEG2RAD = PI/180.D0
      RAD2DEG = 180.D0/PI

C...
C...OPEN STATEMENT FOR UNIT 16 OUTPUT FILE
C...
      OPEN(16,FILE='fort.16')

C...
C...GENERAL PURPOSE FORMAT STATEMENTS
C...
 1112 FORMAT(/,1X,79('_'))
 9973 FORMAT(/,1X,'PROGRAM WILL NOT OVERRIDE INPUT',
     &  //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)

C...
C...PRINT OUT HEADER FOR OUTPUT INCLUDING VERSION NUMBER AND COPYRIGHT
C...
      WRITE(16,1112)
      WRITE(16,1112)
      WRITE(16,1114)
      WRITE(16,1112)
      WRITE(6,1112)
      WRITE(6,1114)
      WRITE(6,1112)
 1114 FORMAT(//,19X,'PROGRAM ADCIRC   VERSION 31.06  ',
     &  //,5X,'AN ADVANCED CIRCULATION MODEL FOR SHELVES, COASTAL ',
     &        'SEAS AND ESTUARIES',
     &  ///,7X,'-  DEVELOPED BY',
     &  //,10X,'R.A. LUETTICH, JR',
     &  /,12X,'UNIVERSITY OF NORTH CAROLINA AT CHAPEL HILL',
     &  /,12X,'INSTITUTE OF MARINE SCIENCES',
     &  //,10X,'J.J. WESTERINK ',
     &  /,12X,'DEPARTMENT OF CIVIL ENGINEERING AND GEOLOGICAL SCIENCES',
     &  /,12X,'UNIVERSITY OF NOTRE DAME',
     &  ///,7X,'-  THE ADCIRC SOURCE CODE IS COPYRIGHTED BY',
     &  //,10X,'R.A. LUETTICH, JR. AND J.J. WESTERINK, 1994-96',
     &  //,7X,'NO PART OF THIS CODE MAY BE REPRODUCED OR REDISTRIBUTED',
     &  /,10X,'WITHOUT THE WRITTEN PERMISSION OF THE AUTHORS',//)

C...
C...WRITE OUT HEADER INFORMATION DESCRIBING HOW THE CODE HAS BE SET UP
C...
      WRITE(16,1210)
 1210 FORMAT(//,1X,'THE ADCIRC SOURCE CODE HAS BEEN CONFIGURED ',
     &             'BY THE PREPROCESSOR AS FOLLOWS:',/)
```

```
C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
      WRITE(16,*) '       - 2D DEPTH INTEGRATED MODEL OPTION'
C2DDI...
C2DDI....END OF 2DDI SPECIFIC STATEMENTS
C2DDI...

C3DVS..
C3DVS..UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D VS MODE.
C3DVS..COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D DSS MODE.
C3DVS..
c     WRITE(16,*) '       - 3D VS MODEL OPTION'
C3DVS..
C3DVS..END OF 3D VS STATEMENTS (MORE FOLLOW BELOW)
C3DVS..

C3DDSS.
C3DDSS.UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE
C3DDSS.COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c     WRITE(16,*) '       - 3D DSS MODEL OPTION'
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
      WRITE(16,*) '       - CODE SETUP TO ALLOW WIND AND PRESSURE ',
     &            'FORCING'
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND...

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
      WRITE(16,*) '        - CODE SETUP TO ALLOW TIDAL POTENTIAL FORCING'
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      WRITE(16,*) '       - CODE SETUP TO COMPUTE MEANS AND ',
     &            'VARIANCES TO CHECK HARMONIC ANALYSIS'
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

CMACHSUN...
CMACHSUN....UNCOMMENT THE FOLLOWING LINES FOR SUN 4 AND SPARC COMPUTERS
CMACHSUN....COMMENT OUT THE FOLLOWING LINES FOR OTHER COMPUTERS
CMACHSUN...
c     WRITE(16,*) '       - CODE SETUP TO RUN ON SUN 4 OR SPARC ',
c    &            'COMPUTERS'
CMACHSUN...
CMACHSUN....END OF SUN SPECIFIC STATEMENTS
CMACHSUN...

CMACH32...
CMACH32...UNCOMMENT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH32...32 BIT MACHINE
```

```
CMACH32...COMMENT OUT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH32...64 BIT MACHINE OR DOUBLE PRECISION ON A 32 BIT MACHINE
CMACH32...
c     WRITE(16,*) '       - CODE SETUP TO RUN ON A 32 BIT COMPUTER'
CMACH32...
CMACH32...END OF 32 BIT MACHINE SPECIFIC STATEMENTS (MORE FOLLOW BELOW)
CMACH32...

CMACH64...
CMACH64...UNCOMMENT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH64...64 BIT MACHINE OR DOUBLE PRECISION ON A 32 BIT MACHINE
CMACH64...COMMENT OUT THE FOLLOWING LINES FOR SINGLE PRECISION ON A
CMACH64...32 BIT MACHINE
CMACH64...
      WRITE(16,*) '       - CODE SETUP TO RUN ON A 64 BIT COMPUTER'
CMACH64...
CMACH64...END OF 64 BIT MACHINE SPECIFIC STATEMENTS (MORE FOLLOW BELOW)
CMACH64...

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...
      WRITE(16,*) '       - CODE OPTIMIZED FOR A VECTOR COMPUTER'
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CVEC....

CSCA...
CSCA...UNCOMMENT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER.
CSCA...COMMENT OUT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER.
CSCA...
c     WRITE(16,*) '       - CODE OPTIMIZED FOR A SCALAR COMPUTER'
CSCA...
CSCA...END OF SCALAR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CSCA....

      WRITE(16,*) '       - NONVECTORIZABLE PARTS OF CODE ',
     &            'OPTIMIZED FOR MEMORY'

CSOLIT...
CSOLIT...UNCOMMENT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLIT...COMMENT OUT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLIT...OR THE DIAGONAL MATRIX SOLVER
CSOLIT...
      WRITE(16,*) '       - CODE CONFIGURED FOR AN ITERATIVE GWCE SOLVER'
CSOLIT...
CSOLIT...END OF ITERATIVE SOLVER SECTION (MORE FOLLOW BELOW)
CSOLIT...

CSOLDIR...
CSOLDIR...UNCOMMENT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLDIR...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIR...OR THE DIAGONAL MATRIX SOLVER
CSOLDIR...
c     WRITE(16,*) '       - CODE CONFIGURED FOR A DIRECT GWCE SOLVER'
CSOLDIR...
CSOLDIR...END OF DIRECT SOLVER STATEMENTS (MORE FOLLOW BELOW)
CSOLDIR...

CSOLDIA...
CSOLDIA...UNCOMMENT THE FOLLOWING LINES TO USE THE DIAGONAL MATRIX SOLVER
CSOLDIA...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIA...OR THE DIRECT MATRIX SOLVER
CSOLDIA...
c     WRITE(16,*) '       - CODE CONFIGURED FOR A DIAGONAL GWCE SOLVER'
CSOLDIA...
CSOLDIA...END OF DIAGONAL SOLVER STATEMENTS (MORE FOLLOW BELOW)
CSOLDIA...
```

```
      WRITE(16,1112)

C...
C...OPEN STATEMENT FOR UNIT 14 AND 15 INPUT FILES
C...
      OPEN(14,FILE='fort.14')
      OPEN(15,FILE='fort.15')

C...
C...INPUT FROM UNIT 15 AND OUTPUT TO UNIT 16 RUN DESCRIPTION AND RUN
C....IDENTIFICATION
C...
      READ(15,'(A32)') RUNDES
      READ(15,'(A24)') RUNID
      WRITE(16,1) RUNDES
  1   FORMAT(//,1X,'RUN DESCRIPTION : ',A32)
      WRITE(16,209) RUNID
209   FORMAT(/,1X,'RUN IDENTIFICATION : ',A24)

C...
C...READ AND PROCESS NFOVER - NONFATAL ERROR OVERRIDE OPTION
C...
      READ(15,*) NFOVER
      WRITE(16,1112)
      WRITE(16,1250)
 1250 FORMAT(//,1X,'GENERAL RUN INFORMATION',/)
      IF(NFOVER.EQ.1) THEN
        WRITE(16,1951) NFOVER
 1951   FORMAT(5X,'NFOVER = ',I2,
     &    /,9X,'NON-FATAL ERRORS WILL BE CORRECTED AND EXECUTION WILL',
     &         ' CONTINUE',
     &    /,9X,'IF NON-FATAL ERRORS ARE DETECTED',/)
        ELSE
        WRITE(16,1952) NFOVER
 1952   FORMAT(/,5X,'NFOVER = ',I3,
     &    /,9X,'NON-FATAL ERRORS WILL STOP EXECUTION ',/)
        ENDIF

C...
C...READ AND PROCESS NABOUT - ABBREVIATED UNIT 16 OUTPUT OPTION
C...
      READ(15,*) NABOUT
      IF(NABOUT.EQ.1) THEN
        WRITE(16,3501) NABOUT
 3501   FORMAT(5X,'NABOUT = ',I2,
     &    /,9X,'ABREVIATED OUTPUT WILL BE PROVIDED TO UNIT 16',
     &    /,9X,'UNIT 14, 21, 22 INPUT DATA WILL NOT BE ECHO PRINTED',/)
        ELSE
        WRITE(16,3502) NABOUT
 3502   FORMAT(/,5X,'NABOUT = ',I3,
     &    /,9X,'DETAILED OUTPUT WILL BE PROVIDED TO UNIT 16',
     &    /,9X,'UNIT 14, 15, 21, 22 INPUT DATA WILL BE ECHO PRINTED',/)
        ENDIF

C...
C...READ AND PROCESS NSCREEN - SCREEN OUTPUT OPTION
C...
      READ(15,*) NSCREEN
      IF(NSCREEN.EQ.1) THEN
        WRITE(16,3561) NSCREEN
3561    FORMAT(5X,'NSCREEN = ',I2,
     &    /,9X,'SCREEN OUTPUT WILL BE PROVIDED TO UNIT 6',/)
        ELSE
        WRITE(16,3562) NSCREEN
3562    FORMAT(/,5X,'NSCREEN = ',I3,
     &    /,9X,'SCREEN OUTPUT WILL NOT BE PROVIDED TO UNIT 6',/)
        ENDIF

C...
```

```
C...READ AND PROCESS IHOT - HOT START OPTION
C...
      READ(15,*) IHOT
      IF((IHOT.NE.0).AND.(IHOT.NE.67).AND.(IHOT.NE.68)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9732)
        WRITE(16,9732)
9732    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF IHOT (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    /,1X,'PLEASE CHECK YOUR INPUT')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9733)
          WRITE(16,9733)
9733      FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &               ' AND SET IHOT EQUAL TO 0 ',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          IHOT=0
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
       IF(IHOT.NE.0) THEN
        WRITE(16,97361) IHOT
97361   FORMAT(/,5X,'THE MODEL WILL BE HOT STARTED ',
     &              'USING INFORMATION ON UNIT ',I2)
        ELSE
        WRITE(16,97371)
97371   FORMAT(//,5X,'THE MODEL WILL BE COLD STARTED')
        ENDIF

C...
C...READ AND PROCESS ICS - CARTESIAN/SPHERICAL COORDINATE OPTION
C...
      READ(15,*) ICS
      IF((ICS.NE.1).AND.(ICS.NE.2)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9734)
        WRITE(16,9734)
9734    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF ICS (A UNIT 15 INPUT PARAMETER)',
     &          ' IS NOT AN ALLOWABLE VALUE',
     &    /,1X,'PLEASE CHECK YOUR INPUT')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9735)
          WRITE(16,9735)
9735      FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &               ' AND SET ICS EQUAL TO 1 ',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          ICS=1
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
       IF(ICS.EQ.1) THEN
        WRITE(16,9736) ICS
9736    FORMAT(/,5X,'ICS = ',I2,
     &    /,9X,'GOVERNING EQUATIONS ARE BASED ON STANDARD ',
     &         'CARTESIAN COORDINATES')
        ELSE
        WRITE(16,9737) ICS
9737    FORMAT(/,5X,'ICS = ',I2,
     &    /,9X,'GOVERNING EQUATIONS ARE BASED ON SPHERICAL ',
     &         'COORDINATES ',
     &    /,9X,'MAPPED USING A CPP PROJECTION')
        ENDIF
```

```
C...
C...READ AND PROCESS IM - 2D/3D MODEL OPTION
C...
      READ(15,*) IM

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
      IF((IM.NE.0).AND.(IM.NE.10)) THEN
C2DDI...
C2DDI....END OF 2DDI SPECIFIC STATEMENTS
C2DDI...

C3DVS..
C3DVS..UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D VS MODE.
C3DVS..COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D DSS MODE.
C3DVS..
c     IF(IM.NE.1) THEN
C3DVS..
C3DVS..END OF 3D VS STATEMENTS (MORE FOLLOW BELOW)
C3DVS..

C3DDSS.
C3DDSS.UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE
C3DDSS.COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c     IF(IM.NE.2) THEN
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

        WRITE(6,9970) IM
        WRITE(16,9970) IM
9970    FORMAT(/,1X,'IM = ',I3,' INCORRECT MODEL TYPE SPECIFIED ',
     &          'IN THE UNIT 15 INPUT FILE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
        STOP
        ENDIF

C...
C...READ AND PROCESS NOLIBF - NONLINEAR BOTTOM FRICTION OPTION
C...
      READ(15,*) NOLIBF
      IF((NOLIBF.LT.0).OR.(NOLIBF.GT.2)) THEN
         IF(NSCREEN.EQ.1) WRITE(6,99711)
         WRITE(16,99711)
99711    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NOLIBF (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
         STOP
         ENDIF
      WRITE(16,9845) NOLIBF
9845  FORMAT(/,5X,'NOLIBF = ',I3)
      IF(NOLIBF.EQ.0) WRITE(16,2050)
2050  FORMAT(9X,'THE MODEL WILL USE LINEAR BOTTOM FRICTION')
      IF(NOLIBF.EQ.1) WRITE(16,2051)
2051  FORMAT(9X,'THE MODEL WILL USE STANDARD QUADRATIC BOTTOM FRICTION')
      IF(NOLIBF.EQ.2) WRITE(16,2052)
2052  FORMAT(9X,'THE MODEL WILL USE STANDARD QUADRATIC BOTTOM FRICTION',
     &          'IN DEEP WATER ',
     &    /,9X,'AND A FRICTION FACTOR THAT INCREASES AS THE DEPTH ',
     &          'DECREASES IN SHALLOW WATER')

C...
C...READ AND PROCESS NOLIFA - NONLINEAR FINITE AMPLITUDE OPTION
C...
```

```
      READ(15,*) NOLIFA
      IF((NOLIFA.LT.0).OR.(NOLIFA.GT.2)) THEN
         IF(NSCREEN.EQ.1) WRITE(6,99712)
         WRITE(16,99712)
99712    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                  'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NOLIFA (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
         STOP
         ENDIF
       WRITE(16,9846) NOLIFA
9846   FORMAT(/,5X,'NOLIFA = ',I3)
       IF(NOLIFA.EQ.0) WRITE(16,2053)
2053   FORMAT(9X,'THE MODEL WILL NOT USE FINITE AMPLITUDE TERMS OR ',
     &           'WETTING AND DRYING')
       IF(NOLIFA.EQ.1) WRITE(16,2054)
2054   FORMAT(9X,'THE MODEL WILL USE FINITE AMPLITUDE TERMS BUT NO ',
     &            'WETTING AND DRYING')
       IF(NOLIFA.EQ.2) WRITE(16,2049)
2049   FORMAT(9X,'THE MODEL WILL USE FINITE AMPLITUDE TERMS AND ',
     &            'WETTING AND DRYING')

C...
C...READ AND PROCESS NOLICA - ADVECTIVE TERM SPATIAL GRADIENT
C...
       READ(15,*) NOLICA
       IF((NOLICA.LT.0).OR.(NOLICA.GT.1)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99713)
          WRITE(16,99713)
99713     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                   'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NOLICA (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
          STOP
          ENDIF
       WRITE(16,9847) NOLICA
9847   FORMAT(/,5X,'NOLICA = ',I3)
       IF(NOLICA.EQ.0) WRITE(16,2055)
2055   FORMAT(9X,'THE MODEL WILL NOT USE SPATIAL DERIVATIVE ',
     &           'COMPONENTS OF THE ADVECTIVE TERMS')
       IF(NOLICA.EQ.1) WRITE(16,2056)
2056   FORMAT(9X,'THE MODEL WILL USE SPATIAL DERIVATIVE ',
     &           'COMPONENTS OF THE ADVECTIVE TERMS')

C...
C...READ AND PROCESS NOLICAT - GWCE ADVECTIVE TERM TIME DERIVATIVE
C...
       READ(15,*) NOLICAT
       IF((NOLICAT.LT.0).OR.(NOLICAT.GT.1)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99715)
          WRITE(16,99715)
99715     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                   'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NOLICAT (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
          STOP
          ENDIF
       IF((NOLIFA.GE.1).AND.(NOLICAT.EQ.0)) THEN
         IF(NSCREEN.EQ.1) WRITE(6,99755)
         WRITE(16,99755)
99755    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                  'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NOLICAT (A UNIT 15 INPUT ',
     &          'PARAMETER) IS INCONSISTENT WITH ',
     &    /,1X,'YOUR SELECTION OF NOLIFA AND MAY LEAD TO MASS ',
     &         'BALANCE PROBLEMS',
     &    /,1X,'AND/OR INCONSISTENCIES')
```

```
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99756)
          WRITE(16,99756)
99756     FORMAT(/,1X,'PROGRAM WILL CONTINUE EXECUTION WITH',
     &               ' YOUR SPECIFIED INPUT VALUE FOR NOLICAT',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
       IF((NOLICA.EQ.1).AND.(NOLICAT.EQ.0)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99755)
          WRITE(16,99755)
          IF(NFOVER.EQ.1) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99756)
            WRITE(16,99756)
            ELSE
            IF(NSCREEN.EQ.1) WRITE(6,9973)
            WRITE(16,9973)
            STOP
            ENDIF
          ENDIF
       WRITE(16,9848) NOLICAT
9848   FORMAT(/,5X,'NOLICAT = ',I3)
       IF(NOLICAT.EQ.0) WRITE(16,2057)
2057   FORMAT(9X,'THE MODEL WILL NOT USE TIME DERIVATIVE COMPONENTS ',
     &  /,9X,'OF THE ADVECTIVE TERMS IN THE GWCE')
       IF(NOLICAT.EQ.1) WRITE(16,2058)
2058   FORMAT(9X,'THE MODEL WILL USE TIME DERIVATIVE COMPONENTS ',
     &  /,9X,'OF THE ADVECTIVE TERMS IN THE GWCE')

C...
C...READ AND PROCESS NWP - SPATIALLY VARYING BOTTOM FRICTION
C...
       READ(15,*) NWP
       IF((NWP.LT.0).OR.(NWP.GT.1)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9974)
          WRITE(16,9974)
9974      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NWP (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
          STOP
          ENDIF
       IF((NWP.EQ.1).AND.(NOLIBF.EQ.2)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9975)
          WRITE(16,9975)
9975      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTIONS OF NWP AND NOLIBF (UNIT 15 INPUT ',
     &          'PARAMETERS) ARE INCOMPATABLE',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
          STOP
          ENDIF
       IF(NWP.EQ.0) THEN
          WRITE(16,231) NWP
231       FORMAT(/,5X,'NWP = ',I2,
     &    /,9X,'SPATIALLY CONSTANT BOTTOM FRICTION RELATIONS ',
     &    /,9X,'WILL BE USED THROUGHOUT THE DOMAIN')
          ELSE
          WRITE(16,232) NWP
232       FORMAT(//,5X,'NWP = ',I2,
     &    /,9X,'SPATIALLY VARYING BOTTOM FRICTION VALUES WILL BE ',
     &    /,9X,'USED; INPUT IS REQUIRED FROM UNIT 21')
          ENDIF

C...
```

```
C...READ AND PROCESS NCOR - SPATIALLY VARYING CORIOLIS PARAMETER
C...
      READ(15,*) NCOR
      IF((NCOR.NE.0).AND.(NCOR.NE.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9976)
        WRITE(16,9976)
9976    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NCOR (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    /,1X,'PLEASE CHECK YOUR INPUT')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9977)
          WRITE(16,9977)
9977      FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &               ' AND SET NCOR EQUAL TO 0 ',
     &       //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          NCOR=0
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
      IF((ICS.EQ.1).AND.(NCOR.EQ.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,99767)
        WRITE(16,99767)
99767   FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NCOR (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT RECOMMENDED WITH',
     &    /,1X,'YOUR SELECTION OF ICS=1 AND MAY LEAD TO GEOMETRIC',
     &         ' DISTORTION PROBLEMS',
     &    /,1X,'FOR LARGE DOMAINS AND/OR OTHER INCONSISTENCIES',
     &    /,1X,'IT IS RECOMMENDED TO USE ICS=2 AND SPHERICAL COORDS')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99768)
          WRITE(16,99768)
99768     FORMAT(/,1X,'PROGRAM WILL CONTINUE EXECUTION WITH',
     &               ' YOUR SPECIFIED INPUT VALUE FOR NCOR ',
     &       //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
      IF(NCOR.EQ.0) THEN
        WRITE(16,233) NCOR
233     FORMAT(/,5X,'NCOR = ',I2,
     &    /,9X,'A CONSTANT VALUE OF THE CORIOLIS PARAMETER WILL BE ',
     &    /,9X,'USED THROUGHOUT THE DOMAIN')
        ELSE
        WRITE(16,234) NCOR
234     FORMAT(/,5X,'NCOR = ',I2,
     &    /,9X,'SPATIALLY VARYING CORIOLIS VALUES WILL BE COMPUTED')
        IF(ICS.EQ.1) THEN
          WRITE(16,9738)
9738      FORMAT(9X,'BASED ON LATITUDES OBTAINED BY APPLYING AN ',
     &              'INVERSE CPP PROJECTION',
     &      /,9X,'TO THE INPUT CARTESIAN COORDINATES')
          ELSE
          WRITE(16,9739)
9739      FORMAT(9X,'BASED ON INPUT LATITUDES')
          ENDIF
        ENDIF

C...
C...READ AND PROCESS NTIP - TIDAL POTENTIAL FORCING
C...
```

```
      READ(15,*) NTIP
      IF((NTIP.LT.0).OR.(NTIP.GT.2)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9978)
        WRITE(16,9978)
9978    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NTIP (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    /,1X,'PLEASE CHECK YOUR INPUT')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9979)
          WRITE(16,9979)
9979      FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &               ' AND SET NTIP EQUAL TO 0 ',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          NTIP=0
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
       IF((ICS.EQ.1).AND.(NTIP.GE.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,99787)
        WRITE(16,99787)
99787   FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NTIP (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT RECOMMENDED WITH',
     &    /,1X,'YOUR SELECTION OF ICS=1 AND MAY LEAD TO GEOMETRIC',
     &         ' DISTORTION PROBLEMS',
     &    /,1X,'FOR LARGE DOMAINS AND/OR OTHER INCONSISTENCIES',
     &    /,1X,'IT IS RECOMMENDED TO USE ICS=2 AND SPHERICAL COORDS')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99788)
          WRITE(16,99788)
99788     FORMAT(/,1X,'PROGRAM WILL CONTINUE EXECUTION WITH',
     &               ' YOUR SPECIFIED INPUT VALUE FOR NTIP ',
     &       //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
       IF(NTIP.EQ.0) THEN
        WRITE(16,235) NTIP
235     FORMAT(/,5X,'NTIP = ',I2,
     &    /,9X,'TIDAL POTENTIAL FORCING IS NOT USED IN THE COMPUTATION')
        ENDIF
       IF(NTIP.GE.1) THEN
        WRITE(16,236) NTIP
236     FORMAT(/,5X,'NTIP = ',I2,
     &    /,9X,'TIDAL POTENTIAL FORCING IS USED IN THE COMPUTATION AND')
        IF(ICS.EQ.1) THEN
          WRITE(16,9748)
9748      FORMAT(9X,'IS BASED ON LONGITUDES/LATITUDES OBTAINED BY ',
     &              'APPLYING AN',
     &      /,9X,'INVERSE CPP PROJECTION TO THE INPUT CARTESIAN ',
     &           'COORDINATES')
          ELSE
          WRITE(16,9749)
9749      FORMAT(9X,'IS BASED ON INPUT LONGITUDES/LATITUDES')
          ENDIF
        ENDIF
       IF(NTIP.EQ.2) WRITE(16,239)
239    FORMAT(9X,'SELF ATTRACTION/LOAD TIDE FORCING USED IN THE ',
     &          'COMPUTATION')

C...
```

```
C...READ AND PROCESS NWS - WIND AND PRESSURE FORCING
C...
      READ(15,*) NWS
      IF((NWS.NE.0).AND.(NWS.NE.1).AND.(NWS.NE.2).AND.(NWS.NE.3).AND.
     &   (NWS.NE.4).AND.(NWS.NE.10)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9980) NWS
        WRITE(16,9980) NWS
9980    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMTER ',
     &          'NWS = ',I8,
     &    /,1X,'IS NOT AN ALLOWABLE VALUE.  PLEASE CHECK YOUR INPUT',
     &    //,1X,'!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!')
        STOP
        ENDIF
      IF(NWS.EQ.0) THEN
        WRITE(16,237) NWS
237     FORMAT(/,5X,'NWS = ',I2,
     &    /,9X,'WIND STRESS OR SURFACE PRESSURE ARE NOT USED TO FORCE',
     &         'THE COMPUTATION')
        ENDIF

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
      IF(NWS.EQ.1) THEN
        WRITE(16,238) NWS
 238    FORMAT(/,5X,'NWS = ',I2,
     &    /,9X,'WIND STRESS AND SURFACE PRESSURE ARE USED TO FORCE',
     &    /,9X,' THE COMPUTATION',
     &    /,9X,'VALUES ARE READ AT ADCIRC GRID NODES FROM UNIT 22',
     &    /,9X,' EVERY MODEL TIME STEP')
        ENDIF
      IF(NWS.EQ.2) THEN
        WRITE(16,2381) NWS
 2381   FORMAT(/,5X,'NWS = ',I2,
     &    /,9X,'WIND STRESS AND SURFACE PRESSURE ARE USED TO FORCE',
     &    /,9X,' THE COMPUTATION',
     &    /,9X,'VALUES ARE READ AT ADCIRC GRID NODES FROM UNIT 22',
     &    /,9X,'INTERPOLATION IN TIME IS DONE TO SYNC THE WIND DATA ',
     &    /,9X,'WITH THE MODEL TIME STEP.')
        ENDIF
      IF(NWS.EQ.3) THEN
        WRITE(16,2382) NWS
 2382   FORMAT(/,5X,'NWS = ',I2,
     &    /,9X,'WIND STRESS ONLY IS USED TO FORCE THE COMPUTATION.',
     &    /,9X,'WIND SPEEDS AND DIRECTIONS ARE READ FROM A FLEET ',
     &    /,9X,'NUMERIC FORMAT FILE AT UNIT 22 AND INTERPOLATED TO',
     &    /,9X,'THE ADCIRC GRID. ',
     &    /,9X,'INTERPOLATION IN TIME IS DONE TO SYNC THE WIND DATA ',
     &    /,9X,'WITH THE MODEL TIME STEP.',
     &    /,9X,'WIND SPEEDS ARE CONVERTED TO STRESS USING THE GARRET ',
     &         'DRAG LAW.')
        ENDIF
      IF(NWS.EQ.4) THEN
        WRITE(16,2383) NWS
 2383   FORMAT(/,5X,'NWS = ',I2,
     &    /,9X,'WIND STRESS AND SURFACE PRESSURE ARE USED TO FORCE',
     &    /,9X,' THE COMPUTATION',
     &    /,9X,'VALUES ARE READ AT SELECTED ADCIRC GRID NODES FROM A ',
     &    /,9X,'PBL/JAG FORMAT FILE AT UNIT 22.',
     &    /,9X,'INTERPOLATION IN TIME IS DONE TO SYNC THE WIND DATA ',
     &    /,9X,'WITH THE MODEL TIME STEP.',
     &    /,9X,'WIND SPEEDS ARE CONVERTED TO STRESS USING THE GARRET ',
     &         'DRAG LAW.')
        ENDIF
      IF(NWS.EQ.10) THEN
        WRITE(16,2384) NWS
 2384   FORMAT(/,5X,'NWS = ',I2,
```

```
     &     /,9X,'WIND STRESS AND SURFACE PRESSURE ARE USED TO FORCE',
     &     /,9X,' THE COMPUTATION',
     &     /,9X,'VALUES ARE READ EVERY 6 HOURS FROM A DIFFERENT FILE ',
     &     /,9X,'FOUND AT UNITS 200, 206, 212, ETC.',
     &     /,9X,'INTERPOLATION IN TIME IS DONE TO SYNC THE WIND DATA ',
     &     /,9X,'WITH THE MODEL TIME STEP AND IN SPACE TO BRING THE ',
     &     /,9X,'WIND DATA FROM A GAUSSIAN GRID TO THE ADCRIC GRID.',
     &     /,9X,'WIND SPEEDS ARE CONVERTED TO STRESS USING THE GARRET ',
     &          'DRAG LAW.')
        ENDIF
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND...

C...
C...READ AND PROCESS NRAMP - WHETHER A RAMP FUNCTION WILL BE USED
C...
      READ(15,*) NRAMP
      IF((NRAMP.NE.0).AND.(NRAMP.NE.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9982)
        WRITE(16,9982)
9982    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NRAMP (A UNIT 15 INPUT ',
     &          'PARAMETER) IS NOT AN ALLOWABLE VALUE',
     &    /,1X,'PLEASE CHECK YOUR INPUT')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9983)
          WRITE(16,9983)
9983      FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &                ' AND SET NRAMP EQUAL TO 0 ',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          NRAMP=0
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF
      IF(NRAMP.EQ.0) THEN
        WRITE(16,240) NRAMP
240     FORMAT(/,5X,'NRAMP = ',I2,
     &    /,9X,'NO RAMP FUNCTION IS USED IN THE COMPUTATION')
        ELSE
        WRITE(16,241) NRAMP
241     FORMAT(/,5X,'NRAMP = ',I2,
     &    /,9X,'A HYPERBOLIC TANGENT RAMP IS APPLIED TO THE FORCING ',
     &         'FUNCTIONS')
        ENDIF

C...
C...PROCESS G - GRAVITY
C...
      READ(15,*) G
      IF((ICS.EQ.1).AND.(G.NE.9.81)) THEN
        IF((NCOR.EQ.1).OR.(NTIP.EQ.1)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99732)
          WRITE(16,99732)
99732     FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                  'INPUT ERROR  !!!!!!!!!',
     &      //,1X,'YOUR SELECTION OF G (A UNIT 15 INPUT ',
     &           'PARAMETER) IS NOT CONSISTENT WITH THE ',
     &      /,1X,'SELECTION OF ICS=1 IN CONJUNCTION WITH NTIP=1 ',
     &          'AND/OR NCOR=1',
     &      /,1X,'PLEASE CHECK YOUR INPUT')
          IF(NFOVER.EQ.1) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99733)
            WRITE(16,99733)
99733       FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &                  ' AND SET G EQUAL TO 9.81 M/SEC*SEC ',
```

```
     &        /,1X,'PLEASE CHECK THAT ALL INPUT HAS SI UNITS',
     &        //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
            G=9.81
            ELSE
            IF(NSCREEN.EQ.1) WRITE(6,9973)
            WRITE(16,9973)
            STOP
            ENDIF
          ENDIF
        ENDIF
       IF((ICS.EQ.2).AND.(G.NE.9.81)) THEN
         IF(NSCREEN.EQ.1) WRITE(6,99735)
         WRITE(16,99735)
99735    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF G (A UNIT 15 INPUT PARAMETER) ',
     &          'IS NOT CONSISTENT WITH THE ',
     &    /,1X,'SELECTION OF ICS=2.   PLEASE CHECK YOUR INPUT')
         IF(NFOVER.EQ.1) THEN
           IF(NSCREEN.EQ.1) WRITE(6,99736)
           WRITE(16,99736)
99736      FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &                ' AND SET G EQUAL TO 9.81 M/SEC*SEC ',
     &      /,1X,'PLEASE CHECK THAT ALL INPUT HAS SI UNITS',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
           G=9.81
           ELSE
           IF(NSCREEN.EQ.1) WRITE(6,9973)
           WRITE(16,9973)
           STOP
           ENDIF
         ENDIF
       WRITE(16,5) G
     5 FORMAT(///,5X,'GRAVITATIONAL CONSTANT G =',F10.5,/)

C...
C...READ AND PROCESS TAU0 - WEIGHTING COEFFICIENT IN THE GWCE
C...
       READ(15,*) TAU0
       WRITE(16,7) TAU0
     7 FORMAT(/,5X,'WEIGHTING COEFFICIENT FOR THE GENERALIZED',
      &          ' WAVE CONTINUITY EQUATION :',
      &  /,5X, 'TAU0 = ',E14.8,2X,'1/sec',/)

C...
C...INPUT FROM UNIT 15 AND OUTPUT TO UNIT 16 TIME INTEGRATION INFORMATION
C....INCLUDING DT,STATIM,REFTIM,AND RNDAY
C...
       WRITE(16,1112)
       WRITE(16,245)
   245 FORMAT(//,1X,'TIME INTEGRATION INFORMATION',//)

C...
C...READ AND PROCESS DT - MODEL TIME STEP
C...
       READ(15,*) DTDP
       DT=DTDP
       WRITE(16,9) DTDP
     9 FORMAT(5X,'TIME STEP =',F12.6,5X,'SECONDS',/)

C...
C...READ AND PROCESS STATIM - SIMULATION STARTING TIME
C...
       READ(15,*) STATIM
       WRITE(16,1113) STATIM
  1113 FORMAT(5X,'STARTING TIME FOR SIMULATION = ',F14.6,' DAYS',/)

C...
C...READ AND PROCESS REFTIM - SIMULATION REFERNCE TIME
C...
```

```
      READ(15,*) REFTIM
      WRITE(16,1115) REFTIM
1115  FORMAT(5X,'REFERENCE TIME FOR SIMULATION = ',F14.6,' DAYS',/)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES FOR NO WIND AND PRESSURE FORCING
CWIND...READ AND PROCESS WREFTIM - SIMULATION REFERNCE TIME
CWIND....AND PARAMETERS DESCRIBING FLEET NUMERIC FILE
CWIND...
      IF(NWS.EQ.2) THEN
        READ(15,*) WTIMINC
        ENDIF
      IF(NWS.EQ.3) THEN
        READ(15,*) IREFYR,IREFMO,IREFDAY,IREFHR,IREFMIN,REFSEC
        WRITE(16,1116) IREFMO,IREFDAY,IREFYR,IREFHR,IREFMIN,REFSEC
 1116   FORMAT(5X,'WIND REFERENCE TIME FOR SIMULATION = ',
     &         I2,'/',I2,'/',I2,'  ',I2,':',I2,':',f7.4,/)
        CALL TIMECONV(IREFYR,IREFMO,IREFDAY,IREFHR,IREFMIN,REFSEC,
     &                                                    WREFTIM)
        REFTIM = 0.
        READ(15,*) NWLAT,NWLON,WLATMAX,WLONMIN,WLATINC,WLONINC,WTIMINC
        ENDIF
      IF(NWS.EQ.4) THEN
        READ(15,*) WTIMINC
        ENDIF
      IF(NWS.EQ.10) THEN
        READ(15,*) NWLAT,NWLON,WTIMINC
        ENDIF
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND...

C...
C...READ AND PROCESS RNDAY - SIMULATION DURATION IN DAYS
C...
      READ(15,*) RNDAY
      WRITE(16,10) RNDAY
10    FORMAT(5X,'TOTAL LENGTH OF NUMERICAL SIMULATION =',F12.4,
     &       5X,'DAYS',/)

C...
C...COMPUTE TOTAL NUMBER OF TIME STEPS NT
C...
      NT=INT(RNDAY*(86400.D0/DTDP)+0.5)
      WRITE(16,1920) NT
1920  FORMAT(5X,'NUMBER OF TIME STEPS  =',I8,/)

C...
C...READ AND PROCESS EFFECTIVE LENGTH OF THE HYPERBOLIC TANGENT RAMP IN DAYS
C...
      READ(15,*) DRAMP
      IF(NRAMP.NE.0) THEN
        WRITE(16,8763) DRAMP
8763    FORMAT(/,5X,'VALUE FOR DRAMP USED IN RAMP EVALUATION =',F12.4,
     &         5X,'DAYS',/)
        DAY=0.0
        WRITE(16,5841)
5841    FORMAT(11X,' DAYS OF SIMULATION',2X,' TIME  ',6X,'  RAMP',/)
        DO IDR=1,24
          RAMP=TANH(DAY*2./DRAMP)
          WRITE(16,5845) DAY,DAY+STATIM,RAMP
5845      FORMAT(15X,F8.2,6X,F8.2,2X,F15.7)
          IF(DAY.LT.3.0) THEN
            DAY=DAY+0.5
            ELSE
            DAY=DAY+1.0
            ENDIF
          END DO
        ENDIF
```

```
C...
C...READ GWCE TIME WEIGHTING FACTORS
C...
      READ(15,*) A00,B00,C00
      WRITE(16,14)
14    FORMAT(//,5X,'TIME WEIGHTING FACTORS IN THE WAVE EQUATION :'/)
      WRITE(16,15) A00,B00,C00
15    FORMAT(9X,'AT TIME LEVEL K+1 : ',F8.5,
     &  /,9X,'AT TIME LEVEL K   : ',F8.5,
     &  /,9X,'AT TIME LEVEL K-1 : ',F8.5,/)

C...
C...READ MINIMUM DEPTH OR WET/DRY PARAMETERS FROM UNIT 15
C...
      IF(NOLIFA.NE.2) THEN
        READ(15,*) H0
        WRITE(16,16) H0
  16    FORMAT(//,5X,'THE BATHYMETRIC DEPTH AT ALL NODES WILL BE ',
     &              'INCREASED TO H0= ',F12.4,' IF NECESSARY'/)
        ENDIF
      IF(NOLIFA.EQ.2) THEN
        READ(15,*) H0,NODEDRYMIN,NODEWETRMP,VELMIN
        LBCODEWD=-1             !NO TANGENTIAL SLIP ALONG W/D BOUNDARIES
        WRITE(16,17) H0,NODEWETRMP,VELMIN,NODEDRYMIN
  17    FORMAT(//,5X,'DRYING WILL OCCUR WHEN THE WATER DEPTH < ',
     &              'H0*(NODEREP/NODEDWETRMP)',
     &         /,5X,'H0 = ',F10.6,
     &         /,5X,'NODEWETRMP = ',I6,
     &         /,5X,'NODEREP = NUMBER OF TIME STEPS SINCE THE NODE WET',
     &        //,5X,'WETTING WILL OCCUR WHEN THE VELOCITY AT ANY WET/',
     &              'DRY INTERFACE NODE',
     &         /,5X,'DIRECTED TOWARD ALL SURROUNDING DRY NODES EXCEEDS',
     &              ' VELMIN = ',F10.5,
     &         /,5X,'AND ALL AFFECTED NODES HAVE BEEN DRY AT LEAST ',
     &              'NODEDRYMIN = ',I6,' TIME STEPS',/)
        ENDIF

C...
C...READ GRID INFORMATION FROM UNITS 14 & 15
C...
      READ(14,'(A24)') AGRID
      READ(14,*) NE,NP
      READ(15,*) SLAM0,SFEA0
      SLAM0=SLAM0*DEG2RAD
      SFEA0=SFEA0*DEG2RAD
      WRITE(16,1112)
      WRITE(16,246)
246   FORMAT(//,1X,'GRID INFORMATION',//)

C...
C...CHECK THAT MNP AND MNE HAVE BEEN SET PROPERLY
C...
      IF(NE.GT.MNE) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9932)
        WRITE(16,9932)
9932    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNE IS EXCEEDED',
     &          ' BY THE INPUT PARAMETER NE',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF
      IF(NP.GT.MNP) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9933)
        WRITE(16,9933)
9933    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNP IS EXCEEDED',
     &          ' BY THE INPUT PARAMETER NP',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
```

```
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF

C...
C...IF ICS=1 INPUT NODAL COORDINATES AND BATHYMETRY FROM UNIT 14
C....IF EITHER NTIP=1 OR NCOR=1, COMPUTE THE INVERSE CPP PROJECTION
C...IF ICS=2 INPUT NODAL COORDINATES AND BATHYMETRY FROM UNIT 14
C....AND COMPUTE CPP PROJECTED COORDINATES
C...
      IF(ICS.EQ.1) THEN
        DO I=1,NP
          READ(14,*) JKI,X(JKI),Y(JKI),DP(JKI)
          IF(JKI.NE.I) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99801)
            WRITE(16,99801)
99801       FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                     'INPUT ERROR  !!!!!!!!!',
     &        //,1X,'YOUR NODE NUMBERING IS NOT SEQUENTIAL ',
     &              'CHECK YOUR UNIT 14 INPUT FILE CAREFULLY',//)
            ENDIF
          IF((NTIP.GE.1).OR.(NCOR.EQ.1)) THEN
            CALL INVCP(X(JKI),Y(JKI),SLAM(JKI),SFEA(JKI),SLAM0,SFEA0)
            ENDIF
          END DO
        ENDIF
      IF(ICS.EQ.2) THEN
        DO I=1,NP
          READ(14,*) JKI,SLAM(JKI),SFEA(JKI),DP(JKI)
          IF(JKI.NE.I) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99801)
            WRITE(16,99801)
            ENDIF
          SLAM(JKI)=DEG2RAD*SLAM(JKI)
          SFEA(JKI)=DEG2RAD*SFEA(JKI)
          CALL CPP(X(JKI),Y(JKI),SLAM(JKI),SFEA(JKI),SLAM0,SFEA0)
          END DO
        ENDIF

C...
C...IF ICS=1 SET THE SFAC VECTOR EQUAL TO UNITY
C...IF ICS=2 COMPUTE THE SFAC VECTOR IN ORDER TO ADJUST EQUATIONS TO CPP
C...COORDINATES
C...
      IF(ICS.EQ.1) THEN
        DO I=1,NP
          SFAC(I)=1.00
          END DO
        ENDIF
      IF(ICS.EQ.2) THEN
        DO I=1,NP
          SFAC(I)=COS(SFEA0)/COS(SFEA(I))
          END DO
        ENDIF

C...
C...IF WETTING AND DRYING WILL NOT BE USED, MAKE SURE ALL BATHYMETRIC DEPTHS ARE
C...> OR = TO H0.
C...
      IF((NOLIFA.EQ.0).OR.(NOLIFA.EQ.1)) THEN
        DO I=1,NP
          IF(DP(I).LT.H0) DP(I)=H0
          END DO
        ENDIF

C...
C...READ THE GLOBAL CONNECTIVITY TABLE FROM UNIT 14
C...COMPUTE ELEMENT AREAS
C...CHECK THAT SUFFICIENT ACCURACY IS PROVIDED BY THE CODE TO HANDLE
C....THE INPUT GRID
```

```
C...CHECK TO INSURE THAT CORRECT CONVENTION HAS BEEN USED FOR INPUTING
C....THE CONNECTIVITY TABLE
C...
      DO I=1,NE
        READ(14,*) JKI,NHY,NM(JKI,1),NM(JKI,2),NM(JKI,3)
        IF(JKI.NE.I) THEN
          IF(NSCREEN.EQ.1) WRITE(6,99802)
          WRITE(16,99802)
99802     FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',
     &      //,1X,'YOUR ELEMENT NUMBERING IS NOT SEQUENTIAL ',
     &      /,1X,'CHECK YOUR UNIT 14 INPUT FILE CAREFULLY',//)
          ENDIF
        X1=X(NM(JKI,1))
        X2=X(NM(JKI,2))
        X3=X(NM(JKI,3))
        Y1=Y(NM(JKI,1))
        Y2=Y(NM(JKI,2))
        Y3=Y(NM(JKI,3))
        AVGXY=(ABS(X1)+ABS(X2)+ABS(X3)+ABS(Y1)+ABS(Y2)+ABS(Y3))/6.
        DIF1R=AVGXY/(((X2-X1)**2+(Y2-Y1)**2)**0.5)
        DIF2R=AVGXY/(((X3-X2)**2+(Y3-Y2)**2)**0.5)
        DIF3R=AVGXY/(((X3-X1)**2+(Y3-Y1)**2)**0.5)
        DIF1R=LOG10(DIF1R)
        DIF2R=LOG10(DIF2R)
        DIF3R=LOG10(DIF3R)
        IF((DIF1R.GT.NPREC).OR.(DIF1R.GT.NPREC).OR.(DIF1R.GT.NPREC))THEN
          IF(NSCREEN.EQ.1) WRITE(6,9898) JKI
          WRITE(16,9898) JKI
9898      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &      //,1X,'ASSUMING THAT THE GRID COORDINATES HAVE 32 BITS ',
     &            '(APPROX 7 DIGITS) OF PRECISION',
     &      /,1X,'A ROBUST MODEL SOLUTION CAN NOT BE GUARANTEED',
     &      /,1X,'AT ELEMENT NO. ',I10,
     &      //,1X,'MORE PRECISION MUST BE USED IN THE GRID, AND, IF ',
     &            ' THE RUN IS',
     &      /,1X,'BEING MADE ON A 32 BIT COMPUTER, ALL COORDINATE ',
     &            'VARIABLES AND ',
     &      /,1X,'ELEMENTAL MATRICIES MUST BE DEFINED AS DOUBLE ',
     &           'PRECISION.'
     &      //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
          STOP
          ENDIF
        AREAS(JKI)=(X1-X3)*(Y2-Y3)+(X3-X2)*(Y1-Y3)     !2 X ACTUAL ELEMENT ARE
        IF(AREAS(JKI).LT.0.0) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9899) JKI
          WRITE(16,9899) JKI
9899      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &      //,1X,'THE CONNECTIVITY FOR ELEMENT ',I6,
     &            '  HAS BEEN INCORRECTLY SPECIFIED ',
     &      /,1X,'CHECK INPUT AND ENSURE THAT COUNTERCLOCKWISE',
     &           ' CONVENTION HAS BEEN USED ',
     &      //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
          STOP
          ENDIF
        END DO

C...
C...OUTPUT TO UNIT 16 GRID INFORMATION INCLUDING AGRID,NE,NP
C....H0 AND NODAL COORDINATES AND BATHYMETRY
C...
      WRITE(16,2039) AGRID
2039  FORMAT(/,5X,'GRID IDENTIFICATION : ',A24,/)
      WRITE(16,3) NP
    3 FORMAT(5X,'TOTAL NUMBER OF NODES =',I6,/)
      WRITE(16,4) NE
    4 FORMAT(5X,'TOTAL NUMBER OF ELEMENTS =',I6,/)
      WRITE(16,13) SLAM0*RAD2DEG,SFEA0*RAD2DEG
   13 FORMAT(5X,'LATITUDE ABOUT WHICH CPP PROJECTION IS CENTERED',
     &       '  SLAM0 = ',F9.4,' DEGREES',
```

```
     &  /,5X,'LONGITUDE ABOUT WHICH CPP PROJECTION IS CENTERED',
     &       '  SFEA0 = ',F9.4,' DEGREES',/)
      IF(NABOUT.NE.1) THEN
        WRITE(16,24)
   24   FORMAT(/,1X,'NODAL COORDINATES AND BATHYMETRY :')
        IF(ICS.EQ.1) THEN
          IF((NTIP.EQ.0).AND.(NCOR.EQ.0)) THEN
            WRITE(16,25)
   25       FORMAT(/,10X,'NODE NO.',10X,'X',20X,'Y',15X,'DP',/)
            DO I=1,NP
              WRITE (16,2008) I,X(I),Y(I),DP(I)
 2008         FORMAT(5X,I6,2(2X,F20.2),2X,F12.2)
              END DO
            ELSE
            WRITE(16,9195)
 9195       FORMAT(/,1X,'   NODE ',7X,'X',14X,'Y',9X,
     &                 'LAMBDA(DEG)',6X,'FEA(DEG)',9X,'DP',/)
            DO I=1,NP
              WRITE (16,9197) I,X(I),Y(I),SLAM(I)*RAD2DEG,
     &                        SFEA(I)*RAD2DEG,DP(I)
9197          FORMAT(1X,I6,2(1X,F14.1),1X,2(1X,E15.7),1X,F8.2)
              END DO
            ENDIF
          ELSE
          WRITE(16,9225)
9225      FORMAT(/,1X,'   NODE ',2X,'LAMBDA(DEG)',5X,'FEA(DEG)',11X,
     &               'XCP',14X,'YCP',11X,'DP',/)
          DO I=1,NP
            WRITE (16,9228) I,SLAM(I)*RAD2DEG,SFEA(I)*RAD2DEG,
     &                      X(I),Y(I),DP(I)
9228        FORMAT(1X,I6,2(1X,F14.8),2(1X,F15.1),1X,F10.2)
            END DO
          ENDIF
        ELSE
        WRITE(16,3511)
3511    FORMAT(/,5X,'NODAL COORDINATES AND BATHYMETRY',
     &              ' INFORMATION IS AVAILABLE IN THE',
     &    /,6X,'UNIT 14 INPUT FILE')
        ENDIF

C...
C...OUTPUT TO UNIT 16 THE GLOBAL CONNECTIVITY TABLE (NODE NUMBERS FOR ELEMENTS)
C...
      IF(NABOUT.NE.1) THEN
        WRITE(16,26)
26      FORMAT(//,5X,'GLOBAL NODE NUMBERS FOR EACH ELEMENT :')
        WRITE(16,27)
27      FORMAT(/,9X,'ELEMENT',8X,'N1',9X,'N2',10X,'N3',/)
        DO I=1,NE
          WRITE(16,2009) I,NM(I,1),NM(I,2),NM(I,3)
2009      FORMAT(8X,4(I7,4X))
          END DO
        ELSE
        WRITE(16,3512)
3512    FORMAT(/,5X,'THE GLOBAL CONNECTIVITY TABLE',
     &              ' INFORMATION IS AVAILABLE IN THE',
     &    /,6X,'UNIT 14 INPUT FILE')
        ENDIF

C...
C...COMPUTE THE NEIGHBOR TABLE
C...
      IF(NSCREEN.EQ.1) WRITE(6,1196)
      WRITE(16,1196)
 1196 FORMAT(/,1X,'THE NEIGHBOR TABLE IS BEING COMPUTED ',/)
      CALL NEIGHB(NE,NP,NNEIGH,NEITAB,NEIMIN,NEIMAX,NSCREEN)
      IF(NSCREEN.EQ.1) WRITE(6,1195) NEIMIN,NEIMAX,NEIMAX
      WRITE(16,1195) NEIMIN,NEIMAX,NEIMAX
 1195 FORMAT(1X,'THE NEIGHBOR TABLE IS COMPLETED ',
     &   /,5X,'THE MINIMUM NUMBER OF NEIGHBORS FOR ANY NODE = ',I3,
```

```
     &    /,5X,'1+THE MAXIMUM NUMBER OF NEIGHBORS FOR ANY NODE = ',I3,
     &    /,5X,'THE PARAMETER MNEI CAN BE SET AS SMALL AS ',I3,/)

C...
C...READ INFORMATION CONCERNING BOTTOM FRICTION COEFFICIENT
C...IF NWP=1, INPUT NODAL FRICTION COEFFICIENTS FROM UNIT 21
C...IF NWP=0, SET NODAL FRICTION COEFFICIENTS EQUAL TO CF
C...
      WRITE(16,1112)
      WRITE(16,2045)
2045  FORMAT(//,' BOTTOM FRICTION INFORMATION',//)

      HBREAK=1.
      FTHETA=1.
      FGAMMA=1.
      IF(NOLIBF.EQ.0) READ(15,*) TAU
      CF=TAU
      IF(NOLIBF.EQ.1) READ(15,*) CF
      IF(NOLIBF.EQ.2) READ(15,*) CF,HBREAK,FTHETA,FGAMMA

      IF(NWP.EQ.0) THEN
        DO I=1,NP
          FRIC(I)=CF
          END DO
        IF(NOLIBF.EQ.2) THEN
          WRITE(16,101) CF,HBREAK,FTHETA,FGAMMA
 101      FORMAT(5X,'HYBRID FRICTION RELATIONSHIP PARAMTERS, CFMIN =',
     &              F12.8,'  HBREAK = ',F8.2,
     &             /,5X,'FTHETA = ',F8.2,'  FGAMMA = ',F10.4,//)
          ENDIF
        IF(NOLIBF.EQ.1) THEN
          WRITE(16,8) CF
8         FORMAT(5X,'NONLINEAR FRICTION COEFFICIENT CF =',F12.8,/)
          ENDIF
        IF(NOLIBF.EQ.0) THEN
          WRITE(16,6) TAU
6         FORMAT(5X,'LINEAR BOTTOM FRICTION TAU =',F12.8,5X,'1/sec'/)
          IF(TAU.NE.TAU0) THEN                !CHECK TAU VALUE AGAINST TAU0
            IF(NSCREEN.EQ.1) WRITE(6,9951)
            WRITE(16,9951)
 9951       FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                     'INPUT ERROR  !!!!!!!!!',
     &        //,1X,'TYPICALLY YOUR INPUT VALUE FOR ',
     &             'TAU0 SHOULD BE SET EQUAL TO TAU')
            ENDIF
          ENDIF
        ENDIF

      IF(NWP.EQ.1) THEN
        OPEN(21,FILE='fort.21')
        READ(21,'(A20)') AFRIC
        DO I=1,NP
          READ(21,*) NHG,FRIC(NHG)
          IF(NHG.NE.I) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99803)
            WRITE(16,99803)
99803       FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ',
     &                     'INPUT ERROR  !!!!!!!!!',
     &        //,1X,'YOUR NODAL FRICTION NUMBERING IS NOT SEQUENTIAL ',
     &        /,1X,'CHECK YOUR UNIT 21 INPUT FILE CAREFULLY',
     &        //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
            STOP
            ENDIF
          END DO
        WRITE(16,3601) AFRIC
 3601   FORMAT(/,5X,'FRICTION FILE IDENTIFICATN : ',A20,/)
        IF(NABOUT.NE.1) THEN
          WRITE(16,2080)
 2080     FORMAT(/,10X,'NODE',5X,'BOTTOM FRICTION FRIC',5X,/)
          DO I=1,NP
```

```
            WRITE(16,2087) I,FRIC(I)
 2087       FORMAT(7X,I6,6X,E15.10)
            END DO
          ELSE
          WRITE(16,3504)
 3504     FORMAT(/,5X,'NODAL BOTTOM FRICTION VALUES ARE AVAILABLE',
     &           /,6X,' IN UNIT 21 INPUT FILE')
          ENDIF
        ENDIF

C...
C...READ IN AND WRITE OUT EDDY VISCOSITY/DIFFUSIVITY COEFFICIENTS
C...
      IF(IM.EQ.10) THEN
        READ(15,*) ESLM,ESLC
        DO I=1,NP
          EVM(I)=ESLM
          EVC(I)=ESLC
          END DO
        WRITE(16,111) ESLM,ESLC
 111    FORMAT(5X,'EVM, EDDY VISCOSITY COEFFICIENT =',E15.8,/,
     &         5X,'EVC, EDDY DIFFUSIVITY COEFFICIENT =',E15.8,//)
        ENDIF
      IF(IM.NE.10) THEN
        READ(15,*) ESLM
        DO I=1,NP
          EVM(I)=ESLM
          END DO
        WRITE(16,11) ESLM
 11     FORMAT(5X,'EVM, EDDY VISCOSITY COEFFICIENT =',E15.8,//)
        ENDIF

C...
C...READ CORIOLIS INFORMATION AND COMPUTE THE CORIOLIS VECTOR
C...OUTPUT RESULTING CORIOLIS INFORMATION
C...
      WRITE(16,1112)
      WRITE(16,2090)
2090  FORMAT(//,1X,'CORIOLIS INFORMATION ',//)

      READ(15,*) CORI
      IF(NCOR.EQ.0) THEN
        DO I=1,NP
          CORIF(I)=CORI
          END DO
        ENDIF
      IF(NCOR.EQ.1) THEN
        DO I=1,NP
          CORIF(I)=2.0*7.29212E-5*SIN(SFEA(I))
          END DO
        ENDIF

      IF(NCOR.EQ.0) THEN
        WRITE(16,12) CORI
12      FORMAT(5X,'CONSTANT CORIOLIS COEFFICIENT =',E15.8,5X,'1/SEC',/)
        ENDIF
      IF(NCOR.EQ.1) THEN
        IF(ICS.EQ.1) THEN
          WRITE(16,3603)
3603      FORMAT(/,5X,'LATITUDES ARE USED TO COMPUTE THE VARIABLE',
     &               ' CORIOLIS PARAMETER AND ARE BASED',
     &      /,7X,'ON AN INVERSE CPP PROJECTION OF THE INPUT COORDS',/)
          ELSE
          WRITE(16,3604)
3604      FORMAT(/,5X,'LATITUDES ARE USED TO COMPUTE VARIABLE CORIOLIS',
     &      /,7X,'AND ARE BASED ON INPUT NODAL COORDINATES',/)
          ENDIF
        IF(NABOUT.NE.1) THEN
          WRITE(16,2092)
2092      FORMAT(/,10X,' NODE ',5X,'NODAL CORIOLIS CORIF',/)
```

```
          DO I=1,NP
            WRITE(16,2096) I,CORIF(I)
2096        FORMAT(7X,I6,10X,E15.9)
            END DO
          ENDIF
        ENDIF

C...
C...READ AND PROCESS INFORMATION ABOUT THE TIDAL POTENTIAL CONSTITUENTS
C...
      READ(15,*) NTIF
C...
C...CHECK DIMENSIONING OF PARAMETER MNTIF
C...
      IF(NTIF.GT.MNTIF) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9901)
        WRITE(16,9901)
9901    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNTIF IS EXCEEDED',
     &          ' BY THE INPUT PARAMETER NTIF',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF
C...
C...READ TIDAL POTENTIAL AMPLITUDE, FREQUENCIES, NODAL FACTORS,
C...EQUILIBRIUM ARGUMENTS AND ALPHANUMERIC LABEL
C...
      DO I=1,NTIF
        READ(15,'(A5)')  TIPOTAG(I)
        READ(15,*)  TPK(I),AMIGT(I),ETRF(I),FFT(I),FACET(I)
        IF(AMIGT(I).EQ.0.) THEN
          PERT(I)=0.
          ELSE
          PERT(I)=2.D0*PI/AMIGT(I)
          ENDIF
        END DO
C...
C...OUTPUT TO UNIT 16 INFORMATION ABOUT TIDAL POTENTIAL FORCING
C....OUTPUT WILL VARY DEPENDING ON VALUES OF NTIP,NTIF AND NCOR
C...
      WRITE(16,1112)
      WRITE(16,2102)
 2102 FORMAT(//,1X,'TIDAL POTENTIAL FORCING INFORMATION ',//)
      WRITE(16,22) NTIF
 22   FORMAT(/,1X,'TIDAL POTENTIAL IS FORCED FOR ',I5,
     &            ' CONSTITUENT(S) ')
      IF(NTIF.GT.0) WRITE(16,23)
 23   FORMAT(/,1X,'AMPLITUDE',4X,'FREQUENCY',5X,
     &            '    ETRF      ','NODAL FACTOR',2X,
     &            'EQU.ARG(DEG)',1X,'CONSTITUENT',/)
      DO I=1,NTIF
        WRITE(16,2107) TPK(I),AMIGT(I),ETRF(I),FFT(I),FACET(I),
     &                 TIPOTAG(I)
 2107   FORMAT(1X,F9.7,1X,F15.12,2X,F10.7,5X,F10.7,1X,F10.3,7X,A5)
        END DO
C...
C...CONVERT FACET(I) VALUES FROM DEGREES TO RADIANS
C...
      DO I=1,NTIF
        FACET(I)=FACET(I)*DEG2RAD
        END DO
C...
C...CHECK CONSISTENCY OF INPUT PARAMETERS NTIF AND NTIP
C...
      IF(((NTIP.EQ.0).AND.(NTIF.NE.0)).OR.((NTIP.NE.0).AND.
     &    (NTIF.EQ.0))) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9961)
        WRITE(16,9961)
9961    FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
```

```
     &                  'INPUT ERROR  !!!!!!!!!',
     &    //,1X,'YOUR SELECTION OF NTIF AND NTIP (UNIT 15 INPUT ',
     &          'PARAMETERS) IS INCONSISTENT',
     &    /,1X,'PLEASE CHECK THESE VALUES')
         IF(NFOVER.EQ.1) THEN
           IF(NSCREEN.EQ.1) WRITE(6,9987)
           WRITE(16,9987)
9987       FORMAT(/,1X,'PROGRAM WILL OVERRIDE THE SPECIFIED ',
     &                 'INPUT AND NEGLECT TIDAL POTENTIAL TERMS',
     &      /,1X,' AND/OR RESET NTIP = 0',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
           NTIP=0
           ELSE
           IF(NSCREEN.EQ.1) WRITE(6,9973)
           WRITE(16,9973)
           STOP
           ENDIF
         GOTO 1893
         ENDIF
C...
C...PRINT OUT LAT/LON VALUES TO BE USED IN COMPUTING TIDAL POTENTIAL
C....IF NOT ALREADY DONE SO IN CORIOLIS SECTION AND TIDAL POTENTIAL
C....IS ACTIVATED WITH NTIP=1
C...
       IF(NTIP.GE.1) THEN
         IF(ICS.EQ.1) THEN
           WRITE(16,3605)
3605       FORMAT(/,5X,'LONGITUDES AND LATITUDES ARE USED TO',
     &                 ' COMPUTE THE TIDAL POTENTIAL FUNCTION',
     &      /,7X,'AND ARE BASED ON AN INVERSE CPP PROJECTION ',
     &           'OF THE INPUT COORDINATES',/)
           ELSE
           WRITE(16,2109)
2109       FORMAT(/,5X,'LONGITUDES AND LATITUDES ARE USED TO',
     &                 ' COMPUTE THE TIDAL POTENTIAL FUNCTION',
     &      /,7X,'AND ARE BASED ON INPUT NODAL COORDINATES ',/)
           ENDIF
         ENDIF

C...
C...INPUT FROM UNIT 15 THE TIDAL FORCING FREQUENCIES ON THE ELEVATION
C....SPECIFIED BOUNDARIES: INCLUDING NBFR, FREQUENCIES, NODAL FACTORS,
C....EQUILIBRIUM ARGUMENTS AND AN ELEVATION BOUNDARY CONDITION
C....ALPHANUMERIC DESCRIPTOR
C...
 1893 READ(15,*) NBFR
C...
C...CHECK TO ENSURE THAT DIMENSIONING PARAMETER MNBFR IS LARGE ENOUGH
C...
       IF(NBFR.GT.MNBFR) THEN
         IF(NSCREEN.EQ.1) WRITE(6,9903)
         WRITE(16,9903)
 9903    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNBFR IS EXCEEDED BY THE ',
     &          'INPUT PARAMETER NBFR',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
         STOP
         ENDIF
       WRITE(16,1112)
       WRITE(16,2106)
 2106 FORMAT(//,1X,'ELEVATION SPECIFIED BOUNDARY FORCING INFORMATION '
     &   ,//)
       WRITE(16,20) NBFR
   20 FORMAT(/,5X,'NUMBER OF BOUNDARY TIDAL FORCING CONSTITUENTS =',I5)
       IF(NBFR.GE.1) WRITE(16,21)
   21 FORMAT(/,7X,'CONSTITUENT #',4X,'FREQUENCY',4X,'NODAL FACTOR',
     &         3X,'EQU.ARG (DEG)',2X,'CONSTITUENT',/)
       DO I=1,NBFR
         READ(15,'(A5)') BOUNTAG(I)
```

```
        READ(15,*) AMIG(I),FF(I),FACE(I)
        WRITE(16,1850) I,AMIG(I),FF(I),FACE(I),BOUNTAG(I)
 1850   FORMAT(12X,I2,6X,F16.12,2X,F10.7,2X,F10.3,10X,A5)
        FACE(I)=FACE(I)*DEG2RAD
        IF(AMIG(I).EQ.0.) THEN
          PER(I)=0.
          ELSE
          PER(I)=2.D0*PI/AMIG(I)
          ENDIF
        END DO
C...
C...INPUT ELEVATION BOUNDARY FORCING NODE NUMBER INFORMATION FROM UNIT 14 AND
C....OUTPUT TO UNIT 16
C...
C...INPUT THE TOTAL NUMBER OF ELEVATION BOUNDARY SEGMENTS
C...
      READ(14,*) NOPE
C...
C...CHECK TO ENSURE THAT THE DIMENSIONING PARAMETER MNOPE HAS BEEN PROPERLY SET
C...
      IF(NOPE.GT.MNOPE) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9936)
        WRITE(16,9936)
9936    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNOPE IS EXCEEDED BY THE ',
     &          'INPUT PARAMETER NOPE',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF
       WRITE(16,1852) NOPE
1852  FORMAT(///,5X,'TOTAL NUMBER OF ELEVATION BOUNDARY FORCING',
     &         ' SEGMENTS ',' = ',I5)
C...
C...INPUT THE TOTAL NUMBER OF ELEVATION BOUNDARY NODES
C...
      READ(14,*) NETA
C...
C...CHECK TO ENSURE THAT THE DIMENSIONING PARAMETER MNETA HAS BEEN PROPERLY SET
C...
      IF(NETA.GT.MNETA) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9937)
        WRITE(16,9937)
 9937   FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNETA IS EXCEEDED BY THE ',
     &          'INPUT PARAMETER NETA',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF
       WRITE(16,1854) NETA
 1854 FORMAT(/,5X,'TOTAL NUMBER OF ELEVATION BOUNDARY FORCING NODES = '
     &   ,I5)
C...
C...INPUT THE NODE NUMBERS ON EACH ELEVATION BOUNDARY FORCING SEGMENT
C...
      JNMM=0
      DO K=1,NOPE
        READ(14,*) NVDLL(K)
        WRITE(16,281) K,NVDLL(K)
  281   FORMAT(//,5X,'TOTAL NUMBER OF NODES ON ELEVATION BOUNDARY',
     &    ' SEGMENT ',2X,I2,2X,'=',1X,I5,/)
        DO I=1,NVDLL(K)
          READ(14,*) NBDV(K,I)
          WRITE(16,1855) NBDV(K,I)
 1855     FORMAT(7X,I7)
          NBD(JNMM+I)=NBDV(K,I)
          END DO
        JNMM=JNMM+NVDLL(K)
        END DO
```

```
C...
C...CHECK TO MAKE SURE THAT JNMM EQUALS NETA
C...
      IF(NETA.NE.JNMM) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9945)
        WRITE(16,9945)
 9945   FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL INPUT ERROR ',
     &                 '!!!!!!!!!!',
     &    //,1X,'THE INPUT PARAMETER NETA FROM UNIT 14 DOES NOT MATCH ',
     &          'THE TOTAL NUMBER OF BOUNDARY NODES',
     &    /,1X,' FROM ALL THE SPECIFIED SEGMENTS COMBINED')
        IF(NFOVER.EQ.1) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9989)
          WRITE(16,9989)
 9989     FORMAT(/,1X,'THE PROGRAM WILL NOW CORRECT THIS ERROR',
     &      /,1X,'PLEASE CHECK YOUR INPUT CAREFULLY !!!',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
          NETA=JNMM
          ELSE
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF
        ENDIF

C...
C...INPUT ELEVATION FORCING CONDITIONS ON ELEVATION BOUNDARIES FOR EACH OF THE
C....ELEVATION BOUNDARY FORCING FREQUENCIES FROM UNIT 15 AND OUTPUT TO UNIT 16
C...
      DO I=1,NBFR
        WRITE(16,29) I,BOUNTAG(I)
   29   FORMAT(////,5X,'ELEVATION BOUNDARY TIDAL FORCING FOR',
     &   ' CONSTITUENT NUMBER',I4,1X,'DESIGNATED : ',A5)
        READ(15,'(A10)') ALPHA
        WRITE(16,31) ALPHA
   31   FORMAT(9X,'VERIFICATION OF CONSTITUENT : ',A10,/)
        WRITE(16,30)
   30   FORMAT(14X,'NODE',11X,'AMPL.',9X,'PHASE(DEG)',/)
        DO J=1,NETA
          READ(15,*) EMO(I,J),EFA(I,J)
          WRITE(16,1870) NBD(J),EMO(I,J),EFA(I,J)
 1870     FORMAT(10X,I8,4X,F14.5,4X,F12.3)
          EFA(I,J)=EFA(I,J)*DEG2RAD
          END DO
        END DO

C.....READ THE MINIMUM INNER ANGLE FOR WHICH VELOCITY AT FLOW BOUNDARY NODES
C.....WILL BE ZEROED IN THE TANGENTIAL DIRECTIONS WHEN NORMAL FLOW IS AN
C.....ESSENTIAL B.C.

      READ(15,*) ANGINN
      WRITE(16,1112)
      WRITE(16,7654) ANGINN
7654  FORMAT(//,5X,'ANGINN = ',F8.2,' DEGREES',
     &  /,5X,'ALL FLOW BOUNDARY NODES WITH NORMAL FLOW AS AN ',
     &       'ESSENTIAL B.C. AND ',
     &  /,9X,'INNER ANGLES LESS THAN ANGINN WILL HAVE BOTH NORMAL ',
     &  /,9X,'AND TANGENTIAL VELOCITY COMPONENTS ZEROED',/)
      COSTSET=COS(ANGINN*DEG2RAD)

C...
C...INPUT FLOW BOUNDARY INFORMATION FROM UNIT 14 AND OUTPUT TO UNIT 16
C...

C.....INTERIOR NODES, LBCODE=-1, COS=0, SIN=1
C.....BOUNDARY NODES, LBCODE=LBCODEI=IBTYPE,
C.....COS & SIN DETERMINED FROM NORMAL DIRECTION IN ALL CASES, ALTHOUGH THIS
C.......INFORMATION IS ONLY USED WHEN NORMAL FLOW IS AN ESSENTIAL B.C. AND
C.......FREE TANGENTIAL SLIP IS ALLOWED.
```

```
C.....INPUT THE TOTAL NUMBER OF FLOW BOUNDARY SEGMENTS

      WRITE(16,1112)
      WRITE(16,1878)
1878  FORMAT(//,1X,'FLOW BOUNDARY INFORMATION ',/)
      READ(14,*) NBOU

C.....CHECK THAT THE DIMENSIONING PARAMETER MNBOU HAS BEEN SET PROPERLY

      IF(NBOU.GT.MNBOU) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9934)
        WRITE(16,9934)
 9934   FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNBOU IS EXCEEDED BY THE ',
     &         'INPUT PARAMETER NBOU',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF
      WRITE(16,1879) NBOU
 1879 FORMAT(//,5X,'THE TOTAL NUMBER OF FLOW BOUNDARY SEGMENTS = ',I5)

C.....INPUT THE TOTAL NUMBER OF FLOW BOUNDARY NODES

      READ(14,*) NVEL

C.....CHECK THAT THE DIMENSIONING PARAMETER MNVEL HAS BEEN SET PROPERLY

      IF(NVEL+1.GT.MNVEL) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9935)
        WRITE(16,9935)
9935    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNVEL IS EXCEEDED BY THE ',
     &         'INPUT PARAMETER NVEL+1',
     &    /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF
      WRITE(16,1881) NVEL
1881  FORMAT(/,5X,'THE TOTAL NUMBER OF FLOW BOUNDARY NODES = ',I5)

C.....INPUT THE NUMBER OF NODES IN THE NEXT FLOW BOUNDARY SEGMENT
C.....AND THE BOUNDARY TYPE

      JGW=0
      JME=0
      NFLUXF=0
      NFLUXB=0
      NFLUXIB=0
      NVELEXT=0
      DO K=1,NBOU
        READ(14,*) NVELL(K),IBTYPE
C.......CHECK THAT IBTYPE PARAMETER HAS BEEN SET PROPERLY
        IF(    (IBTYPE.NE.0).AND.(IBTYPE.NE.10).AND.(IBTYPE.NE.20)
     &    .AND.(IBTYPE.NE.1).AND.(IBTYPE.NE.11).AND.(IBTYPE.NE.21)
     &    .AND.(IBTYPE.NE.2).AND.(IBTYPE.NE.12).AND.(IBTYPE.NE.22)
     &    .AND.(IBTYPE.NE.3).AND.(IBTYPE.NE.13).AND.(IBTYPE.NE.23)
     &    .AND.(IBTYPE.NE.4).AND.(IBTYPE.NE.24).AND.(IBTYPE.NE.30)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9985) K
          WRITE(16,9985) K
 9985     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &      //,1X,'THE FLOW BOUNDARY TYPE PARAMETER IBTYPE ',
     &           'HAS NOT BEEN CORRECTLY SET FOR ',
     &      /,1X,'FLOW BOUNDARY SEGMENT NO. ',I8,
     &      /,1X,'USER MUST CORRECT UNIT 14 INPUT FILE',
     &      //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
          STOP
        ENDIF
C.......WRITE OUT INFORMATION TO UNIT 16
        IF((IBTYPE.EQ.4).OR.(IBTYPE.EQ.24)) THEN
```

```
          WRITE(16,28) K,NVELL(K),K,2*NVELL(K)
   28     FORMAT(///,5X,'TOTAL NUMBER OF PAIRS FOR FLOW BOUNDARY',
     &                ' SEGMENT',2X,I2,2X,'=',2X,I5,/,
     &              5X,'TOTAL NUMBER OF NODES FOR FLOW BOUNDARY',
     &                ' SEGMENT',2X,I2,2X,'=',2X,I5)
        ELSE
          WRITE(16,128) K,NVELL(K)
  128     FORMAT(///,5X,'TOTAL NUMBER OF NODES FOR FLOW BOUNDARY',
     &                ' SEGMENT',2X,I2,2X,'=',2X,I5)
        ENDIF
C.......CONTINUE PROCESSING FLOW BOUNDARY INFORMATION
        IF(IBTYPE.EQ.0) THEN
          WRITE(16,2340)
 2340     FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'NO NORMAL FLOW AS AN ESSENTIAL B.C.',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
          ENDIF
        IF(IBTYPE.EQ.1) THEN
          WRITE(16,2341)
 2341     FORMAT(5X,'THIS SEGMENT IS AN INTERNAL BOUNDARY WITH:',/,
     &           7X,'NO NORMAL FLOW AS AN ESSENTIAL B.C.',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.2) THEN
          NFLUXF=1
          WRITE(16,2342)
 2342     FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'SPECIFIED NORMAL FLOW AS AN ESSENTIAL B.C.',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.3) THEN
          NFLUXB=1
          WRITE(16,2344)
 2344     FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'A BARRIER WHICH ALLOWS FREE SURFACE',
     &           ' SUPERCRITICAL OUTFLOW',/,
     &           7X,'FROM THE DOMAIN ONCE THE BARRIER HAS BEEN',
     &           ' OVERTOPPED',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.4) THEN
          NFLUXIB=1
          WRITE(16,2345)
 2345     FORMAT(5X,'THIS SEGMENT IS AN INTERNAL BARRIER BOUNDARY:',/,
     &           7X,'WITH CROSS BARRIER FLOW TREATED AS AN ESSENTIAL ',
     &           ' NORMAL FLOW BOUNDARY CONDITION',/,
     &           7X,'WHICH LEAVES/ENTERS THE DOMAIN ON ONE SIDE OF ',
     &           ' THE BARRIER AND ENTERS/LEAVES THE DOMAIN ON THE ',/,
     &           7X,'CORRESPONDING OPPOSITE SIDE OF THE BARRIER ',/,
     &           7X,'FLOW RATE AND DIRECTION ARE BASED ON BARRIER ',
     &           ' HEIGHT, SURFACE WATER ELEVATION',/,
     &           7X,'ON BOTH SIDES OF THE BARRIER, BARRIER COEFFICIENT',
     &           ' AND THE APPROPRIATE BARRIER FLOW FORMULA',/,
     &           7X,'FREE TANGENTIAL SLIP IS ALLOWED',/)
        ENDIF
        IF(IBTYPE.EQ.10) THEN
          WRITE(16,2350)
 2350     FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'NO NORMAL FLOW AS AN ESSENTIAL B.C.',/,
     &           7X,'AND NO TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.11) THEN
          WRITE(16,2351)
 2351     FORMAT(5X,'THIS SEGMENT IS AN INTERNAL BOUNDARY WITH:',/,
     &           7X,'NO NORMAL FLOW AS AN ESSENTIAL B.C.',/,
     &           7X,'AND NO TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.12) THEN
          NFLUXF=1
          WRITE(16,2352)
```

```
2352      FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'SPECIFIED NORMAL FLOW AS AN ESSENTIAL B.C.',/,
     &           7X,'AND NO TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.13) THEN
          NFLUXB=1
          WRITE(16,2354)
2354      FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'A BARRIER WHICH ALLOWS FREE SURFACE',
     &           ' SUPERCRITICAL OUTFLOW',/,
     &           7X,'FROM THE DOMAIN ONCE THE BARRIER HAS BEEN',
     &           ' OVERTOPPED',/,
     &           7X,'AND NO TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.20) THEN
          WRITE(16,2360)
2360      FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'NO NORMAL FLOW AS A NATURAL B.C.',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.21) THEN
          WRITE(16,2361)
2361      FORMAT(5X,'THIS SEGMENT IS AN INTERNAL BOUNDARY WITH:',/,
     &           7X,'NO NORMAL FLOW AS A NATURAL B.C.',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.22) THEN
          NFLUXF=1
          WRITE(16,2362)
2362      FORMAT(5X,'THIS SEGMENT IS A EXTERNAL BOUNDARY WITH:',/,
     &           7X,'SPECIFIED NORMAL FLOW AS A NATURAL B.C.',/,
     &           7X,'AND FREE TANGENTIAL SLIP',/)
        ENDIF
        IF(IBTYPE.EQ.23) THEN
          NFLUXB=1
          WRITE(16,2356)
2356      FORMAT(5X,'THIS SEGMENT IS AN EXTERNAL BOUNDARY WITH:',/,
     &           7X,'A BARRIER WHICH ALLOWS FREE SURFACE',
     &           ' SUPERCRITICAL OUTFLOW',/,
     &           7X,'FROM THE DOMAIN ONCE THE BARRIER HAS BEEN',
     &           ' OVERTOPPED',/,
     &           7X,' IMPLEMENTED AS A NATURAL BOUNDARY CONDITION'
     &           ,7X,'FREE TANGENTIAL SLIP IS ALSO ALLOWED',/)
        ENDIF
        IF(IBTYPE.EQ.24) THEN
          NFLUXIB=1
          WRITE(16,2357)
2357      FORMAT(5X,'THIS SEGMENT IS AN INTERNAL BARRIER BOUNDARY:',/,
     &           7X,'WITH CROSS BARRIER FLOW TREATED AS A NATURAL ',
     &           ' NORMAL FLOW BOUNDARY CONDITION',/,
     &           7X,'WHICH LEAVES/ENTERS THE DOMAIN ON ONE SIDE OF ',
     &           ' THE BARRIER AND ENTERS/LEAVES THE DOMAIN ON THE ',/,
     &           7X,'CORRESPONDING OPPOSITE SIDE OF THE BARRIER ',/,
     &           7X,'FLOW RATE AND DIRECTION ARE BASED ON BARRIER ',
     &           ' HEIGHT, SURFACE WATER ELEVATION',/,
     &           7X,'ON BOTH SIDES OF THE BARRIER, BARRIER COEFFICIENT',
     &           ' AND THE APPROPRIATE BARRIER FLOW FORMULA',/,
     &           7X,'FREE TANGENTIAL SLIP IS ALLOWED',/)
        ENDIF
        IF(IBTYPE.EQ.30) THEN
          NFLUXRBC=1
          WRITE(16,2355)
2355      FORMAT(5X,'THIS SEGMENT IS AN OUTWARD RADIATING BOUNDARY:',/,
     &           7X,'NORMAL FLUX IS A NATURAL B.C. IN GWCE',/,
     &           7X,'NORMAL AND TANGENTIAL VELOCITY ARE COMPUTED FROM ',
     &              'THE MOMENTUM EQNS.',/)
        ENDIF


C.....INPUT INFORMATION FOR VARIOUS TYPES OF FLOW BOUNDARY SEGMENTS
```

```
C.......INPUT THE STANDARD NODE NUMBERS FOR THE Kth FLOW BOUNDARY SEGMENT
        IF((IBTYPE.NE.3).AND.(IBTYPE.NE.13).AND.(IBTYPE.NE.23).AND.
     &   (IBTYPE.NE.4).AND.(IBTYPE.NE.24)) THEN
          DO I=1,NVELL(K)
            READ(14,*) NBVV(K,I)
          END DO
          NPRBI=1
        ENDIF
C.......INPUT THE NODE NUMBERS FOR THE Kth EXTERNAL BARRIER BOUNDARY SEGMENT
C........ALSO INPUT THE ELEVATION OF THE EXTERNAL BARRIER NODES ABOVE
C........THE GEOID AND THE COEFFICIENT OF FREE SURFACE SUPERCRITICAL
C........FLOW ALONG WITH EACH EXTERNAL BARRIER BOUNDARY NODE FROM UNIT 14
        IF((IBTYPE.EQ.3).OR.(IBTYPE.EQ.13).OR.(IBTYPE.EQ.23)) THEN
          DO I=1,NVELL(K)
            READ(14,*) NBVV(K,I),BARLANHTR(I),BARLANCFSPR(I)
          END DO
          NPRBI=1
        ENDIF
C.......INPUT THE NODE NUMBERS FOR THE Kth INTERNAL BARRIER BOUNDARY SEGMENT
C........ALSO INPUT CONNECTION NODE NUMBER AND ELEVATION OF THE INTERNAL BARRIER
C........NODES ABOVE THE GEOID AND THE COEFFICIENTS OF FREE SURFACE SUPERCRITICAL
C........AND SUBCRITICAL FLOW ALONG WITH EACH INTERNAL BARRIER BOUNDARY NODE FROM
C........UNIT 14
        IF((IBTYPE.EQ.4).OR.(IBTYPE.EQ.24)) THEN
          DO I=1,NVELL(K)
            READ(14,*) NBVV(K,I),IBCONNR(I),BARINHTR(I),BARINCFSBR(I)
     &                      ,BARINCFSPR(I)
          END DO
          NPRBI=2
        ENDIF

C.....PROCESS INFORMATION FOR VARIOUS TYPES OF FLOW BOUNDARY SEGMENTS

        DO IPRBI=1,NPRBI

C.........LOAD PAIRED NODES INTO PRIMARY PROCESSING VECTORS AND RESET
C..........CONNECTING NODES FOR BACK FACE
C..........THUS BACK/CONNECTING NODES ARE BEING LOADED AS PRIMARY NODES
C..........AND FRONT NODES ARE RELOADED AS CONNECTING NODES
C..........NOTE THAT THE CLOCKWISE ORIENTATION OF ISLAND TYPE BOUNDARIES
C..........IS BEING MAINTAINED WHEN BACK NODES ARE RELOADED AS PRIMARY NODES
C..........ADDITIONAL INTERNAL BARRIER BOUNDARY INFORMATION IS ALSO RESET
          IF(IPRBI.EQ.2) THEN
            DO I=1,NVELL(K)
              NTRAN1(I)=NBVV(K,I)
              NTRAN2(I)=IBCONNR(I)
              BTRAN3(I)=BARINHTR(I)
              BTRAN4(I)=BARINCFSBR(I)
              BTRAN5(I)=BARINCFSPR(I)
            END DO
            DO I=1,NVELL(K)
              NBVV(K,I)=NTRAN2(NVELL(K)+1-I)
              IBCONNR(I)=NTRAN1(NVELL(K)+1-I)
              BARINHTR(I)=BTRAN3(NVELL(K)+1-I)
              BARINCFSBR(I)=BTRAN4(NVELL(K)+1-I)
              BARINCFSPR(I)=BTRAN5(NVELL(K)+1-I)
            END DO
          ENDIF

C.........WRITE OUT ADDITIONAL HEADER FOR INTERNAL BARRIER BOUNDARIES

          IF((IBTYPE.EQ.4).OR.(IBTYPE.EQ.24)) THEN
            IF(IPRBI.EQ.1) THEN
              WRITE(16,1842)
 1842         FORMAT(/,5X,'FRONT FACE OF INTERNAL BARRIER BOUNDARY',/)
            ELSE
              WRITE(16,1843)
 1843         FORMAT(/,5X,'BACK FACE OF INTERNAL BARRIER BOUNDARY',/)
            ENDIF
          ENDIF
```

```
C.........WRITE OUT GENERAL HEADER FOR BOUNDARY INFORMATION

          WRITE(16,1841)
 1841     FORMAT('     JGW     JME     ME2GW    NODE #  BNDRY CODE    INNER',
     &         ' ANGLE',7X,'COS',13X,'SIN',9X,'0.667*BNDRY LEN',/)

C.........COMPLETE THE BOUNDARY ARRAY FOR THE Kth FLOW BOUNDARY SEGMENT

          NBVV(K,0)=NBVV(K,1)                              !UNCLOSED EXTERNAL
          IF((IBTYPE.EQ.1).OR.(IBTYPE.EQ.11).OR.(IBTYPE.EQ.21)) THEN
            IF(NBVV(K,NVELL(K)).NE.NBVV(K,1)) THEN    !CLOSE AN UNCLOSED INTERNAL
              NVELL(K)=NVELL(K)+1
              NBVV(K,NVELL(K))=NBVV(K,1)
              ENDIF
            ENDIF
          IF(NBVV(K,NVELL(K)).EQ.NBVV(K,1)) THEN      !CLOSED EXTERNAL OR INTERNAL
            NBVV(K,0)=NBVV(K,NVELL(K)-1)
            ENDIF
          NBVV(K,NVELL(K)+1)=NBVV(K,NVELL(K))

C.........PUT BOUNDARY INFORMATION INTO 2 TYPES OF ARRAYS, ONE FOR THE GWCE B.C.
C.........AND ONE FOR THE MOMENTUM EQUATION B.C.
C.........THE GWCE ARRAYS INCLUDE EVERY NODE IN THE UNIT 14 FILE, I.E., NODES
C.........ARE REPEATED WHERE SPECIFIED NORMAL FLOW AND NO NORMAL FLOW BOUNDARIES
C.........MEET AND AT THE BEGINNING AND END OF CLOSED EXTERNAL BOUNDARIES AND
C.........ISLANDS.
C.........THE MOMENTUM EQUATION ARRAYS ARE KEYED TO THE GWCE ARRAYS VIA THE
C.........ARRAY ME2GW WHICH INDICATES THE LOCATION IN THE GWCE ARRAYS THAT
C.........THE APPROPRIATE M.E. VALUE LIES.
C.........THE M.E. ARRAYS DO NOT REPEAT NODES THAT ARE DUPLICATED IN THE
C.........UNIT 14 FILE, I.E., WHEN SPECIFIED NORMAL FLOW AND NO NORMAL FLOW
C.........BOUNDARIES MEET, THE SPECIFIED NORMAL FLOW BOUNDARY CONDITION TAKES
C.........PRECEDENT.  ALSO THE BEGINNING AND ENDING NODES OF CLOSED EXTERNAL
C.........AND ISLAND BOUNDARIES ARE NOT REPEATED.

          DO I=1,NVELL(K)

C.........SET UP THE GWCE BOUNDARY ARRAYS WHICH CONSIST OF
C.........BOUNDARY NODE NUMBERS
C.........BOUNDARY CODES
C.........0.66667*LENGTH OF EACH BOUNDARY SEGMENT.  NOTE, THE LENGTH OF THE LAST
C.........BOUNDARY SEGMENT ON EACH BOUNDARY SHOULD BE ZERO

            JGW=JGW+1
            IF((IBTYPE.EQ.0).OR.(IBTYPE.EQ.10).OR.(IBTYPE.EQ.20).OR.
     &         (IBTYPE.EQ.2).OR.(IBTYPE.EQ.12).OR.(IBTYPE.EQ.22).OR.
     &         (IBTYPE.EQ.3).OR.(IBTYPE.EQ.13).OR.(IBTYPE.EQ.23).OR.
     &         (IBTYPE.EQ.30)) THEN
              NVELEXT=NVELEXT+1
              ENDIF
            NBV(JGW)=NBVV(K,I)
            NBVI=NBVV(K,I)
            NBVJ=NBVV(K,I+1)
            DELX=X(NBVJ)-X(NBVI)
            DELY=Y(NBVJ)-Y(NBVI)
            BNDLEN2O3(JGW)=2.D0*(SQRT(DELX*DELX+DELY*DELY))/3.D0

C...........COMPUTE THE INCLUDED ANGLE AND TEST TO DETERMINE WHETHER TO ZERO
C...........TANGENTIAL VELOCITIES
C...........NOTE:.IMPLEMENTATION FOR ICS=2 REQUIRES COMPUTING ALL COORDINATES IN
C...........A LOCALIZED SYSTEM (I.E. THE TRANSFORMATION IS CENTERED AT X0,Y0)

            IF(ICS.EQ.1) THEN
              XL0=X(NBVV(K,I))
              XL1=X(NBVV(K,I-1))
              XL2=X(NBVV(K,I+1))
              YL0=Y(NBVV(K,I))
              YL1=Y(NBVV(K,I-1))
              YL2=Y(NBVV(K,I+1))
```

```
            ELSE
              CALL CPP(XL0,YL0,SLAM(NBVV(K,I)),SFEA(NBVV(K,I)),
     &                       SLAM(NBVV(K,I)),SFEA(NBVV(K,I)))
              CALL CPP(XL1,YL1,SLAM(NBVV(K,I-1)),SFEA(NBVV(K,I-1)),
     &                       SLAM(NBVV(K,I)),SFEA(NBVV(K,I)))
              CALL CPP(XL2,YL2,SLAM(NBVV(K,I+1)),SFEA(NBVV(K,I+1)),
     &                       SLAM(NBVV(K,I)),SFEA(NBVV(K,I)))
            ENDIF

C...........NOTE: INTERIOR ANGLE AT ENDS OF BOUNDARIES MUST BE EQUAL, EITHER:
C...........A FICTICIOUSLY LARGE VALUE IF THE BOUNDARY IS NOT CLOSED OR
C...........A TRUE VALUE IF THE BOUNDARY IS CLOSED

            THETA=0.
            IF((I.EQ.1).AND.(NBVV(K,I).EQ.NBVV(K,I-1))) THEN
              THETA1=-9999999.
              THETA=THETA1
              COSTHETA1=COSTSET
              COSTHETA=COSTHETA1
              CROSS1=0.
              CROSS=CROSS1
            ENDIF
            IF(I.EQ.NVELL(K)) THEN
              THETA=THETA1
              COSTHETA=COSTHETA1
              CROSS=CROSS1
            ENDIF
            IF(THETA.EQ.0.) THEN
              VL1X=XL1-XL0
              VL1Y=YL1-YL0
              VL2X=XL2-XL0
              VL2Y=YL2-YL0
              DOTVEC=VL1X*VL2X+VL1Y*VL2Y
              VECNORM=(SQRT(VL1X**2+VL1Y**2))*(SQRT(VL2X**2+VL2Y**2))
              COSTHETA=DOTVEC/VECNORM
              IF(COSTHETA.GT.1.0) COSTHETA=1.0
              IF(COSTHETA.LT.-1.0) COSTHETA=-1.0
              THETA=RAD2DEG*ACOS(COSTHETA)
              CROSS=-VL1X*VL2Y+VL2X*VL1Y
              IF(CROSS.LT.0) THETA=360.-THETA
              IF(I.EQ.1) THEN
                THETA1=THETA
                COSTHETA1=COSTHETA
                CROSS1=CROSS
              ENDIF
            ENDIF

C...........CHECK WHETHER ANGLE IS LESS THAN MINIMUM ANGLE, IF SO CHANGE THE
C...........BOUNDARY CODE TO ZERO TANGENTIAL VELOCITIES

            LBCODEI(JGW)=IBTYPE
            IF((COSTHETA.GT.COSTSET).AND.(CROSS.GT.0.0)) THEN
              IF(IBTYPE.EQ.0) LBCODEI(JGW)=10
              IF(IBTYPE.EQ.1) LBCODEI(JGW)=11
              IF(IBTYPE.EQ.2) LBCODEI(JGW)=12
              IF(IBTYPE.EQ.3) LBCODEI(JGW)=13
              IF((IBTYPE.GE.0).AND.(IBTYPE.LE.3)) THEN
                WRITE(16,1856) NBVV(K,I),THETA
 1856           FORMAT(2X,I7,4X,'THE INNER ANGLE = ',F8.2,1X,
     &                 'TANGENTIAL SLIP WILL BE ZEROED')
              ENDIF
            ENDIF

C...........COMPUTE COS AND SIN OF OUTWARD NORMAL REGARDLESS OF BOUNDARY TYPE

            X1=X(NBVV(K,I-1))
            X2=X(NBVV(K,I+1))
            Y1=Y(NBVV(K,I-1))
            Y2=Y(NBVV(K,I+1))
            XL=SQRT((X1-X2)**2+(Y1-Y2)**2)
```

```
            CSII(JGW)=SFAC(NBVV(K,I))*(Y2-Y1)/XL
            SIII(JGW)=(X1-X2)/XL

C...........SET UP THE MOMENTUM EQUATION BOUNDARY ARRAY WHICH CONSISTS OF
C...........A KEY TO THE GWCE BOUNDARY CONDITION ARRAY

            IF(I.EQ.1) THEN                !DEAL WITH FIRST NODE IN L.B. SEG
              IF(JGW.EQ.1) THEN                          !VERY FIRST L.B. SEG
                JME=JME+1                                !M.E. USES IT
                ME2GW(JME)=JGW
              ENDIF
              IF(JGW.NE.1) THEN
                IF(NBV(JGW).NE.NBV(JGW-1)) THEN      !L.B. SEGS DON'T OVERLAP
                  JME=JME+1                              !M.E. USES IT
                  ME2GW(JME)=JGW
                ENDIF
                IF(NBV(JGW).EQ.NBV(JGW-1)) THEN      !L.B. SEGS OVERLAP
                  IF((LBCODEI(JGW).EQ.2) .OR.              ! M.E. USES IT ONLY
     &               (LBCODEI(JGW).EQ.12).OR.            ! IF IT IS
     &               (LBCODEI(JGW).EQ.22).OR.            ! SPECIFIED FLOW,
     &               (LBCODEI(JGW).EQ.3) .OR.            ! AN OVERFLOW BARRIER
     &               (LBCODEI(JGW).EQ.13).OR.            ! OR A RADIATION
     &               (LBCODEI(JGW).EQ.23).OR.            ! BOUNDARY
     &               (LBCODEI(JGW).EQ.30)) ME2GW(JME)=JGW
                ENDIF
              ENDIF
            ENDIF
            IF((I.GT.1).AND.(I.LT.NVELL(K))) THEN    !IF NOT FIRST OR
              JME=JME+1                              !LAST NODE
              ME2GW(JME)=JGW                         !M.E. USES IT
            ENDIF
            IF(I.EQ.NVELL(K)) THEN         !DEAL WITH LAST NODE ON BOUNDARY
              IF((NBV(JGW).NE.NBVV(K,1)).AND.        !IF UNCLOSED BOUNDARY
     &           (NBV(JGW).NE.NBV(1))) THEN              !M.E. USES IT
                JME=JME+1
                ME2GW(JME)=JGW
              ENDIF
              IF(NBVV(K,I).EQ.NBV(1)) THEN           !IF OVERLAPS WITH VERY FIRST
                IF((LBCODEI(JGW).EQ.2) .OR.              ! L.B. NODE
     &               (LBCODEI(JGW).EQ.12).OR.                ! M.E. USES IT ONLY IF IT IS
     &               (LBCODEI(JGW).EQ.22).OR.            ! SPECIFIED FLOW,
     &               (LBCODEI(JGW).EQ.3) .OR.            ! AN OVERFLOW BARRIER OR
     &               (LBCODEI(JGW).EQ.13).OR.                ! A RADIATION
     &               (LBCODEI(JGW).EQ.23).OR.            ! BOUNDARY
     &               (LBCODEI(JGW).EQ.30)) ME2GW(1)=JGW
              ENDIF
            ENDIF

C...........LOAD EXTERNAL BARRIER BOUNDARY INFORMATION INTO THE CORRECT VECTORS
            IF((IBTYPE.EQ.3).OR.(IBTYPE.EQ.13).OR.(IBTYPE.EQ.23)) THEN
              BARLANHT(JGW)=BARLANHTR(I)
              BARLANCFSP(JGW)=BARLANCFSPR(I)
            ENDIF

C...........LOAD INTERNAL BARRIER BOUNDARY INFORMATION INTO THE CORRECT VECTORS
            IF((IBTYPE.EQ.4).OR.(IBTYPE.EQ.24)) THEN
              IBCONN(JGW)=IBCONNR(I)
              BARINHT(JGW)=BARINHTR(I)
              BARINCFSB(JGW)=BARINCFSBR(I)
              BARINCFSP(JGW)=BARINCFSPR(I)
            ENDIF


C...........WRITE OUT BOUNDARY CONDITION ARRAY INFORMATION

            WRITE(16,1857) JGW,JME,ME2GW(JME),NBV(JGW),LBCODEI(JGW),
     &                     THETA,CSII(JGW),SIII(JGW),BNDLEN2O3(JGW)
 1857       FORMAT(1X,I6,1X,I6,1X,I6,3X,I6,3X,I4,9X,F8.2,2X,E16.8,1X,
     &                                          E16.8,2X,E16.8)
```

```
C..........CHECK EXTERNAL BARRIER HEIGHTS AGAINST DEPTHS
          IF((IBTYPE.EQ.3).OR.(IBTYPE.EQ.13).OR.(IBTYPE.EQ.23)) THEN
            IF(BARLANHT(JGW).LT.-DP(NBV(JGW))) THEN
              IF(NSCREEN.EQ.1) WRITE(6,8367) JGW,NBV(JGW),
     &           BARLANHT(JGW),DP(NBV(JGW))
              WRITE(16,8367) JGW,NBV(JGW),BARLANHT(JGW),DP(NBV(JGW))
 8367         FORMAT(////,1X,'!!!!!!!!!!  FATAL INPUT ERROR   !!!'
     &           ,'!!!!!!!!!!',//,
     &           1X,'AT BOUNDARY NODE NO.',I6,' (GLOBAL NODE NO.',
     &           I6, ' AND OF EXTERNAL BARRIER TYPE) ',/,
     &           2X,'THE EXTERNAL BARRIER HEIGHT = ',E12.5,
     &           2X,'IS EXCEEDED BY THE DEPTH SPECIFIED AT ',/,2X
     &           ,'THE ASSOCIATED GLOBAL NODE = ',E12.5,/,2X,
     &           'USER MUST SPECIFY CONSISTENT BARRIER HEIGHTS',
     &           ' AND DEPTHS')
              IF(NSCREEN.EQ.1) WRITE(6,9973)
              WRITE(16,9973)
              STOP
            ENDIF
          ENDIF
C..........CHECK INTERNAL BARRIER HEIGHTS AGAINST DEPTHS
          IF((IBTYPE.EQ.4).OR.(IBTYPE.EQ.24)) THEN
            IF(BARINHT(JGW).LT.-DP(NBV(JGW))) THEN
              IF(NSCREEN.EQ.1) WRITE(6,8368) JGW,NBV(JGW),
     &           BARINHT(JGW),DP(NBV(JGW))
              WRITE(16,8368) JGW,NBV(JGW),BARINHT(JGW),DP(NBV(JGW))
 8368         FORMAT(////,1X,'!!!!!!!!!!  FATAL INPUT ERROR   !!!'
     &           ,'!!!!!!!!!!',//,
     &           1X,'AT BOUNDARY NODE NO.',I6,' (GLOBAL NODE NO. ',
     &           I6,' AND OF INTERNAL BARRIER TYPE) ',/,
     &           2X,'THE INTERNAL BARRIER HEIGHT = ',E12.5,
     &           2X,'IS EXCEEDED BY THE DEPTH SPECIFIED AT ',/,2X
     &           ,'THE ASSOCIATED GLOBAL NODE = ',E12.5,/,2X,
     &           'USER MUST SPECIFY CONSISTENT BARRIER HEIGHTS',
     &           ' AND DEPTHS')
              IF(NSCREEN.EQ.1) WRITE(6,9973)
              WRITE(16,9973)
              STOP
            ENDIF
          ENDIF

C..........CHECK FOR OVERLAPPING OF AN INTERNAL BARRIER BOUNDARY WITH
C..........ANY EXTERNAL BARRIER BOUNDARY. IF THIS DOES OCCUR, TAKE
C..........APPROPRIATE ACTION
          IF((IBTYPE.EQ.4).OR.(IBTYPE.EQ.24)) THEN
            DO ICK=1,NVELEXT
C.............CHECK IF OVERLAP EXISTS
              IF(NBV(ICK).EQ.NBV(JGW)) THEN
C..............CHECK FOR ILLEGAL OVERLAPS
                IF((LBCODEI(ICK).EQ.2).OR.(LBCODEI(ICK).EQ.3).OR.
     &             (LBCODEI(ICK).EQ.12).OR.(LBCODEI(ICK).EQ.13).OR.
     &             (LBCODEI(ICK).EQ.22).OR.(LBCODEI(ICK).EQ.23)) THEN
                  IF(NSCREEN.EQ.1) WRITE(6,8567) JGW,NBV(JGW),ICK,
     &               NBV(ICK)
                  WRITE(16,8567) JGW,NBV(JGW),ICK,NBV(ICK)
 8567             FORMAT(////,1X,'!!!!!!!!!!  FATAL INPUT ERROR   !!!'
     &               ,'!!!!!!!!!!',//,
     &               1X,'BOUNDARY NODE NO. ',I6,' (GLOBAL NODE NO. ',
     &               I9, 'AND OF INTERNAL BARRIER TYPE) ',/,
     &               2X,'OVERLAPS BOUNDARY NODE NO.',I6,' (GLOBAL NODE'
     &               ,' NO.',I6,' )',/,
     &               2X,'THIS IS AN ILLEGAL TYPE OVERLAP !! - INTERNAL '
     &               ,'BARRIER BOUNDARIES CAN ONLY OVERLAP WITH ',
     &               'NO NORMAL FLOW EXTERNAL BOUNDARIES',/
     &               2X,'(I.E. IBTYPE=0,10,20)')
                  IF(NSCREEN.EQ.1) WRITE(6,9973)
                  WRITE(16,9973)
                  STOP
                ENDIF
C..............CHECK FOR OVERLAPS WHICH REQUIRE ADJUSTMENTS OF BOUNDARY
```

```
C.................CODE ON THE EXTERNAL BOUNDARY
                  IF((IBTYPE.EQ.4).AND.(LBCODEI(ICK).EQ.0)) THEN
                    WRITE(16,8568) JGW,ICK,ICK
 8568               FORMAT(1X,'DUE TO LEGAL OVERLAPPING OF ',
     &                'BOUNDARY NODE',I7,' (WHICH IS AN ESSENTIAL INTER'
     &                ,'NAL BARRIER BOUNDARY NODE)', /,2X,
     &                'AND BOUNDARY NODE',I7,' (WHICH IS AN ESSENTIAL ',
     &                'EXTERNAL NO NORMAL FLOW WITH SLIP BOUNDARY',
     &                ' NODE),',/,2X,
     &                'THE BOUNDARY TYPE FOR BOUNDARY NODE ',I7,
     &                ' IS BEING RESET TO IBTYPE=20',/,2X,
     &                '(NATURAL NO NORMAL FLOW WITH SLIP BOUNDARY) ')
                    LBCODEI(ICK)=20
                  ENDIF
                  IF((IBTYPE.EQ.4).AND.(LBCODEI(ICK).EQ.10)) THEN
                    WRITE(16,8569) JGW,ICK,ICK
 8569               FORMAT(1X,'DUE TO LEGAL OVERLAPPING OF ',
     &                'BOUNDARY NODE ',I7,' (WHICH IS AN ESSENTIAL INTER'
     &                ,'NAL BARRIER BOUNDARY NODE)', /,2X,
     &                'AND BOUNDARY NODE',I7,' (WHICH IS AN ESSENTIAL ',
     &                'EXTERNAL NO NORMAL FLOW WITH NO SLIP BOUNDARY',
     &                ' NODE),',/,2X,
     &                'THE BOUNDARY TYPE FOR BOUNDARY NODE ',I7,
     &                ' IS BEING RESET TO IBTYPE=20',/,2X,
     &                '(NATURAL NO NORMAL FLOW WITH SLIP BOUNDARY) ')
                    LBCODEI(ICK)=20
                  ENDIF
                  IF((IBTYPE.EQ.24).AND.(LBCODEI(ICK).EQ.10)) THEN
                    WRITE(16,8570) JGW,ICK,ICK
 8570               FORMAT(1X,'DUE TO LEGAL OVERLAPPING OF ',
     &                'BOUNDARY NODE',I7,' (WHICH IS A NATURAL INTERNAL'
     &                ,' BARRIER BOUNDARY NODE)', /,2X,
     &                'AND BOUNDARY NODE',I7,' (WHICH IS AN ESSENTIAL ',
     &                'EXTERNAL NO NORMAL FLOW WITH NO SLIP BOUNDARY',
     &                ' NODE),',/,2X,
     &                'THE BOUNDARY TYPE FOR BOUNDARY NODE',I7,
     &                ' IS BEING RESET TO IBTYPE=0',/,2X,
     &                '(ESSENTIAL NO NORMAL FLOW WITH SLIP BOUNDARY) ')
                    LBCODEI(ICK)=0
                  ENDIF
                ENDIF
              END DO
            ENDIF

          END DO
        END DO
      END DO


C.....ONCE ALL FLOW BOUNDARY NODES HAVE BEEN PROCESSED, CHECK TO MAKE SURE
C.....THAT JGW LE MNVEL.  NOTE, JME MUST BE < JGW.

      IF(MNVEL.LT.JGW) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9947)
        WRITE(16,9947)
 9947   FORMAT(////,1X,'!!!!!!!!!!  FATAL INPUT ERROR   !!!!!!!!!!!!',
     &    //,1X,'THE DIMENSION PARAMETER MNVEL IS LESS THAN ',
     &          'THE TOTAL NUMBER OF FLOW BOUNDARY NODES',
     &    /,1X,'FROM ALL THE SPECIFIED FLOW SEGMENTS COMBINED',/)
        IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
        ENDIF

      NVEL=JGW
      NVELME=JME

C.....TRANSFER FLOW BOUNDARY INFORMATION INTO NODAL ARRAYS

      DO I=1,NP
```

```
        LBCODE(I)=0
        CSI(I)=0.
        SII(I)=1.
        END DO

      DO I=1,NVELME
        J=ME2GW(I)
        LBCODE(NBV(J))=LBCODEI(J)
        CSI(NBV(J))=CSII(J)
        SII(NBV(J))=SIII(J)
        QNAM(1,J)=0.
        QNPH(1,J)=0.
        END DO

C...IF ANY NON ZERO NORMAL FLOW BOUNDARIES WERE SPECIFIED, (NFLUXF=1)
C.....READ FORCING INFORMATION FROM UNIT 15 FILE

      IF(NFLUXF.EQ.1) THEN

C.....INPUT FROM THE NUMBER OF FREQUENCIES PRESENT IN NORMAL FLOW FORCING
C......DATA.  IF THIS = 0, NORMAL FLOW DATA IS READ IN FROM THE FORT.20 FILE.

        READ(15,*) NFFR

C.....CHECK TO ENSURE THAT DIMENSIONING PARAMETER MNFFR IS LARGE ENOUGH

        IF(NFFR.GT.MNFFR) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9904)
          WRITE(16,9904)
 9904     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &    //,1X,'THE DIMENSIONING PARAMETER MNFFR IS EXCEEDED BY THE ',
     &          'INPUT PARAMETER NFFR',
     &     /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &    //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
          STOP
          ENDIF

C.....READ IN AND WRITE OUT INFO ON SPECIFIED NORMAL FLOW BOUNDARIES

        WRITE(16,1112)
        WRITE(16,2200)
 2200   FORMAT(//,1X,'NORMAL FLOW BOUNDARY FORCING INFORMATION ',//)
        IF(NFFR.EQ.0) THEN
          WRITE(16,2201)
 2201     FORMAT(/,5X,'NORMAL FLOW VALUES WILL BE READ FROM UNIT 20 ',
     &    /,9X,'INTERPOLATION IN TIME IS DONE TO SYNC THE FLOW DATA ',
     &    /,9X,'WITH THE MODEL TIME STEP.')
            READ(15,*) FTIMINC
          ENDIF
        IF(NFFR.NE.0) THEN
          WRITE(16,2202) NFFR
 2202     FORMAT(/,5X,'NUMBER OF PERIODIC NORMAL FLOW CONSTITUENTS =',
     &                                                          I5)
          WRITE(16,2203)
 2203     FORMAT(/,7X,'CONSTITUENT #',4X,'FREQUENCY',4X,'NODAL FACTOR',
     &         3X,'EQU.ARG (DEG)',2X,'CONSTITUENT',/)
          DO I=1,NFFR
            READ(15,'(A5)') FBOUNTAG(I)
            READ(15,*) FAMIG(I),FFF(I),FFACE(I)
            WRITE(16,2204) I,FAMIG(I),FFF(I),FFACE(I),FBOUNTAG(I)
 2204       FORMAT(12X,I2,6X,F16.12,2X,F10.7,2X,F10.3,10X,A5)
            FFACE(I)=FFACE(I)*DEG2RAD
            IF(FAMIG(I).EQ.0.) THEN
              FPER(I)=0.
              ELSE
              FPER(I)=2.D0*PI/FAMIG(I)
              ENDIF
            END DO

C.......INPUT PERIODIC NORMAL FLOW FORCING CONDITIONS ON DESIGNATED FLOW BOUNDARIES
```

```
C........FOR EACH OF THE FORCING FREQUENCIES FROM UNIT 15 AND OUTPUT TO UNIT 16

          DO I=1,NFFR
            WRITE(16,2206) I,FBOUNTAG(I)
 2206       FORMAT(////,5X,'PERIODIC NORMAL FLOW CONSTITUENT ',
     &                     'NUMBER',I4,1X,'DESIGNATED : ',A5)
            READ(15,'(A10)') ALPHA
            WRITE(16,31) ALPHA
            WRITE(16,30)
            DO J=1,NVEL
              IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                              .OR.(LBCODEI(J).EQ.22)) THEN
                READ(15,*) QNAM(I,J),QNPH(I,J)
                WRITE(16,2205) NBV(J),QNAM(I,J),QNPH(I,J)
 2205           FORMAT(10X,I8,4X,F14.5,4X,F12.3)
                QNPH(I,J)=QNPH(I,J)*DEG2RAD
                ENDIF
              END DO
            END DO
          ENDIF
        ENDIF

C...IF ANY EXTERNAL BARRIER BOUNDARIES WERE SPECIFIED, (NFLUXB=1)
C.....WRITE OUT EXTERNAL BARRIER BOUNDARY INFORMATION TO UNIT 16 FILE
C.....NOTE THAT THIS INFORMATION WAS READ IN FROM THE UNIT 14 FILE

      IF(NFLUXB.EQ.1) THEN

C.....WRITE OUT INFO ON SPECIFIED EXTERNAL BARRIER BOUNDARIES

        WRITE(16,1112)
        WRITE(16,2220)
 2220   FORMAT(//,1X,'EXTERNAL BARRIER BOUNDARY INFORMATION ',/)

C.......OUTPUT ELEVATION OF EXTERNAL BARRIER NODES ABOVE THE GEOID AND
C.........THE COEFFICIENT OF FREE SURFACE SUPERCRITICAL FLOW AT
C.........DESIGNATED EXTERNAL BARRIER BOUNDARY NODES TO UNIT 16
        WRITE(16,2224)
 2224   FORMAT(//,9X,'NODE',10X,'BARRIER HEIGHT',
     &                    6X,'SUPER-CRIT. EXTERNAL BAR. COEF.',/)
        DO J=1,NVEL
          IF((LBCODEI(J).EQ.3).OR.(LBCODEI(J).EQ.13)
     &       .OR.(LBCODEI(J).EQ.23)) THEN
          WRITE(16,2225) NBV(J),BARLANHT(J),BARLANCFSP(J)
 2225     FORMAT(5X,I8,6X,F14.5,15X,F12.3)
          ENDIF
        END DO
      ENDIF

C...IF ANY INTERNAL BARRIER BOUNDARIES WERE SPECIFIED, (NFLUXIB=1)
C.....WRITE INTERNAL BARRIER BOUNDARY INFORMATION TO UNIT 16 FILE

      IF(NFLUXIB.EQ.1) THEN

C.....WRITE OUT INFO ON SPECIFIED INTERNAL BARRIER BOUNDARIES

        WRITE(16,1112)
        WRITE(16,2320)
 2320   FORMAT(//,1X,'INTERNAL BARRIER BOUNDARY INFORMATION ',/)

C.......WRITE CONNECTION NODE NUMBER AND ELEVATION OF THE INTERNAL BARRIER
C.........NODES ABOVE THE GEOID AND THE COEFFICIENTS OF FREE SURFACE SUPERCRITICAL
C.........AND SUBCRITICAL FLOW AT DESIGNATED INTERNAL BARRIER BOUNDARY NODES
C.........TO UNIT 16 (NOTE THAT THIS INFORMATION WAS INPUT FROM THE UNIT 14
C.........FILE WITH BOUNDARY NODE INFORMATION)
        WRITE(16,2324)
 2324   FORMAT(//,9X,'NODE',6X,'CONNECTED NODE',6X,'BARRIER HEIGHT',
     &               4X,'SUB-CRIT. INT. BAR. COEF.',
     &               4X,'SUPER-CRIT. INT. BAR. COEF.',/)
        DO J=1,NVEL
```

```
          IF((LBCODEI(J).EQ.4).OR.(LBCODEI(J).EQ.24)) THEN
            WRITE(16,2325) NBV(J),IBCONN(J),BARINHT(J),
     &                     BARINCFSB(J),BARINCFSP(J)
 2325       FORMAT(5X,I8,7X,I8,6X,F14.5,12X,F12.3,17X,F12.3)
          ENDIF
        END DO
      ENDIF

C...
C...READ IN INFORMATION CONCERNING OUTPUT REQUIREMENTS FROM UNIT 15 AND
C...OUTPUT THIS TO UNIT 16
C...
      WRITE(16,1112)
      WRITE(16,3000)
3000  FORMAT(//,1X,'OUTPUT INFORMATION WILL BE PROVIDED AS'
     &  ,' FOLLOWS :')

C...
C...INPUT INFORMATION FOR ELEVATION RECORDING STATIONS
C...

C....READ IN NOUTE,TOUTSE,TOUTFE,NSPOOLE : IF ABS(NOUTE)>0, INTERPOLATED
C....ELEVATIONS AT ELEVATION STATIONS ARE SPOOLED TO UNIT 61 EVERY NSPOOLE
C....TIME STEPS BETWEEN TIMES TOUTSE AND TOUTFE

      READ(15,*) NOUTE,TOUTSE,TOUTFE,NSPOOLE
      WRITE(16,3001) NOUTE
 3001 FORMAT(///,1X,'ELEVATION RECORDING STATION OUTPUT : ',
     &        //,5X,'NOUTE = ',I2)

C....CHECK INPUT PARAMETER NOUTE

      IF(ABS(NOUTE).GT.2) THEN
        IF(NSCREEN.EQ.1) WRITE(6,3002)
        WRITE(16,3002)
 3002   FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &                ' NOUTE',
     &            /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
        IF(NSCREEN.EQ.1) WRITE(6,9973)
        WRITE(16,9973)
        STOP
        ENDIF

C....IF STATION ELEVATION OUTPUT WILL NOT BE GENERATED

      IF(NOUTE.EQ.0) THEN
        WRITE(16,3003)
 3003   FORMAT(/,5X,'NO OUTPUT WILL BE SPOOLED AT ELEVATION ',
     &              'RECORDING STATIONS')
        ENDIF

C....IF STATION ELEVATION OUTPUT WILL BE GENERATED

      IF(NOUTE.NE.0) THEN

C......COMPUTE NTCYSE, NTCYFE, WHICH = TOUTSE AND TOUTFE IN TIMESTEPS

        NTCYSE=INT((TOUTSE-STATIM)*(86400.D0/DTDP)+0.5)
        NTCYFE=INT((TOUTFE-STATIM)*(86400.D0/DTDP)+0.5)
        IF(NTCYFE.GT.NT) NTCYFE=NT

C......COMPUTE NTRSPE = THE NO. OF DATA SETS TO BE SPOOLED TO UNIT 61

        IF(NSPOOLE.EQ.0) NTRSPE=0
        IF(NSPOOLE.NE.0) NTRSPE=INT((NTCYFE-NTCYSE)/NSPOOLE)

C......WRITE TOUTSE,TOUTFE,NTCYSE,NTCYFE,NSPOOLE TO UNIT 16

        WRITE(16,3004) TOUTSE,NTCYSE,TOUTFE,NTCYFE,NSPOOLE
```

```
     &               ' DAY(S) RELATIVE',
     &          /,9X,'TO THE STARTING TIME OR',I9,
     &               ' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'DATA RECORDS WILL STOP AFTER TOUTFE =',F8.3,
     &               ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &            I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 61 EVERY',
     &               ' NSPOOLE =',I8,' TIME STEPS')
        IF(ABS(NOUTE).EQ.1) WRITE(16,3005)
 3005   FORMAT(/,5X,'UNIT 61 FORMAT WILL BE ASCII')
        IF(ABS(NOUTE).EQ.2) WRITE(16,3006)
 3006   FORMAT(/,5X,'UNIT 61 FORMAT WILL BE BINARY')
        ENDIF

C....REGARDLESS OF WHETHER NOUTE=0, READ IN THE NUMBER OF ELEVATION
C....RECORDING STATIONS

      READ(15,*) NSTAE
      WRITE(16,3007) NSTAE
 3007 FORMAT(///,5X,'NUMBER OF INPUT ELEVATION RECORDING STATIONS = ',
     &           I5)
      IF(NSTAE.GT.0) THEN
        IF(ICS.EQ.1) WRITE(16,3008)
 3008   FORMAT(/,7X,'STATION #   ELEMENT',9X,'X',13X,'Y',/)
        IF(ICS.EQ.2) WRITE(16,3009)
 3009   FORMAT(/,5X,'STATION   ELEMENT',3X,'LAMBDA(DEG)',
     &          4X,'FEA(DEG)',10X,'XCP',12X,'YCP',/)
        ENDIF

C....CHECK TO ENSURE THAT THE DIMENSIONING PARAMETER MNSTAE HAS BEEN SET PROPERLY

      IF(NSTAE.GT.MNSTAE) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9938)
        WRITE(16,9938)
9938    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &         //,1X,'THE DIMENSIONING PARAMETER MNSTAE IS EXCEEDED ',
     &              'BY THE INPUT PARAMETER NSTAE ',
     &          /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &       //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF

C....INPUT COORDINATES OF ELEVATION RECORDING STATIONS THEN COMPUTE
C....THE ELEMENT NO. THE STATION LIES IN

      DO I=1,NSTAE
        NNE(I)=0
        IF(ICS.EQ.1) THEN
          READ(15,*) XEL(I),YEL(I)
          ELSE
          READ(15,*) SLEL(I),SFEL(I)
          SLEL(I)=SLEL(I)*DEG2RAD
          SFEL(I)=SFEL(I)*DEG2RAD
          CALL CPP(XEL(I),YEL(I),SLEL(I),SFEL(I),SLAM0,SFEA0)
          ENDIF
        AEMIN=1.0E+25
        KMIN=0
        DO K=1,NE
          N1=NM(K,1)
          N2=NM(K,2)
          N3=NM(K,3)
          X1=X(N1)
          X2=X(N2)
          X3=X(N3)
          X4=XEL(I)
          Y1=Y(N1)
          Y2=Y(N2)
          Y3=Y(N3)
          Y4=YEL(I)
```

```
          A2=(X4-X1)*(Y3-Y1)-(Y4-Y1)*(X3-X1)
          A3=(Y4-Y1)*(X2-X1)-(X4-X1)*(Y2-Y1)
          AA=ABS(A1)+ABS(A2)+ABS(A3)
          AE=ABS(AA-AREAS(K))/AREAS(K)
          IF(AE.LT.AEMIN) THEN
            AEMIN=AE
            KMIN=K
            ENDIF
          IF(AE.LT.1.0E-5) NNE(I)=K
          END DO

        IF(NNE(I).EQ.0) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9784) I
          WRITE(16,9784) I
9784      FORMAT(///,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',//
     &           ,1X,'ELEVATION RECORDING STATION ',I6,' DOES NOT LIE',
     &               ' WITHIN ANY ELEMENT IN THE DEFINED',
     &           /,1X,'COMPUTATIONAL DOMAIN,   PLEASE CHECK THE INPUT',
     &               ' COORDINATES FOR THIS STATION')
          IF(NFOVER.EQ.1) THEN
            IF(NSCREEN.EQ.1) WRITE(6,9790) AEMIN
            WRITE(16,9790) AEMIN
9790        FORMAT(/,1X,'PROGRAM WILL ESTIMATE NEAREST ELEMENT',
     &           /,1X,'PROXIMITY INDEX FOR THIS STATION EQUALS ',E15.6,
     &           //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
            NNE(I)=KMIN
            ELSE
            IF(NSCREEN.EQ.1) WRITE(6,9791) AEMIN
            WRITE(16,9791) AEMIN
9791        FORMAT(/,1X,'PROGRAM WILL NOT CORRECT ERROR ',
     &                 'SINCE NON-FATAL ERROR OVERIDE OPTION, NFOVER,',
     &           /,1X,'HAS BEEN SELECTED EQUAL TO 0',
     &           /,1X,'PROXIMITY INDEX FOR THIS STATION EQUALS ',E15.6,
     &           //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',
     &           //)
            STOP
            ENDIF
          ENDIF

        IF(ICS.EQ.1) THEN
          WRITE(16,1880) I,NNE(I),XEL(I),YEL(I)
1880      FORMAT(8X,I3,6X,I7,2(2X,F14.2))
          ELSE
          WRITE(16,1883) I,NNE(I),SLEL(I)*RAD2DEG,
     &                   SFEL(I)*RAD2DEG,XEL(I),YEL(I)
1883      FORMAT(6X,I3,4X,I7,2(2X,F13.8),2X,2(1X,F13.2))
          ENDIF

C....PRE-COMPUTE INFORMATION REQUIRED TO INTERPOLATE AT ELEV. RECORDING STATIONS

        N1=NM(NNE(I),1)
        N2=NM(NNE(I),2)
        N3=NM(NNE(I),3)
        X1=X(N1)
        X2=X(N2)
        X3=X(N3)
        X4=XEL(I)
        Y1=Y(N1)
        Y2=Y(N2)
        Y3=Y(N3)
        Y4=YEL(I)
        STAIE1(I)=((X4-X3)*(Y2-Y3)+(X2-X3)*(Y3-Y4))/AREAS(NNE(I))
        STAIE2(I)=((X4-X1)*(Y3-Y1)-(Y4-Y1)*(X3-X1))/AREAS(NNE(I))
        STAIE3(I)=(-(X4-X1)*(Y2-Y1)+(Y4-Y1)*(X2-X1))/AREAS(NNE(I))

        END DO

C...
```

```
C...INPUT INFORMATION FOR VELOCITY RECORDING STATIONS
C...

C....READ IN NOUTV,TOUTSV,TOUTFV,NSPOOLV : IF NOUTV<>0,INTERPOLATED VELOCITIES AT
C....VELOCITY STATIONS ARE SPOOLED TO UNIT 62 EVERY NSPOOLV TIME STEPS BETWEEN
C....TIMES TOUTSV AND TOUTFV; IF ABS(NOUTV)=2, OUTPUT WILL BE BINARY

      READ(15,*) NOUTV,TOUTSV,TOUTFV,NSPOOLV
      WRITE(16,3101) NOUTV
 3101 FORMAT(////,1X,'VELOCITY RECORDING STATION OUTPUT : ',
     &          //,5X,'NOUTV = ',I2)

C....CHECK INPUT PARAMETER NOUTV

      IF(ABS(NOUTV).GT.2) THEN
        IF(NSCREEN.EQ.1) WRITE(6,3102)
        WRITE(16,3102)
 3102   FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &                ' NOUTV',
     &            /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
        IF(NSCREEN.EQ.1) WRITE(6,9973)
        WRITE(16,9973)
        STOP
        ENDIF

C....IF STATION VELOCITY OUTPUT WILL NOT BE GENERATED

      IF(NOUTV.EQ.0) THEN
        WRITE(16,3103)
 3103   FORMAT(///,5X,'NO OUTPUT WILL BE SPOOLED AT VELOCITY',
     &               ' RECORDING STATIONS')
        ENDIF

C....IF STATION VELOCITY OUTPUT WILL BE GENERATED

      IF(NOUTV.NE.0) THEN

C......COMPUTE NTCYSV, NTCYFV, WHICH = TOUTSV AND TOUTFV IN TIME STEPS

        NTCYSV=INT((TOUTSV-STATIM)*(86400.D0/DTDP) + 0.5)
        NTCYFV=INT((TOUTFV-STATIM)*(86400.D0/DTDP) + 0.5)
        IF(NTCYFV.GT.NT) NTCYFV=NT

C......CALCULATE NTRSPV = THE NO. OF DATA SETS TO BE SPOOLED TO UNIT 62

        IF(NSPOOLV.EQ.0) NTRSPV=0
        IF(NSPOOLV.NE.0) NTRSPV=INT((NTCYFV-NTCYSV)/NSPOOLV)

C......WRITE NOUTV,TOUTSV,TOUTFV,NTCYSV,NTCYFV,NSPOOLV TO UNIT 16

        WRITE(16,3104) TOUTSV,NTCYSV,TOUTFV,NTCYFV,NSPOOLV
 3104   FORMAT(/,5X,'DATA RECORDS WILL START AFTER TOUTSV =',F8.3,
     &              ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'DATA RECORDS WILL STOP AFTER TOUTFV =',F8.3,
     &              ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 62 EVERY ',
     &              ' NSPOOLV =',I8,' TIME STEPS')
        IF(ABS(NOUTV).EQ.1) WRITE(16,3105)
 3105   FORMAT(/,5X,'UNIT 62 FORMAT WILL BE ASCII')
        IF(ABS(NOUTV).EQ.2) WRITE(16,3106)
 3106   FORMAT(/,5X,'UNIT 62 FORMAT WILL BE BINARY')
        ENDIF

C....REGARDLESS OF WHETHER NOUTV=0, READ IN THE NUMBER OF VELOCITY
C....RECORDING STATIONS

      READ(15,*) NSTAV
```

```
      WRITE(16,3107) NSTAV
 3107 FORMAT(////,5X,'NUMBER OF INPUT VELOCITY RECORDING STATIONS = ',
     &            I5)
      IF(NSTAV.GT.0) THEN
        IF(ICS.EQ.1) WRITE(16,3108)
 3108   FORMAT(/,7X,'STATION #   ELEMENT',9X,'X',13X,'Y',/)
        IF(ICS.EQ.2) WRITE(16,3109)
 3109   FORMAT(/,5X,'STATION   ELEMENT',3X,'LAMBDA(DEG)',
     &             4X,'FEA(DEG)',10X,'XCP',12X,'YCP',/)
        ENDIF

C....CHECK TO ENSURE THAT THE DIMENSIONING PARAMETER MNSTAV HAS BEEN SET PROPERLY

      IF(NSTAV.GT.MNSTAV) THEN
        IF(NSCREEN.EQ.1) WRITE(6,9939)
        WRITE(16,9939)
9939    FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'THE DIMENSIONING PARAMETER MNSTAV IS EXCEEDED',
     &              ' BY THE INPUT PARAMETER NSTAV ',
     &            /,1X,'USER MUST RE-DIMENSION PROGRAM',//,1X,
     &              '!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
        STOP
        ENDIF

C....INPUT COORDINATES OF VELOCITY RECORDING STATIONS
C....THEN COMPUTE ELEMENT NO. WITHIN WHICH STATION LIES

      DO I=1,NSTAV
        NNV(I)=0
        IF(ICS.EQ.1) THEN
          READ(15,*) XEV(I),YEV(I)
          ELSE
          READ(15,*) SLEV(I),SFEV(I)
          SLEV(I)=SLEV(I)*DEG2RAD
          SFEV(I)=SFEV(I)*DEG2RAD
          CALL CPP(XEV(I),YEV(I),SLEV(I),SFEV(I),SLAM0,SFEA0)
          ENDIF
        AEMIN=1.0E+25
        KMIN=0
        DO K=1,NE
          N1=NM(K,1)
          N2=NM(K,2)
          N3=NM(K,3)
          X1=X(N1)
          X2=X(N2)
          X3=X(N3)
          X4=XEV(I)
          Y1=Y(N1)
          Y2=Y(N2)
          Y3=Y(N3)
          Y4=YEV(I)
          A1=(X4-X3)*(Y2-Y3)+(X2-X3)*(Y3-Y4)
          A2=(X4-X1)*(Y3-Y1)-(Y4-Y1)*(X3-X1)
          A3=(Y4-Y1)*(X2-X1)-(X4-X1)*(Y2-Y1)
          AA=ABS(A1)+ABS(A2)+ABS(A3)
          AE=ABS(AA-AREAS(K))/AREAS(K)
          IF(AE.LT.AEMIN) THEN
            AEMIN=AE
            KMIN=K
            ENDIF
          IF(AE.LT.1.0E-5) NNV(I)=K
          END DO

        IF(NNV(I).EQ.0) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9786) I
          WRITE(16,9786) I
9786      FORMAT(///,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &                 'INPUT ERROR  !!!!!!!!!',//
     &               ,1X,'VELOCITY RECORDING STATION ',I6,' DOES NOT LIE'
     &                  ,' WITHIN ANY ELEMENT IN THE DEFINED',
```

```
     &             /,1X,'COMPUTATIONAL DOMAIN,   PLEASE CHECK THE INPUT'
     &                 ,' COORDINATES FOR THIS STATION')
           IF(NFOVER.EQ.1) THEN
             IF(NSCREEN.EQ.1) WRITE(6,9790) AEMIN
             WRITE(16,9790) AEMIN
             NNV(I)=KMIN
             ELSE
             IF(NSCREEN.EQ.1) WRITE(6,9791) AEMIN
             WRITE(16,9791) AEMIN
             STOP
             ENDIF
           ENDIF

         IF(ICS.EQ.1) THEN
           WRITE(16,1880) I,NNV(I),XEV(I),YEV(I)
           ELSE
           WRITE(16,1883) I,NNV(I),SLEV(I)*RAD2DEG,SFEV(I)*RAD2DEG,
     &                    XEV(I),YEV(I)
           ENDIF

C....PRE-COMPUTE INFORMATION REQUIRED TO INTERPOLATE AT VEL. RECORDING STATIONS

         N1=NM(NNV(I),1)
         N2=NM(NNV(I),2)
         N3=NM(NNV(I),3)
         X1=X(N1)
         X2=X(N2)
         X3=X(N3)
         X4=XEV(I)
         Y1=Y(N1)
         Y2=Y(N2)
         Y3=Y(N3)
         Y4=YEV(I)
         STAIV1(I)=((X4-X3)*(Y2-Y3)+(X2-X3)*(Y3-Y4))/AREAS(NNV(I))
         STAIV2(I)=((X4-X1)*(Y3-Y1)-(Y4-Y1)*(X3-X1))/AREAS(NNV(I))
         STAIV3(I)=(-(X4-X1)*(Y2-Y1)+(Y4-Y1)*(X2-X1))/AREAS(NNV(I))

         END DO

C...
C...IF TRANSPORT IS INCLUDED IN THE RUN, INPUT INFORMATION FOR CONCENTRATION
C...RECORDING STATIONS
C...
       NOUTC=0
       IF(IM.EQ.10) THEN

C.....READ IN NOUTC,TOUTSC,TOUTFC,NSPOOLC : IF NOUTC<>0,INTERPOLATED
C.....CONCENTRATIONS ARE SPOOLED TO UNIT 71 EVERY NSPOOLC TIME STEPS
C.....BETWEEN TIMES TOUTSC AND TOUTFC; IF ABS(NOUTC)=2, OUTPUT WILL BE BINARY

         READ(15,*) NOUTC,TOUTSC,TOUTFC,NSPOOLC
         WRITE(16,3201) NOUTC
 3201    FORMAT(///,1X,'CONCENTRATION RECORDING STATION OUTPUT : ',
     &          //,5X,'NOUTC = ',I2)

C.....CHECK INPUT PARAMETER NOUTC

         IF(ABS(NOUTC).GT.2) THEN
           IF(NSCREEN.EQ.1) WRITE(6,3202)
           WRITE(16,3202)
 3202      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &                ' NOUTC',
     &            /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
           IF(NSCREEN.EQ.1) WRITE(6,9973)
           WRITE(16,9973)
           STOP
           ENDIF

C.....IF STATION CONCENTRATION OUTPUT WILL NOT BE GENERATED
```

```
        IF(NOUTC.EQ.0) THEN
          WRITE(16,3203)
 3203     FORMAT(/,5X,'NO OUTPUT WILL BE SPOOLED AT CONCENTRATION',
     &                ' RECORDING STATIONS')
          ENDIF

C.....IF STATION CONCENTRATION OUTPUT WILL BE GENERATED

        IF(NOUTC.NE.0) THEN

C.......COMPUTE NTCYSC, NTCYFC, WHICH = TOUTSC AND TOUTFC IN TIMESTEPS

          NTCYSC=INT((TOUTSC-STATIM)*(86400.D0/DTDP) + 0.5)
          NTCYFC=INT((TOUTFC-STATIM)*(86400.D0/DTDP) + 0.5)
          IF(NTCYFC.GT.NT) NTCYFC=NT

C.......COMPUTE NTRSPC = THE NO. OF DATA SETS TO BE SPOOLED TO UNIT 71

          IF(NSPOOLC.EQ.0) NTRSPC=0
          IF(NSPOOLC.NE.0) NTRSPC=INT((NTCYFC-NTCYSC)/NSPOOLC)

C.......WRITE TOUTSC,TOUTFC,NTCYSC,NTCYFC,NSPOOLC TO UNIT 16

          WRITE(16,3204) TOUTSC,NTCYSC,TOUTFC,NTCYFC,NSPOOLC
 3204     FORMAT(/,5X,'DATA RECORDS WILL START AFTER TOUTSC =',F8.3,
     &                ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &             I9,' TIME STEPS INTO THE SIMULATION',
     &           //,5X,'DATA RECORDS WILL STOP AFTER TOUTFC =',F8.3,
     &                ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &             I9,' TIME STEPS INTO THE SIMULATION',
     &           //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 71 EVERY',
     &                ' NSPOOLC =',I8,' TIME STEPS')
          IF(ABS(NOUTC).EQ.1) WRITE(16,3205)
 3205     FORMAT(/,5X,'UNIT 71 FORMAT WILL BE ASCII')
          IF(ABS(NOUTC).EQ.2) WRITE(16,3206)
 3206     FORMAT(/,5X,'UNIT 71 FORMAT WILL BE BINARY')
          ENDIF

C.....REGARDLESS OF WHETHER NOUTC=0, READ IN THE NUMBER OF CONCENTRATION
C.....RECORDING STATIONS

        READ(15,*) NSTAC
        WRITE(16,3207) NSTAC
 3207   FORMAT(///,5X,'NUMBER OF INPUT CONCENTRATION RECORDING ',
     &               'STATIONS = ',I5)
        IF(NSTAC.GT.0) THEN
          IF(ICS.EQ.1) WRITE(16,3208)
 3208     FORMAT(/,7X,'STATION #   ELEMENT',9X,'X',13X,'Y',/)
          IF(ICS.EQ.2) WRITE(16,3209)
 3209     FORMAT(/,5X,'STATION   ELEMENT',3X,'LAMBDA(DEG)',
     &            4X,'FEA(DEG)',10X,'XCP',12X,'YCP',/)
          ENDIF

C.....CHECK TO ENSURE THAT THE DIMENSIONING PARAMETER MNSTAC HAS BEEN SET PROPERLY

        IF(NSTAC.GT.MNSTAC) THEN
          IF(NSCREEN.EQ.1) WRITE(6,9940)
          WRITE(16,9940)
 9940     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'THE DIMENSIONING PARAMETER MNSTAC IS EXCEEDED ',
     &                'BY THE INPUT PARAMETER NSTAC ',
     &            /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &        //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
          STOP
          ENDIF

C.....INPUT COORDINATES OF CONCENTRATION RECORDING STATIONS
C.....THEN COMPUTE ELEMENT NO. WITHIN WHICH STATION LIES
```

```
        DO I=1,NSTAC
          NNC(I)=0
          IF(ICS.EQ.1) THEN
            READ(15,*) XEC(I),YEC(I)
            ELSE
            READ(15,*) SLEC(I),SFEC(I)
            SLEC(I)=SLEC(I)*DEG2RAD
            SFEC(I)=SFEC(I)*DEG2RAD
            CALL CPP(XEC(I),YEC(I),SLEC(I),SFEC(I),SLAM0,SFEA0)
            ENDIF
          AEMIN=1.0E+25
          KMIN=0
          DO K=1,NE
            N1=NM(K,1)
            N2=NM(K,2)
            N3=NM(K,3)
            X1=X(N1)
            X2=X(N2)
            X3=X(N3)
            X4=XEC(I)
            Y1=Y(N1)
            Y2=Y(N2)
            Y3=Y(N3)
            Y4=YEC(I)
            A1=(X4-X3)*(Y2-Y3)+(X2-X3)*(Y3-Y4)
            A2=(X4-X1)*(Y3-Y1)-(Y4-Y1)*(X3-X1)
            A3=(Y4-Y1)*(X2-X1)-(X4-X1)*(Y2-Y1)
            AA=ABS(A1)+ABS(A2)+ABS(A3)
            AE=ABS(AA-AREAS(K))/AREAS(K)
            IF(AE.LT.AEMIN) THEN
              AEMIN=AE
              KMIN=K
            ENDIF
            IF(AE.LT.1.0E-5) NNC(I)=K
            END DO

          IF(NNC(I).EQ.0) THEN
            IF(NSCREEN.EQ.1) WRITE(6,9785) I
            WRITE(16,9785) I
9785        FORMAT(///,1X,'!!!!!!!!!!  WARNING - NONFATAL INPUT ERROR ',
     &           '!!!!!!!!!!',//,
     &           ' CONCENTRATION RECORDING STATION ',I6,' DOES NOT LIE'
     &          ,' WITHIN ANY ELEMENT IN THE DEFINED',/,
     &           ' COMPUTATIONAL DOMAIN,   PLEASE CHECK THE INPUT',
     &           ' COORDINATES FOR THIS STATION')
            IF(NFOVER.EQ.1) THEN
              IF(NSCREEN.EQ.1) WRITE(6,9790) AEMIN
              WRITE(16,9790) AEMIN
              NNC(I)=KMIN
              ELSE
              IF(NSCREEN.EQ.1) WRITE(6,9791) AEMIN
              WRITE(16,9791) AEMIN
              STOP
              ENDIF
            ENDIF

          IF(ICS.EQ.1) THEN
            WRITE(16,1880) I,NNC(I),XEC(I),YEC(I)
            ELSE
            WRITE(16,1883) I,NNC(I),SLEC(I)*RAD2DEG,
     &                     SFEC(I)*RAD2DEG,XEC(I),YEC(I)
            ENDIF

C.....PRE-COMPUTE INFORMATION REQUIRED TO INTERPOLATE AT CONCENTRATION
C.....RECORDING STATIONS

          N1=NM(NNC(I),1)
          N2=NM(NNC(I),2)
          N3=NM(NNC(I),3)
          X1=X(N1)
```

```
          X2=X(N2)
          X3=X(N3)
          X4=XEL(I)
          Y1=Y(N1)
          Y2=Y(N2)
          Y3=Y(N3)
          Y4=YEL(I)
          STAIC1(I)=((X4-X3)*(Y2-Y3)+(X2-X3)*(Y3-Y4))/AREAS(NNC(I))
          STAIC2(I)=((X4-X1)*(Y3-Y1)-(Y4-Y1)*(X3-X1))/AREAS(NNC(I))
          STAIC3(I)=(-(X4-X1)*(Y2-Y1)+(Y4-Y1)*(X2-X1))/AREAS(NNC(I))

          END DO
        ENDIF

C...
C...INPUT INFORMATION ABOUT GLOBAL ELEVATION DATA OUTPUT
C...

C....READ IN NOUTGE,TOUTSGE,TOUTFGE,NSPOOLGE : IF NOUTGE<>0, GLOBAL ELEV.
C....OUTPUT IS SPOOLED TO UNIT 63 EVERY NSPOOLGE TIME STEPS BETWEEN
C....TIMES TOUTSGE AND TOUTFGE; IF ABS(NOUTGE)=2, OUTPUT WILL BE BINARY

      READ(15,*) NOUTGE,TOUTSGE,TOUTFGE,NSPOOLGE
      WRITE(16,3301) NOUTGE
 3301 FORMAT(////,1X,'GLOBAL NODAL ELEVATION INFORMATION OUTPUT: ',
     &          //,5X,'NOUTGE = ',I2)

C....CHECK INPUT PARAMETER NOUTGE

      IF(ABS(NOUTGE).GT.2) THEN
        IF(NSCREEN.EQ.1) WRITE(6,3302)
        WRITE(16,3302)
 3302   FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &                ' NOUTGE',
     &            /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
        IF(NSCREEN.EQ.1) WRITE(6,9973)
        WRITE(16,9973)
        STOP
        ENDIF

C....IF GLOBAL ELEVATION OUTPUT WILL NOT BE GENERATED

      IF(NOUTGE.EQ.0) THEN
        WRITE(16,3303)
 3303   FORMAT(///,5X,'NO GLOBAL ELEVATION OUTPUT WILL BE SPOOLED')
        ENDIF

C....IF GLOBAL ELEVATION OUTPUT WILL BE GENERATED

      IF(NOUTGE.NE.0) THEN

C......COMPUTE NTCYSGE, NTCYFGE, WHICH = TOUTSGE AND TOUTFGE IN TIMESTEPS

        NTCYSGE=INT((TOUTSGE-STATIM)*(86400.D0/DTDP) + 0.5)
        NTCYFGE=INT((TOUTFGE-STATIM)*(86400.D0/DTDP) + 0.5)
        IF(NTCYFGE.GT.NT) NTCYFGE=NT

C......CALCULATE NDSETSE = THE # OF DATA SETS TO BE SPOOLED TO UNIT 63

        IF(NSPOOLGE.EQ.0) NDSETSE=0
        IF(NSPOOLGE.NE.0) NDSETSE=INT((NTCYFGE-NTCYSGE)/NSPOOLGE)

C......WRITE NOUTGE,TOUTSGE,TOUTFGE,NTCYSGE,NTCYFGE,NSPOOLGE TO UNIT 16

        WRITE(16,3304) TOUTSGE,NTCYSGE,TOUTFGE,NTCYFGE,NSPOOLGE
 3304   FORMAT(/,5X,'DATA RECORDS WILL START AFTER TOUTSGE =',F8.3,
     &              ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'DATA RECORDS WILL STOP AFTER TOUTFGE =',F8.3,
```

```
     &              ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 63 EVERY ',
     &              'NSPOOLGE =',I8,' TIME STEPS')
        IF(ABS(NOUTGE).EQ.1) WRITE(16,3305)
 3305   FORMAT(/,5X,'UNIT 63 FORMAT WILL BE ASCII')
        IF(ABS(NOUTGE).EQ.2) WRITE(16,3306)
 3306   FORMAT(/,5X,'UNIT 63 FORMAT WILL BE BINARY')
        ENDIF

C...
C...INPUT INFORMATION ABOUT GLOBAL VELOCITY DATA OUTPUT
C...

C....READ IN NOUTGV,TOUTSGV,TOUTFGV,NSPOOLGV : IF NOUTGV<>0, GLOBAL VEL.
C....OUTPUT IS SPOOLED TO UNIT 64 EVERY NSPOOLGV TIME STEPS BETWEEN
C....TIMES TOUTSGV AND TOUTFGV; IF ABS(NOUTGV)=2, OUTPUT WILL BE BINARY

        READ(15,*) NOUTGV,TOUTSGV,TOUTFGV,NSPOOLGV
        WRITE(16,3351) NOUTGV
 3351   FORMAT(////,1X,'GLOBAL NODAL VELOCITY INFORMATION OUTPUT : ',
     &          //,5X,'NOUTGV = ',I2)

C....CHECK INPUT PARAMETER NOUTGV

        IF(ABS(NOUTGV).GT.2) THEN
          IF(NSCREEN.EQ.1) WRITE(6,3352)
          WRITE(16,3352)
 3352     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!!',
     &          //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &                ' NOUTGV',
     &           /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF

C....IF GLOBAL VELOCITY OUTPUT WILL NOT BE GENERATED

        IF(NOUTGV.EQ.0) THEN
          WRITE(16,3353)
 3353     FORMAT(///,5X,'NO GLOBAL VELOCITY OUTPUT WILL BE SPOOLED')
          ENDIF

C....IF GLOBAL VELOCITY OUTPUT WILL BE GENERATED

        IF(NOUTGV.NE.0) THEN

C......COMPUTE NTCYSGV, NTCYFGV, WHICH = TOUTSGV AND TOUTFGV IN TIMESTEPS

          NTCYSGV=INT((TOUTSGV-STATIM)*(86400.D0/DTDP) + 0.5)
          NTCYFGV=INT((TOUTFGV-STATIM)*(86400.D0/DTDP) + 0.5)
          IF(NTCYFGV.GT.NT) NTCYFGV=NT

C......CALCULATE NDSETSV = THE # OF DATA SETS TO BE SPOOLED TO UNIT 64

          IF(NSPOOLGV.EQ.0) NDSETSV=0
          IF(NSPOOLGV.NE.0) NDSETSV=INT((NTCYFGV-NTCYSGV)/NSPOOLGV)

C......WRITE NOUTGV,TOUTSGV,TOUTFGV,NTCYSGV,NTCYFGV,NSPOOLGV TO UNIT 16

          WRITE(16,3354) TOUTSGV,NTCYSGV,TOUTFGV,NTCYFGV,NSPOOLGV
 3354     FORMAT(/,5X,'DATA RECORDS WILL START AFTER TOUTSGV =',F8.3,
     &                ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'DATA RECORDS WILL STOP AFTER TOUTFGV =',F8.3,
     &                ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &         //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 64 EVERY ',
     &                'NSPOOLGV =',I8,' TIME STEPS')
```

```
          IF(ABS(NOUTGV).EQ.1) WRITE(16,3355)
 3355     FORMAT(/,5X,'UNIT 64 FORMAT WILL BE ASCII')
          IF(ABS(NOUTGV).EQ.2) WRITE(16,3356)
 3356     FORMAT(/,5X,'UNIT 64 FORMAT WILL BE BINARY')
          ENDIF

C...
C...IF TRANSPORT IS INCLUDED IN THE RUN, INPUT INFORMATION ABOUT GLOBAL
C...CONCENTRATION DATA OUTPUT
C...
      NOUTGC=0
      IF(IM.EQ.10) THEN

C.....READ IN NOUTGC,TOUTSGC,TOUTFGC,NSPOOLGC : IF NOUTGC<>0, GLOBAL
C.....CONCENTRATION OUTPUT IS SPOOLED TO UNIT 73 EVERY NSPOOLGC TIME STEPS
C.....BETWEEN TIMES TOUTSGC AND TOUTFGC; IF ABS(NOUTGC)=2, OUTPUT WILL BE BINARY

        READ(15,*) NOUTGC,TOUTSGC,TOUTFGC,NSPOOLGC
        WRITE(16,3401) NOUTGC
 3401   FORMAT(////,1X,'GLOBAL NODAL CONCENTRATION INFORMATION OUTPUT:',
     &          //,5X,'NOUTGC = ',I2)

C.....CHECK INPUT PARAMETER NOUTGC

        IF(ABS(NOUTGC).GT.2) THEN
          IF(NSCREEN.EQ.1) WRITE(6,3402)
          WRITE(16,3402)
 3402     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!!',
     &           //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &             ' NOUTGC',
     &            /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
          IF(NSCREEN.EQ.1) WRITE(6,9973)
          WRITE(16,9973)
          STOP
          ENDIF

C.....IF GLOBAL CONCENTRATION OUTPUT WILL NOT BE GENERATED

        IF(NOUTGC.EQ.0) THEN
          WRITE(16,3403)
 3403     FORMAT(///,5X,'NO GLOBAL CONCENTRATION OUTPUT WILL BE ',
     &                 'SPOOLED')
          ENDIF

C.....IF GLOBAL CONCENTRATION OUTPUT WILL BE GENERATED

        IF(NOUTGC.NE.0) THEN

C.......COMPUTE NTCYSGC, NTCYFGC, WHICH = TOUTSGC AND TOUTFGC IN TIMESTEPS

          NTCYSGC=INT((TOUTSGC-STATIM)*(86400.D0/DTDP) + 0.5)
          NTCYFGC=INT((TOUTFGC-STATIM)*(86400.D0/DTDP) + 0.5)
          IF(NTCYFGC.GT.NT) NTCYFGC=NT

C.......CALCULATE NDSETSC = THE # OF DATA SETS TO BE SPOOLED TO UNIT 73

          IF(NSPOOLGC.EQ.0) NDSETSC=0
          IF(NSPOOLGC.NE.0) NDSETSC=INT((NTCYFGC-NTCYSGC)/NSPOOLGC)

C.......WRITE NOUTGC,TOUTSGC,TOUTFGC,NTCYSGC,NTCYFGC,NSPOOLGC TO UNIT 16

          WRITE(16,3404) TOUTSGC,NTCYSGC,TOUTFGC,NTCYFGC,NSPOOLGC
 3404     FORMAT(/,5X,'DATA RECORDS WILL START AFTER TOUTSGC =',F8.3,
     &               ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &          //,5X,'DATA RECORDS WILL STOP AFTER TOUTFGC =',F8.3,
     &               ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &           I9,' TIME STEPS INTO THE SIMULATION',
     &          //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 73 EVERY ',
     &               'NSPOOLGC =',I8,' TIME STEPS')
```

```
          IF(ABS(NOUTGC).EQ.1) WRITE(16,3405)
 3405     FORMAT(/,5X,'UNIT 73 FORMAT WILL BE ASCII')
          IF(ABS(NOUTGC).EQ.2) WRITE(16,3406)
 3406     FORMAT(/,5X,'UNIT 73 FORMAT WILL BE BINARY')
          ENDIF

        ENDIF

C...
C...IF NWS>0   INPUT INFORMATION ABOUT GLOBAL WIND STRESS DATA OUTPUT
C...
      IF(NWS.GT.0) THEN

C......READ IN NOUTGW,TOUTSGW,TOUTFGW,NSPOOLGW : IF NOUTGW<>0, GLOBAL WIND
C......OUTPUT IS SPOOLED TO UNIT 74 EVERY NSPOOLGW TIME STEPS BETWEEN
C......TIMES TOUTSGW AND TOUTFGW; IF ABS(NOUTGW)=2, OUTPUT WILL BE BINARY

        READ(15,*) NOUTGW,TOUTSGW,TOUTFGW,NSPOOLGW
        WRITE(16,3451) NOUTGW
 3451   FORMAT(////,1X,'GLOBAL WIND STRESS INFORMATION OUTPUT : ',
     &           //,5X,'NOUTGW = ',I2)

C......CHECK INPUT PARAMETER NOUTGW

        IF(ABS(NOUTGW).GT.2) THEN
          IF(NSCREEN.EQ.1) WRITE(6,3452)
          WRITE(16,3452)
 3452     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &           //,1X,'YOUR SELECTION OF THE UNIT 15 INPUT PARAMETER',
     &                ' NOUTGW',
     &            /,1X,'IS NOT AN ALLOWABLE VALUE.  CHECK YOUR INPUT!!')
          IF(NSCREEN.EQ.1) WRITE(6,3453)
          WRITE(16,3453)
          STOP
          ENDIF

C......IF GLOBAL WIND STRESS OUTPUT WILL NOT BE GENERATED

        IF(NOUTGW.EQ.0) THEN
          WRITE(16,3453)
 3453     FORMAT(///,5X,'NO GLOBAL WIND STRESS OUTPUT WILL BE SPOOLED')
          ENDIF

C........IF GLOBAL WIND STRESS OUTPUT WILL BE GENERATED

        IF(NOUTGW.NE.0) THEN

C........COMPUTE NTCYSGW, NTCYFGW, WHICH = TOUTSGW AND TOUTFGW IN TIMESTEPS

          NTCYSGW=INT((TOUTSGW-STATIM)*(86400.D0/DTDP) + 0.5)
          NTCYFGW=INT((TOUTFGW-STATIM)*(86400.D0/DTDP) + 0.5)
          IF(NTCYFGW.GT.NT) NTCYFGW=NT

C........CALCULATE NDSETSW = THE # OF DATA SETS TO BE SPOOLED TO UNIT 74

          IF(NSPOOLGW.EQ.0) NDSETSW=0
          IF(NSPOOLGW.NE.0) NDSETSW=INT((NTCYFGW-NTCYSGW)/NSPOOLGW)

C........WRITE NOUTGW,TOUTSGW,TOUTFGW,NTCYSGW,NTCYFGW,NSPOOLGW TO UNIT 16

          WRITE(16,3454) TOUTSGW,NTCYSGW,TOUTFGW,NTCYFGW,NSPOOLGW
 3454     FORMAT(/,5X,'DATA RECORDS WILL START AFTER TOUTSGW =',F8.3,
     &                ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &             I9,' TIME STEPS INTO THE SIMULATION',
     &           //,5X,'DATA RECORDS WILL STOP AFTER TOUTFGW =',F8.3,
     &                ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',
     &             I9,' TIME STEPS INTO THE SIMULATION',
     &           //,5X,'INFORMATION WILL BE SPOOLED TO UNIT 74 EVERY ',
     &                'NSPOOLGW =',I8,' TIME STEPS')
          IF(ABS(NOUTGW).EQ.1) WRITE(16,3455)
```

```
 3455      FORMAT(/,5X,'UNIT 74 FORMAT WILL BE ASCII')
           IF(ABS(NOUTGW).EQ.2) WRITE(16,3456)
 3456      FORMAT(/,5X,'UNIT 74 FORMAT WILL BE BINARY')
           ENDIF

         ENDIF

C...
C...READ AND CHECK INFORMATION ABOUT HARMONIC ANALYSIS OF MODEL RESULTS
C...
         READ(15,*) NHARFR
         WRITE(16,99392) NHARFR
99392    FORMAT(////,1X,'HARMONIC ANALYSIS INFORMATION OUTPUT : ',
     &    //,5X,'HARMONIC ANALYSIS PERFORMED FOR ',I4,' CONSTITUENTS',/)
         IF(NHARFR.LT.0) THEN
           IF(NSCREEN.EQ.1) WRITE(6,99391)
           WRITE(16,99391)
99391      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &     //,1X,'YOUR SELECTION OF NHARFR (A UNIT 15 '
     &       ,'INPUT PARAMETER) IS NOT AN ALLOWABLE VALUE',/,1X,
     &       'PLEASE CHECK YOUR INPUT',
     &     //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
           STOP
         ENDIF
         IF(NHARFR.GT.MNHARF) THEN
           IF(NSCREEN.EQ.1) WRITE(6,99381)
           WRITE(16,99381)
99381      FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     +     //,1X,'THE DIMENSIONING PARAMETER MNHARF IS EXCEEDED'
     +     ,' BY THE INPUT PARAMETER NHARFR ',
     +     /,1X,'USER MUST RE-DIMENSION PROGRAM',
     +     //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
           STOP
         ENDIF
         IF(NHARFR.GT.0) WRITE(16,2330)
2330     FORMAT(/,7X,'FREQUENCY',4X,'NODAL FACTOR',6X,'EQU.ARG(DEG)',
     +           1X,'CONSTITUENT',/)
         DO 1201 I=1,NHARFR
           READ(15,'(A10)') HAFNAM(I)
           READ(15,*) HAFREQ(I),HAFF(I),HAFACE(I)
           WRITE(16,2331) HAFREQ(I),HAFF(I),HAFACE(I),HAFNAM(I)
2331       FORMAT(4X,F15.12,2X,F10.7,5X,F10.3,7X,A10)
1201       CONTINUE

C....READ IN INTERVAL INFORMATION FOR HARMONIC ANALYSIS
C....COMPUTE THAS AND THAF IN TERMS OF THE NUMBER OF TIME STEPS

         READ(15,*) THAS,THAF,NHAINC,FMV
         ITHAS=INT((THAS-STATIM)*(86400.D0/DTDP) + 0.5)
         THAS=ITHAS*DTDP/86400.D0 + STATIM
         ITHAF=INT((THAF-STATIM)*(86400.D0/DTDP) + 0.5)
         THAF=ITHAF*DTDP/86400.D0 + STATIM
         ITMV = ITHAF - (ITHAF-ITHAS)*FMV
         IF(NHARFR.GT.0) THEN
           WRITE(16,34634) THAS,ITHAS,THAF,ITHAF,NHAINC
34634      FORMAT(/,5X,'HARMONIC ANALYSIS WILL START AFTER THAS =',F8.3,
     &     ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',I9,
     &     ' TIME STEPS INTO THE SIMULATION',
     &     //,5X,'HARMONIC ANALYSIS WILL STOP AFTER THAF =',F8.3,
     &     ' DAY(S) RELATIVE',/,9X,'TO THE STARTING TIME OR',I9,
     &     ' TIME STEPS INTO THE SIMULATION'
     &     ,//,5X,'INFORMATION WILL BE ANALYZED EVERY ',
     &     'NHAINC =',I8,' TIME STEPS.')
           WRITE(16,34639) FMV*100.,ITMV
34639      FORMAT(/,5X,'MEANS AND VARIANCES WILL BE COMPUTED FOR THE ',
     &           'FINAL ',F10.5,' %',/9X,'OF THE HARMONIC ANALYSIS ',
     &           'PERIOD OR AFTER ',I9,' TIME STEPS INTO THE ',
     &           'SIMULATION.',/9X,' RESULTS ARE WRITTEN TO UNIT 55.')

         ELSE
```

```
        WRITE(16,34645)
34645   FORMAT(///,5X,'NO HARMONIC ANALYSIS WILL BE DONE')
      ENDIF

C....READ IN AND WRITE OUT INFORMATION ON WHERE HARMONIC ANALYSIS WILL BE DONE

      READ(15,*) NHASE,NHASV,NHAGE,NHAGV
      IF((NHASE.LT.0).OR.(NHASE.GT.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,99661)
        WRITE(16,99661)
99661     FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &     'INPUT ERROR  !!!!!!!!!',//
     &     ,1X,'YOUR SELECTION OF NHASE (A UNIT 15 '
     &     ,'INPUT PARAMETER) IS NOT AN ALLOWABLE VALUE',/,1X,
     &     'PLEASE CHECK YOUR INPUT')
          IF(NFOVER.EQ.1) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99671)
            WRITE(16,99671)
99671       FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &      ' AND SET NHASE EQUAL TO 0 ',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
            NHASE=0
          ELSE
            IF(NSCREEN.EQ.1) WRITE(6,9973)
            WRITE(16,9973)
            STOP
          ENDIF
      ENDIF
      IF(NHASE.EQ.1) THEN
        WRITE(16,34641)
34641   FORMAT(///,5X,'STATION ELEVATION HARMONIC ANAL WILL BE ',
     &                'WRITTEN TO UNIT 51')
      ENDIF
      IF((NHASV.LT.0).OR.(NHASV.GT.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,99662)
        WRITE(16,99662)
99662     FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &     'INPUT ERROR  !!!!!!!!!',//
     &     ,1X,'YOUR SELECTION OF NHASV (A UNIT 15 '
     &     ,'INPUT PARAMETER) IS NOT AN ALLOWABLE VALUE',/,1X,
     &     'PLEASE CHECK YOUR INPUT')
          IF(NFOVER.EQ.1) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99672)
            WRITE(16,99672)
99672       FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &      ' AND SET NHASV EQUAL TO 0 ',
     &      //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
            NHASV=0
          ELSE
            IF(NSCREEN.EQ.1) WRITE(6,9973)
            WRITE(16,9973)
            STOP
          ENDIF
      ENDIF
      IF(NHASV.EQ.1) THEN
        WRITE(16,34642)
34642   FORMAT(///,5X,'STATION VELOCITY HARMONIC ANAL WILL BE ',
     &                'WRITTEN TO UNIT 52')
      ENDIF
      IF((NHAGE.LT.0).OR.(NHAGE.GT.1)) THEN
        IF(NSCREEN.EQ.1) WRITE(6,99663)
        WRITE(16,99663)
99663     FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &     'INPUT ERROR  !!!!!!!!!',//
     &     ,1X,'YOUR SELECTION OF NHAGE (A UNIT 15 '
     &     ,'INPUT PARAMETER) IS NOT AN ALLOWABLE VALUE',/,1X,
     &     'PLEASE CHECK YOUR INPUT')
          IF(NFOVER.EQ.1) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99673)
            WRITE(16,99673)
```

```
99673         FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &        ' AND SET NHAGE EQUAL TO 0 ',
     &        //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
              NHAGE=0
            ELSE
              IF(NSCREEN.EQ.1) WRITE(6,9973)
              WRITE(16,9973)
              STOP
            ENDIF
          ENDIF
          IF(NHAGE.EQ.1) THEN
            WRITE(16,34643)
34643       FORMAT(///,5X,'GLOBAL ELEVATION HARMONIC ANAL WILL BE ',
     &                  'WRITTEN TO UNIT 53')
          ENDIF
          IF((NHAGV.LT.0).OR.(NHAGV.GT.1)) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99664)
            WRITE(16,99664)
99664         FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &         'INPUT ERROR  !!!!!!!!!',//
     &        ,1X,'YOUR SELECTION OF NHAGV (A UNIT 15 '
     &         ,'INPUT PARAMETER) IS NOT AN ALLOWABLE VALUE',/,1X,
     &         'PLEASE CHECK YOUR INPUT')
              IF(NFOVER.EQ.1) THEN
                IF(NSCREEN.EQ.1) WRITE(6,99674)
                WRITE(16,99674)
99674           FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &          ' AND SET NHAGV EQUAL TO 0 ',
     &          //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
                NHAGV=0
              ELSE
                IF(NSCREEN.EQ.1) WRITE(6,9973)
                WRITE(16,9973)
                STOP
              ENDIF
          ENDIF
          IF(NHAGV.EQ.1) THEN
            WRITE(16,34644)
34644       FORMAT(///,5X,'GLOBAL VELOCITY HARMONIC ANAL WILL BE ',
     &                  'WRITTEN TO UNIT 54')
          ENDIF

C....ESTABLISH INDICATOR OF WHETHER ANY HARMONIC ANALYSIS WILL BE DONE

          IHARIND=NHARFR*(NHASE+NHASV+NHAGE+NHAGV)
          IF(IHARIND.GT.0) IHARIND=1

C...
C...INPUT INFORMATION ABOUT HOT START OUTPUT
C...
          READ(15,*) NHSTAR,NHSINC
          WRITE(16,99655)
99655     FORMAT(////,1X,'HOT START OUTPUT INFORMATION OUTPUT : ')
          IF((NHSTAR.LT.0).OR.(NHSTAR.GT.1)) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99665)
            WRITE(16,99665)
99665         FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL ',
     &         'INPUT ERROR  !!!!!!!!!',//
     &        ,1X,'YOUR SELECTION OF NHSTAR (A UNIT 15 '
     &         ,'INPUT PARAMETER) IS NOT AN ALLOWABLE VALUE',/,1X,
     &         'PLEASE CHECK YOUR INPUT')
              IF(NFOVER.EQ.1) THEN
                IF(NSCREEN.EQ.1) WRITE(6,99675)
                WRITE(16,99675)
99675           FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &          ' AND SET NHSTAR EQUAL TO 0 ',
     &          //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
                NHSTAR=0
              ELSE
                IF(NSCREEN.EQ.1) WRITE(6,9973)
```

```
              WRITE(16,9973)
              STOP
            ENDIF
          ENDIF
          IF(NHSTAR.EQ.1) THEN
            WRITE(16,34636) NHSINC
34636       FORMAT(/,5X,'HOT START OUTPUT WILL BE WRITTEN TO UNIT',
     &       ' 67 OR 68 EVERY ',I5,' TIME STEPS')
          ELSE
            WRITE(16,34646)
34646       FORMAT(///,5X,'NO HOT START OUTPUT WILL BE GENERATED')
          ENDIF
          IF((IHOT.EQ.0).OR.(IHOT.EQ.68)) IHSFIL=67
          IF(IHOT.EQ.67) IHSFIL=68

C...
C...INPUT INFORMATION ABOUT SOLVER
C...
          READ(15,*) ITITER,ISLDIA,CONVCR,ITMAX
          WRITE(16,99656)
99656     FORMAT(//,1X,'SOLVER INFORMATION OUTPUT : ')
          IF((ITITER.LT.-1).OR.(ITITER.GT.7)) THEN
            IF(NSCREEN.EQ.1) WRITE(6,99380)
            WRITE(16,99380)
99380       FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     +     //,1X,'AN INVALID SOLVER WAS SELECTED. ITITER MUST BE -1 TO 7',
     +       //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
            STOP
          ENDIF

CSOLIT...
CSOLIT...UNCOMMENT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLIT...COMMENT OUT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLIT...
          IF((ISLDIA.LT.0).OR.(ISLDIA.GT.5)) THEN
            IF(NSCREEN.EQ.1) WRITE(6,9920)
            WRITE(16,9920)
 9920       FORMAT(////,1X,'!!!!!!!!!!  WARNING - NONFATAL INPUT ERROR',
     &           ' !!!!!!!!!!',//,1X,'ISLDIA (A UNIT 15 INPUT PARAMETER) ',
     &           'MUST BE 0-5',/,1X,'PLEASE CHECK YOUR INPUT')
            IF(NFOVER.EQ.1) THEN
              IF(NSCREEN.EQ.1) WRITE(6,9921)
              WRITE(16,9921)
 9921         FORMAT(/,1X,'PROGRAM WILL OVERRIDE SPECIFIED INPUT',
     &                    ' AND SET ISLDIA EQUAL TO 0 ',
     &              //,1X,'!!!!!! EXECUTION WILL CONTINUE !!!!!!',//)
              ISLDIA=0
              ELSE
              IF(NSCREEN.EQ.1) WRITE(6,9973)
              WRITE(16,9973)
              STOP
              ENDIF
            ENDIF
          IF(ITITER.EQ.1) WRITE(16,9922)
 9922     FORMAT(/,5X,'THE JCG ITERATIVE SOLVER WILL BE USED')
          IF(ITITER.EQ.2) WRITE(16,9923)
 9923     FORMAT(/,5X,'THE JSI ITERATIVE SOLVER WILL BE USED')
          IF(ITITER.EQ.3) WRITE(16,9924)
 9924     FORMAT(/,5X,'THE SOR ITERATIVE SOLVER WILL BE USED')
          IF(ITITER.EQ.4) WRITE(16,9925)
 9925     FORMAT(/,5X,'THE SSORCG ITERATIVE SOLVER WILL BE USED')
          IF(ITITER.EQ.5) WRITE(16,9926)
 9926     FORMAT(/,5X,'THE SSORSI ITERATIVE SOLVER WILL BE USED')
          IF(ITITER.EQ.6) WRITE(16,9927)
 9927     FORMAT(/,5X,'THE RSCG ITERATIVE SOLVER WILL BE USED')
          IF(ITITER.EQ.7) WRITE(16,9928)
 9928     FORMAT(/,5X,'THE RSSI ITERATIVE SOLVER WILL BE USED')

          CALL DFAULT(IPARM,RPARM)
          IPARM(1)=ITMAX
```

```
        IPARM(2)=ISLDIA
        OPEN(33,FILE='fort.33')
        IPARM(4)=33
        RPARM(1)=CONVCR
        IF(ITITER.EQ.1) NW = 4*NP + 4*ITMAX
        IF(ITITER.EQ.2) NW = 2*NP
        IF(ITITER.EQ.3) NW = 2*NP
        IF(ITITER.EQ.4) NW = 6*NP + 4*ITMAX
        IF(ITITER.EQ.5) NW = 5*NP
        IF(ITITER.EQ.6) NW = NP + 3*NP + 4*ITMAX
        IF(ITITER.EQ.7) NW = NP + NP
        IF(ITITER.GE.6) IPARM(9)=-1
CSOLIT...
CSOLIT...END OF ITERATIVE SOLVER SECTION (MORE FOLLOW BELOW)
CSOLIT...

CSOLDIR...
CSOLDIR...UNCOMMENT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLDIR...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIR...CALCULATE THE MATRIX HALF BAND WIDTH AND CHECK PROGRAM DIMENSIONING
CSOLDIR...
c       WRITE(16,9950)
c9950   FORMAT(/,5X,'THE BANDED DIRECT SOLVER WILL BE USED')
c       NBW=0
c       DO I=1,NE
c         MAX=MAX0(NM(I,1),NM(I,2),NM(I,3))
c         MIN=MIN0(NM(I,1),NM(I,2),NM(I,3))
c         MDF=MAX-MIN
c         IF(NBW.LE.MDF) NBW=MDF
c         END DO
c       WRITE(16,2025) NBW
c2025   FORMAT(/,5X,'MAXIMUM NODAL POINT DIFFERENCE ',
c    &               'AND HALF BANDWIDTH NBW = ',I6,/)
c       IF(NBW.GT.MNBW) THEN
c         IF(NSCREEN.EQ.1) WRITE(6,9905) NBW
c         WRITE(16,9905) NBW
c9905     FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
c    &             //,1X,'THE DIMENSIONING PARAMETER MNBW IS EXCEEDED',
c    &                 ' BY THE ACTUAL HALF',
c    &              /,1X,' BANDWIDTH, NBW, OF THE INPUT GRID',
c    &              /,1X,'USER MUST RE-DIMENSION PROGRAM SO THAT MNBW',
c    &                 ' EQUALS AT LEAST ',I6,'.',
c    &       //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
c           STOP
c           ENDIF
c
c       MBW=2*NBW+1            !TOTAL BANDWIDTH OF MATRIX
CSOLDIR...
CSOLDIR...END OF DIRECT SOLVER STATEMENTS (MORE FOLLOW BELOW)
CSOLDIR...

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...READ IN THE SELF ATTRACTION/LOAD TIDE INFO FROM UNIT 23
CTIP...
        IF(NTIP.EQ.2) THEN
          DO I=1,NTIF
            READ(23,9930)
 9930       FORMAT(///)
            DO J=1,NP
              READ(23,*) JJ,SALTAMP(I,JJ),SALTPHA(I,JJ)
              SALTPHA(I,JJ)=SALTPHA(I,JJ)*DEG2RAD
              END DO
            END DO
          ELSE
          DO I=1,NTIF
            DO J=1,NP
              SALTAMP(I,J)=0.
              SALTPHA(I,J)=0.
```

```
              END DO
            END DO
          ENDIF
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C...
C...SPECIFY IF GWCE WILL BE LUMPED
C...

      ILUMP=0

CLUMP...
CLUMP...UNCOMMENT THIS SECTION TO LUMP THE GWCE MATRIX.  COMMENT OUT THIS SECTION
CLUMP...TO LEAVE THE GWCE MATRIX IN CONSISTENT FORM.
CLUMP...
c     ILUMP=1
CLUMP...
CLUMP...
CLUMP...

C.....INITIALIZE AVERAGING FOR INTERNAL BARRIER WATER LEVELS
C......BARAVGWT=0.000 -> NO AVERAGING PERFORMED
      BARAVGWT=0.000
      IBSTART=0
      DO I=1,NVEL
        RBARWL1AVG(I)=0.D0
        RBARWL2AVG(I)=0.D0
      END DO
C.....INITIALIZE NIBNODECODE(I)
      DO I=1,NP
        NIBNODECODE(I)=0
      END DO

C...
C...*************** COLD START PROGRAM SETUP SECTION ***************
C...
      IF(IHOT.EQ.0) THEN
        ITHS = 0
C...
C....SET AT REST INITIAL CONDITION OVER WHOLE DOMAIN
C...
        DO I=1,NP
          UU1(I) =0.D0
          VV1(I) =0.D0
          UU2(I) =0.D0
          VV2(I) =0.D0
          ETA2(I)=0.D0
          ETAS(I)=0.D0
          NODEREP(I)=MAX0(NODEWETRMP,NODEDRYMIN)
          NNODECODE(I)=1
          IF(NOLIFA.EQ.2) THEN
            IF(DP(I).LE.H0) THEN
              NNODECODE(I)=0
              ETA2(I)=H0-DP(I)
              ENDIF
            ENDIF
          ETA1(I)=ETA2(I)
          CH1(I)=0.
          END DO

C....INITIALIZE THE NORMAL FLOW BOUNDARY CONDITION

        DO I=1,NVEL
          QN2(I)=0.D0
          QN1(I)=0.D0
          QN0(I)=0.D0
          END DO
```

```
        IF((NFLUXF.EQ.1).AND.(NFFR.EQ.0)) THEN
          IF(NSCREEN.EQ.1) WRITE(6,1112)
          WRITE(16,1112)
          IF(NSCREEN.EQ.1) WRITE(6,1979)
          WRITE(16,1979)
 1979     FORMAT(/,1X,'NORMAL FLOW INFORMATION READ FROM UNIT 20',/)
          OPEN(20,FILE='fort.20')
          DO J=1,NVEL
                  QNIN1(J)=0.D0
            IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                           .OR.(LBCODEI(J).EQ.22))
     &                                                READ(20,*) QNIN1(J)
            END DO
          DO J=1,NVEL
              QNIN2(J)=0.D0
            IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                           .OR.(LBCODEI(J).EQ.22))
     &                                                READ(20,*) QNIN2(J)
            END DO
          QTIME1 = (STATIM - REFTIM)*86400.D0
          QTIME2 = QTIME1 + FTIMINC
          ENDIF

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...INPUT WIND INFORMATION FROM UNIT 22
CWIND...IF FLEET NUMERIC WIND DATA IS USED, FIND BEGINNING TIME IN FILE,
CWIND....NOTE: CAN'T DEAL WITH WIND THAT STARTS AFTER WREFTIM!!!!!!!!!!!!
CWIND....READ IN AND INTERPOLATE IN SPACE ONTO THE ADCIRC GRID THE
CWIND....TIME LEVEL 1 AND LEVEL 2 WIND FIELDS
CWIND...
        DO I=1,NP
          WSX1(I)=0.D0
          WSY1(I)=0.D0
          PR1(I) =0.D0
          WSX2(I)=0.D0
          WSY2(I)=0.D0
          PR2(I) =0.D0
          END DO

        IF(NSCREEN.EQ.1) WRITE(6,1112)
        WRITE(16,1112)
        IF(NSCREEN.EQ.1) WRITE(6,1980)
        WRITE(16,1980)
 1980   FORMAT(/,1X,'WIND (AND PRESSURE) INFORMATION READ.',/)

        IF(NWS.EQ.2) THEN
          OPEN(22,FILE='fort.22')
          READ(22,*) (NHG,WVNX1(NHG),WVNY1(NHG),PRN1(NHG),I=1,NP)
          READ(22,*) (NHG,WVNX2(NHG),WVNY2(NHG),PRN2(NHG),I=1,NP)
          WTIME1 = (STATIM - REFTIM)*86400.D0
          WTIME2 = WTIME1 + WTIMINC
          ENDIF

        IF(NWS.EQ.3) THEN
          OPEN(22,FILE='fort.22')
 2222     CALL NWS3GET(X,Y,SLAM,SFEA,WVNX2,WVNY2,IWTIME,IWYR,WTIMED,NP,
     &                 NWLON,NWLAT,WLATMAX,WLONMIN,WLATINC,WLONINC,ICS)
          IF(IWYR.NE.IREFYR) THEN
            IWTIMEP=IWTIME
            DO I=1,NP
              WVNX1(I)=WVNX2(I)
              WVNY1(I)=WVNY2(I)
              END DO
            GOTO 2222
            ENDIF
          IF(WTIMED.LE.WREFTIM) THEN
            IWTIMEP=IWTIME
            DO I=1,NP
```

```
            WVNX1(I)=WVNX2(I)
            WVNY1(I)=WVNY2(I)
            END DO
          GOTO 2222
          ENDIF
        IF(NSCREEN.EQ.1) WRITE(6,*)'FOUND WIND DATA AT TIME= ',IWTIMEP
        WRITE(16,*) 'FOUND WIND DATA AT TIME= ',IWTIMEP
        IF(NSCREEN.EQ.1) WRITE(6,*)'FOUND WIND DATA AT TIME= ',IWTIME
        WRITE(16,*) 'FOUND WIND DATA AT TIME= ',IWTIME
        WTIME2=WTIMED-WREFTIM                    !CAST INTO MODEL TIME REFRENCE
        WTIME1=WTIME2-WTIMINC
        ENDIF

      IF(NWS.EQ.4) THEN
        OPEN(22,FILE='fort.22')
        WTIME1 = (STATIM - REFTIM)*86400.D0
        WTIME2=WTIME1+WTIMINC
        CALL NWS4GET(WVNX1,WVNY1,PRN1,NP,1000.,G)
        CALL NWS4GET(WVNX2,WVNY2,PRN2,NP,1000.,G)
        ENDIF

      IF(NWS.EQ.10) THEN
        WTIME1=(STATIM-REFTIM)*86400.D0
        WTIME2=WTIME1+WTIMINC
        NWSGGWI=0
        CALL NWS10GET(NWSGGWI,SLAM,SFEA,WVNX1,WVNY1,PRN1,NP,1000.,G,
     &                NWLON,NWLAT)
        NWSGGWI=1
        CALL NWS10GET(NWSGGWI,SLAM,SFEA,WVNX2,WVNY2,PRN2,NP,1000.,G,
     &                NWLON,NWLAT)
        ENDIF
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND...

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
      DO I=1,NP
        TIP2(I)=0.D0
        END DO
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

CWETDRY...
CWETDRY...THE FOLLOWING LINES ARE FOR WETTING AND DRYING
CWETDRY...DETERMINE INITIAL WET/DRY INTERFACE NODES
CWETDRY...
      IF(NOLIFA.EQ.2) THEN
        DO I=1,NP                                !MAKE FIRST CUT AT DETERMINING
          IF(NNODECODE(I).EQ.1) THEN             !WET/DRY INTERFACE NODES.  IF
            DO J=2,NNEIGH(I)                     !A WET NODE IS CONNECTED TO A DRY
              NEINOD=NEITAB(I,J)                 !NODE IT MUST BE AN INTERFACE NODE
              IF(NNODECODE(NEINOD).EQ.0) NNODECODE(I)=-1
              END DO
            ENDIF
          END DO
        DO I=1,NP                                !DRY WET/DRY INTERFACE NODES
          IF(NNODECODE(I).EQ.-1) THEN            !THAT ARE NOT CONNECTED TO AT
            IWET=0                               !LEAST ONE NODE THAT HAS
            DO J=2,NNEIGH(I)                     !FUNCTIONING VELOCITY
              NEINOD=NEITAB(I,J)
              IF((NNODECODE(NEINOD).EQ.1).AND.(LBCODE(NEINOD).GE.0))
     &                                                    IWET=1
              IF((NNODECODE(NEINOD).EQ.-1).AND.(LBCODEWD.EQ.1)) IWET=1
              END DO
            IF(IWET.EQ.0) THEN
              NNODECODE(I)=0
```

```
                DO J=2,NNEIGH(I)                    !MAKE ALL CONNECTED WET NODES (NO
                  NEINOD=NEITAB(I,J)                !BNDRY NODES) INTO INTERFACE NODES
                  IF(NNODECODE(NEINOD).EQ.1) NNODECODE(NEINOD)=-1
                  END DO
                ENDIF
              ENDIF
            END DO
          ENDIF
CWETDRY...
CWETDRY...END WET/DRY SECTION
CWETDRY...

C...
C......INITIALIZE 3D SOLUTION
C...

C3DVS..
C3DVS..UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D VS MODE.
C3DVS..COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D DSS MODE.
C3DVS..
c     CALL VSSTUP(DT,STATIM,NBYTE,RUNDES,RUNID,AGRID,NT)
C3DVS..
C3DVS..END OF 3D VS STATEMENTS (MORE FOLLOW BELOW)
C3DVS..

C3DDSS.
C3DDSS.UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE
C3DDSS.COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c     CALL DSSSTUP(DT,STATIM,NBYTE,RUNDES,RUNID,AGRID,NT)
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

C...
C....INITILIZE ELEVATION STATION SPOOL COUNTER
C....OPEN ELEVATION STATION OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NTRSPE (NO. OF DATA PTS. AT EACH
C....ELEVATION STATION), NSTAE, DT*NSPOOLE, NSPOOLE, IRTYPE
C...
        NSCOUE=0
        IESTP=0

 3220   FORMAT(1X,A32,2X,A24,2X,A24)
 3645   FORMAT(1X,I10,1X,I10,1X,E15.7,1X,I5,1X,I5)

        IF(ABS(NOUTE).EQ.1) THEN
          OPEN(61,FILE='fort.61')
          WRITE(61,3220) RUNDES,RUNID,AGRID
          WRITE(61,3645) NTRSPE,NSTAE,DTDP*NSPOOLE,NSPOOLE,1
          IESTP=2
          ENDIF

        IF(ABS(NOUTE).EQ.2) THEN
          OPEN(61,FILE='fort.61',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(61,REC=IESTP+I) RDES4(I)
              ENDDO
            IESTP=IESTP+8
            DO I=1,6
              WRITE(61,REC=IESTP+I) RID4(I)
              ENDDO
            IESTP=IESTP+6
            DO I=1,6
              WRITE(61,REC=IESTP+I) AID4(I)
              ENDDO
            IESTP=IESTP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
```

```
            DO I=1,4
              WRITE(61,REC=IESTP+I) RDES8(I)
              ENDDO
            IESTP=IESTP+4
            DO I=1,3
              WRITE(61,REC=IESTP+I) RID8(I)
              ENDDO
            IESTP=IESTP+3
            DO I=1,3
              WRITE(61,REC=IESTP+I) AID8(I)
              ENDDO
            IESTP=IESTP+3
            ENDIF
          WRITE(61,REC=IESTP+1) NTRSPE
          WRITE(61,REC=IESTP+2) NSTAE
          WRITE(61,REC=IESTP+3) DT*NSPOOLE
          WRITE(61,REC=IESTP+4) NSPOOLE
          WRITE(61,REC=IESTP+5) 1
          IESTP=IESTP+5
          CLOSE(61)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(61,FILE='fort.61',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF

C...
C....INITILIZE VELOCITY STATION SPOOL COUNTER
C....OPEN VELOCITY STATION OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NTRSPV (NO. OF DATA PTS. AT EACH
C....VELOCITY STATION), NSTAV, DT*NSPOOLV, NSPOOLV, IRTYPE
C...
        NSCOUV=0
        IVSTP=0

        IF(ABS(NOUTV).EQ.1) THEN
          OPEN(62,FILE='fort.62')
          WRITE(62,3220) RUNDES,RUNID,AGRID
          WRITE(62,3645) NTRSPV,NSTAV,DTDP*NSPOOLV,NSPOOLV,2
          IVSTP=2
          ENDIF

        IF(ABS(NOUTV).EQ.2) THEN
          OPEN(62,FILE='fort.62',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(62,REC=IVSTP+I) RDES4(I)
              ENDDO
            IVSTP=IVSTP+8
            DO I=1,6
              WRITE(62,REC=IVSTP+I) RID4(I)
              ENDDO
            IVSTP=IVSTP+6
            DO I=1,6
              WRITE(62,REC=IVSTP+I) AID4(I)
              ENDDO
            IVSTP=IVSTP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(62,REC=IVSTP+I) RDES8(I)
              ENDDO
            IVSTP=IVSTP+4
            DO I=1,3
              WRITE(62,REC=IVSTP+I) RID8(I)
              ENDDO
            IVSTP=IVSTP+3
            DO I=1,3
              WRITE(62,REC=IVSTP+I) AID8(I)
              ENDDO
            IVSTP=IVSTP+3
            ENDIF
          WRITE(62,REC=IVSTP+1) NTRSPV
```

```
          WRITE(62,REC=IVSTP+2) NSTAV
          WRITE(62,REC=IVSTP+3) DT*NSPOOLV
          WRITE(62,REC=IVSTP+4) NSPOOLV
          WRITE(62,REC=IVSTP+5) 2
          IVSTP=IVSTP+5
          CLOSE(62)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(62,FILE='fort.62',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF

C...
C....INITILIZE CONCENTRATION STATION SPOOL COUNTER
C....OPEN ELEVATION STATION OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NTRSPC (NO. OF DATA PTS. AT EACH
C....CONCENTRATION STATION), NSTAC, DT*NSPOOLC, NSPOOLC, IRTYPE
C...
        NSCOUC=0
        ICSTP=0

        IF(ABS(NOUTC).EQ.1) THEN
          OPEN(71,FILE='fort.71')
          WRITE(71,3220) RUNDES,RUNID,AGRID
          WRITE(71,3645) NTRSPC,NSTAC,DTDP*NSPOOLC,NSPOOLC,1
          ICSTP=2
          ENDIF

        IF(ABS(NOUTC).EQ.2) THEN
          OPEN(71,FILE='fort.71',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(71,REC=ICSTP+I) RDES4(I)
              ENDDO
            ICSTP=ICSTP+8
            DO I=1,6
              WRITE(71,REC=ICSTP+I) RID4(I)
              ENDDO
            ICSTP=ICSTP+6
            DO I=1,6
              WRITE(71,REC=ICSTP+I) AID4(I)
              ENDDO
            ICSTP=ICSTP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(71,REC=ICSTP+I) RDES8(I)
              ENDDO
            ICSTP=ICSTP+4
            DO I=1,3
              WRITE(71,REC=ICSTP+I) RID8(I)
              ENDDO
            ICSTP=ICSTP+3
            DO I=1,3
              WRITE(61,REC=ICSTP+I) AID8(I)
              ENDDO
            ICSTP=ICSTP+3
            ENDIF
          WRITE(71,REC=ICSTP+1) NTRSPC
          WRITE(71,REC=ICSTP+2) NSTAC
          WRITE(71,REC=ICSTP+3) DT*NSPOOLC
          WRITE(71,REC=ICSTP+4) NSPOOLC
          WRITE(71,REC=ICSTP+5) 1
          ICSTP=ICSTP+5
          CLOSE(71)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(71,FILE='fort.71',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF

C...
C....INITILIZE GLOBAL ELEVATION SPOOL COUNTER
C....OPEN GLOBAL ELEVATION OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NDSETSE
C....(NO. OF GLOBAL ELEVATION DATA SETS TO BE SPOOLED),
```

```
C....NP, DT*NSPOOLGE, NSPOOLGE, IRTYPE
C...
        NSCOUGE=0
        IGEP=0

        IF(ABS(NOUTGE).EQ.1) THEN
          OPEN(63,FILE='fort.63')
          WRITE(63,3220) RUNDES,RUNID,AGRID
          WRITE(63,3645) NDSETSE,NP,DTDP*NSPOOLGE,NSPOOLGE,1
          IGEP=2
          ENDIF

        IF(ABS(NOUTGE).EQ.2) THEN
          OPEN(63,FILE='fort.63',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(63,REC=IGEP+I) RDES4(I)
              ENDDO
            IGEP=IGEP+8
            DO I=1,6
              WRITE(63,REC=IGEP+I) RID4(I)
              ENDDO
            IGEP=IGEP+6
            DO I=1,6
              WRITE(63,REC=IGEP+I) AID4(I)
              ENDDO
            IGEP=IGEP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(63,REC=IGEP+I) RDES8(I)
              ENDDO
            IGEP=IGEP+4
            DO I=1,3
              WRITE(63,REC=IGEP+I) RID8(I)
              ENDDO
            IGEP=IGEP+3
            DO I=1,3
              WRITE(63,REC=IGEP+I) AID8(I)
              ENDDO
            IGEP=IGEP+3
            ENDIF
          WRITE(63,REC=IGEP+1) NDSETSE
          WRITE(63,REC=IGEP+2) NP
          WRITE(63,REC=IGEP+3) DT*NSPOOLGE
          WRITE(63,REC=IGEP+4) NSPOOLGE
          WRITE(63,REC=IGEP+5) 1
          IGEP=IGEP+5
          CLOSE(63)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(63,FILE='fort.63',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF

C...
C....INITILIZE GLOBAL VELOCITY SPOOL COUNTER
C....OPEN GLOBAL VELOCITY OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NDSETSV
C....(NO. OF GLOBAL VELOCITY DATA SETS TO BE SPOOLED),
C....NP, DT*NSPOOLGV, NSPOOLGV, IRTYPE
C...
        NSCOUGV=0
        IGVP=0

        IF(ABS(NOUTGV).EQ.1) THEN
          OPEN(64,FILE='fort.64')
          WRITE(64,3220) RUNDES,RUNID,AGRID
          WRITE(64,3645) NDSETSV,NP,DTDP*NSPOOLGV,NSPOOLGV,2
          IGVP=2
          ENDIF

        IF(ABS(NOUTGV).EQ.2) THEN
```

```
          OPEN(64,FILE='fort.64',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(64,REC=IGVP+I) RDES4(I)
              ENDDO
            IGVP=IGVP+8
            DO I=1,6
              WRITE(64,REC=IGVP+I) RID4(I)
              ENDDO
            IGVP=IGVP+6
            DO I=1,6
              WRITE(64,REC=IGVP+I) AID4(I)
              ENDDO
            IGVP=IGVP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(64,REC=IGVP+I) RDES8(I)
              ENDDO
            IGVP=IGVP+4
            DO I=1,3
              WRITE(64,REC=IGVP+I) RID8(I)
              ENDDO
            IGVP=IGVP+3
            DO I=1,3
              WRITE(64,REC=IGVP+I) AID8(I)
              ENDDO
            IGVP=IGVP+3
            ENDIF
          WRITE(64,REC=IGVP+1) NDSETSV
          WRITE(64,REC=IGVP+2) NP
          WRITE(64,REC=IGVP+3) DT*NSPOOLGV
          WRITE(64,REC=IGVP+4) NSPOOLGV
          WRITE(64,REC=IGVP+5) 2
          IGVP=IGVP+5
          CLOSE(64)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(64,FILE='fort.64',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF

C...
C....INITILIZE GLOBAL CONCENTRATION SPOOL COUNTER
C....OPEN GLOBAL CONCENTRATION OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NDSETSC
C....(NO. OF GLOBAL CONCENTRATION DATA SETS TO BE SPOOLED),
C....NP, DT*NSPOOLGC, NSPOOLGC, IRTYPE
C...
        NSCOUGC=0
        IGCP=0

        IF(ABS(NOUTGC).EQ.1) THEN
          OPEN(73,FILE='fort.73')
          WRITE(73,3220) RUNDES,RUNID,AGRID
          WRITE(73,3645) NDSETSC,NP,DTDP*NSPOOLGC,NSPOOLGC,1
          IGCP=2
          ENDIF

        IF(ABS(NOUTGC).EQ.2) THEN
          OPEN(73,FILE='fort.73',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(73,REC=IGCP+I) RDES4(I)
              ENDDO
            IGCP=IGCP+8
            DO I=1,6
              WRITE(73,REC=IGCP+I) RID4(I)
              ENDDO
            IGCP=IGCP+6
            DO I=1,6
              WRITE(73,REC=IGCP+I) AID4(I)
              ENDDO
```

```
            IGCP=IGCP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(73,REC=IGCP+I) RDES8(I)
              ENDDO
            IGCP=IGCP+4
            DO I=1,3
              WRITE(73,REC=IGCP+I) RID8(I)
              ENDDO
            IGCP=IGCP+3
            DO I=1,3
              WRITE(73,REC=IGCP+I) AID8(I)
              ENDDO
            IGCP=IGCP+3
            ENDIF
          WRITE(73,REC=IGCP+1) NDSETSC
          WRITE(73,REC=IGCP+2) NP
          WRITE(73,REC=IGCP+3) DT*NSPOOLGC
          WRITE(73,REC=IGCP+4) NSPOOLGC
          WRITE(73,REC=IGCP+5) 1
          IGCP=IGCP+5
          CLOSE(73)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(73,FILE='fort.73',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF

C...
C....INITILIZE GLOBAL WIND STRESS SPOOL COUNTER
C....OPEN GLOBAL WIND STRESS OUTPUT FILE
C....WRITE OUT HEADER INFORMATION INCLUDING NDSETSW
C....(NO. OF GLOBAL WIND STRESS DATA SETS TO BE SPOOLED),
C....NP, DT*NSPOOLGW, NSPOOLGW, IRTYPE
C...
        NSCOUGW=0
        IGWP=0

        IF(ABS(NOUTGW).EQ.1) THEN
          OPEN(74,FILE='fort.74')
          WRITE(74,3220) RUNDES,RUNID,AGRID
          WRITE(74,3645) NDSETSW,NP,DTDP*NSPOOLGW,NSPOOLGW,2
          IGWP=2
          ENDIF

        IF(ABS(NOUTGW).EQ.2) THEN
          OPEN(74,FILE='fort.74',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(74,REC=IGWP+I) RDES4(I)
              ENDDO
            IGWP=IGWP+8
            DO I=1,6
              WRITE(74,REC=IGWP+I) RID4(I)
              ENDDO
            IGWP=IGWP+6
            DO I=1,6
              WRITE(74,REC=IGWP+I) AID4(I)
              ENDDO
            IGWP=IGWP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(74,REC=IGWP+I) RDES8(I)
              ENDDO
            IGWP=IGWP+4
            DO I=1,3
              WRITE(74,REC=IGWP+I) RID8(I)
              ENDDO
            IGWP=IGWP+3
            DO I=1,3
              WRITE(74,REC=IGWP+I) AID8(I)
```

```
              ENDDO
             IGWP=IGWP+3
             ENDIF
           WRITE(74,REC=IGWP+1) NDSETSW
           WRITE(74,REC=IGWP+2) NP
           WRITE(74,REC=IGWP+3) DT*NSPOOLGW
           WRITE(74,REC=IGWP+4) NSPOOLGW
           WRITE(74,REC=IGWP+5) 2
           IGWP=IGWP+5
           CLOSE(74)                    ! DO THIS TO FLUSH THE WRITE BUFFER
           OPEN(74,FILE='fort.74',ACCESS='DIRECT',RECL=NBYTE)
           ENDIF

C...
C....INITIALIZE HARMONIC ANALYSIS MATRICIES, MEAN AND SQUARE VECTORS
C...
        IF(IHARIND.EQ.1) THEN
           ICHA=0
           CALL HACOLDS
           IF(NHASE.EQ.1) CALL HACOLDSES(NSTAE)
           IF(NHASV.EQ.1) CALL HACOLDSVS(NSTAV)
           IF(NHAGE.EQ.1) CALL HACOLDSEG(NP)
           IF(NHAGV.EQ.1) CALL HACOLDSVG(NP)
CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
           DO I=1,NP
             ELAV(I)=0.D0
             XVELAV(I)=0.D0
             YVELAV(I)=0.D0
             ELVA(I)=0.D0
             XVELVA(I)=0.D0
             YVELVA(I)=0.D0
             END DO
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

           ENDIF

C...
C...END COLD START SECTION
C...
        ENDIF


C...
C...*************** HOT START PROGRAM SETUP SECTION ***************
C...
      IF(IHOT.NE.0) THEN

C...
C......READ IN 2DDI HOT START INITIAL CONDITION OVER WHOLE DOMAIN
C......THIS FILE ALWAYS HAS A RECL=8 BECAUSE IT IS ASSUMED THAT THE HARMONIC
C......ANALYSIS IS ALWAYS DONE IN 64 BITS, EVEN ON A WORKSTATION
C...
        IF(IHOT.EQ.67) OPEN(67,FILE='fort.67',ACCESS='DIRECT',RECL=8)
        IF(IHOT.EQ.68) OPEN(68,FILE='fort.68',ACCESS='DIRECT',RECL=8)
        IHOTSTP=1
        READ(IHOT,REC=IHOTSTP) IMHS
        IHOTSTP=2
        READ(IHOT,REC=IHOTSTP) TIME
        IHOTSTP=3
        READ(IHOT,REC=IHOTSTP) ITHS
        DO I=1,NP
          READ(IHOT,REC=IHOTSTP+1) ETA1(I)
          READ(IHOT,REC=IHOTSTP+2) ETA2(I)
          READ(IHOT,REC=IHOTSTP+3) UU2(I)
          READ(IHOT,REC=IHOTSTP+4) VV2(I)
```

```
            IHOTSTP=IHOTSTP+4
            IF(IMHS.EQ.10) THEN
              READ(IHOT,REC=IHOTSTP+1) CH1(I)
              IHOTSTP=IHOTSTP+1
              ENDIF
            READ(IHOT,REC=IHOTSTP+1) NNODECODE(I)
            IHOTSTP=IHOTSTP+1
            ETAS(I)=ETA2(I)-ETA1(I)
            NODEREP(I)=MAX0(NODEWETRMP,NODEDRYMIN)
            END DO

          RAMP2=1.0D0
          RAMP1=1.0D0
          IF(NRAMP.EQ.1) THEN
            RAMP1=TANH((2.D0*(ITHS-1)*DTDP/86400.D0)/DRAMP)
            RAMP2=TANH((2.D0*ITHS*DTDP/86400.D0)/DRAMP)
            ENDIF

C
C....SET POSITIONS IN BOUNDARY CONDITION, WIND AND OUTPUT FILES
C
          WRITE(16,1112)
          WRITE(16,1794)
1794      FORMAT(//,' INFORMATION ABOUT RE-STARTING THE TIME SERIES',
     &             ' OUTPUT FILES (UNITS 61-64,71,74),',
     &    /,' WIND/PRESSURE FILE (UNIT 22) AND FLOW BOUNDARY CONDITION',
     &      ' FILE (UNIT 20)',//)

C......INITIALLY, ZERO OUT THE NORMAL FLOW ON ALL BOUNDARIES

          DO I=1,NVEL
            QN2(I)=0.D0
            QN1(I)=0.D0
            QN0(I)=0.D0
            END DO

C......RESTART THE PERIODIC NORMAL FLOW BOUNDARY CONDITION

          IF((NFLUXF.EQ.1).AND.(NFFR.GT.0)) THEN
            DO J=1,NFFR
              IF(FPER(J).EQ.0.) THEN
                NCYC=0.
                ELSE
                NCYC=INT(TIME/FPER(J))
                ENDIF
              ARGJ1=FAMIG(J)*(TIME-DTDP-NCYC*FPER(J))+FFACE(J)
              ARGJ2=FAMIG(J)*(TIME-NCYC*FPER(J))+FFACE(J)
              RFF1=FFF(J)*RAMP1
              RFF2=FFF(J)*RAMP2
              DO I=1,NVEL
                ARG1=ARGJ1-QNPH(J,I)
                ARG2=ARGJ2-QNPH(J,I)
                QN1(I)=QN1(I)+QNAM(J,I)*RFF1*COS(ARG1)
                QN2(I)=QN2(I)+QNAM(J,I)*RFF2*COS(ARG2)
                END DO
              END DO
              ENDIF

C......FIND PROPER PLACE IN THE APERIODIC NORMAL FLOW BOUNDARY CONDITION FILE

          IF((NFLUXF.EQ.1).AND.(NFFR.EQ.0)) THEN
              DO J=1,NVEL
                QNIN2(J)=0.D0
                QNIN1(J)=0.D0
                END DO
            IF(NSCREEN.EQ.1) WRITE(6,1112)
            WRITE(16,1112)
            IF(NSCREEN.EQ.1) WRITE(6,1978)
            WRITE(16,1978)
 1978       FORMAT(/,1X,'LOCATING NORMAL FLOW INFORMATION IN UNIT 20',/)
```

```
          OPEN(20,FILE='fort.20')
          QTIME1=(STATIM-REFTIM)*86400.D0
          QTIME2=QTIME1+FTIMINC
          DO J=1,NVEL
            QNIN1(J)=0.D0
            IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                           .OR.(LBCODEI(J).EQ.22))
     &                                                READ(20,*) QNIN1(J)
            END DO
          DO J=1,NVEL
            QNIN2(J)=0.D0
            IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                           .OR.(LBCODEI(J).EQ.22))
     &                                                READ(20,*) QNIN2(J)
            END DO
          DO IT=1,ITHS-1
            TIMEIT=IT*DTDP + (STATIM - REFTIM)*86400.D0
            IF(TIMEIT.GT.QTIME2) THEN
              QTIME1=QTIME2
              QTIME2=QTIME2+FTIMINC
              DO J=1,NVEL
                IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                               .OR.(LBCODEI(J).EQ.22)) THEN
                  QNIN1(J)=QNIN2(J)
                  READ(20,*) QNIN2(J)
                  ENDIF
                END DO
              ENDIF
            END DO
          QTRATIO=(TIMEIT-QTIME1)/FTIMINC
          DO I=1,NVEL
            QN1(I)=RAMP1*(QNIN1(I)+QTRATIO*(QNIN2(I)-QNIN1(I)))
            END DO
          IF(TIME.GT.QTIME2) THEN
            QTIME1=QTIME2
            QTIME2=QTIME2+FTIMINC
            DO J=1,NVEL
              IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
     &                             .OR.(LBCODEI(J).EQ.22)) THEN
                QNIN1(J)=QNIN2(J)
                READ(20,*) QNIN2(J)
                ENDIF
              END DO
            ENDIF
          QTRATIO=(TIME-QTIME1)/FTIMINC
          DO I=1,NVEL
            QN2(I)=RAMP2*(QNIN1(I)+QTRATIO*(QNIN2(I)-QNIN1(I)))
            END DO
          ENDIF

C...
C...RESTART SUPERCRITICAL OUTWARD NORMAL FLOW OVER SPECIFIED
C....EXTERNAL BARRIER BOUNDARY NODES
C...
          IF(NFLUXB.EQ.1) THEN
            DO I=1,NVEL
              IF((LBCODEI(I).EQ.3).OR.(LBCODEI(I).EQ.13)
     &          .OR.(LBCODEI(I).EQ.23)) THEN
                NNBB=NBV(I)
                RBARWL=2.D0*(ETA1(NNBB)-BARLANHT(I))/3.D0
                IF(RBARWL.GT.0.0D0) THEN
                  QN1(I)=-RAMP1*BARLANCFSP(I)*RBARWL*(RBARWL*G)**0.5D0
                ENDIF
                RBARWL=2.D0*(ETA2(NNBB)-BARLANHT(I))/3.D0
                IF(RBARWL.GT.0.0D0) THEN
                  QN2(I)=-RAMP2*BARLANCFSP(I)*RBARWL*(RBARWL*G)**0.5D0
                ENDIF
              ENDIF
            END DO
          ENDIF
```

```
C...
C...RESTART INWARD/OUTWARD NORMAL FLOW OVER SPECIFIED
C....INTERNAL BARRIER BOUNDARY NODES
C...
        IF(NFLUXIB.EQ.1) THEN
          DO I=1,NVEL
            IF((LBCODEI(I).EQ.4).OR.(LBCODEI(I).EQ.24)) THEN
              NNBB1=NBV(I)       ! GLOBAL NODE NUMBER ON THIS SIDE OF BARRIER
              NNBB2=IBCONN(I)    ! GLOBAL NODE NUMBER ON OPPOSITE SIDE OF BARRIER
C.............RESET INFORMATION FOR K-1 TIME LEVEL
              RBARWL1=ETA1(NNBB1)-BARINHT(I)
              RBARWL2=ETA1(NNBB2)-BARINHT(I)
              RBARWL1F=2.0D0*RBARWL1/3.0D0
              RBARWL2F=2.0D0*RBARWL2/3.0D0
              IF((RBARWL1.LT.0.0).AND.(RBARWL2.LT.0.0)) THEN ! WATER LEVEL BELOW BARRIER
                QN1(I)=0.0                                   ! NO FLOW
                GOTO 1998
              ENDIF
              IF(RBARWL1.EQ.RBARWL2) THEN ! WATER LEVEL EQUAL ON BOTH SIDES OF BARRIER
                QN1(I)=0.0                ! NO FLOW
                GOTO 1998
              ENDIF
              IF(RBARWL1.GT.RBARWL2) THEN ! WATER LEVEL GREATER ON THIS SIDE OF BARRIER
                IF(RBARWL2.GT.RBARWL1F) THEN ! OUTWARD SUBCRITICAL FLOW
                  QN1(I)=-RAMP1*BARINCFSB(I)*RBARWL2*
     &                   (2*G*(RBARWL1-RBARWL2))**0.5D0
                  GOTO 1998
                ELSE                         ! OUTWARD SUPERCRITICAL FLOW
                  QN1(I)=-RAMP1*BARINCFSP(I)*RBARWL1F*
     &                   (RBARWL1F*G)**0.5D0
                  GOTO 1998
                ENDIF
              ENDIF
              IF(RBARWL2.GT.RBARWL1) THEN  ! WATER LEVEL LOWER ON THIS SIDE OF BARRIER
                IF(RBARWL1.GT.RBARWL2F) THEN ! INWARD SUBCRITICAL FLOW
                  QN1(I)=RAMP1*BARINCFSB(I)*RBARWL1*
     &                  (2*G*(RBARWL2-RBARWL1))**0.5D0
                  GOTO 1998
                ELSE                          ! INWARD SUPERCRITICAL FLOW
                  QN1(I)=RAMP1*BARINCFSP(I)*RBARWL2F*(RBARWL2F*G)**0.5D0
                  GOTO 1998
                ENDIF
              ENDIF
1998          CONTINUE
C.............RESET INFORMATION FOR K TIME LEVEL
              RBARWL1=ETA2(NNBB1)-BARINHT(I)
              RBARWL2=ETA2(NNBB2)-BARINHT(I)
              RBARWL1F=2.0D0*RBARWL1/3.0D0
              RBARWL2F=2.0D0*RBARWL2/3.0D0
              IF((RBARWL1.LT.0.0).AND.(RBARWL2.LT.0.0)) THEN ! WATER LEVEL BELOW BARRIER
                QN2(I)=0.0                                   ! NO FLOW
                GOTO 1999
              ENDIF
              IF(RBARWL1.EQ.RBARWL2) THEN ! WATER LEVEL EQUAL ON BOTH SIDES OF BARRIER
                QN2(I)=0.0                ! NO FLOW
                GOTO 1999
              ENDIF
              IF(RBARWL1.GT.RBARWL2) THEN ! WATER LEVEL GREATER ON THIS SIDE OF BARRIER
                IF(RBARWL2.GT.RBARWL1F) THEN ! OUTWARD SUBCRITICAL FLOW
                  QN2(I)=-RAMP2*BARINCFSB(I)*RBARWL2*
     &                   (2*G*(RBARWL1-RBARWL2))**0.5D0
                  GOTO 1999
                ELSE                         ! OUTWARD SUPERCRITICAL FLOW
                  QN2(I)=-RAMP2*BARINCFSP(I)*RBARWL1F*
     &                   (RBARWL1F*G)**0.5D0
                  GOTO 1999
                ENDIF
              ENDIF
              IF(RBARWL2.GT.RBARWL1) THEN !WATER LEVEL LOWER ON THIS SIDE OF BARRIER
```

```
              IF(RBARWL1.GT.RBARWL2F) THEN ! INWARD SUBCRITICAL FLOW
                QN2(I)=RAMP2*BARINCFSB(I)*RBARWL1*
     &                 (2*G*(RBARWL2-RBARWL1))**0.5D0
                GOTO 1999
              ELSE                          ! INWARD SUPERCRITICAL FLOW
                QN2(I)=RAMP2*BARINCFSP(I)*RBARWL2F*(RBARWL2F*G)**0.5D0
                GOTO 1999
              ENDIF
            ENDIF
1999        CONTINUE
          ENDIF
        END DO
      ENDIF

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...COMPUTE OR READ THE WIND FILE AT TIME OF HOT START AND STICK INTO
CWIND...WSX2(I),WSY2(I),PR2(I)
CWIND...
      IF(NWS.EQ.1) THEN
        OPEN(22,FILE='fort.22')
        WTIME1 = (STATIM - REFTIM)*86400.D0
        WTIME2 = WTIME1 + WTIMINC
        DO J=1,ITHS
          DO I=1,NP
            READ(22,*) NHG,WSX2(NHG),WSY2(NHG),PR2(NHG)
            END DO
          END DO
        DO I=1,NP
          WSX2(NHG)=RAMP2*WSX2(NHG)
          WSY2(NHG)=RAMP2*WSY2(NHG)
          PR2(NHG)=RAMP2*PR2(NHG)
          END DO
        ENDIF

      IF(NWS.EQ.2) THEN
        OPEN(22,FILE='fort.22')
        WTIME1 = (STATIM - REFTIM)*86400.D0
        WTIME2 = WTIME1 + WTIMINC
        READ(22,*) (NHG,WVNX2(NHG),WVNY2(NHG),PRN2(NHG),I=1,NP)
        DO IT=1,ITHS
          TIMEIT=IT*DTDP + (STATIM - REFTIM)*86400.D0
          IF(TIMEIT.GT.WTIME2) THEN
            WTIME1=WTIME2
            WTIME2=WTIME2+WTIMINC
            DO I=1,NP
              WVNX1(I)=WVNX2(I)
              WVNY1(I)=WVNY2(I)
              PRN1(I)=PRN2(I)
              READ(22,*) NHG,WVNX2(NHG),WVNY2(NHG),PRN2(NHG)
              END DO
            ENDIF
          END DO
        WTRATIO=(TIME-WTIME1)/WTIMINC
        DO I=1,NP
          WSX2(I)=RAMP2*(WVNX1(I)+WTRATIO*(WVNX2(I)-WVNX1(I)))
          WSY2(I)=RAMP2*(WVNY1(I)+WTRATIO*(WVNY2(I)-WVNY1(I)))
          PR2(I)=RAMP2*(PRN1(I)+WTRATIO*(PRN2(I)-PRN1(I)))
          END DO
        ENDIF

      IF(NWS.EQ.3) THEN
        OPEN(22,FILE='fort.22')
 2223   CALL NWS3GET(X,Y,SLAM,SFEA,WVNX2,WVNY2,IWTIME,IWYR,WTIMED,NP,
     &                NWLON,NWLAT,WLATMAX,WLONMIN,WLATINC,WLONINC,ICS)
        IF(IWYR.NE.IREFYR) THEN
          IWTIMEP=IWTIME
          DO I=1,NP
            WVNX1(I)=WVNX2(I)
```

```
          WVNY1(I)=WVNY2(I)
          END DO
        GOTO 2223
        ENDIF
      IF(WTIMED.LE.WREFTIM) THEN
        IWTIMEP=IWTIME
        DO I=1,NP
          WVNX1(I)=WVNX2(I)
          WVNY1(I)=WVNY2(I)
          END DO
        GOTO 2223
        ENDIF
      IF(NSCREEN.EQ.1) WRITE(6,*)'FOUND WIND DATA AT TIME= ',IWTIMEP
      WRITE(16,*) 'FOUND WIND DATA AT TIME =',IWTIMEP
      IF(NSCREEN.EQ.1) WRITE(6,*)'FOUND WIND DATA AT TIME= ',IWTIME
      WRITE(16,*) 'FOUND WIND DATA AT TIME =',IWTIME
      WTIME2=WTIMED-WREFTIM                    !CAST INTO MODEL TIME REFRENCE
      WTIME1=WTIME2-WTIMINC
      DO IT=1,ITHS
        TIMEIT=IT*DTDP + (STATIM - REFTIM)*86400.D0
        IF(TIMEIT.GT.WTIME2) THEN
          WTIME1=WTIME2
          WTIME2=WTIME2+WTIMINC
          DO I=1,NP
            WVNX1(I)=WVNX2(I)
            WVNY1(I)=WVNY2(I)
            END DO
          CALL NWS3GET(X,Y,SLAM,SFEA,WVNX2,WVNY2,IWTIME,IWYR,WTIMED,
     &          NP,NWLON,NWLAT,WLATMAX,WLONMIN,WLATINC,WLONINC,ICS)
          IF(NSCREEN.EQ.1) WRITE(6,*)'WIND FILE ADVANCED TO TIME',
     &                               ' = ', IWTIME
          WRITE(16,*) 'WIND FILE ADVANCED TO TIME = ',IWTIME
          ENDIF
        END DO
      WTRATIO=(TIME-WTIME1)/WTIMINC
      DO I=1,NP                                !INTERPOLATE IN TIME
        WINDX = WVNX1(I) + WTRATIO*(WVNX2(I)-WVNX1(I))
        WINDY = WVNY1(I) + WTRATIO*(WVNY2(I)-WVNY1(I))
        WINDMAG=SQRT(WINDX*WINDX+WINDY*WINDY)
        WDRAGCO = 0.001*(0.75+0.067*WINDMAG)
        IF(WDRAGCO.GT.0.003) WDRAGCO=0.003
        WSX2(I)=RAMP2*0.001293*WDRAGCO*WINDX*WINDMAG
        WSY2(I)=RAMP2*0.001293*WDRAGCO*WINDY*WINDMAG
        END DO
      ENDIF

    IF(NWS.EQ.4) THEN
      OPEN(22,FILE='fort.22')
      WTIME1 = (STATIM - REFTIM)*86400.D0
      WTIME2 = WTIME1 + WTIMINC
      CALL NWS4GET(WVNX2,WVNY2,PRN2,NP,1000.,G)
      DO IT=1,ITHS
        TIMEIT=IT*DTDP + (STATIM - REFTIM)*86400.D0
        IF(TIMEIT.GT.WTIME2) THEN
          WTIME1=WTIME2
          WTIME2=WTIME2+WTIMINC
          DO I=1,NP
            WVNX1(I)=WVNX2(I)
            WVNY1(I)=WVNY2(I)
            PRN1(I)=PRN2(I)
            END DO
          CALL NWS4GET(WVNX2,WVNY2,PRN2,NP,1000.,G)
          ENDIF
        END DO
      WTRATIO=(TIME-WTIME1)/WTIMINC
      DO I=1,NP
        WINDX = WVNX1(I) + WTRATIO*(WVNX2(I)-WVNX1(I))
        WINDY = WVNY1(I) + WTRATIO*(WVNY2(I)-WVNY1(I))
        WINDMAG = SQRT(WINDX*WINDX+WINDY*WINDY)
        WDRAGCO = 0.001*(0.75+0.067*WINDMAG)
```

```
            IF(WDRAGCO.GT.0.003) WDRAGCO=0.003
            WSX2(I) = RAMP2*0.001293*WDRAGCO*WINDX*WINDMAG
            WSY2(I) = RAMP2*0.001293*WDRAGCO*WINDY*WINDMAG
            PR2(I)=RAMP2*(PRN1(I)+WTRATIO*(PRN2(I)-PRN1(I)))
            END DO
          ENDIF

        IF(NWS.EQ.10) THEN
          WTIME1=TIME
          WTIME2=WTIME1+WTIMINC
          NWSGGWI=0
          CALL NWS10GET(NWSGGWI,SLAM,SFEA,WVNX1,WVNY1,PRN1,NP,1000.,G,
     &                  NWLON,NWLAT)
          NWSGGWI=1
          CALL NWS10GET(NWSGGWI,SLAM,SFEA,WVNX2,WVNY2,PRN2,NP,1000.,G,
     &                  NWLON,NWLAT)
          WTRATIO=(TIME-WTIME1)/WTIMINC
          DO I=1,NP
            WINDX = WVNX1(I) + WTRATIO*(WVNX2(I)-WVNX1(I))
            WINDY = WVNY1(I) + WTRATIO*(WVNY2(I)-WVNY1(I))
            WINDMAG = SQRT(WINDX*WINDX+WINDY*WINDY)
            WDRAGCO = 0.001*(0.75+0.067*WINDMAG)
            IF(WDRAGCO.GT.0.003) WDRAGCO=0.003
            WSX2(I) = RAMP2*0.001293*WDRAGCO*WINDX*WINDMAG
            WSY2(I) = RAMP2*0.001293*WDRAGCO*WINDY*WINDMAG
            PR2(I)=RAMP2*(PRN1(I)+WTRATIO*(PRN2(I)-PRN1(I)))
            END DO
          ENDIF
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...COMPUTE THE TIDAL POTENTIAL AT THE TIME OF HOT START
CTIP...
        IF(NTIP.GE.1) THEN
          DO J=1,NTIF
            IF(PERT(J).EQ.0.) THEN
              NCYC=0
              ELSE
              NCYC=INT(TIME/PERT(J))
              ENDIF
            ARGT=AMIGT(J)*(TIME-NCYC*PERT(J))+FACET(J)
            TPMUL=RAMP*ETRF(J)*TPK(J)*FFT(J)
            SALTMUL=RAMP*FFT(J)
            NA=NINT(0.00014/AMIGT(J))
            IF(NA.EQ.1) THEN                          !SEMI-DIURNAL SPECIES
              DO I=1,NP
                ARGTP=ARGT+2.*SLAM(I)
                ARGSALT=ARGT-SALTPHA(J,I)
                CCSFEA=COS(SFEA(I))
                CCSFEA=CCSFEA*CCSFEA
                TIP2(I)=TIP2(I)+TPMUL*CCSFEA*COS(ARGTP)
     &                          +SALTMUL*SALTAMP(J,I)*COS(ARGSALT)
                END DO
              ENDIF
            IF(NA.EQ.2) THEN
              DO I=1,NP
                ARGTP=ARGT+SLAM(I)
                ARGSALT=ARGT-SALTPHA(J,I)
                S2SFEA=SIN(2.*SFEA(I))
                TIP2(I)=TIP2(I)+TPMUL*S2SFEA*COS(ARGTP)
     &                          +SALTMUL*SALTAMP(J,I)*COS(ARGSALT)
                END DO
              ENDIF
            END DO
          ENDIF
CTIP...
```

```
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C...
C....IF RESTARTING THE ELEVATION STATION OUTPUT FILE, GO TO THE PROPER PLACE
C....IN THE FILE.  OTHERWISE ZERO OUT NSCOUE.
C...
        READ(IHOT,REC=IHOTSTP+1) IESTP
        READ(IHOT,REC=IHOTSTP+2) NSCOUE
        IHOTSTP=IHOTSTP+2
        IF(NOUTE.GT.0) WRITE(16,1040) IESTP,NSCOUE
 1040   FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN ELEVATION ',
     &                  'STATION FILE BY THE TIME OF THE HOT START',
     &             /,8X,'SPOOL COUNTER = ',I6)
        IF(NOUTE.LT.0) THEN
          IESTP=0
          NSCOUE=0
          IF((NTCYSE.LT.ITHS).AND.(NSPOOLE.GT.0)) THEN
            NTCYSE=NTCYSE+((ITHS-NTCYSE)/NSPOOLE)*NSPOOLE
            IF(NTCYSE.LT.ITHS) NTCYSE=NTCYSE+NSPOOLE
            IF(NSPOOLE.NE.0) NTRSPE=(NTCYFE-NTCYSE)/NSPOOLE
            ENDIF
          WRITE(16,1041)
 1041     FORMAT(//,' A NEW ELEVATION STATION FILE WILL BE STARTED')
          ENDIF

        IF(NOUTE.EQ.-2) THEN
          OPEN(61,FILE='fort.61',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(61,REC=IESTP+I) RDES4(I)
              ENDDO
            IESTP=IESTP+8
            DO I=1,6
              WRITE(61,REC=IESTP+I) RID4(I)
              ENDDO
            IESTP=IESTP+6
            DO I=1,6
              WRITE(61,REC=IESTP+I) AID4(I)
              ENDDO
            IESTP=IESTP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(61,REC=IESTP+I) RDES8(I)
              ENDDO
            IESTP=IESTP+4
            DO I=1,3
              WRITE(61,REC=IESTP+I) RID8(I)
              ENDDO
            IESTP=IESTP+3
            DO I=1,3
              WRITE(61,REC=IESTP+I) AID8(I)
              ENDDO
            IESTP=IESTP+3
            ENDIF
          WRITE(61,REC=IESTP+1) NTRSPE
          WRITE(61,REC=IESTP+2) NSTAE
          WRITE(61,REC=IESTP+3) DT*NSPOOLE
          WRITE(61,REC=IESTP+4) NSPOOLE
          WRITE(61,REC=IESTP+5) 1
          IESTP=IESTP+5
          CLOSE(61)                        ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(61,FILE='fort.61',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF
        IF(NOUTE.EQ.-1) THEN
          OPEN(61,FILE='fort.61')
          WRITE(61,3220) RUNDES,RUNID,AGRID
          WRITE(61,3645) NTRSPE,NSTAE,DTDP*NSPOOLE,NSPOOLE,1
          IESTP=2
```

```
          ENDIF
        IF(NOUTE.EQ.1) THEN
          OPEN(61,FILE='fort.61')
          DO I=1,IESTP
 1050       FORMAT(1X)
            READ(61,1050)
            ENDDO
          ENDIF
        IF(NOUTE.EQ.2)
     &                OPEN(61,FILE='fort.61',ACCESS='DIRECT',RECL=NBYTE)

C...
C....GO TO THE PROPER PLACE IN THE VELOCITY STATION OUTPUT FILE
C...
        READ(IHOT,REC=IHOTSTP+1) IVSTP
        READ(IHOT,REC=IHOTSTP+2) NSCOUV
        IHOTSTP=IHOTSTP+2
        IF(NOUTV.GT.0) WRITE(16,1042) IVSTP,NSCOUV
 1042   FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN VELOCITY ',
     &                  'STATION FILE BY THE TIME OF THE HOT START',
     &          /,8X,'SPOOL COUNTER =',I6)
        IF(NOUTV.LT.0) THEN
          IVSTP=0
          NSCOUV=0
          IF((NTCYSV.LT.ITHS).AND.(NSPOOLV.GT.0)) THEN
            NTCYSV=NTCYSV+((ITHS-NTCYSV)/NSPOOLV)*NSPOOLV
            IF(NTCYSV.LT.ITHS) NTCYSV=NTCYSV+NSPOOLV
            NTRSPV=(NTCYFV-NTCYSV)/NSPOOLV
            ENDIF
          WRITE(16,1043)
 1043     FORMAT(//,' A NEW VELOCITY STATION FILE WILL BE STARTED')
          ENDIF

        IF(NOUTV.EQ.-2) THEN
          OPEN(62,FILE='fort.62',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(62,REC=IVSTP+I) RDES4(I)
              ENDDO
            IVSTP=IVSTP+8
            DO I=1,6
              WRITE(62,REC=IVSTP+I) RID4(I)
              ENDDO
            IVSTP=IVSTP+6
            DO I=1,6
              WRITE(62,REC=IVSTP+I) AID4(I)
              ENDDO
            IVSTP=IVSTP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(62,REC=IVSTP+I) RDES8(I)
              ENDDO
            IVSTP=IVSTP+4
            DO I=1,3
              WRITE(62,REC=IVSTP+I) RID8(I)
              ENDDO
            IVSTP=IVSTP+3
            DO I=1,3
              WRITE(62,REC=IVSTP+I) AID8(I)
              ENDDO
            IVSTP=IVSTP+3
            ENDIF
          WRITE(62,REC=IVSTP+1) NTRSPV
          WRITE(62,REC=IVSTP+2) NSTAV
          WRITE(62,REC=IVSTP+3) DT*NSPOOLV
          WRITE(62,REC=IVSTP+4) NSPOOLV
          WRITE(62,REC=IVSTP+5) 2
          IVSTP=IVSTP+5
          CLOSE(62)                    ! DO THIS TO FLUSH THE WRITE BUFFER
```

```
          OPEN(62,FILE='fort.62',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF
        IF(NOUTV.EQ.-1) THEN
          OPEN(62,FILE='fort.62')
          WRITE(62,3220) RUNDES,RUNID,AGRID
          WRITE(62,3645) NTRSPV,NSTAV,DTDP*NSPOOLV,NSPOOLV,2
          IVSTP=2
          ENDIF
        IF(NOUTV.EQ.1) THEN
          OPEN(62,FILE='fort.62')
          DO I=1,IVSTP
            READ(62,1050)
            ENDDO
          ENDIF
        IF(NOUTV.EQ.2)
     &                OPEN(62,FILE='fort.62',ACCESS='DIRECT',RECL=NBYTE)

C...
C....GO TO THE PROPER PLACE IN THE CONCENTRATION STATION OUTPUT FILE
C...
        READ(IHOT,REC=IHOTSTP+1) ICSTP
        READ(IHOT,REC=IHOTSTP+2) NSCOUC
        IHOTSTP=IHOTSTP+2
        IF(NOUTC.GT.0) WRITE(16,1044) ICSTP,NSCOUC
 1044   FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN CONCENTRATION ',
     &                  'STATION FILE BY THE TIME OF THE HOT START',
     &          /,8X,'SPOOL COUNTER = ',I6)
        IF(NOUTC.LT.0) THEN
          ICSTP=0
          NSCOUC=0
          IF((NTCYSC.LT.ITHS).AND.(NSPOOLC.GT.0)) THEN
            NTCYSC=NTCYSC+((ITHS-NTCYSC)/NSPOOLC)*NSPOOLC
            IF(NTCYSC.LT.ITHS) NTCYSC=NTCYSC+NSPOOLC
            NTRSPC=(NTCYFC-NTCYSC)/NSPOOLC
            ENDIF
          WRITE(16,1045)
 1045     FORMAT(//,' A NEW CONCENTRATION STATION FILE WILL BE STARTED')
          ENDIF

        IF(NOUTC.EQ.-2) THEN
          OPEN(71,FILE='fort.71',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(71,REC=ICSTP+I) RDES4(I)
              ENDDO
            ICSTP=ICSTP+8
            DO I=1,6
              WRITE(71,REC=ICSTP+I) RID4(I)
              ENDDO
            ICSTP=ICSTP+6
            DO I=1,6
              WRITE(71,REC=ICSTP+I) AID4(I)
              ENDDO
            ICSTP=ICSTP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(71,REC=ICSTP+I) RDES8(I)
              ENDDO
            ICSTP=ICSTP+4
            DO I=1,3
              WRITE(71,REC=ICSTP+I) RID8(I)
              ENDDO
            ICSTP=ICSTP+3
            DO I=1,3
              WRITE(61,REC=ICSTP+I) AID8(I)
              ENDDO
            ICSTP=ICSTP+3
            ENDIF
          WRITE(71,REC=ICSTP+1) NTRSPC
```

```
          WRITE(71,REC=ICSTP+2) NSTAC
          WRITE(71,REC=ICSTP+3) DT*NSPOOLC
          WRITE(71,REC=ICSTP+4) NSPOOLC
          WRITE(71,REC=ICSTP+5) 1
          ICSTP=ICSTP+5
          CLOSE(71)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(71,FILE='fort.71',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF
        IF(NOUTC.EQ.-1) THEN
          OPEN(71,FILE='fort.71')
          WRITE(71,3220) RUNDES,RUNID,AGRID
          WRITE(71,3645) NTRSPC,NSTAC,DTDP*NSPOOLC,NSPOOLC,1
          ICSTP=2
          ENDIF
        IF(NOUTC.EQ.1) THEN
          OPEN(71,FILE='fort.71')
          DO I=1,ICSTP
            READ(71,1050)
            ENDDO
          ENDIF
        IF(NOUTC.EQ.2)
     &               OPEN(71,FILE='fort.71',ACCESS='DIRECT',RECL=NBYTE)

C...
C....GO TO THE PROPER PLACE IN THE GLOBAL ELEVATION OUTPUT FILE
C...
        READ(IHOT,REC=IHOTSTP+1) IGEP
        READ(IHOT,REC=IHOTSTP+2) NSCOUGE
        IHOTSTP=IHOTSTP+2
        IF(NOUTGE.GT.0) WRITE(16,1046) IGEP,NSCOUGE
 1046   FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN THE GLOBAL ',
     &                  'ELEVATION FILE BY THE TIME OF THE HOT START',
     &          /,8X,'SPOOL COUNTER =',I6)
        IF(NOUTGE.LT.0) THEN
          IGEP=0
          NSCOUGE=0
          IF((NTCYSGE.LT.ITHS).AND.(NSPOOLGE.GT.0)) THEN
            NTCYSGE=NTCYSGE+((ITHS-NTCYSGE)/NSPOOLGE)*NSPOOLGE
            IF(NTCYSGE.LT.ITHS) NTCYSGE=NTCYSGE+NSPOOLGE
            NDSETSE=(NTCYFGE-NTCYSGE)/NSPOOLGE
            ENDIF
          WRITE(16,1047)
 1047     FORMAT(//,' A NEW GLOBAL ELEVATION FILE WILL BE STARTED')
          ENDIF

        IF(NOUTGE.EQ.-2) THEN
          OPEN(63,FILE='fort.63',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(63,REC=IGEP+I) RDES4(I)
              ENDDO
            IGEP=IGEP+8
            DO I=1,6
              WRITE(63,REC=IGEP+I) RID4(I)
              ENDDO
            IGEP=IGEP+6
            DO I=1,6
              WRITE(63,REC=IGEP+I) AID4(I)
              ENDDO
            IGEP=IGEP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(63,REC=IGEP+I) RDES8(I)
              ENDDO
            IGEP=IGEP+4
            DO I=1,3
              WRITE(63,REC=IGEP+I) RID8(I)
              ENDDO
            IGEP=IGEP+3
```

```
            DO I=1,3
              WRITE(63,REC=IGEP+I) AID8(I)
              ENDDO
            IGEP=IGEP+3
            ENDIF
          WRITE(63,REC=IGEP+1) NDSETSE
          WRITE(63,REC=IGEP+2) NP
          WRITE(63,REC=IGEP+3) DT*NSPOOLGE
          WRITE(63,REC=IGEP+4) NSPOOLGE
          WRITE(63,REC=IGEP+5) 1
          IGEP=IGEP+5
          CLOSE(63)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(63,FILE='fort.63',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF
        IF(NOUTGE.EQ.-1) THEN
          OPEN(63,FILE='fort.63')
          WRITE(63,3220) RUNDES,RUNID,AGRID
          WRITE(63,3645) NDSETSE,NP,DTDP*NSPOOLGE,NSPOOLGE,1
          IGEP=2
          ENDIF
        IF(NOUTGE.EQ.1) THEN
          OPEN(63,FILE='fort.63')
          DO I=1,IGEP
            READ(63,1050)
            ENDDO
          ENDIF
        IF(NOUTGE.EQ.2)
     &               OPEN(63,FILE='fort.63',ACCESS='DIRECT',RECL=NBYTE)

C...
C....GO TO THE PROPER PLACE IN THE GLOBAL VELOCITY OUTPUT FILE
C...
        READ(IHOT,REC=IHOTSTP+1) IGVP
        READ(IHOT,REC=IHOTSTP+2) NSCOUGV
        IHOTSTP=IHOTSTP+2
        IF(NOUTGV.GT.0) WRITE(16,1048) IGVP,NSCOUGV
 1048   FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN THE GLOBAL ',
     &                  'VELOCITY FILE BY THE TIME OF THE HOT START',
     &          /,8X,'SPOOL COUNTER =',I6)
        IF(NOUTGV.LT.0) THEN
          IGVP=0
          NSCOUGV=0
          IF((NTCYSGV.LT.ITHS).AND.(NSPOOLGV.GT.0)) THEN
            NTCYSGV=NTCYSGV+((ITHS-NTCYSGV)/NSPOOLGV)*NSPOOLGV
            IF(NTCYSGV.LT.ITHS) NTCYSGV=NTCYSGV+NSPOOLGV
            NDSETSV=(NTCYFGV-NTCYSGV)/NSPOOLGV
            ENDIF
          WRITE(16,1049)
 1049     FORMAT(//,' A NEW GLOBAL VELOCITY FILE WILL BE STARTED')
          ENDIF

        IF(NOUTGV.EQ.-2) THEN
          OPEN(64,FILE='fort.64',ACCESS='DIRECT',RECL=NBYTE)
          IF(NBYTE.EQ.4) THEN
            DO I=1,8
              WRITE(64,REC=IGVP+I) RDES4(I)
              ENDDO
            IGVP=IGVP+8
            DO I=1,6
              WRITE(64,REC=IGVP+I) RID4(I)
              ENDDO
            IGVP=IGVP+6
            DO I=1,6
              WRITE(64,REC=IGVP+I) AID4(I)
              ENDDO
            IGVP=IGVP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(64,REC=IGVP+I) RDES8(I)
```

```
            ENDDO
          IGVP=IGVP+4
          DO I=1,3
            WRITE(64,REC=IGVP+I) RID8(I)
            ENDDO
          IGVP=IGVP+3
          DO I=1,3
            WRITE(64,REC=IGVP+I) AID8(I)
            ENDDO
          IGVP=IGVP+3
          ENDIF
        WRITE(64,REC=IGVP+1) NDSETSV
        WRITE(64,REC=IGVP+2) NP
        WRITE(64,REC=IGVP+3) DT*NSPOOLGV
        WRITE(64,REC=IGVP+4) NSPOOLGV
        WRITE(64,REC=IGVP+5) 2
        IGVP=IGVP+5
        CLOSE(64)                    ! DO THIS TO FLUSH THE WRITE BUFFER
        OPEN(64,FILE='fort.64',ACCESS='DIRECT',RECL=NBYTE)
        ENDIF
      IF(NOUTGV.EQ.-1) THEN
        OPEN(64,FILE='fort.64')
        WRITE(64,3220) RUNDES,RUNID,AGRID
        WRITE(64,3645) NDSETSV,NP,DTDP*NSPOOLGV,NSPOOLGV,2
        IGVP=2
        ENDIF
       IF(NOUTGV.EQ.1) THEN
        OPEN(64,FILE='fort.64')
        DO I=1,IGVP
          READ(64,1050)
          ENDDO
        ENDIF
       IF(NOUTGV.EQ.2)
     &              OPEN(64,FILE='fort.64',ACCESS='DIRECT',RECL=NBYTE)

C...
C....GO TO THE PROPER PLACE IN THE GLOBAL CONCENTRATION OUTPUT FILE
C...
       READ(IHOT,REC=IHOTSTP+1) IGCP
       READ(IHOT,REC=IHOTSTP+2) NSCOUGC
       IHOTSTP=IHOTSTP+2
       IF(NOUTGC.GT.0) WRITE(16,1053) IGCP,NSCOUGC
 1053  FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN THE GLOBAL ',
     &           'CONCENTRATION FILE BY THE TIME OF THE HOT START',
     &         /,8X,'SPOOL COUNTER =',I6)
       IF(NOUTGC.LT.0) THEN
         IGCP=0
         NSCOUGC=0
         IF((NTCYSGC.LT.ITHS).AND.(NSPOOLGC.GT.0)) THEN
           NTCYSGC=NTCYSGC+((ITHS-NTCYSGC)/NSPOOLGC)*NSPOOLGC
           IF(NTCYSGC.LT.ITHS) NTCYSGC=NTCYSGC+NSPOOLGC
           NDSETSC=(NTCYFGC-NTCYSGC)/NSPOOLGC
           ENDIF
         WRITE(16,1054)
 1054    FORMAT(//,' A NEW GLOBAL CONCENTRATION FILE WILL BE STARTED')
         ENDIF

        IF(NOUTGC.EQ.-2) THEN
         OPEN(73,FILE='fort.73',ACCESS='DIRECT',RECL=NBYTE)
         IF(NBYTE.EQ.4) THEN
           DO I=1,8
             WRITE(73,REC=IGCP+I) RDES4(I)
             ENDDO
           IGCP=IGCP+8
           DO I=1,6
             WRITE(73,REC=IGCP+I) RID4(I)
             ENDDO
           IGCP=IGCP+6
           DO I=1,6
             WRITE(73,REC=IGCP+I) AID4(I)
```

```
            ENDDO
          IGCP=IGCP+6
          ENDIF
        IF(NBYTE.EQ.8) THEN
          DO I=1,4
            WRITE(73,REC=IGCP+I) RDES8(I)
            ENDDO
          IGCP=IGCP+4
          DO I=1,3
            WRITE(73,REC=IGCP+I) RID8(I)
            ENDDO
          IGCP=IGCP+3
          DO I=1,3
            WRITE(73,REC=IGCP+I) AID8(I)
            ENDDO
          IGCP=IGCP+3
          ENDIF
        WRITE(73,REC=IGCP+1) NDSETSC
        WRITE(73,REC=IGCP+2) NP
        WRITE(73,REC=IGCP+3) DT*NSPOOLGC
        WRITE(73,REC=IGCP+4) NSPOOLGC
        WRITE(73,REC=IGCP+5) 1
        IGCP=IGCP+5
        CLOSE(73)                    ! DO THIS TO FLUSH THE WRITE BUFFER
        OPEN(73,FILE='fort.73',ACCESS='DIRECT',RECL=NBYTE)
        ENDIF
      IF(NOUTGC.EQ.-1) THEN
        OPEN(73,FILE='fort.73')
        WRITE(73,3220) RUNDES,RUNID,AGRID
        WRITE(73,3645) NDSETSC,NP,DTDP*NSPOOLGC,NSPOOLGC,1
        IGCP=2
        ENDIF
      IF(NOUTGC.EQ.1) THEN
        OPEN(73,FILE='fort.73')
        DO I=1,IGCP
          READ(73,1050)
          ENDDO
        ENDIF
      IF(NOUTGC.EQ.2)
     &          OPEN(73,FILE='fort.73',ACCESS='DIRECT',RECL=NBYTE)

C...
C....GO TO THE PROPER PLACE IN THE GLOBAL WIND STRESS OUTPUT FILE
C...
      READ(IHOT,REC=IHOTSTP+1) IGWP
      READ(IHOT,REC=IHOTSTP+2) NSCOUGW
      IHOTSTP=IHOTSTP+2
      IF(NOUTGW.GT.0) WRITE(16,1055) IGWP,NSCOUGW
 1055 FORMAT(//,1X,I6,' LINES OR RECORDS WRITTEN IN THE GLOBAL ',
     &           'WIND STRESS FILE BY THE TIME OF THE HOT START',
     &          /,8X,'SPOOL COUNTER =',I6)
      IF(NOUTGW.LT.0) THEN
        IGWP=0
        NSCOUGW=0
        IF((NTCYSGW.LT.ITHS).AND.(NSPOOLGW.GT.0)) THEN
          NTCYSGW=NTCYSGW+((ITHS-NTCYSGW)/NSPOOLGW)*NSPOOLGW
          IF(NTCYSGW.LT.ITHS) NTCYSGW=NTCYSGW+NSPOOLGW
          NDSETSW=(NTCYFGW-NTCYSGW)/NSPOOLGW
          ENDIF
        WRITE(16,1056)
 1056   FORMAT(//,' A NEW GLOBAL WIND FILE WILL BE STARTED')
        ENDIF

      IF(NOUTGW.EQ.-2) THEN
        OPEN(74,FILE='fort.74',ACCESS='DIRECT',RECL=NBYTE)
        IF(NBYTE.EQ.4) THEN
          DO I=1,8
            WRITE(74,REC=IGWP+I) RDES4(I)
            ENDDO
          IGWP=IGWP+8
```

ADCIRC part 2

```
            DO I=1,6
              WRITE(74,REC=IGWP+I) RID4(I)
              ENDDO
            IGWP=IGWP+6
            DO I=1,6
              WRITE(74,REC=IGWP+I) AID4(I)
              ENDDO
            IGWP=IGWP+6
            ENDIF
          IF(NBYTE.EQ.8) THEN
            DO I=1,4
              WRITE(74,REC=IGWP+I) RDES8(I)
              ENDDO
            IGWP=IGWP+4
            DO I=1,3
              WRITE(74,REC=IGWP+I) RID8(I)
              ENDDO
            IGWP=IGWP+3
            DO I=1,3
              WRITE(74,REC=IGWP+I) AID8(I)
              ENDDO
            IGWP=IGWP+3
            ENDIF
          WRITE(74,REC=IGWP+1) NDSETSW
          WRITE(74,REC=IGWP+2) NP
          WRITE(74,REC=IGWP+3) DT*NSPOOLGW
          WRITE(74,REC=IGWP+4) NSPOOLGW
          WRITE(74,REC=IGWP+5) 2
          IGWP=IGWP+5
          CLOSE(74)                    ! DO THIS TO FLUSH THE WRITE BUFFER
          OPEN(74,FILE='fort.74',ACCESS='DIRECT',RECL=NBYTE)
          ENDIF
        IF(NOUTGW.EQ.-1) THEN
          OPEN(74,FILE='fort.74')
          WRITE(74,3220) RUNDES,RUNID,AGRID
          WRITE(74,3645) NDSETSW,NP,DTDP*NSPOOLGW,NSPOOLGW,2
          IGWP=2
          ENDIF
        IF(NOUTGW.EQ.1) THEN
          OPEN(74,FILE='fort.74')
          DO I=1,IGWP
            READ(74,1050)
            ENDDO
          ENDIF
        IF(NOUTGW.EQ.2)
     &               OPEN(74,FILE='fort.74',ACCESS='DIRECT',RECL=NBYTE)

C...
C......HOT START INFORMATION FOR HARMONIC ANALYSIS
C...
        IF(IHARIND.EQ.1) THEN
          IHABEG=ITHAS+NHAINC
C...
C........IF HARMONIC ANALYSIS HAS NOT BEGUN, COLD START THE HARMONIC ANALYSIS
C...
          IF(ITHS.LT.IHABEG) THEN
            ICHA=0
            CALL HACOLDS
            IF(NHASE.EQ.1) CALL HACOLDSES(NSTAE)
            IF(NHASV.EQ.1) CALL HACOLDSVS(NSTAV)
            IF(NHAGE.EQ.1) CALL HACOLDSEG(NP)
            IF(NHAGV.EQ.1) CALL HACOLDSVG(NP)

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
            DO I=1,NP
              ELAV(I)=0.D0
              XVELAV(I)=0.D0
```

```
              YVELAV(I)=0.D0
              ELVA(I)=0.D0
              XVELVA(I)=0.D0
              YVELVA(I)=0.D0
              END DO
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

            ENDIF

C...
C........IF HARMONIC ANALYSIS HAS ALREADY BEGUN, READ IN HOT START
C........HARMONIC ANALYSIS, MEAN AND SQUARE INFO
C...
          IF(ITHS.GT.ITHAS) THEN
            IHOTSTP=IHOTSTP+1
            READ(IHOT,REC=IHOTSTP) ICHA
            ENDIF
          IF(ITHS.GE.IHABEG) THEN
            CALL HAHOTS(NSTAE,NSTAV,NP,NHASE,NHASV,NHAGE,NHAGV,NSCREEN,
     &                                                    IHOTSTP,IHOT)
            IF(NHASE.EQ.1) CALL HAHOTSES(NSTAE,IHOTSTP,IHOT)
            IF(NHASV.EQ.1) CALL HAHOTSVS(NSTAV,IHOTSTP,IHOT)
            IF(NHAGE.EQ.1) CALL HAHOTSEG(NP,IHOTSTP,IHOT)
            IF(NHAGV.EQ.1) CALL HAHOTSVG(NP,IHOTSTP,IHOT)
            ENDIF

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...Read in Means and Squares
CHARMV...
          IF((FMV.NE.0.).AND.(ITHS.GT.ITMV)) THEN
            IHOTSTP=IHOTSTP+1
            READ(IHOT,REC=IHOTSTP) NTSTEPS
            IF(NHAGE.EQ.1) THEN
              DO I=1,NP
                READ(IHOT,REC=IHOTSTP+1) ELAV(I)
                READ(IHOT,REC=IHOTSTP+2) ELVA(I)
                IHOTSTP=IHOTSTP+2
                ENDDO
              ENDIF
            IF(NHAGV.EQ.1) THEN
              DO I=1,NP
                READ(IHOT,REC=IHOTSTP+1) XVELAV(I)
                READ(IHOT,REC=IHOTSTP+2) YVELAV(I)
                READ(IHOT,REC=IHOTSTP+3) XVELVA(I)
                READ(IHOT,REC=IHOTSTP+4) YVELVA(I)
                IHOTSTP=IHOTSTP+4
                ENDDO
              ENDIF
            ENDIF
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

          ENDIF


C3DVS..3D HOT START NOT IMPLIMENTED
C3DVS..UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D VS MODE.
C3DVS..COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D DSS MODE.
C3DVS..
c     ! CALL VSSTUP(DT,STATIM,NBYTE,RUNDES,RUNID,AGRID,NT)
C3DVS..
C3DVS..END OF 3D VS STATEMENTS (MORE FOLLOW BELOW)
C3DVS..

C3DDSS.3D HOT START NOT IMPLIMENTED
```

```
C3DDSS.UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE
C3DDSS.COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c      ! CALL DSSSTUP(DT,STATIM,NBYTE,RUNDES,RUNID,AGRID,NT)
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

C...
C...END HOT START SECTION
C...
        CLOSE(IHOT)
        ENDIF

C...
C...DETERMINE THE NUMBER OF POTENTIALLY ACTIVE ELEMENTS ATTACHED TO EACH NODE
C...
      DO I=1,NP
        MJU(I)=0
        NODECODE(I)=NNODECODE(I)
        END DO

      DO IE=1,NE
        NM1=NM(IE,1)
        NM2=NM(IE,2)
        NM3=NM(IE,3)
        MJU(NM1)=MJU(NM1)+ABS(NNODECODE(NM1))
        MJU(NM2)=MJU(NM2)+ABS(NNODECODE(NM2))
        MJU(NM3)=MJU(NM3)+ABS(NNODECODE(NM3))
        END DO

C...
C...***************** SET FLAGS AND COEFFICIENTS USED IN TIME STEPPING ******************
C...

C...NONLINEAR FLAGS

      IF(NOLIBF.EQ.0) THEN
        IFNLBF=0
        IFLINBF=1
        IFHYBF=0
        ENDIF
      IF(NOLIBF.EQ.1) THEN
        IFNLBF=1
        IFLINBF=0
        IFHYBF=0
        ENDIF
      IF(NOLIBF.EQ.2) THEN
        IFNLBF=0
        IFLINBF=0
        IFHYBF=1
        ENDIF
      IF(NOLIFA.EQ.0) THEN
        IFNLFA=0
        ELSE
        IFNLFA=1
        ENDIF
      IF(NOLICA.EQ.0) THEN
        IFNLCT=0
        ELSE
        IFNLCT=1
        ENDIF
      IF(NOLICAT.EQ.0) THEN
        IFNLCAT=0
        ELSE
        IFNLCAT=1
        ENDIF

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
```

```
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...SET WIND MULTIPLIER FLAG
CWIND...
      IFWIND=1
      IF(IM.EQ.1) IFWIND=0
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

C...CONSTANT COEFFICIENTS

      TT0L=((1.0+0.5*DT*TAU0)/DT)/DT
      GA00=G*A00
      TT0R=((0.5*TAU0*DT-1.0)/DT)/DT
      GC00=G*C00
      TADVODT=IFNLCAT/DT
      GB00A00=G*(B00+A00)
      GFAO2=G*IFNLFA/2.D0
      GO3=G/3.D0
      DTO2=DT/2.D0
      DT2=DT*2.D0
      GDTO2=G*DT/2.D0
      SADVDTO3=IFNLCT*DT/3.D0

C...
C...************************ BEGIN TIME STEPPING ************************
C...
      WRITE(16,1112)
      WRITE(16,17931)
      IF(NSCREEN.EQ.1) WRITE(6,1112)
      IF(NSCREEN.EQ.1) WRITE(6,17931)
17931 FORMAT(//,1X,'LIMITED RUNTIME INFORMATION SECTION ',//)

      NCCHANGE=1
      NWET=1

      DO 100 IT=ITHS+1,NT

C...
C...ALL CALCULATIONS ARE MADE FOR TIME WHICH INCLUDES THE REFTIM.
C...
        TIME=IT*DTDP + (STATIM - REFTIM)*86400.D0
C...
C....COMPUTE TIME OUTPUT IS TAGGED WITH.  THIS DOES NOT INCLUDE REFTIM.
C...
        TIMEOUT=IT*DTDP + STATIM*86400.D0

C...
C...SHIFT THE DEPTH AVERAGED VELOCITIES, BOTTOM STRESS, WIND STRESS, SURFACE PRESSURE AN
C...TIDAL POTENTIALS TO PREVIOUS TIME STEP.
C...ZERO OUT THE NEW FORCING TERMS, LOAD VECTORS (QW - GWCE, QU,QV - MOM) AND RESPONSES
C...
        DO I=1,NP
          UU1(I)=UU2(I)
          VV1(I)=VV2(I)
          QW(I)=0.D0
          QU(I)=0.D0
          QV(I)=0.D0
          IF(IM.EQ.10) THEN
            QB(I)=0.D0
            QA(I)=0.D0
            ENDIF

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
          UV1=SQRT(UU1(I)*UU1(I)+VV1(I)*VV1(I))
          IF(NNODECODE(I).NE.1) UV1=VELMIN
          HH1=DP(I)+IFNLFA*ETA2(I)
```

```
          TK(I)=FRIC(I)*(IFLINBF + (UV1/HH1)*(IFNLBF + IFHYBF*
     &                    (1+(HBREAK/HH1)**FTHETA)**(FGAMMA/FTHETA)))
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
          WSX1(I)=WSX2(I)
          WSX2(I)=0.D0
          WSY1(I)=WSY2(I)
          WSY2(I)=0.D0
          PR1(I)=PR2(I)
          PR2(I)=0.D0
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
          TIP1(I)=TIP2(I)
          TIP2(I)=0.D0
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

          END DO

C...
C...SHIFT THE SPECIFIED NORMAL FLOW BOUNDARY CONDITION TO PREVIOUS TIME STEPS.
C...ZERO OUT THE NEW SPECIFIED NORMAL FLOW BOUNDARY CONDITION
C...
        DO I=1,NVEL
          QN0(I)=QN1(I)
          QN1(I)=QN2(I)
          QN2(I)=0.D0
          END DO

C...
C...RECOMPUTE THE GWCE SYSTEM MATRIX AT THE FIRST TIME STEP OR IF ANY WETTING OR DRYING
C...OCCURRED IN THE PREVIOUS TIME STEP.
C...

        IF(NCCHANGE.EQ.1) THEN
          NCCHANGE=0
          EP=0.0
          IF(NSCREEN.EQ.1) WRITE(6,3806)
          WRITE(16,3806)
3806      FORMAT(/,1X,'RE-SETTING GWCE SYSTEM MATRIX',/)

CSOLIT...
CSOLIT...UNCOMMENT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLIT...COMMENT OUT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLIT...OR THE DIAGNOL MATRIX SOLVER.
CSOLIT...
          DO I=1,NP
            DO J=1,NEIMAX
              COEF(I,J)=0.0
              END DO
            END DO

          DO IE=1,NE
            NMI1=NM(IE,1)
            NMI2=NM(IE,2)
            NMI3=NM(IE,3)
            NMJ1=NMI1
```

```
NMJ2=NMI2
NMJ3=NMI3
NC1=ABS(NNODECODE(NMI1))
NC2=ABS(NNODECODE(NMI2))
NC3=ABS(NNODECODE(NMI3))
NCELE=NC1*NC2*NC3

SFACPP=(SFAC(NMI1)+SFAC(NMI2)+SFAC(NMI3))/3.

AREAIE=AREAS(IE)
FDX1 = (Y(NMI2)-Y(NMI3))*SFACPP
FDX2 = (Y(NMI3)-Y(NMI1))*SFACPP
FDX3 = (Y(NMI1)-Y(NMI2))*SFACPP
FDY1 = X(NMI3)-X(NMI2)
FDY2 = X(NMI1)-X(NMI3)
FDY3 = X(NMI2)-X(NMI1)
FDX1OA=FDX1/AREAIE              !dphi1/dx
FDY1OA=FDY1/AREAIE              !dphi1/dy
FDX2OA=FDX2/AREAIE              !dphi2/dx
FDY2OA=FDY2/AREAIE              !dphi2/dy
FDX3OA=FDX3/AREAIE              !dphi3/dx
FDY3OA=FDY3/AREAIE              !dphi3/dy
AH=(DP(NMI1)+DP(NMI2)+DP(NMI3))/3.
DXX11=FDX1OA*FDX1               !<2*(dphi1/dx)*(dphi1/dx)>
DYY11=FDY1OA*FDY1               !<2*(dphi1/dy)*(dphi1/dy)>
DXXYY11=DXX11+DYY11
DXYH11=AH*DXXYY11
DXX12=FDX1OA*FDX2               !<2*(dphi1/dx)*(dphi2/dx)>
DYY12=FDY1OA*FDY2               !<2*(dphi1/dy)*(dphi2/dy)>
DXXYY12=DXX12+DYY12
DXYH12=AH*DXXYY12
DXX13=FDX1OA*FDX3               !<2*(dphi1/dx)*(dphi3/dx)>
DYY13=FDY1OA*FDY3               !<2*(dphi1/dy)*(dphi3/dy)>
DXXYY13=DXX13+DYY13
DXYH13=AH*DXXYY13
DXYH21=DXYH12
DXX22=FDX2OA*FDX2               !<2*(dphi2/dx)*(dphi2/dx)>
DYY22=FDY2OA*FDY2               !<2*(dphi2/dy)*(dphi2/dy)>
DXXYY22=DXX22+DYY22
DXYH22=AH*DXXYY22
DXX23=FDX2OA*FDX3               !<2*(dphi2/dx)*(dphi3/dx)>
DYY23=FDY2OA*FDY3               !<2*(dphi2/dy)*(dphi3/dy)>
DXXYY23=DXX23+DYY23
DXYH23=AH*DXXYY23
DXYH31=DXYH13
DXYH32=DXYH23
DXX33=FDX3OA*FDX3               !<2*(dphi3/dx)*(dphi3/dx)>
DYY33=FDY3OA*FDY3               !<2*(dphi3/dy)*(dphi3/dy)>
DXXYY33=DXX33+DYY33
DXYH33=AH*DXXYY33
AO6=AREAIE/6.
AO12=AREAIE/12.
FDDD=(1+ILUMP)*AO6              !2*<phi*phj> diagonal terms
FDDOD=(1-ILUMP)*AO12            !2*<phi*phj> off diagonal terms

DO JN=2,NEIMAX
   IF(NEITAB(NMI1,JN).EQ.NMJ2) J12=JN
   IF(NEITAB(NMI1,JN).EQ.NMJ3) J13=JN
   IF(NEITAB(NMI2,JN).EQ.NMJ1) J21=JN
   IF(NEITAB(NMI2,JN).EQ.NMJ3) J23=JN
   IF(NEITAB(NMI3,JN).EQ.NMJ1) J31=JN
   IF(NEITAB(NMI3,JN).EQ.NMJ2) J32=JN
   END DO

COEF(NMI1,1)  =COEF(NMI1,1)  +(TT0L*FDDD +GA00*DXYH11)*NCELE
COEF(NMI1,J12)=COEF(NMI1,J12)+(TT0L*FDDOD+GA00*DXYH12)*NCELE
COEF(NMI1,J13)=COEF(NMI1,J13)+(TT0L*FDDOD+GA00*DXYH13)*NCELE
COEF(NMI2,J21)=COEF(NMI2,J21)+(TT0L*FDDOD+GA00*DXYH21)*NCELE
COEF(NMI2,1)  =COEF(NMI2,1)  +(TT0L*FDDD +GA00*DXYH22)*NCELE
COEF(NMI2,J23)=COEF(NMI2,J23)+(TT0L*FDDOD+GA00*DXYH23)*NCELE
```

```
            COEF(NMI3,J31)=COEF(NMI3,J31)+(TT0L*FDDOD+GA00*DXYH31)*NCELE
            COEF(NMI3,J32)=COEF(NMI3,J32)+(TT0L*FDDOD+GA00*DXYH32)*NCELE
            COEF(NMI3,1)  =COEF(NMI3,1)  +(TT0L*FDDD +GA00*DXYH33)*NCELE

            END DO

        !MODIFY THE MATRIX "COEF" BY IMPOSING THE ELEVATION SPECIFIED BOUNDARY
        !CONDITIONS WHILE MAINTAINING THE SYMMETRY OF THE SYSTEM

           DO I=1,NP
             EP=EP+COEF(I,1)**2
             END DO
           EP=SQRT(EP/NP)
           DO I=1,NETA
             COEF(NBD(I),1)=EP
             DO J=2,NNEIGH(NBD(I))
               COEF(NBD(I),J)=0.0
               END DO
             END DO
           DO I=1,NETA
             DO J=2,NNEIGH(NBD(I))
               DO IJ=2,NNEIGH(NEITAB(NBD(I),J))
                 IF(NBD(I).EQ.NEITAB(NEITAB(NBD(I),J),IJ)) THEN
                   OBCCOEF(I,J-1)=COEF(NEITAB(NBD(I),J),IJ)
                   COEF(NEITAB(NBD(I),J),IJ)=0.0
                   ENDIF
                 END DO
               END DO
             END DO

        !CHECK THAT ALL THE DIAGONAL ELEMENTS IN "COEF" ARE > 0.

           DO I=1,NP
             IF(COEF(I,1).EQ.0.) COEF(I,1)=EP
             IF(COEF(I,1).LT.0.) THEN
               IF(NSCREEN.EQ.1) WRITE(6,1019) I,COEF(I,1)
               WRITE(16,1019) I,COEF(I,1)
 1019          FORMAT(/,1X,'!!!!!!!!!  WARNING !!!!!!!!',
     &              /,1X,'THE DIAGONAL TERM IN THE EQUATION FOR NODE ',I10,
     &               '= ',E15.6,' AND IS < 0',/)
               END IF
             END DO
CSOLIT...
CSOLIT...END OF ITERATIVE SOLVER SECTION (MORE FOLLOW BELOW)
CSOLIT...

CSOLDIR...
CSOLDIR...UNCOMMENT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLDIR...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIR...OR THE DIAGNOL MATRIX SOLVER.
CSOLDIR...
c          DO I=1,NP
c            DO J=1,NEIMAX
c              COEF(I,J)=0.0
c              END DO
c            END DO
c
c          DO IE=1,NE
c            NMI1=NM(IE,1)
c            NMI2=NM(IE,2)
c            NMI3=NM(IE,3)
c            NMJ1=NMI1
c            NMJ2=NMI2
c            NMJ3=NMI3
c            NC1=ABS(NNODECODE(NMI1))
c            NC2=ABS(NNODECODE(NMI2))
c            NC3=ABS(NNODECODE(NMI3))
c            NCELE=NC1*NC2*NC3
c
c            SFACPP=(SFAC(NMI1)+SFAC(NMI2)+SFAC(NMI3))/3.
```

```
c
c             AREAIE=AREAS(IE)
c             FDX1 = (Y(NMI2)-Y(NMI3))*SFACPP
c             FDX2 = (Y(NMI3)-Y(NMI1))*SFACPP
c             FDX3 = (Y(NMI1)-Y(NMI2))*SFACPP
c             FDY1 = X(NMI3)-X(NMI2)
c             FDY2 = X(NMI1)-X(NMI3)
c             FDY3 = X(NMI2)-X(NMI1)
c             FDX1OA=FDX1/AREAIE                !dphi1/dx
c             FDY1OA=FDY1/AREAIE                !dphi1/dy
c             FDX2OA=FDX2/AREAIE                !dphi2/dx
c             FDY2OA=FDY2/AREAIE                !dphi2/dy
c             FDX3OA=FDX3/AREAIE                !dphi3/dx
c             FDY3OA=FDY3/AREAIE                !dphi3/dy
c             AH=(DP(NMI1)+DP(NMI2)+DP(NMI3))/3.
c             DXX11=FDX1OA*FDX1                 !<2*(dphi1/dx)*(dphi1/dx)>
c             DYY11=FDY1OA*FDY1                 !<2*(dphi1/dy)*(dphi1/dy)>
c             DXXYY11=DXX11+DYY11
c             DXYH11=AH*DXXYY11
c             DXX12=FDX1OA*FDX2                 !<2*(dphi1/dx)*(dphi2/dx)>
c             DYY12=FDY1OA*FDY2                 !<2*(dphi1/dy)*(dphi2/dy)>
c             DXXYY12=DXX12+DYY12
c             DXYH12=AH*DXXYY12
c             DXX13=FDX1OA*FDX3                 !<2*(dphi1/dx)*(dphi3/dx)>
c             DYY13=FDY1OA*FDY3                 !<2*(dphi1/dy)*(dphi3/dy)>
c             DXXYY13=DXX13+DYY13
c             DXYH13=AH*DXXYY13
c             DXYH21=DXYH12
c             DXX22=FDX2OA*FDX2                 !<2*(dphi2/dx)*(dphi2/dx)>
c             DYY22=FDY2OA*FDY2                 !<2*(dphi2/dy)*(dphi2/dy)>
c             DXXYY22=DXX22+DYY22
c             DXYH22=AH*DXXYY22
c             DXX23=FDX2OA*FDX3                 !<2*(dphi2/dx)*(dphi3/dx)>
c             DYY23=FDY2OA*FDY3                 !<2*(dphi2/dy)*(dphi3/dy)>
c             DXXYY23=DXX23+DYY23
c             DXYH23=AH*DXXYY23
c             DXYH31=DXYH13
c             DXYH32=DXYH23
c             DXX33=FDX3OA*FDX3                 !<2*(dphi3/dx)*(dphi3/dx)>
c             DYY33=FDY3OA*FDY3                 !<2*(dphi3/dy)*(dphi3/dy)>
c             DXXYY33=DXX33+DYY33
c             DXYH33=AH*DXXYY33
c             AO6=AREAIE/6.
c             AO12=AREAIE/12.
c             FDDD=(1+ILUMP)*AO6                !2*<phi*phj> diagonal terms
c             FDDOD=(1-ILUMP)*AO12              !2*<phi*phj> off diagonal terms
c
c             DO JN=2,NEIMAX
c                IF(NEITAB(NMI1,JN).EQ.NMJ2) J12=JN
c                IF(NEITAB(NMI1,JN).EQ.NMJ3) J13=JN
c                IF(NEITAB(NMI2,JN).EQ.NMJ1) J21=JN
c                IF(NEITAB(NMI2,JN).EQ.NMJ3) J23=JN
c                IF(NEITAB(NMI3,JN).EQ.NMJ1) J31=JN
c                IF(NEITAB(NMI3,JN).EQ.NMJ2) J32=JN
c                END DO
c
c             COEF(NMI1,1)  =COEF(NMI1,1)  +(TT0L*FDDD +GA00*DXYH11)*NCELE
c             COEF(NMI1,J12)=COEF(NMI1,J12)+(TT0L*FDDOD+GA00*DXYH12)*NCELE
c             COEF(NMI1,J13)=COEF(NMI1,J13)+(TT0L*FDDOD+GA00*DXYH13)*NCELE
c             COEF(NMI2,J21)=COEF(NMI2,J21)+(TT0L*FDDOD+GA00*DXYH21)*NCELE
c             COEF(NMI2,1)  =COEF(NMI2,1)  +(TT0L*FDDD +GA00*DXYH22)*NCELE
c             COEF(NMI2,J23)=COEF(NMI2,J23)+(TT0L*FDDOD+GA00*DXYH23)*NCELE
c             COEF(NMI3,J31)=COEF(NMI3,J31)+(TT0L*FDDOD+GA00*DXYH31)*NCELE
c             COEF(NMI3,J32)=COEF(NMI3,J32)+(TT0L*FDDOD+GA00*DXYH32)*NCELE
c             COEF(NMI3,1)  =COEF(NMI3,1)  +(TT0L*FDDD +GA00*DXYH33)*NCELE
c
c             END DO
c
c       !MODIFY THE MATRIX "COEF" BY IMPOSING THE ELEVATION SPECIFIED BOUNDARY
c       !CONDITIONS WHILE MAINTAINING THE SYMMETRY OF THE SYSTEM
```

```
c
c          DO I=1,NP
c            EP=EP+COEF(I,1)**2
c            END DO
c          EP=SQRT(EP/NP)
c          DO I=1,NETA
c            COEF(NBD(I),1)=EP
c            DO J=2,NNEIGH(NBD(I))
c              COEF(NBD(I),J)=0.0
c              END DO
c            END DO
c          DO I=1,NETA
c            DO J=2,NNEIGH(NBD(I))
c              DO IJ=2,NNEIGH(NEITAB(NBD(I),J))
c                IF(NBD(I).EQ.NEITAB(NEITAB(NBD(I),J),IJ)) THEN
c                  OBCCOEF(I,J-1)=COEF(NEITAB(NBD(I),J),IJ)
c                  COEF(NEITAB(NBD(I),J),IJ)=0.0
c                  ENDIF
c                END DO
c              END DO
c            END DO
c
c       !LOAD THE GLOBAL WAVE EQUATION MATRIX INTO THE
c       !BANDED STORAGE MATRIX "ABD" , FACTORIZE THE MATRIX "ABD"
c       !AND COMPUTE THE RECRIPROCAL OF THE CONDITION NUMBER
c       !USING LINPACK.
c
c          DO I=1,NP
c            DO J=1,3*NBW+1
c              ABD(J,I)=0.0
c              END DO
c            END DO
c          DO I=1,NP
c          IF(COEF(I,1).EQ.0.) COEF(I,1)=EP
c            DO J=1,NNEIGH(I)
c              LRC=MBW+I-NEITAB(I,J)
c              ABD(LRC,NEITAB(I,J))=COEF(I,J)
c              END DO
c            END DO
c          CALL SGBCO(ABD,3*MNBW+1,NP,NBW,NBW,IPV,COND,ZX)
c          IF(NSCREEN.EQ.1) WRITE(6,1985) COND
c          WRITE(16,1985) COND
c1985      FORMAT(1X,'THE RECIPROCAL OF THE CONDITION NUMBER',
c     &              ' OF THE GWCE MATRIX =',E15.5,/)
c          IF(COND.LT.1.0E-5) THEN
c            IF(NSCREEN.EQ.1) WRITE(6,9988)
c            WRITE(16,9988)
c9988        FORMAT(/,1X,'!!!!!!!!!!  WARNING  !!!!!!!!!',/,
c     &               1X,'THE GWCE MATRIX IS POORLY CONDITIONED',/,
c     &               1X,'THE SOLUTION MAY NOT BE ACCURATE',/)
c            ENDIF
c
CSOLDIR...
CSOLDIR...END OF DIRECT SOLVER SECTION (MORE FOLLOW BELOW)
CSOLDIR...

CSOLDIA...
CSOLDIA...UNCOMMENT THE FOLLOWING LINES TO USE THE DIAGONAL MATRIX SOLVER
CSOLDIA...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIA...OR THE DIRECT MATRIX SOLVER.
CSOLDIA...
c          DO I=1,NP
c            COEF(I)=0.0
c            END DO
c
c          DO IE=1,NE
c            NMI1=NM(IE,1)
c            NMI2=NM(IE,2)
c            NMI3=NM(IE,3)
c            NC1=ABS(NNODECODE(NMI1))
```

```
c             NC2=ABS(NNODECODE(NMI2))
c             NC3=ABS(NNODECODE(NMI3))
c             NCELE=NC1*NC2*NC3
c             AREAIE=AREAS(IE)
c             AO6=AREAIE/6.
c             AO12=AREAIE/12.
c             FDDD=(1+ILUMP)*AO6              !2*<phi*phj> diagonal terms
c             COEF(NMI1)=COEF(NMI1)+TT0L*FDDD*NCELE
c             COEF(NMI2)=COEF(NMI2)+TT0L*FDDD*NCELE
c             COEF(NMI3)=COEF(NMI3)+TT0L*FDDD*NCELE
c             END DO
c
c       !MODIFY THE MATRIX "COEF" BY IMPOSING THE ELEVATION BOUNDARY CONDITIONS
c
c           DO I=1,NETA
c             COEF(NBD(I))=1.0
c             END DO
c
c           DO I=1,NP
c             IF(COEF(I).EQ.0.) COEF(I)=1.0
c             END DO
CSOLDIA...
CSOLDIA...END OF DIAGONAL SOLVER SECTION (MORE FOLLOW BELOW)
CSOLDIA...

            ENDIF           !END OF GWCE MATRIX SETUP


C...
C...DEFINE RAMP FUNCTION FOR BOUNDARY ELEVATION FORCING, WIND AND PRESSURE
C....FORCING AND TIDAL POTENTIAL FORCING
C...
          RAMP=1.0
          IF(NRAMP.EQ.1) RAMP=TANH((2.D0*IT*DTDP/86400.D0)/DRAMP)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...UPDATE THE WIND STRESS AND SURFACE PRESSURE AND READ IN NEW VALUES FROM
CWIND...UNIT 22.  APPLY RAMP FUNCTION.
CWIND...
          IF(NWS.EQ.1) THEN
            DO I=1,NP
              READ(22,*) NHG,WSX2(NHG),WSY2(NHG),PR2(NHG)
              WSX2(NHG)=RAMP*WSX2(NHG)
              WSY2(NHG)=RAMP*WSY2(NHG)
              PR2(NHG)=RAMP*PR2(NHG)
              END DO
            ENDIF

          IF(NWS.EQ.2) THEN
            IF(TIME.GT.WTIME2) THEN
              WTIME1=WTIME2
              WTIME2=WTIME2+WTIMINC
              DO I=1,NP
                WVNX1(I)=WVNX2(I)
                WVNY1(I)=WVNY2(I)
                PRN1(I)=PRN2(I)
                READ(22,*) NHG,WVNX2(NHG),WVNY2(NHG),PRN2(NHG)
                END DO
              ENDIF
            WTRATIO=(TIME-WTIME1)/WTIMINC
            DO I=1,NP
              WSX2(I)=RAMP*(WVNX1(I)+WTRATIO*(WVNX2(I)-WVNX1(I)))
              WSY2(I)=RAMP*(WVNY1(I)+WTRATIO*(WVNY2(I)-WVNY1(I)))
              PR2(I)=RAMP*(PRN1(I)+WTRATIO*(PRN2(I)-PRN1(I)))
              END DO
            ENDIF

          IF(NWS.EQ.3) THEN
```

```
      IF(TIME.GT.WTIME2) THEN
        WTIME1=WTIME2
        WTIME2=WTIME2+WTIMINC
        DO I=1,NP
          WVNX1(I)=WVNX2(I)
          WVNY1(I)=WVNY2(I)
          END DO
        CALL NWS3GET(X,Y,SLAM,SFEA,WVNX2,WVNY2,IWTIME,IWYR,WTIMED,
     &               NP,NWLON,NWLAT,WLATMAX,WLONMIN,WLATINC,WLONINC,ICS)
        ENDIF
      WTRATIO=(TIME-WTIME1)/WTIMINC
      DO I=1,NP
        WINDX = WVNX1(I) + WTRATIO*(WVNX2(I)-WVNX1(I))
        WINDY = WVNY1(I) + WTRATIO*(WVNY2(I)-WVNY1(I))
        WINDMAG = SQRT(WINDX*WINDX+WINDY*WINDY)
        WDRAGCO = 0.001*(0.75+0.067*WINDMAG)
        IF(WDRAGCO.GT.0.003) WDRAGCO=0.003
        WSX2(I) = RAMP*0.001293*WDRAGCO*WINDX*WINDMAG
        WSY2(I) = RAMP*0.001293*WDRAGCO*WINDY*WINDMAG
        END DO
      ENDIF

    IF(NWS.EQ.4) THEN
      IF(TIME.GT.WTIME2) THEN
        WTIME1=WTIME2
        WTIME2=WTIME2+WTIMINC
        DO I=1,NP
          WVNX1(I)=WVNX2(I)
          WVNY1(I)=WVNY2(I)
          PRN1(I)=PRN2(I)
          END DO
        CALL NWS4GET(WVNX2,WVNY2,PRN2,NP,1000.,G)
        ENDIF
      WTRATIO=(TIME-WTIME1)/WTIMINC
      DO I=1,NP
        WINDX = WVNX1(I) + WTRATIO*(WVNX2(I)-WVNX1(I))
        WINDY = WVNY1(I) + WTRATIO*(WVNY2(I)-WVNY1(I))
        WINDMAG = SQRT(WINDX*WINDX+WINDY*WINDY)
        WDRAGCO = 0.001*(0.75+0.067*WINDMAG)
        IF(WDRAGCO.GT.0.003) WDRAGCO=0.003
        WSX2(I) = RAMP*0.001293*WDRAGCO*WINDX*WINDMAG
        WSY2(I) = RAMP*0.001293*WDRAGCO*WINDY*WINDMAG
        PR2(I)=RAMP*(PRN1(I)+WTRATIO*(PRN2(I)-PRN1(I)))
        END DO
      ENDIF

    IF(NWS.EQ.10) THEN
      IF(TIME.GT.WTIME2) THEN
        WTIME1=WTIME2
        WTIME2=WTIME2+WTIMINC
        DO I=1,NP
          WVNX1(I)=WVNX2(I)
          WVNY1(I)=WVNY2(I)
          PRN1(I)=PRN2(I)
          END DO
        NWSGGWI=NWSGGWI+1
        CALL NWS10GET(NWSGGWI,SLAM,SFEA,WVNX2,WVNY2,PRN2,NP,1000.,G,
     &                NWLON,NWLAT)
        ENDIF
      WTRATIO=(TIME-WTIME1)/WTIMINC
      DO I=1,NP
        WINDX = WVNX1(I) + WTRATIO*(WVNX2(I)-WVNX1(I))
        WINDY = WVNY1(I) + WTRATIO*(WVNY2(I)-WVNY1(I))
        WINDMAG = SQRT(WINDX*WINDX+WINDY*WINDY)
        WDRAGCO = 0.001*(0.75+0.067*WINDMAG)
        IF(WDRAGCO.GT.0.003) WDRAGCO=0.003
        WSX2(I) = RAMP*0.001293*WDRAGCO*WINDX*WINDMAG
        WSY2(I) = RAMP*0.001293*WDRAGCO*WINDY*WINDMAG
        PR2(I)=RAMP*(PRN1(I)+WTRATIO*(PRN2(I)-PRN1(I)))
        END DO
```

```
          ENDIF
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...THE EARTH TIDE POTENTIAL REDUCTION FACTOR, ETRF(J) HAS BEEN
CTIP....INCORPORATED INTO THIS CALCULATION.
CTIP...
        IF(NTIP.GE.1) THEN
          DO J=1,NTIF
            IF(PERT(J).EQ.0.) THEN
              NCYC=0
              ELSE
              NCYC=INT(TIME/PERT(J))
              ENDIF
            ARGT=AMIGT(J)*(TIME-NCYC*PERT(J))+FACET(J)
            TPMUL=RAMP*ETRF(J)*TPK(J)*FFT(J)
            SALTMUL=RAMP*FFT(J)
            NA=NINT(0.00014/AMIGT(J))
            IF(NA.EQ.1) THEN                          !SEMI-DIURNAL SPECIES
              DO I=1,NP
                ARGTP=ARGT+2.*SLAM(I)
                ARGSALT=ARGT-SALTPHA(J,I)
                CCSFEA=COS(SFEA(I))
                CCSFEA=CCSFEA*CCSFEA
                TIP2(I)=TIP2(I)+TPMUL*CCSFEA*COS(ARGTP)
     &                          +SALTMUL*SALTAMP(J,I)*COS(ARGSALT)
              END DO
              ENDIF
            IF(NA.EQ.2) THEN                          !DIURNAL SPECIES
              DO I=1,NP
                ARGTP=ARGT+SLAM(I)
                ARGSALT=ARGT-SALTPHA(J,I)
                S2SFEA=SIN(2.*SFEA(I))
                TIP2(I)=TIP2(I)+TPMUL*S2SFEA*COS(ARGTP)
     &                          +SALTMUL*SALTAMP(J,I)*COS(ARGSALT)
              END DO
              ENDIF
            END DO
          ENDIF
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C...
C...COMPUTE SPECIFIED NORMAL FLOW BOUNDARY CONDITION
C...
        IF(NFLUXF.EQ.1) THEN
          DO J=1,NFFR
            IF(FPER(J).EQ.0.) THEN
              NCYC=0.
              ELSE
              NCYC=INT(TIME/FPER(J))
              ENDIF
            ARGJ=FAMIG(J)*(TIME-NCYC*FPER(J))+FFACE(J)
            RFF=FFF(J)*RAMP
            DO I=1,NVEL
              ARG=ARGJ-QNPH(J,I)
              QN2(I)=QN2(I)+QNAM(J,I)*RFF*COS(ARG)
              END DO
            END DO
          IF(NFFR.EQ.0) THEN
            IF(TIME.GT.QTIME2) THEN
              QTIME1=QTIME2
              QTIME2=QTIME2+FTIMINC
              DO J=1,NVEL
                IF((LBCODEI(J).EQ.2).OR.(LBCODEI(J).EQ.12)
```

```
     &                                .OR.(LBCODEI(J).EQ.22)) THEN
                  QNIN1(J)=QNIN2(J)
                  READ(20,*) QNIN2(J)
                  ENDIF
                END DO
              ENDIF
            QTRATIO=(TIME-QTIME1)/FTIMINC
            DO I=1,NVEL
              QN2(I)=RAMP*(QNIN1(I)+QTRATIO*(QNIN2(I)-QNIN1(I)))
              END DO
            ENDIF
          ENDIF

C...
C...COMPUTE DISCHARGE CONTRIBUTION FROM RADIATION BOUNDARY CONDITION
C...
        IF(NFLUXRBC.EQ.1) THEN
          DO J=1,NVEL
            IF(LBCODEI(J).EQ.30) THEN
              NNBB=NBV(J)
              HH1=DP(NNBB)+IFNLFA*ETA2(NNBB)
              UN1=UU1(NNBB)*CSII(J)+VV1(NNBB)*SIII(J)
              QN1(J)=HH1*UN1
              ENDIF
            END DO
          ENDIF

C...
C...COMPUTE SUPERCRITICAL OUTWARD NORMAL FLOW OVER SPECIFIED
C....EXTERNAL BARRIER BOUNDARY NODES
C...
        IF(NFLUXB.EQ.1) THEN
          DO I=1,NVEL
            IF((LBCODEI(I).EQ.3).OR.(LBCODEI(I).EQ.13).OR.
     &         (LBCODEI(I).EQ.23)) THEN
              NNBB=NBV(I)
              RBARWL=2.D0*(ETA2(NNBB)-BARLANHT(I))/3.D0
              IF(RBARWL.GT.0.0D0) THEN
                QN2(I)=-RAMP*BARLANCFSP(I)*RBARWL*(RBARWL*G)**0.5D0
              ELSE
                QN2(I)=0.0D0
              ENDIF
            ENDIF
          END DO
        ENDIF

C...
C...COMPUTE INWARD/OUTWARD NORMAL FLOW OVER SPECIFIED
C....INTERNAL BARRIER BOUNDARY NODES
C...
        IF(NFLUXIB.EQ.1) THEN
          DO I=1,NVEL
            IF((LBCODEI(I).EQ.4).OR.(LBCODEI(I).EQ.24)) THEN
              NNBB1=NBV(I)       ! GLOBAL NODE NUMBER ON THIS SIDE OF BARRIER
              NNBB2=IBCONN(I)    ! GLOBAL NODE NUMBER ON OPPOSITE SIDE OF BARRIER
              IF(IBSTART.EQ.0)THEN
                RBARWL1AVG(I)=ETA2(NNBB1)-BARINHT(I)
                RBARWL2AVG(I)=ETA2(NNBB2)-BARINHT(I)
                IBSTART=1
              ENDIF
              RBARWL1=RBARWL1AVG(I)
              RBARWL2=RBARWL2AVG(I)
              RBARWL1F=2.0D0*RBARWL1/3.0D0
              RBARWL2F=2.0D0*RBARWL2/3.0D0
              QN2(I)=0.0
              IF((RBARWL1.LT.0.0).AND.(RBARWL2.LT.0.0)) THEN
C..............WATER LEVEL ON BOTH SIDES OF BARRIER BELOW BARRIER -> CASE 1
                QN2(I)=0.0
                GOTO 1034
              ENDIF
```

```
              IF(RBARWL1.EQ.RBARWL2) THEN
C.............WATER LEVEL EQUAL ON BOTH SIDES OF BARRIER -> CASE 2
                QN2(I)=0.0
                GOTO 1034
              ENDIF
              IF((RBARWL1.GT.RBARWL2).AND.(RBARWL1.GT.0)) THEN
C.............WATER LEVEL GREATER ON THIS SIDE OF THE BARRIER AND IS SUCH
C.............THAT OVERTOPPING IS OCCURING
                IF(RBARWL2.GT.RBARWL1F) THEN
C...............OUTWARD SUBCRITICAL FLOW -> CASE 3
                  QN2(I)=-RAMP*ABS(NODECODE(NNBB1))*ABS(NODECODE(NNBB2))
     &                    *BARINCFSB(I)*RBARWL2*
     &                    (2*G*(RBARWL1-RBARWL2))**0.5D0
                ELSE
C...............OUTWARD SUPERCRITICAL FLOW -> CASE 4
                  QN2(I)=-RAMP*ABS(NODECODE(NNBB1))*ABS(NODECODE(NNBB2))
     &                    *BARINCFSP(I)*RBARWL1F*(RBARWL1F*G)**0.5D0
                ENDIF
                GOTO 1034
              ENDIF
              IF((RBARWL2.GT.RBARWL1).AND.(RBARWL2.GT.0)) THEN
C.............WATER LEVEL LOWER ON THIS SIDE OF BARRIER AND IS SUCH
C.............THAT OVERTOPPING IS OCCURING
                IF(RBARWL1.GT.RBARWL2F) THEN
C...............INWARD SUBCRITICAL FLOW -> CASE 5
                  QN2(I)=RAMP*ABS(NODECODE(NNBB1))*ABS(NODECODE(NNBB2))
     &                    *BARINCFSB(I)*RBARWL1*
     &                    (2*G*(RBARWL2-RBARWL1))**0.5D0
                ELSE
C...............INWARD SUPERCRITICAL FLOW -> CASE 6
                  QN2(I)=RAMP*ABS(NODECODE(NNBB1))*ABS(NODECODE(NNBB2))
     &                    *BARINCFSP(I)*RBARWL2F*(RBARWL2F*G)**0.5D0
                ENDIF
                GOTO 1034
              ENDIF
1034          CONTINUE
            ENDIF
          END DO
        ENDIF

C...
C...COMPLETE THE THE LOAD VECTOR QW FOR THE GWCE ELEMENT BY ELEMENT
C...BY FORMING TEMPORARY VECTORS AND THEN ASSEMBLING AT THE END.
C...THE FOLLOWING ASSEMBLY LOOPS HAVE ALL BEEN UNROLLED TO OPTIMIZE
C...VECORIZATION
C...
        DO 1037 IE=1,NE
C...
C...SET NODAL VALUES FOR EACH ELEMENT
C...
          NM1=NM(IE,1)
          NM2=NM(IE,2)
          NM3=NM(IE,3)
          NC1=ABS(NNODECODE(NM1))
          NC2=ABS(NNODECODE(NM2))
          NC3=ABS(NNODECODE(NM3))
          NCELE=NC1*NC2*NC3
          E0N1=ETA1(NM1)
          E0N2=ETA1(NM2)
          E0N3=ETA1(NM3)
          E1N1=ETA2(NM1)
          E1N2=ETA2(NM2)
          E1N3=ETA2(NM3)
          E1N1SQ=E1N1*E1N1
          E1N2SQ=E1N2*E1N2
          E1N3SQ=E1N3*E1N3
          ESN1=ETAS(NM1)
          ESN2=ETAS(NM2)
          ESN3=ETAS(NM3)
          U1N1=UU1(NM1)
```

```
            U1N2=UU1(NM2)
            U1N3=UU1(NM3)
            V1N1=VV1(NM1)
            V1N2=VV1(NM2)
            V1N3=VV1(NM3)
            HH1N1=DP(NM1)+IFNLFA*E1N1
            HH1N2=DP(NM2)+IFNLFA*E1N2
            HH1N3=DP(NM3)+IFNLFA*E1N3
            HHU1N1=HH1N1*U1N1
            HHU1N2=HH1N2*U1N2
            HHU1N3=HH1N3*U1N3
            HHV1N1=HH1N1*V1N1
            HHV1N2=HH1N2*V1N2
            HHV1N3=HH1N3*V1N3
            SFACPP=(SFAC(NM1)+SFAC(NM2)+SFAC(NM3))/3.

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
            WSXN1=WSX1(NM1)
            WSXN2=WSX1(NM2)
            WSXN3=WSX1(NM3)
            WSYN1=WSY1(NM1)
            WSYN2=WSY1(NM2)
            WSYN3=WSY1(NM3)
            PR1N1=PR1(NM1)
            PR1N2=PR1(NM2)
            PR1N3=PR1(NM3)
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
            TIPN1=TIP1(NM1)
            TIPN2=TIP1(NM2)
            TIPN3=TIP1(NM3)
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
            TK1N1=TK(NM1)
            TK1N2=TK(NM2)
            TK1N3=TK(NM3)
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c           BSXN1=BSX1(NM1)
c           BSXN2=BSX1(NM2)
c           BSXN3=BSX1(NM3)
c           BSYN1=BSY1(NM1)
c           BSYN2=BSY1(NM2)
c           BSYN3=BSY1(NM3)
c           DVV1N1=DVV1(NM1)
c           DVV1N2=DVV1(NM2)
c           DVV1N3=DVV1(NM3)
c           DUV1N1=DUV1(NM1)
c           DUV1N2=DUV1(NM2)
```

```
c          DUV1N3=DUV1(NM3)
c          DUU1N1=DUU1(NM1)
c          DUU1N2=DUU1(NM2)
c          DUU1N3=DUU1(NM3)
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...
C...COMPUTE ELEMENT AVERAGED QUANTITIES
C...
           AH=(DP(NM1)+DP(NM2)+DP(NM3))/3.
           GHPP=GO3*(HH1N1+HH1N2+HH1N3)
           UPP=(U1N1+U1N2+U1N3)/3.
           VPP=(V1N1+V1N2+V1N3)/3.
           UHPP3=HHU1N1+HHU1N2+HHU1N3
           VHPP3=HHV1N1+HHV1N2+HHV1N3
           UHPP=UHPP3/3.
           VHPP=VHPP3/3.
           EVMPP=(EVM(NM1)+EVM(NM2)+EVM(NM3))/3.
           CORIFPP=(CORIF(NM1)+CORIF(NM2)+CORIF(NM3))/3.

C...
C...COMPUTE ELEMENTAL COEFFICIENTS
C...
           AREAIE=AREAS(IE)
           FDX1 = (Y(NM2)-Y(NM3))*SFACPP
           FDX2 = (Y(NM3)-Y(NM1))*SFACPP
           FDX3 = (Y(NM1)-Y(NM2))*SFACPP
           FDY1 = X(NM3)-X(NM2)
           FDY2 = X(NM1)-X(NM3)
           FDY3 = X(NM2)-X(NM1)
           FDX1OA=FDX1/AREAIE              !dphi1/dx
           FDY1OA=FDY1/AREAIE              !dphi1/dy
           FDX2OA=FDX2/AREAIE              !dphi2/dx
           FDY2OA=FDY2/AREAIE              !dphi2/dy
           FDX3OA=FDX3/AREAIE              !dphi3/dx
           FDY3OA=FDY3/AREAIE              !dphi3/dy

           DDX1=FDX1/3.                    !<2*(dphi1/dx)*phij> j=1,2,3
           DDY1=FDY1/3.                    !<2*(dphi1/dy)*phij> j=1,2,3
           DXX11=FDX1OA*FDX1               !<2*(dphi1/dx)*(dphi1/dx)>
           DYY11=FDY1OA*FDY1               !<2*(dphi1/dy)*(dphi1/dy)>
           DXY11=FDX1OA*FDY1               !<2*(dphi1/dx)*(dphi1/dy)>
           DXXYY11=DXX11+DYY11
           DXYH11=AH*DXXYY11
           DXX12=FDX1OA*FDX2               !<2*(dphi1/dx)*(dphi2/dx)>
           DYY12=FDY1OA*FDY2               !<2*(dphi1/dy)*(dphi2/dy)>
           DXY12=FDX1OA*FDY2               !<2*(dphi1/dx)*(dphi2/dy)>
           DXXYY12=DXX12+DYY12
           DXYH12=AH*DXXYY12
           DXX13=FDX1OA*FDX3               !<2*(dphi1/dx)*(dphi3/dx)>
           DYY13=FDY1OA*FDY3               !<2*(dphi1/dy)*(dphi3/dy)>
           DXY13=FDX1OA*FDY3               !<2*(dphi1/dx)*(dphi3/dy)>
           DXXYY13=DXX13+DYY13
           DXYH13=AH*DXXYY13

           DDX2=FDX2/3.                    !<2*(dphi2/dx)*phij> j=1,2,3
           DDY2=FDY2/3.                    !<2*(dphi2/dy)*phij> j=1,2,3
           DXX21=DXX12                     !<2*(dphi2/dx)*(dphi1/dx)>
           DYY21=DYY12                     !<2*(dphi2/dy)*(dphi1/dy)>
           DXY21=FDX2OA*FDY1               !<2*(dphi2/dx)*(dphi1/dy)>
           DXXYY21=DXXYY12
           DXYH21=DXYH12
           DXX22=FDX2OA*FDX2               !<2*(dphi2/dx)*(dphi2/dx)>
           DYY22=FDY2OA*FDY2               !<2*(dphi2/dy)*(dphi2/dy)>
           DXY22=FDX2OA*FDY2               !<2*(dphi2/dx)*(dphi2/dy)>
           DXXYY22=DXX22+DYY22
           DXYH22=AH*DXXYY22
           DXX23=FDX2OA*FDX3               !<2*(dphi2/dx)*(dphi3/dx)>
```

```
            DYY23=FDY2OA*FDY3                !<2*(dphi2/dy)*(dphi3/dy)>
            DXY23=FDX2OA*FDY3                !<2*(dphi2/dx)*(dphi3/dy)>
            DXXYY23=DXX23+DYY23
            DXYH23=AH*DXXYY23

            DDX3=FDX3/3.                     !<2*(dphi3/dx)*phij> j=1,2,3
            DDY3=FDY3/3.                     !<2*(dphi3/dy)*phij> j=1,2,3
            DXX31=DXX13                      !<2*(dphi3/dx)*(dphi1/dx)>
            DYY31=DYY13                      !<2*(dphi3/dy)*(dphi1/dy)>
            DXY31=FDX3OA*FDY1                !<2*(dphi3/dx)*(dphi1/dy)>
            DXXYY31=DXXYY13
            DXYH31=DXYH13
            DXX32=DXX23                      !<2*(dphi3/dx)*(dphi2/dx)>
            DYY32=DYY23                      !<2*(dphi3/dy)*(dphi2/dy)>
            DXY32=FDX3OA*FDY2                !<2*(dphi3/dx)*(dphi2/dy)>
            DXXYY32=DXXYY23
            DXYH32=DXYH23
            DXX33=FDX3OA*FDX3                !<2*(dphi3/dx)*(dphi3/dx)>
            DYY33=FDY3OA*FDY3                !<2*(dphi3/dy)*(dphi3/dy)>
            DXY33=FDX3OA*FDY3                !<2*(dphi3/dx)*(dphi3/dy)>
            DXXYY33=DXX33+DYY33
            DXYH33=AH*DXXYY33

            AO6=AREAIE/6.
            AO12=AREAIE/12.
            FDDD=(1+ILUMP)*AO6               !2*<phi*phj> diagonal terms
            FDDOD=(1-ILUMP)*AO12             !2*<phi*phj> off diagonal terms

C...
C...COMPUTE THE RHS GWCE FORCING AND PUT INTO QTEMA VECTOR FOR NODE NM1
C...
            QTEMA1=
C...TRANSIENT AND TAU0 TERMS FROM LHS
     &          -(FDDD*ESN1+FDDOD*ESN2+FDDOD*ESN3)*TT0R

C...FREE SURFACE TERMS FROM LHS (TIME LEVEL K-1)
     &          -(DXYH11*E0N1+DXYH12*E0N2+DXYH13*E0N3)*GC00

C...FREE SURFACE TERMS FROM LHS (TIME LEVEL K)
     &          -(DXYH11*E1N1+DXYH12*E1N2+DXYH13*E1N3)*GB00A00

C...BOTTOM FRICTION

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
     &          +(TAU0-TK1N1)*(HHU1N1*DDX1+HHV1N1*DDY1)
     &          +(TAU0-TK1N2)*(HHU1N2*DDX1+HHV1N2*DDY1)
     &          +(TAU0-TK1N3)*(HHU1N3*DDX1+HHV1N3*DDY1)
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c    &          +TAU0*(UHPP3*DDX1+VHPP3*DDY1)
c    &          -(BSXN1+BSXN2+BSXN3)*DDX1-(BSYN1+BSYN2+BSYN3)*DDY1
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...CORIOLIS FORCE
     &          +CORIFPP*(VHPP3*DDX1-UHPP3*DDY1)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
```

```
CWIND...
     &          +(WSXN1+WSXN2+WSXN3)*DDX1+(WSYN1+WSYN2+WSYN3)*DDY1
     &          -GHPP*(PR1N1*DXXYY11+PR1N2*DXXYY12+PR1N3*DXXYY13)
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
     &          +GHPP*(TIPN1*DXXYY11+TIPN2*DXXYY12+TIPN3*DXXYY13)
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C...LATERAL VISCOSITY TERM
     &          -EVMPP*(DXXYY11*ESN1+DXXYY12*ESN2+DXXYY13*ESN3)
C...FINITE AMPLITUDE
     &          -GFAO2*(E1N1SQ*DXXYY11+E1N2SQ*DXXYY12+E1N3SQ*DXXYY13)
C...ADVECTIVE TERMS
     &          -IFNLCT*(UHPP*(U1N1*DXX11+U1N2*DXX21+U1N3*DXX31
     &                        +V1N1*DXY11+V1N2*DXY21+V1N3*DXY31)
     &                  +VHPP*(U1N1*DXY11+U1N2*DXY12+U1N3*DXY13
     &                        +V1N1*DYY11+V1N2*DYY21+V1N3*DYY31))

C...ADVECTIVE TERMS (TIME DERIVATIVE PORTION IN GWCE) WHICH MUST
C....BE BUNDLED IN WITH THE FINITE AMPLITUDE TERMS IN ORDER TO GET GOOD MASS
C....CONSERVATION WHEN THE ADVECTIVE TERMS ARE SHUT DOWN
     &          +TADVODT*(UPP*DDX1+VPP*DDY1)*(ESN1+ESN2+ESN3)

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
C3D....DISPERSION TERMS
c    &          -IFNLCT*(DUU1N1*DXX11+DUU1N2*DXX12+DUU1N3*DXX13
c    &                  +DUV1N1*DXY11+DUV1N2*DXY12+DUV1N3*DXY13
c    &                  +DUV1N1*DXY11+DUV1N2*DXY21+DUV1N3*DXY31
c    &                  +DVV1N1*DYY11+DVV1N2*DYY12+DVV1N3*DYY13)
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...
C...COMPUTE THE RHS GWCE FORCING AND PUT INTO QTEMA VECTOR FOR NODE NM2
C...
          QTEMA2=
C...TRANSIENT AND TAU0 TERMS FROM LHS
     &          -(FDDOD*ESN1+FDDD*ESN2+FDDOD*ESN3)*TT0R

C...FREE SURFACE TERMS FROM LHS (TIME LEVEL K-1)
     &          -(DXYH12*E0N1+DXYH22*E0N2+DXYH23*E0N3)*GC00

C...FREE SURFACE TERMS FROM LHS (TIME LEVEL K)
     &          -(DXYH12*E1N1+DXYH22*E1N2+DXYH23*E1N3)*GB00A00

C...BOTTOM FRICTION

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
     &          +(TAU0-TK1N1)*(HHU1N1*DDX2+HHV1N1*DDY2)
     &          +(TAU0-TK1N2)*(HHU1N2*DDX2+HHV1N2*DDY2)
     &          +(TAU0-TK1N3)*(HHU1N3*DDX2+HHV1N3*DDY2)
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...
```

```
C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c     &          +TAU0*(UHPP3*DDX2+VHPP3*DDY2)
c     &          -(BSXN1+BSXN2+BSXN3)*DDX2-(BSYN1+BSYN2+BSYN3)*DDY2
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...CORIOLIS FORCE
      &          +CORIFPP*(VHPP3*DDX2-UHPP3*DDY2)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
      &          +(WSXN1+WSXN2+WSXN3)*DDX2+(WSYN1+WSYN2+WSYN3)*DDY2
      &          -GHPP*(PR1N1*DXXYY21+PR1N2*DXXYY22+PR1N3*DXXYY23)
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
      &          +GHPP*(TIPN1*DXXYY21+TIPN2*DXXYY22+TIPN3*DXXYY23)
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C...LATERAL VISCOSITY TERM
      &          -EVMPP*(DXXYY12*ESN1+DXXYY22*ESN2+DXXYY23*ESN3)
C...FINITE AMPLITUDE
      &          -GFAO2*(E1N1SQ*DXXYY21+E1N2SQ*DXXYY22+E1N3SQ*DXXYY23)
C...ADVECTIVE TERMS
      &          -IFNLCT*(UHPP*(U1N1*DXX12+U1N2*DXX22+U1N3*DXX32
      &                        +V1N1*DXY12+V1N2*DXY22+V1N3*DXY32)
      &                  +VHPP*(U1N1*DXY21+U1N2*DXY22+U1N3*DXY23
      &                        +V1N1*DYY12+V1N2*DYY22+V1N3*DYY32))
C...ADVECTIVE TERMS (TIME DERIVATIVE PORTION IN GWCE) WHICH MUST
C....BE BUNDLED IN WITH THE FINITE AMPLITUDE TERMS IN ORDER TO GET GOOD MASS
C....CONSERVATION WHEN THE ADVECTIVE TERMS ARE SHUT DOWN
      &          +TADVODT*(UPP*DDX2+VPP*DDY2)*(ESN1+ESN2+ESN3)

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
C3D....DISPERSION TERMS
c     &          -IFNLCT*(DUU1N1*DXX12+DUU1N2*DXX22+DUU1N3*DXX23
c     &                  +DUV1N1*DXY21+DUV1N2*DXY22+DUV1N3*DXY23
c     &                  +DUV1N1*DXY12+DUV1N2*DXY22+DUV1N3*DXY32
c     &                  +DVV1N1*DYY12+DVV1N2*DYY22+DVV1N3*DYY23)
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...
C...COMPUTE THE RHS GWCE FORCING AND PUT INTO QTEMA VECTOR FOR NODE NM1
C...
          QTEMA3=
C...TRANSIENT AND TAU0 TERMS FROM LHS
      &          -(FDDOD*ESN1+FDDOD*ESN2+FDDD*ESN3)*TT0R

C...FREE SURFACE TERMS FROM LHS (TIME LEVEL K-1)
      &          -(DXYH13*E0N1+DXYH23*E0N2+DXYH33*E0N3)*GC00

C...FREE SURFACE TERMS FROM LHS (TIME LEVEL K)
```

```
     &          -(DXYH13*E1N1+DXYH23*E1N2+DXYH33*E1N3)*GB00A00

C...BOTTOM FRICTION

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
     &          +(TAU0-TK1N1)*(HHU1N1*DDX3+HHV1N1*DDY3)
     &          +(TAU0-TK1N2)*(HHU1N2*DDX3+HHV1N2*DDY3)
     &          +(TAU0-TK1N3)*(HHU1N3*DDX3+HHV1N3*DDY3)
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c    &          +TAU0*(UHPP3*DDX3+VHPP3*DDY3)
c    &          -(BSXN1+BSXN2+BSXN3)*DDX3-(BSYN1+BSYN2+BSYN3)*DDY3
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...CORIOLIS FORCE
     &          +CORIFPP*(VHPP3*DDX3-UHPP3*DDY3)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
     &          +(WSXN1+WSXN2+WSXN3)*DDX3+(WSYN1+WSYN2+WSYN3)*DDY3
     &          -GHPP*(PR1N1*DXXYY31+PR1N2*DXXYY32+PR1N3*DXXYY33)
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
     &          +GHPP*(TIPN1*DXXYY31+TIPN2*DXXYY32+TIPN3*DXXYY33)
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

C...LATERAL VISCOSITY TERM
     &          -EVMPP*(DXXYY13*ESN1+DXXYY23*ESN2+DXXYY33*ESN3)
C...FINITE AMPLITUDE
     &          -GFAO2*(E1N1SQ*DXXYY31+E1N2SQ*DXXYY32+E1N3SQ*DXXYY33)
C...ADVECTIVE TERMS
     &          -IFNLCT*(UHPP*(U1N1*DXX13+U1N2*DXX23+U1N3*DXX33
     &                        +V1N1*DXY13+V1N2*DXY23+V1N3*DXY33)
     &                  +VHPP*(U1N1*DXY31+U1N2*DXY32+U1N3*DXY33
     &                        +V1N1*DYY13+V1N2*DYY23+V1N3*DYY33))
C...ADVECTIVE TERMS (TIME DERIVATIVE PORTION IN GWCE) WHICH MUST
C....BE BUNDLED IN WITH THE FINITE AMPLITUDE TERMS IN ORDER TO GET GOOD MASS
C....CONSERVATION WHEN THE ADVECTIVE TERMS ARE SHUT DOWN
     &          +TADVODT*(UPP*DDX3+VPP*DDY3)*(ESN1+ESN2+ESN3)

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
C3D....DISPERSION TERMS
c    &          -IFNLCT*(DUU1N1*DXX13+DUU1N2*DXX23+DUU1N3*DXX33
c    &                  +DUV1N1*DXY31+DUV1N2*DXY32+DUV1N3*DXY33
c    &                  +DUV1N1*DXY13+DUV1N2*DXY23+DUV1N3*DXY33
c    &                  +DVV1N1*DYY13+DVV1N2*DYY23+DVV1N3*DYY33)
```

```
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...
          QTEMA(IE,1)=QTEMA1*NCELE
          QTEMA(IE,2)=QTEMA2*NCELE
          QTEMA(IE,3)=QTEMA3*NCELE
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CVEC....

CSCA...
CSCA...UNCOMMENT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER.
CSCA...COMMENT OUT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER.
CSCA...NOTE: THESE LINES FINALIZE THE ASSEMBLY PROCESS FOR QW
CSCA...       ON A SCALAR COMPUTER USING THE TEMPORARY VECTORS
CSCA...
c         QW(NM1)=QW(NM1)+QTEMA1*NCELE
c         QW(NM2)=QW(NM2)+QTEMA2*NCELE
c         QW(NM3)=QW(NM3)+QTEMA3*NCELE
CSCA...
CSCA...END OF SCALAR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CSCA....

1037      CONTINUE

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...NOTE: THESE LINES FINALIZE THE ASSEMBLY PROCESS FOR QW
CVEC...       ON A VECTOR COMPUTER USING THE TEMPORARY VECTORS
CVEC...
        DO IE=1,NE
          NM1=NM(IE,1)
          NM2=NM(IE,2)
          NM3=NM(IE,3)
          QW(NM1)=QW(NM1)+QTEMA(IE,1)
          QW(NM2)=QW(NM2)+QTEMA(IE,2)
          QW(NM3)=QW(NM3)+QTEMA(IE,3)
          END DO
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CVEC....

C...
C...SAVE THE ELEVATION AT THE PAST TIME STEP INTO ETA1 AND ZERO ETA2
C...
        DO I=1,NP
          ETA1(I)=ETA2(I)
          ETA2(I)=0.0
          END DO

C...
C...IMPOSE ELEVATION BOUNDARY CONDITIONS TO LOAD VECTOR QW(I)
C...NOTE; EP IS THE RMS OF ALL THE DIAGONAL MEMBERS IN THE GWCE.  IT IS USED TO SCALE
C...       THE DIAGONAL ELEMENT FOR THE ELEVATION SPECIFIED BOUNDARY NODES AND THEREFORE
C...       MUST ALSO BE USED TO SCALE THE RHS OF THE EQUATIONS
C...
        DO J=1,NBFR
          IF(PER(J).EQ.0.) THEN
            NCYC=0.
            ELSE
            NCYC=INT(TIME/PER(J))
            ENDIF
          ARGJ=AMIG(J)*(TIME-NCYC*PER(J))+FACE(J)
          RFF=FF(J)*RAMP
```

```
            DO I=1,NETA
              ARG=ARGJ-EFA(J,I)
              NBDI=NBD(I)
              ETA2(NBDI)=ETA2(NBDI)+EMO(J,I)*RFF*COS(ARG)
              END DO
            END DO

C...
C...IMPOSE NORMAL FLOW OR RADIATION BOUNDARY CONDITIONS ALONG FLOW BOUNDARY TO LOAD
C...VECTOR QW(I) NOTE, THESE VALUES ALL MUST BE MULTIPLIED BY 2 SINCE ALL ELEMENTAL
C...COEFFICIENTS HAVE BEEN.

          IF((NFLUXF.EQ.1).OR.(NFLUXB.EQ.1).OR.(NFLUXIB.EQ.1)
     &                                       .OR.(NFLUXRBC.EQ.1)) THEN
            NBDJ=NBV(1)
            IF(LBCODEI(1).LE.29) QFORCEJ=(QN2(1)-QN0(1))/DT2 + TAU0*QN1(1)
            IF(LBCODEI(1).EQ.30) THEN
              HH1=DP(NBDJ)+IFNLFA*ETA2(NBDJ)
              CELERITY=SQRT(G*HH1)
              QFORCEJ=-CELERITY*ETAS(NBDJ)/DT - TAU0*QN1(1)
              ENDIF
            DO J=2,NVEL
              NBDI=NBDJ
              NBDJ=NBV(J)
              QFORCEI=QFORCEJ
              IF(LBCODEI(J).LE.29) QFORCEJ=(QN2(J)-QN0(J))/DT2+TAU0*QN1(J)
              IF(LBCODEI(J).EQ.30) THEN
                HH1=DP(NBDJ)+IFNLFA*ETA2(NBDJ)
                CELERITY=SQRT(G*HH1)
                QFORCEJ=-CELERITY*ETAS(NBDJ)/DT - TAU0*QN1(J)
                ENDIF
              NCI=ABS(NNODECODE(NBDI))
              NCJ=ABS(NNODECODE(NBDJ))
              NCBND=NCI*NCJ
              BNDLEN2O3NC=NCBND*BNDLEN2O3(J-1)
              QW(NBDI)=QW(NBDI) + BNDLEN2O3NC*(QFORCEI+QFORCEJ/2.D0)
              QW(NBDJ)=QW(NBDJ) + BNDLEN2O3NC*(QFORCEJ+QFORCEI/2.D0)
              END DO
            ENDIF

CSOLDIA...
CSOLDIA...UNCOMMENT THE FOLLOWING LINES TO USE THE DIAGONAL MATRIX SOLVER
CSOLDIA...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIA...OR THE DIRECT MATRIX SOLVER.
CSOLDIA...
c         DO I=1,NETA
c           NBDI=NBD(I)
c           ETAS(NBDI)=ETA2(NBDI)-ETA1(NBDI)
c           QW(NBDI)=ETAS(NBDI)*ABS(NNODECODE(NBDI))
c           END DO
CSOLDIA...
CSOLDIA...END OF DIAGONAL SOLVER SECTION
CSOLDIA...

CSOLIT...
CSOLIT...UNCOMMENT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLIT...COMMENT OUT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLIT...OR THE DIAGNOL MATRIX SOLVER.
CSOLIT...
          DO I=1,NETA
            NBDI=NBD(I)
            ETAS(NBDI)=ETA2(NBDI)-ETA1(NBDI)
            QW(NBDI)=ETAS(NBDI)*ABS(NNODECODE(NBDI))*EP
            DO J=2,NNEIGH(NBDI)
              QW(NEITAB(NBDI,J))=QW(NEITAB(NBDI,J))
     &                           -ETAS(NBDI)*OBCCOEF(I,J-1)
              END DO
            END DO
CSOLIT...
CSOLIT...END OF ITERATIVE SOLVER SECTION FOR GWCE
```

```
CSOLIT...

CSOLDIR...
CSOLDIR...UNCOMMENT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLDIR...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIR...OR THE DIAGNOL MATRIX SOLVER.
CSOLDIR...
c       DO I=1,NETA
c         NBDI=NBD(I)
c         ETAS(NBDI)=ETA2(NBDI)-ETA1(NBDI)
c         QW(NBDI)=ETAS(NBDI)*ABS(NNODECODE(NBDI))*EP
c         DO J=2,NNEIGH(NBDI)
c           QW(NEITAB(NBDI,J))=QW(NEITAB(NBDI,J))
c    &                         -ETAS(NBDI)*OBCCOEF(I,J-1)
c           END DO
c         END DO
CSOLDIR...
CSOLDIR...END OF DIRECT SOLVER SECTION
CSOLDIR...


C...
C...SOLVE GWCE FOR ELEVATION AT NEW TIME LEVEL
C...

CSOLDIR...
CSOLDIR...UNCOMMENT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLDIR...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIR...OR THE DIAGNOL MATRIX SOLVER.
CSOLDIR...
c       CALL SGBSL(ABD,3*MNBW+1,NP,NBW,NBW,IPV,QW,0)
c       DO I=1,NP
c         ETAS(I)=QW(I)
c         ETA2(I)=ABS(NNODECODE(I))*ETAS(I)+ETA1(I)       !COMPUTE NEW ELEVATIONS
c         END DO
c       NUMITR=0
CSOLDIR...
CSOLDIR...END OF DIRECT SOLVER SECTION
CSOLDIR...

CSOLIT...
CSOLIT...UNCOMMENT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLIT...COMMENT OUT THE FOLLOWING LINES TO USE THE DIRECT MATRIX SOLVER
CSOLIT...OR THE DIAGNOL MATRIX SOLVER.
CSOLIT...
        IPARM(1)=ITMAX
        IF(ITITER.EQ.1) CALL JCG(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        IF(ITITER.EQ.2) CALL JSI(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        IF(ITITER.EQ.3) CALL SOR(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        IF(ITITER.EQ.4) CALL SSORCG(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        IF(ITITER.EQ.5) CALL SSORSI(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        IF(ITITER.EQ.6) CALL RSCG(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        IF(ITITER.EQ.7) CALL RSSI(NP,MNP,MNEI,NEITAB,COEF,QW,ETAS,
     &                           IWKSP,NW,WKSP,IPARM,RPARM,IER)
        NUMITR=IPARM(1)
        DO I=1,NP
          ETA2(I)=ABS(NNODECODE(I))*ETAS(I)+ETA1(I)       !COMPUTE NEW ELEVATIONS
          END DO
CSOLIT...
CSOLIT...END OF ITERATIVE SOLVER SECTION FOR GWCE
CSOLIT...

CSOLDIA...
CSOLDIA...UNCOMMENT THE FOLLOWING LINES TO USE THE DIAGONAL MATRIX SOLVER
```

```
CSOLDIA...COMMENT OUT THE FOLLOWING LINES TO USE THE ITERATIVE MATRIX SOLVER
CSOLDIA...OR THE DIRECT MATRIX SOLVER.
CSOLDIA...
c       DO I=1,NP
c         ETAS(I)=QW(I)/COEF(I)
c         ETA2(I)=ABS(NNODECODE(I))*ETAS(I)+ETA1(I)       !COMPUTE NEW ELEVATIONS
c         END DO
c       NUMITR=0
CSOLDIA...
CSOLDIA...END OF DIAGONAL SOLVER SECTION
CSOLDIA...

CWETDRY...
CWETDRY...THE FOLLOWING LINES ARE FOR WETTING AND DRYING
CWETDRY...
CWETDRY...DRYING CRITERIA
CWETDRY...
CWETDRY...NOTE:NODEREP IS THE NUMBER OF TIME STEPS SINCE A NODE LAST CHANGED STATE
CWETDRY...
CWETDRY...A FULLY WET NODE OR A NODE THAT IS ON THE WET/DRY INTERFACE CAN DRY FOR
CWETDRY...TWO REASONS.
CWETDRY...1.) IF THE DEPTH FALLS BELOW H0*(NODEREP/NODEWETRMP)
CWETDRY .....WHERE NODEWETRMP IS THE NUMBER OF TIME STEPS OVER WHICH THE MINIMUM
CWETDRY......DEPTH CRITERIA IS RAMPED UP TO H0.
CWETDRY......THE MINIMUM DEPTH IS RAMPED IN THIS WAY TO KEEP SMALL OSCILLATIONS IN
CWETDRY......WATER LEVEL AFTER A NODE WETS FROM CAUSING THE NODE TO IMMEDIATELY DRY.
CWETDRY...2.) IF ALL SURROUNDING NODES ARE EITHER DRY OR ON THE WET/DRY BOUNDARY.
CWETDRY......IN THIS CASE THE NODE IS DRIED DUE TO BECOMING LANDLOCKED.
CWETDRY...NOTE: IF NIBNODECODE(I)=1, A NODE WILL NOT BE ALLOWED TO DRY SINCE
CWETDRY......FLOW ACTIVELY COMING OVER AN INTERNAL BARRIER BOUNDARY
CWETDRY...
        IF(NOLIFA.EQ.2) THEN

      !DRY NODES ACCORDING TO UPDATED ELEVATIONS AND PREVIOUS NODECODE

          NDRY=0
          DO I=1,NP
            NODEREP(I)=NODEREP(I)+1
            IF(ABS(NNODECODE(I)).EQ.1) THEN
              HTOT=DP(I)+ETA2(I)
              H0TEST=H0*NODEREP(I)/NODEWETRMP
              IF(H0TEST.GT.H0) H0TEST=H0
              IF(HTOT.LE.H0TEST) THEN
                IF(HTOT.LE.0.00001) ETA2(I)=H0TEST-DP(I)
                IF(NIBNODECODE(I).EQ.0) THEN
                  NDRY=NDRY+1
                  NNODECODE(I)=0
                  NODEREP(I)=0
                  IF(NSCREEN.EQ.1) WRITE(6,9880) I,HTOT
                  WRITE(16,9880) I,HTOT
 9880             FORMAT(' !!! NODE ',I6,' DRIED, WATER DEPTH = ',F10.5)
                ENDIF
              ENDIF
            ENDIF
          END DO

      !IF ANY DRYING OCCURRED

          IF(NDRY.GT.0) THEN
            DO I=1,NP                                    !MAKE FIRST CUT AT DETERMINING
              IF(NNODECODE(I).NE.0) THEN                 !WET/DRY INTERFACE NODES.
                NNODEC=1                                 !ASSUME NODE IS WET. MAKE IT AN
                DO J=2,NNEIGH(I)                         !INTERFACE NODE IF IT IS CONNECTE
                  NEINOD=NEITAB(I,J)                     !TO A DRY NODE
                  IF(NNODECODE(NEINOD).EQ.0) NNODEC=-1
                  END DO
                IF(NNODEC.NE.NNODECODE(I)) THEN
                  IF(NNODEC.EQ.-1) THEN
                    IF(NSCREEN.EQ.1) WRITE(6,9881) I
                    WRITE(16,9881) I
```

```
9881              FORMAT(' !!! NODE ',I6,' BECAME AN INTERFACE NODE',
     &                                                ' FROM DRYING')
                  ENDIF
                IF(NNODEC.EQ.1) THEN
                  IF(NSCREEN.EQ.1) WRITE(6,9882) I
                  WRITE(16,9882) I
9882              FORMAT(' !!! NODE ',I6,' WETTED DURING THE DRYING ',
     &                   'PROCESS.  THIS MUST BE AN ERROR !!!')
                  ENDIF
                NNODECODE(I)=NNODEC
                NODEREP(I)=0
                ENDIF
              ENDIF
            END DO

          DO I=1,NP                                    !DRY WET/DRY INTERFACE NODES
            IF(NNODECODE(I).EQ.-1) THEN                !THAT ARE NOT CONNECTED TO AT
              IWET=0                                   !LEAST ONE NODE THAT HAS
              DO J=2,NNEIGH(I)                         !FUNCTIONING VELOCITY
                NEINOD=NEITAB(I,J)
                IF((NNODECODE(NEINOD).EQ.1).AND.(LBCODE(NEINOD).GE.0))
     &                                                   IWET=1
                IF((NNODECODE(NEINOD).EQ.-1).AND.(LBCODEWD.EQ.1))
     &                                                   IWET=1
                END DO
              IF(IWET.EQ.0) THEN
                IF(NSCREEN.EQ.1) WRITE(6,9883) I
                WRITE(16,9883) I
9883            FORMAT(' !!! NODE ',I6,' DRIED FROM LANDLOCKING')
                NNODECODE(I)=0
                NODEREP(I)=0
                DO J=2,NNEIGH(I)                        !MAKE ALL CONNECTED WET NODES
                  NEINOD=NEITAB(I,J)                    !BNDRY NODES) INTO INTERFACE N
                  IF(NNODECODE(NEINOD).EQ.1) THEN
                    IF(NSCREEN.EQ.1) WRITE(6,9884) NEINOD
                    WRITE(16,9884) NEINOD
9884                FORMAT(' !!! NODE ',I6,' BECAME AN INTERFACE ',
     &                                  'NODE FROM LANDLOCKING')
                    NNODECODE(NEINOD)=-1
                    NODEREP(NEINOD)=0
                    ENDIF
                  END DO
                ENDIF
              ENDIF
            END DO
          ENDIF

      !IF ANY NODES WETTED DURING THE PREVIOUS TIME STEP OR DRIED THIS TIME
      !STEP, TRANSFER NODE CODE AND RECOMPUTE HOW MANY POTENTIALLY ACTIVE
      !ELEMENTS EACH NODE IS CONNECTED TO FOR USE IN THE MOMENTUM EQUATION

        IF((NDRY.GT.0).OR.(NWET.GT.0)) THEN
          DO I=1,NP
            NODECODE(I)=NNODECODE(I)
            MJU(I)=0
            END DO
          DO IE=1,NE
            NM1=NM(IE,1)
            NM2=NM(IE,2)
            NM3=NM(IE,3)
            MJU(NM1)=MJU(NM1)+ABS(NNODECODE(NM1))
            MJU(NM2)=MJU(NM2)+ABS(NNODECODE(NM2))
            MJU(NM3)=MJU(NM3)+ABS(NNODECODE(NM3))
            END DO
          DO I=1,NP
            IF(MJU(I).EQ.0) MJU(I)=1
            END DO
          ENDIF

        ENDIF
```

```
CWETDRY...
CWETDRY...END WET/DRY SECTION
CWETDRY...

C...
C...UPDATE LOAD VECTOR OF DEPTH AVERAGED MOMENTUM EQUATION QU(I) AND QV(I)
C....NOTE: QU, QV AND AUV ARE ZEROED OUT AT THE TOP OF THE TIME STEPPING LOOP.
C...

C.....FIRST TREAT THE NON-LUMPED PART OF THE EQUATIONS.
C......THESE LOOPS HAVE BEEN UNROLLED TO OPTIMIZE VECTORIZATION

        DISPERX=0.
        DISPERY=0.

        DO 9999 IE=1,NE
C...
C...SET NODAL VALUES FOR EACH ELEMENT
C...
          NM1=NM(IE,1)
          NM2=NM(IE,2)
          NM3=NM(IE,3)
          U1N1=UU1(NM1)
          U1N2=UU1(NM2)
          U1N3=UU1(NM3)
          V1N1=VV1(NM1)
          V1N2=VV1(NM2)
          V1N3=VV1(NM3)

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....
c         DVV1N1=DVV1(NM1)
c         DVV1N2=DVV1(NM2)
c         DVV1N3=DVV1(NM3)
c         DUV1N1=DUV1(NM1)
c         DUV1N2=DUV1(NM2)
c         DUV1N3=DUV1(NM3)
c         DUU1N1=DUU1(NM1)
c         DUU1N2=DUU1(NM2)
c         DUU1N3=DUU1(NM3)
C3D....
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

          ESN1=ETAS(NM1)
          ESN2=ETAS(NM2)
          ESN3=ETAS(NM3)
          HH1N1=DP(NM1)+IFNLFA*ETA1(NM1)
          HH1N2=DP(NM2)+IFNLFA*ETA1(NM2)
          HH1N3=DP(NM3)+IFNLFA*ETA1(NM3)
          SFACPP=(SFAC(NM1)+SFAC(NM2)+SFAC(NM3))/3.

          AREAIE=AREAS(IE)
          FDX1=(Y(NM2)-Y(NM3))*SFACPP     !b1
          FDX2=(Y(NM3)-Y(NM1))*SFACPP     !b2
          FDX3=(Y(NM1)-Y(NM2))*SFACPP     !b3
          FDY1=X(NM3)-X(NM2)              !a1
          FDY2=X(NM1)-X(NM3)              !a2
          FDY3=X(NM2)-X(NM1)              !a3
          FDX1OA=FDX1/AREAIE              !dphi1/dx
          FDY1OA=FDY1/AREAIE              !dphi1/dy
          FDX2OA=FDX2/AREAIE              !dphi2/dx
          FDY2OA=FDY2/AREAIE              !dphi2/dy
          FDX3OA=FDX3/AREAIE              !dphi3/dx
          FDY3OA=FDY3/AREAIE              !dphi3/dy

          DDX1=FDX1/3.                    !<2*(dphi1/dx)*phij> j=1,2,3
          DDY1=FDY1/3.                    !<2*(dphi1/dy)*phij> j=1,2,3
```

```
            DXX11=FDX1OA*FDX1                !<2*(dphi1/dx)*(dphi1/dx)>
            DYY11=FDY1OA*FDY1                !<2*(dphi1/dy)*(dphi1/dy)>
            DXXYY11=DXX11+DYY11
            DXX12=FDX1OA*FDX2                !<2*(dphi1/dx)*(dphi2/dx)>
            DYY12=FDY1OA*FDY2                !<2*(dphi1/dy)*(dphi2/dy)>
            DXXYY12=DXX12+DYY12
            DXX13=FDX1OA*FDX3                !<2*(dphi1/dx)*(dphi3/dx)>
            DYY13=FDY1OA*FDY3                !<2*(dphi1/dy)*(dphi3/dy)>
            DXXYY13=DXX13+DYY13

            DDX2=FDX2/3.                     !<2*(dphi2/dx)*phij> j=1,2,3
            DDY2=FDY2/3.                     !<2*(dphi2/dy)*phij> j=1,2,3
            DXXYY21=DXXYY12
            DXX22=FDX2OA*FDX2                !<2*(dphi2/dx)*(dphi2/dx)>
            DYY22=FDY2OA*FDY2                !<2*(dphi2/dy)*(dphi2/dy)>
            DXXYY22=DXX22+DYY22
            DXX23=FDX2OA*FDX3                !<2*(dphi2/dx)*(dphi3/dx)>
            DYY23=FDY2OA*FDY3                !<2*(dphi2/dy)*(dphi3/dy)>
            DXXYY23=DXX23+DYY23

            DDX3=FDX3/3.                     !<2*(dphi3/dx)*phij> j=1,2,3
            DDY3=FDY3/3.                     !<2*(dphi3/dy)*phij> j=1,2,3
            DXXYY31=DXXYY13
            DXXYY32=DXXYY23
            DXX33=FDX3OA*FDX3                !<2*(dphi3/dx)*(dphi3/dx)>
            DYY33=FDY3OA*FDY3                !<2*(dphi3/dy)*(dphi3/dy)>
            DXXYY33=DXX33+DYY33

            FIIN=AREAIE/3.D0                 !2*<phi*phj> lumped

C...
C...COMPUTE NODAL VALUES FOR CERTAIN ELEMENTAL COEFFICIENTS
C...
            VCOEF3N1=ETA1(NM1)+ETA2(NM1)
            VCOEF3N2=ETA1(NM2)+ETA2(NM2)
            VCOEF3N3=ETA1(NM3)+ETA2(NM3)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
            VCOEF3N1=VCOEF3N1+PR1(NM1)+PR2(NM1)
            VCOEF3N2=VCOEF3N2+PR1(NM2)+PR2(NM2)
            VCOEF3N3=VCOEF3N3+PR1(NM3)+PR2(NM3)
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

CTIP...
CTIP...UNCOMMENT THE FOLLOWING LINES TO USE TIDAL POTENTIAL FORCING
CTIP...COMMENT OUT THE FOLLOWING LINES IF NO TIDAL POTENTIAL FORCING
CTIP...
            VCOEF3N1=VCOEF3N1-TIP1(NM1)-TIP2(NM1)
            VCOEF3N2=VCOEF3N2-TIP1(NM2)-TIP2(NM2)
            VCOEF3N3=VCOEF3N3-TIP1(NM3)-TIP2(NM3)
CTIP...
CTIP...END OF TIDAL POTENTIAL FORCING STATEMENTS (MORE FOLLOW BELOW)
CTIP....

            VCOEF3N1=-VCOEF3N1*GDTO2
            VCOEF3N2=-VCOEF3N2*GDTO2
            VCOEF3N3=-VCOEF3N3*GDTO2

C...
C...COMPUTE ELEMENT AVERAGES QUANTITIES
C...
            UPPDT=SADVDTO3*(U1N1+U1N2+U1N3)
            VPPDT=SADVDTO3*(V1N1+V1N2+V1N3)
            UPPDTDDX1=UPPDT*DDX1
            UPPDTDDX2=UPPDT*DDX2
```

```
            UPPDTDDX3=UPPDT*DDX3
            VPPDTDDY1=VPPDT*DDY1
            VPPDTDDY2=VPPDT*DDY2
            VPPDTDDY3=VPPDT*DDY3
            EVMPP=(EVM(NM1)+EVM(NM2)+EVM(NM3))/3.
            EVMPPDT=EVMPP*DT

C...
C...ASSEMBLE PARTIAL PRODUCTS
C...
            VCOEF3X=VCOEF3N1*DDX1+VCOEF3N2*DDX2+VCOEF3N3*DDX3
            VCOEF3Y=VCOEF3N1*DDY1+VCOEF3N2*DDY2+VCOEF3N3*DDY3
            ADVECX=(UPPDTDDX1+VPPDTDDY1)*U1N1
     &             +(UPPDTDDX2+VPPDTDDY2)*U1N2
     &             +(UPPDTDDX3+VPPDTDDY3)*U1N3
            ADVECY=(UPPDTDDX1+VPPDTDDY1)*V1N1
     &             +(UPPDTDDX2+VPPDTDDY2)*V1N2
     &             +(UPPDTDDX3+VPPDTDDY3)*V1N3

C3D....
C3D....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D MODE
C3D....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI MODE
C3D....DISPERSION TERMS
C3D....
c           HPP=(HH1N1+HH1N2+HH1N3)/3.
c           DTOHPP=DT/HPP
c           DISPERX=IFNLCT*DTOHPP*(DUU1N1*DDX1+DUU1N2*DDX2+DUU1N3*DDX3
c     &                           +DUV1N1*DDY1+DUV1N2*DDY2+DUV1N3*DDY3)
c           DISPERY=IFNLCT*DTOHPP*(DUV1N1*DDX1+DUV1N2*DDX2+DUV1N3*DDX3
c     &                           +DVV1N1*DDY1+DVV1N2*DDY2+DVV1N3*DDY3)
C3D...
C3D....END OF 3D STATEMENTS (MORE FOLLOW BELOW)
C3D....

C...
C...LOAD NON-LUMPED ELEMENTAL COMPONENTS FOR X-MOMENTUM EQUATION INTO QTEMA
C....VECTOR FOR NODE NM1
C...
            QTEMA1=(
C...SURFACE GRADIENT, ATMOSPHERIC PRESSURE AND TIDAL POTENTIAL TERMS
     &           VCOEF3X
C...LATERAL VISCOUS TERMS
     &          -EVMPPDT*(DXXYY11*U1N1+DXXYY12*U1N2+DXXYY13*U1N3)
C...ADVECTIVE TERMS
     &          -ADVECX
C...DISPERSION (3D)
     &          +DISPERX
C...COMMON DIVISION OPERATION
     &                                                              )/FIIN
C...
C...LOAD NON-LUMPED ELEMENTAL COMPONENTS FOR X-MOMENTUM EQUATION INTO QTEMA
C....VECTOR FOR NODE NM2
C...
            QTEMA2=(
C...SURFACE GRADIENT, ATMOSPHERIC PRESSURE AND TIDAL POTENTIAL TERMS
     &           VCOEF3X
C...LATERAL VISCOUS TERMS]
     &          -EVMPPDT*(DXXYY21*U1N1+DXXYY22*U1N2+DXXYY23*U1N3)
C...ADVECTIVE TERMS
     &          -ADVECX
C...DISPERSION (3D)
     &          +DISPERX
C...COMMON DIVISION OPERATION
     &                                                              )/FIIN
C...
C...LOAD NON-LUMPED ELEMENTAL COMPONENTS FOR X-MOMENTUM EQUATION INTO QTEMA
C....VECTOR FOR NODE NM3
C...
            QTEMA3=(
C...SURFACE GRADIENT, ATMOSPHERIC PRESSURE AND TIDAL POTENTIAL TERMS
```

```
     &           VCOEF3X
C...LATERAL VISCOUS TERMS
     &          -EVMPPDT*(DXXYY31*U1N1+DXXYY32*U1N2+DXXYY33*U1N3)
C...ADVECTIVE TERMS
     &          -ADVECX
C...DISPERSION (3D)
     &          +DISPERX
C...COMMON DIVISION OPERATION
     &                                                              )/FIIN
C...
C...LOAD NON-LUMPED ELEMENTAL COMPONENTS FOR Y-MOMENTUM EQUATION INTO QTEMB
C....VECTOR FOR NODE NM1
C...
          QTEMB1=(
C...SURFACE GRADIENT, ATMOSPHERIC PRESSURE AND TIDAL POTENTIAL TERMS
     &           VCOEF3Y
C...LATERAL VISCOUS TERMS
     &          -EVMPPDT*(DXXYY11*V1N1+DXXYY12*V1N2+DXXYY13*V1N3)
C...ADVECTIVE TERMS
     &          -ADVECY
C...DISPERSION (3D)
     &          +DISPERY
C...COMMON DIVISION OPERATION
     &                                                              )/FIIN
C...
C...LOAD NON-LUMPED ELEMENTAL COMPONENTS FOR Y-MOMENTUM EQUATION INTO QTEMB
C....VECTOR FOR NODE NM2
C...
          QTEMB2=(
C...SURFACE GRADIENT, ATMOSPHERIC PRESSURE AND TIDAL POTENTIAL TERMS
     &           VCOEF3Y
C...LATERAL VISCOUS TERMS
     &          -EVMPPDT*(DXXYY21*V1N1+DXXYY22*V1N2+DXXYY23*V1N3)
C...ADVECTIVE TERMS
     &          -ADVECY
C...DISPERSION (3D)
     &          +DISPERY
C...COMMON DIVISION OPERATION
     &                                                              )/FIIN
C...
C...LOAD NON-LUMPED ELEMENTAL COMPONENTS FOR Y-MOMENTUM EQUATION INTO QTEMB
C....VECTOR FOR NODE NM3
C...
          QTEMB3=(
C...SURFACE GRADIENT, ATMOSPHERIC PRESSURE AND TIDAL POTENTIAL TERMS
     &           VCOEF3Y
C...LATERAL VISCOUS TERMS
     &          -EVMPPDT*(DXXYY31*V1N1+DXXYY32*V1N2+DXXYY33*V1N3)
C...ADVECTIVE TERMS
     &          -ADVECY
C...DISPERSION (3D)
     &          +DISPERY
C...COMMON DIVISION OPERATION
     &                                                              )/FIIN

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...
          QTEMA(IE,1)=QTEMA1
          QTEMA(IE,2)=QTEMA2
          QTEMA(IE,3)=QTEMA3
          QTEMB(IE,1)=QTEMB1
          QTEMB(IE,2)=QTEMB2
          QTEMB(IE,3)=QTEMB3
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CVEC....

CSCA...
```

```
CSCA...UNCOMMENT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CSCA...COMMENT OUT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CSCA...NOTE: THESE LINES FINALIZE THE ASSEMBLY PROCESS FOR QU, QV AND QUV
CSCA...      ON A SCALAR COMPUTER USING THE TEMPORARY VECTORS
CSCA...
c          QU(NM1)=QU(NM1)+QTEMA1
c          QU(NM2)=QU(NM2)+QTEMA2
c          QU(NM3)=QU(NM3)+QTEMA3
c          QV(NM1)=QV(NM1)+QTEMB1
c          QV(NM2)=QV(NM2)+QTEMB2
c          QV(NM3)=QV(NM3)+QTEMB3
CSCA...
CSCA...END OF SCALAR COMPUTER STATEMENTS
CSCA....

 9999   CONTINUE

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...NOTE: THESE LINES FINALIZE THE ASSEMBLY PROCESS FOR QU, QV AND AUV
CVEC...      ON A VECTOR COMPUTER USING THE TEMPORARY VECTORS
CVEC...
        DO IE=1,NE
          NM1=NM(IE,1)
          NM2=NM(IE,2)
          NM3=NM(IE,3)
          QU(NM1)=QU(NM1)+QTEMA(IE,1)
          QU(NM2)=QU(NM2)+QTEMA(IE,2)
          QU(NM3)=QU(NM3)+QTEMA(IE,3)
          QV(NM1)=QV(NM1)+QTEMB(IE,1)
          QV(NM2)=QV(NM2)+QTEMB(IE,2)
          QV(NM3)=QV(NM3)+QTEMB(IE,3)
          END DO
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS
CVEC....

C...
C...UPDATE MOMENTUM EQUATION LHS COEFFICIENTS AND LOAD VECTORS AT EACH NODE
C...BY DIVIDING BY NUMBER OF ELEMENTS THE NODE IS ASSOCIATED WITH AND ADDING
C...IN LUMPED TERMS AND BOTTOM FRICTION AND TAKING ACCOUNT OF THE BOUNDARY
C...CONDITION
C...
        WSX=0.D0
        WSY=0.D0
        DO I=1,NP
          NCI=ABS(NNODECODE(I))
          QU(I)=QU(I)/MJU(I)
          QV(I)=QV(I)/MJU(I)
          HH1=DP(I)+IFNLFA*ETA1(I)
          HH2=DP(I)+IFNLFA*ETA2(I)

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...
          WSX=DTO2*IFWIND*(WSX1(I)/HH1+WSX2(I)/HH2)
          WSY=DTO2*IFWIND*(WSY1(I)/HH1+WSY2(I)/HH2)
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS (MORE FOLLOW BELOW)
CWIND....

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI....TRANSIENT, CORIOLIS BOTTOM FRICTION AND FREE SURFACE STRESS
C2DDI....
          VCOEF1=DTO2*TK(I)
          VCOEF2=DTO2*CORIF(I)
```

```
          QU(I)=NCI*(QU(I)+WSX+(1.D0-VCOEF1)*UU1(I)+VCOEF2*VV1(I))
          QV(I)=NCI*(QV(I)+WSY+(1.D0-VCOEF1)*VV1(I)-VCOEF2*UU1(I))
          AUV11(I)=1.D0+VCOEF1*NCI
          AUV12(I)=-VCOEF2*NCI
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

C3DDSS.
C3DDSS....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE.
C3DDSS....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS....TRANSIENT, CORIOLIS AND BOTTOM FRICTION TERMS
C3DDSS.
c         VCOEF2=DTO2*CORIF(I)
c         QU(I)=(QU(I)+WSX+UU1(I)+VCOEF2*VV1(I)-DTO2*BSX1(I)/HH1)*NCI
c         QV(I)=(QV(I)+WSY+VV1(I)-VCOEF2*UU1(I)-DTO2*BSY1(I)/HH1)*NCI
c         AUV11(I)=1.D0
c         AUV12(I)=-VCOEF2*NCI
c         AUV13(I)=(DTO2/HH2)*NCI
c         AUV14(I)=0.D0
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

          END DO

C...
C...IF A SPECIFIED NORMAL FLOW ESSENTIAL BOUNDARY CONDITION IS USED,
C....IMPOSE NORMAL FLOW BOUNDARY CONDITIONS TO MOMENTUM MATRIX AND LOAD VECTOR
C....NOTE:THIS HAS BEEN SPECIALLY FORMULATED TO MATAIN THE BASIC
C....STRUCTURE OF THE LHS MATRIX (I.E., AUV11=AUV22; AUV12=-AUV21)
C...

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI....
        DO J=1,NVELME
          I=ME2GW(J)
          NBDI=NBV(I)
          HH2=DP(NBDI)+IFNLFA*ETA2(NBDI)
          NCI=ABS(NNODECODE(NBDI))
          IF((LBCODEI(I).GE.0).AND.(LBCODEI(I).LE.9)) THEN   !ESSENTIAL NORMAL FLOW
            VELNORM=-QN2(I)/HH2                              !AND FREE TANGENTIAL SLIP
            QU(NBDI)=(SIII(I)*QU(NBDI)-CSII(I)*QV(NBDI)
     &                              -VELNORM*AUV12(NBDI))*NCI  !TANGENTIAL EQN
            QV(NBDI)=VELNORM*AUV11(NBDI)*NCI                 !NORMAL EQN
            AUV12(NBDI)=-CSII(I)*AUV11(NBDI)
            AUV11(NBDI)=SIII(I)*AUV11(NBDI)
            ENDIF
          IF((LBCODEI(I).GE.10).AND.(LBCODEI(I).LE.19)) THEN !ESSENTIAL NORMAL FLOW
            VELNORM=-QN2(I)/HH2                              !AND NO TANGENTIAL SLIP
            VELTAN=0.D0
            QU(NBDI)=VELTAN*NCI                              !TANGENTIAL EQN
            QV(NBDI)=VELNORM*NCI                             !NORMAL EQN
            AUV11(NBDI)=SIII(I)
            AUV12(NBDI)=-CSII(I)
            ENDIF
          END DO
C2DDI...
C2DDI....END OF 2DDI STATEMENTS (MORE FOLLOW BELOW)
C2DDI...

C3DVS..
C3DVS..UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D VS MODE.
C3DVS..COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D DSS MODE.
C3DVS..
c       DO I=1,NP
c         IF(LBCODE(I).EQ.1) THEN      !SLIP LATERAL BOUNDARY CONDITION
c           QU(I)=SII(I)*QU(I)-CSI(I)*QV(I)
```

```
c           QV(I)=VELNORM
c           ENDIF
c         IF(LBCODE(I).EQ.-1) THEN     !NO SLIP LATERAL BOUNDARY CONDITION
c           QU(I)=0.D0
c           QV(I)=0.D0
c           ENDIF
c         ENDDO
C3DVS..
C3DVS..END OF 3D VS STATEMENTS (MORE FOLLOW BELOW)
C3DVS..

C3DDSS.
C3DDSS....UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE.
C3DDSS....COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS.
c       DO I=1,NP
c         IF(LBCODE(I).EQ.1) THEN      !SLIP LATERAL BOUNDARY CONDITION
c           QUNORM=-VELNORM*AUV12(I)*NCI
c           QVNORM=VELNORM*AUV11(I)*NCI
c           AUV12(I)=-CSI(I)*AUV11(I)
c           AUV11(I)=SII(I)*AUV11(I)
c           QU(I)=SII(I)*QU(I)-CSI(I)*QV(I)+QUNORM
c           QV(I)=QVNORM
c           AUV14(I)=-CSI(I)*AUV13(I)
c           AUV13(I)=SII(I)*AUV13(I)
c           ENDIF
c         IF(LBCODE(I).EQ.-1) THEN     !NO SLIP LATERAL BOUNDARY CONDITION
c           AUV11(I)=1.D0
c           AUV12(I)=0.D0
c           AUV14(I)=0.D0
c           AUV13(I)=0.D0
c           QU(I)=0.D0
c           QV(I)=0.D0
c           ENDIF
c         END DO
C3DDSS.
C3DDSS.END OF 3D DSS STATEMENTS (MORE FOLLOW BELOW)
C3DDSS.

C...
C...SOLVE FOR VELOCITY AT NEW LEVEL  (K+1)
C...

C2DDI...
C2DDI....UNCOMMENT THE FOLLOWING LINES FOR THE 2DDI VERSION OF THE CODE
C2DDI....COMMENT OUT THE FOLLOWING LINES FOR THE 3D VERSIONS OF THE CODE
C2DDI...
        DO I=1,NP
          AUV22=AUV11(I)
          AUV21=-AUV12(I)
          DDU=AUV11(I)*AUV22-AUV12(I)*AUV21
          UU2(I)=(QU(I)*AUV22-QV(I)*AUV12(I))/DDU
          VV2(I)=(QV(I)*AUV11(I)-QU(I)*AUV21)/DDU
          END DO
C2DDI...
C2DDI....END OF 2DDI STATEMENTS
C2DDI...


CWETDRY...
CWETDRY...THE FOLLOWING LINES ARE FOR WETTING AND DRYING
CWETDRY...
CWETDRY...WETTING CRITERIA
CWETDRY...
CWETDRY...NOTE: NODEREP IS THE NUMBER OF TIME STEPS SINCE A NODE LAST CHANGED STATE
CWETDRY....
CWETDRY....A NODE THAT IS ON THE WET/DRY INTERFACE WETS FOR TWO REASONS
CWETDRY....1.) IF THE VELOCITY COMPUTED AT THAT NODE HAS COMPONENTS THAT ARE > VELMIN
CWETDRY........TOWARD ALL NEIGHBORING DRY NODES AND NODEREP > NODEDRYMIN AT THE NODE
CWETDRY........UNDER CONSIDERATION AND ALL SURROUNDING DRY NODES.
```

```
CWETDRY........WHEN A NODE MOVES FROM THE WET/DRY INTERFACE TO BECOME A FULLY WET
CWETDRY........INTERIOR NODE, ALL SURROUNDING DRY NODES BECOME WET/DRY INTERFACE
CWETDRY........NODES.
CWETDRY....2.) IF ALL SURROUNDING NODES ARE EITHER WET OR ON THE WET/DRY BOUNDARY.
CWETDRY........IN THIS CASE THE NODE IS WET DUE TO BECOMING WATERLOGGED.
CWETDRY...
        IF(NOLIFA.EQ.2) THEN

       !WET NODES BY CONVERTING A WET/DRY INTERFACE NODE INTO A FULLY FUNCTIONAL
       !WET NODE IF NODEREP > NODEDRYMIN AND THE VELOCITY AT THAT NODE IS TOWARD
       !EACH SURROUNDING DRY NODE. MAKE INTERFACE NODES OUT OF ALL SURROUNDING
       !DRY NODES.

          NWET=0
          DO I=1,NP
            IF(NODECODE(I).EQ.-1) THEN
              IFAV=0
              IF(NODEREP(I).GT.NODEDRYMIN) THEN
                IFAV=1
                DO J=2,NNEIGH(I)
                  NEINOD=NEITAB(I,J)
                  IF(NODECODE(NEINOD).EQ.0) THEN
                    IF(NODEREP(NEINOD).LT.NODEDRYMIN) IFAV=0
      !             ETABAL=ETA2(I)-ETA2(NEINOD)
      !             IF(ETABAL.LE.0) IFAV=0
                    DELY=Y(NEINOD)-Y(I)
                    DELX=X(NEINOD)-X(I)
                    DIST=SQRT(DELX*DELX+DELY*DELY)
                    VELRES=(UU2(I)*DELX+VV2(I)*DELY)/DIST
                    IF(VELRES.LE.VELMIN) IFAV=0
                    ENDIF
                  END DO
                ENDIF
              IF(IFAV.EQ.1) THEN
                NWET=NWET+1
                IF(NSCREEN.EQ.1) WRITE(6,9890) I
                WRITE(16,9890) I
 9890           FORMAT(' !!! NODE ',I6,' WETTED ')
                NNODECODE(I)=1
                NODEREP(I)=0
                DO J=2,NNEIGH(I)
                  NEINOD=NEITAB(I,J)
                  IF(NODECODE(NEINOD).EQ.0) THEN
                    IF(NSCREEN.EQ.1) WRITE(6,9891) NEINOD
                    WRITE(16,9891) NEINOD
 9891               FORMAT(' !!! NODE ',I6,' BECAME AN INTERFACE NODE ',
     &                                          'DUE TO WETTING ')
                    NNODECODE(NEINOD)=-1
                    NODEREP(NEINOD)=0
                    ENDIF
                  END DO
                ENDIF
              IF(IFAV.EQ.0) THEN
                UU2(I)=0.
                VV2(I)=0.
                ENDIF
              ENDIF
            END DO

C...
C.....RESET GLOBAL ARRAY TO KEEP REGIONS BEHIND BARRIER SET IN DRYING
C.....SECTION AFTER THE GWCE EQUATION IS SOLVED
C.....NIBNODECODE(I)=0  ALLOWS A NODE TO BE TURNED OFF AFTER THE
C.....GWCE IS SOLVED
C.....NIBNODECODE(I)=1  DOES NOT ALLOW A NODE TO BE TURNED OFF
C.....AFTER THE GWCE IS SOLVED
C...
            IF(NFLUXIB.EQ.1) THEN
              DO I=1,NP
                NIBNODECODE(I)=0
```

```
              END DO
            ENDIF
C...
C.....PREPARE INWARD/OUTWARD NORMAL FLOW OVER SPECIFIED
C.....INTERNAL BARRIER BOUNDARY NODES AND DETERMINE WHICH
C.....ELEMENTS SHOULD BE WETTED
C...
            IF(NFLUXIB.EQ.1) THEN
              DO I=1,NVEL
                IF((LBCODEI(I).EQ.4).OR.(LBCODEI(I).EQ.24)) THEN
                  NNBB1=NBV(I)     ! GLOBAL NODE NO. ON THIS SIDE OF BARRIER
                  NNBB2=IBCONN(I) ! GLOBAL NODE NO. ON OPPOSITE SIDE OF BARRIER
                  RBARWL1AVG(I)=(ETA2(NNBB1)-BARINHT(I)+BARAVGWT
     &                          *RBARWL1AVG(I))/(1+BARAVGWT)
                  RBARWL2AVG(I)=(ETA2(NNBB2)-BARINHT(I)+BARAVGWT
     &                          *RBARWL2AVG(I))/(1+BARAVGWT)
                  RBARWL1=RBARWL1AVG(I)
                  RBARWL2=RBARWL2AVG(I)
                  RBARWL1F=2.0D0*RBARWL1/3.0D0
                  RBARWL2F=2.0D0*RBARWL2/3.0D0
C.................WATER LEVEL LOWER ON THIS SIDE OF BARRIER AND OVERTOPPING
C.................IS OCCURING
                  IF((RBARWL2.GT.RBARWL1).AND.(RBARWL2.GT.0)) THEN
C.....................RESET NIBNODECODE SUCH THAT NO NODE ATTACHED TO AN INTERNAL
C.....................BOUNDARY THAT IS ACTIVELY RECEIVING FLOW WILL NOT BE ALLOWED TO
C.....................TURN OFF. ALSO ANY NEIGHBORING NODES ARE NOT ALLOWED TO
C.....................BE TURNED OFF.
                    NIBNODECODE(NNBB1)=1
                    DO J=2,NNEIGH(NNBB1)
                      NEINOD=NEITAB(NNBB1,J)
                      NIBNODECODE(NEINOD)=1
                    END DO
C.....................IF FLOW IS INTO A DRY ELEMENT, RE-SET NNODECODE VALUES TO
C.....................ACCOMODATE INFLOW INTO/AND WETTING OF THE REGION
                    IF(NNODECODE(NNBB1).LE.0) THEN
                      NNODECODE(NNBB1)=1
                      NODEREP(NNBB1)=0
                      IF(NSCREEN.EQ.1) WRITE(6,9896) NNBB1
                      WRITE(16,9896) NNBB1
 9896                 FORMAT(' !!! NODE ',I6,' WETTED DUE TO INTERNAL',
     &                  ' BARRIER OVERTOPPING')
                      DO J=2,NNEIGH(NNBB1)
                        NEINOD=NEITAB(NNBB1,J)
                        NODEREP(NEINOD)=0
                        IF(NNODECODE(NEINOD).EQ.0) THEN
                          NNODECODE(NEINOD)=-1
                          IF(NSCREEN.EQ.1) WRITE(6,9897) NEINOD
                          WRITE(16,9897) NEINOD
 9897                     FORMAT(' !!! NODE ',I6,' BECAME AN INTERFACE',
     &                      ' NODE DUE TO WETTING OF AN INTERNAL ',
     &                      'BARRIER BOUNDARY')
                        ENDIF
                      END DO
                      NWET=NWET+1
                    ENDIF
                  ENDIF
                ENDIF
              END DO
            ENDIF

          !IF WETTING OCCURRED

            IF(NWET.GT.0) THEN

            !WET INTERFACE NODES THAT ARE NOT CONNECTED TO AT LEAST ONE DRY NODE

              DO I=1,NP
                IF(NNODECODE(I).EQ.-1) THEN
                  IDRY=0
                  DO J=2,NNEIGH(I)
```

```
                  NEINOD=NEITAB(I,J)
                  IF(NNODECODE(NEINOD).EQ.0) IDRY=IDRY+1
                  END DO
                IF(IDRY.EQ.0) THEN
                  IF(NSCREEN.EQ.1) WRITE(6,9892) I
                  WRITE(16,9892) I
 9892             FORMAT(' !!! NODE ',I6,' WETTED DUE TO WATERLOGGING')
                  NNODECODE(I)=1
                  NODEREP(I)=0
                  ENDIF
                ENDIF
              END DO

          !UPDATE THE NODECODE ARRAY

            DO I=1,NP
              NODECODE(I)=NNODECODE(I)
              END DO

            ENDIF

       !IF ANY NODES WETTED OR DRIED, SET GWCE MATRIX CHANGE FLAG

          IF((NDRY.GT.0).OR.(NWET.GT.0)) NCCHANGE=1

          ENDIF
CWETDRY...
CWETDRY...END WET/DRY SECTION
CWETDRY...

C3DVS..
C3DVS...UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D VS MODE.
C3DVS...COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D DSS MODE.
C3DVS..
c     CALL VSSOL(IT,TIME,DT)
C3DVS...
C3DVS...NOTE, WSX1, WSX2, WSY1, WSY2 MUST BE TREATED AS SCALARS IN 3D
C3DVS...SUBROUTINES IF NO WIND FORCING IS USED!!!!!
C3DVS...END OF 3D VS STATEMENTS
C3DVS...

C3DDSS...
C3DDSS...UNCOMMENT THE FOLLOWING LINES TO RUN THE CODE IN 3D DSS MODE.
C3DDSS...COMMENT OUT THE FOLLOWING LINES TO RUN THE CODE IN 2DDI OR 3D VS MODE.
C3DDSS...
c     CALL DSSSOL(IT,TIME,DT)
C3DDSS...
C3DDSS...NOTE, WSX1, WSX2, WSY1, WSY2 MUST BE TREATED AS SCALARS IN 3D
C3DDSS...SUBROUTINES IF NO WIND FORCING IS USED!!!!!
C3DDSS...END OF 3D DSS STATEMENTS
C3DDSS...

C...
C...IF TRANSPORT IS INCLUDED SOLVE FOR THE CONCENTRATION
C...NOTE: THE VARIABLE CH1(I) IS ACTUALLY C*H
C...
        IF(IM.EQ.10) THEN

C....COMPUTE SOURCE/SINK TERM AT THE NODES USING CLASSICAL COHESIVE
C....SEDIMENT TRANSPORT RELATIONS

          rho0  = 1025.0     ! reference density of seawater [kg/m^3]
          WS = 0.0001        ! particle fall velocity [m/s]
          CBEDSTRD = 0.15    ! critical shear stress for deposition [N/m^2]
          CCRITD = 0.30      ! critical concentration for hindered settling [kg/m^3]
          ECONST = 0.00001   ! erosion rate constant [kg/m^2/sec]
          CBEDSTRE = 0.4     ! critical shear stress for erosion [N/m^2]

          DO I=1,NP
            UV1=SQRT(UU1(I)*UU1(I)+VV1(I)*VV1(I))
```

```
            HH1=DP(I)+IFNLFA*ETA1(I)
            BEDSTR=HH1*UV1*TK(I)*rho0                    !in N/m^2
            C1=CH1(I)/HH1

C.....Calculate the deposition rate using Krone's (1962) formulation:
C.....dC/dt = -P*WSMOD*C/D     where
C.....WSMOD=WS           when C < Ccrit  and
C.....WSMOD=K*C**1.33    when C > Ccrit
C.....D is the average depth through which particles settle D = H/2,
C.....H is the water depth
C.....C is the depth-averaged sediment concentration,
C.....P is the sticking probability  P = (1-BEDSTR/CBEDSTRD),
C.....CBEDSTRD is the critical bottom stress above which no deposition occurs.
C.....It was assumed that the constant K could be backed out by setting
C.....WSMOD = WS when C = Ccrit.

            WSMOD=WS
            IF(C1.GT.CCRITD) WSMOD=WS*(C1/CCRITD)**1.33
            HSD=0.
            IF(BEDSTR.LT.CBEDSTRD) HSD=-(2.*WSMOD*C1)*
     &                                  (1.0-BEDSTR/CBEDSTRD)
            IF(HSD.GT.0.) HSD=0.

C.....Calculate the surface erosion rate for cohesive sediment using
C.....the Ariathurai et at. (1977) adaption of Partheniades' (1962) findings

            HSE=0.
            IF(BEDSTR.GT.CBEDSTRE) HSE=ECONST*(BEDSTR/CBEDSTRE-1.0)

C.....Determine the total source sink term

            SOURSIN(I)=HSD+HSE
            END DO

C....UPDATE THE TRANSPORT EQUATION ELEMENT BY ELEMENT BY FORMING TEMPORARY
C....VECTORS AND THEN ASSEMBLING.
C....NOTE: QB(I), QA(I) ARE ZEROED OUT AT THE TOP OF THE TIME STEPPING LOOP.
C....AGAIN THESE LOOPS HAVE BEEN UNROLLED TO OPTIMIZE VECTORIZATION

        DO 1057 IE=1,NE

C.....SET NODAL VALUES FOR EACH ELEMENT

          NM1=NM(IE,1)
          NM2=NM(IE,2)
          NM3=NM(IE,3)
          NC1=ABS(NNODECODE(NM1))
          NC2=ABS(NNODECODE(NM2))
          NC3=ABS(NNODECODE(NM3))
          NCELE=NC1*NC2*NC3
          U1N1=UU1(NM1)
          U1N2=UU1(NM2)
          U1N3=UU1(NM3)
          V1N1=VV1(NM1)
          V1N2=VV1(NM2)
          V1N3=VV1(NM3)
          CH1N1=CH1(NM1)
          CH1N2=CH1(NM2)
          CH1N3=CH1(NM3)
          EVC1=EVC(NM1)
          EVC2=EVC(NM2)
          EVC3=EVC(NM3)
          SS1N1=SOURSIN(NM1)
          SS1N2=SOURSIN(NM2)
          SS1N3=SOURSIN(NM3)
          HH1N1=DP(NM1)+IFNLFA*ETA1(NM1)
          HH1N2=DP(NM2)+IFNLFA*ETA1(NM2)
          HH1N3=DP(NM3)+IFNLFA*ETA1(NM3)
          SFACPP=(SFAC(NM1)+SFAC(NM2)+SFAC(NM3))/3.
```

```
C.....COMPUTE ELEMENTAL MATRICIES

          AREAIE=AREAS(IE)                !2*element area
          FDX1=(Y(NM2)-Y(NM3))*SFACPP     !b1
          FDX2=(Y(NM3)-Y(NM1))*SFACPP     !b2
          FDX3=(Y(NM1)-Y(NM2))*SFACPP     !b3
          FDY1=X(NM3)-X(NM2)              !a1
          FDY2=X(NM1)-X(NM3)              !a2
          FDY3=X(NM2)-X(NM1)              !a3
          FDX1OA=FDX1/AREAIE              !dphi1/dx
          FDY1OA=FDY1/AREAIE              !dphi1/dy
          FDX2OA=FDX2/AREAIE              !dphi2/dx
          FDY2OA=FDY2/AREAIE              !dphi2/dy
          FDX3OA=FDX3/AREAIE              !dphi3/dx
          FDY3OA=FDY3/AREAIE              !dphi3/dy

          DDX1=FDX1/3.                    !<2*(dphi1/dx)*phij> j=1,2,3
          DDY1=FDY1/3.                    !<2*(dphi1/dy)*phij> j=1,2,3
          DXX11=FDX1OA*FDX1               !<2*(dphi1/dx)*(dphi1/dx)>
          DYY11=FDY1OA*FDY1               !<2*(dphi1/dy)*(dphi1/dy)>
          DXXYY11=DXX11+DYY11
          DXX12=FDX1OA*FDX2               !<2*(dphi1/dx)*(dphi2/dx)>
          DYY12=FDY1OA*FDY2               !<2*(dphi1/dy)*(dphi2/dy)>
          DXXYY12=DXX12+DYY12
          DXX13=FDX1OA*FDX3               !<2*(dphi1/dx)*(dphi3/dx)>
          DYY13=FDY1OA*FDY3               !<2*(dphi1/dy)*(dphi3/dy)>
          DXXYY13=DXX13+DYY13

          DDX2=FDX2/3.                    !<2*(dphi2/dx)*phij> j=1,2,3
          DDY2=FDY2/3.                    !<2*(dphi2/dy)*phij> j=1,2,3
          DXX22=FDX2OA*FDX2               !<2*(dphi2/dx)*(dphi2/dx)>
          DYY22=FDY2OA*FDY2               !<2*(dphi2/dy)*(dphi2/dy)>
          DXXYY22=DXX22+DYY22
          DXX23=FDX2OA*FDX3               !<2*(dphi2/dx)*(dphi3/dx)>
          DYY23=FDY2OA*FDY3               !<2*(dphi2/dy)*(dphi3/dy)>
          DXXYY23=DXX23+DYY23

          DDX3=FDX3/3.                    !<2*(dphi3/dx)*phij> j=1,2,3
          DDY3=FDY3/3.                    !<2*(dphi3/dy)*phij> j=1,2,3
          DXX33=FDX3OA*FDX3               !<2*(dphi3/dx)*(dphi3/dx)>
          DYY33=FDY3OA*FDY3               !<2*(dphi3/dy)*(dphi3/dy)>
          DXXYY33=DXX33+DYY33

          LUMPT=1                         !=1/0; LUMP/DO NOT LUMP THE TRANSPORT EQN
          FDDD=(1+LUMPT)*AREAIE/6.D0      !<2*(phii*phij) i=j>
          FDDOD=(1-LUMPT)*AREAIE/12.D0    !<2*(phii*phij) i<>j>
          FDDDODT=FDDD/DTDP
          FDDODODT=FDDOD/DTDP

C.....COMPUTE ELEMENT AVERAGES QUANTITIES

          UEA=(U1N1+U1N2+U1N3)/3.
          VEA=(V1N1+V1N2+V1N3)/3.
          HEA=(HH1N1+HH1N2+HH1N3)/3.
          EVCEA=(EVC1+EVC2+EVC3)/3.
          DHDX=HH1N1*FDX1OA+HH1N2*FDX2OA+HH1N3*FDX3OA
          DHDY=HH1N1*FDY1OA+HH1N2*FDY2OA+HH1N3*FDY3OA
          UPEA=UEA+DHDX*EVCEA/HEA
          VPEA=VEA+DHDY*EVCEA/HEA

C.....ASSEMBLE PARTIAL PRODUCT

          CHSUM=CH1N1+CH1N2+CH1N3

C.....LOAD ELEMENTAL COMPONENTS FOR TRANSPORT EQUATION INTO QTEMA1 AND
C.....QTEMB1 VECTORS FOR NODE NM1

          QTEMB1=                         !LOAD VECTOR
C......TRANSIENT TERM (EITHER LUMPED OR CONSISTENT)
     &             FDDDODT*CH1N1+FDDODODT*(CH1N2+CH1N3)
```

```
C......LATERAL SGS TERMS
     &            -EVCEA*(DXXYY11*CH1N1+DXXYY12*CH1N2+DXXYY13*CH1N3)
C......ADVECTIVE TERMS
     &            +(UPEA*DDX1+VPEA*DDY1)*CHSUM
C......SOURCE SINK TERMS (EITHER LUMPED OR CONSISTENT)
     &            +FDDD*SS1N1+FDDOD*(SS1N2+SS1N3)
          QTEMA1=                          !LHS VECTOR
C......TRANSIENT TERM (LUMPED)
     &             FDDDODT+2.*FDDODODT

C.....LOAD ELEMENTAL COMPONENTS FOR TRANSPORT EQUATION INTO QTEMA2 AND
C.....QTEMB2 VECTOR FOR NODE NM2

          QTEMB2=                          !LOAD VECTOR
C......TRANSIENT TERM (EITHER LUMPED OR CONSISTENT)
     &             FDDDODT*CH1N2+FDDODODT*(CH1N1+CH1N3)
C......LATERAL SGS TERMS
     &            -EVCEA*(DXXYY12*CH1N1+DXXYY22*CH1N2+DXXYY23*CH1N3)
C......ADVECTIVE TERMS
     &            +(UPEA*DDX2+VPEA*DDY2)*CHSUM
C......SOURCE SINK TERMS (EITHER LUMPED OR CONSISTENT)
     &            +FDDD*SS1N2+FDDOD*(SS1N1+SS1N3)
          QTEMA2=                          !LHS VECTOR
C......TRANSIENT TERM (LUMPED)
     &             FDDDODT+2.*FDDODODT

C.....LOAD ELEMENTAL COMPONENTS FOR TRANSPORT EQUATION INTO QTEMA3 AND
C.....QTEMB3 VECTOR FOR NODE NM3

          QTEMB3=                          !LOAD VECTOR
C......TRANSIENT TERM (EITHER LUMPED OR CONSISTENT)
     &             FDDDODT*CH1N3+FDDODODT*(CH1N1+CH1N2)
C......LATERAL SGS TERMS
     &            -EVCEA*(DXXYY13*CH1N1+DXXYY23*CH1N2+DXXYY33*CH1N3)
C......ADVECTIVE TERMS
     &            +(UPEA*DDX3+VPEA*DDY3)*CHSUM
C......SOURCE SINK TERMS (EITHER LUMPED OR CONSISTENT)
     &            +FDDD*SS1N3+FDDOD*(SS1N1+SS1N2)
          QTEMA3=                          !LHS VECTOR
C......TRANSIENT TERM (LUMPED)
     &             FDDDODT+2.*FDDODODT

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...
          QTEMB(IE,1)=QTEMB1*NCELE         !LOAD VECTOR
          QTEMB(IE,2)=QTEMB2*NCELE         !LOAD VECTOR
          QTEMB(IE,3)=QTEMB3*NCELE         !LOAD VECTOR
          QTEMA(IE,1)=QTEMA1*NCELE         !LUMPED LHS MATRIX
          QTEMA(IE,2)=QTEMA2*NCELE         !LUMPED LHS MATRIX
          QTEMA(IE,3)=QTEMA3*NCELE         !LUMPED LHS MATRIX
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS (MORE FOLLOW BELOW)
CVEC....

CSCA...
CSCA...UNCOMMENT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CSCA...COMMENT OUT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CSCA...NOTE: THESE LINES FINALIZE THE ASSEMBLY PROCESS FOR QC AND QA
CSCA...       ON A SCALAR COMPUTER USING THE TEMPORARY VECTORS
CSCA...
c         QB(NM1)=QB(NM1)+QTEMB1*NCELE     !LOAD VECTOR
c         QB(NM2)=QB(NM2)+QTEMB2*NCELE     !LOAD VECTOR
c         QB(NM3)=QB(NM3)+QTEMB3*NCELE     !LOAD VECTOR
c         QA(NM1)=QA(NM1)+QTEMA1*NCELE     !LUMPED LHS MATRIX
c         QA(NM2)=QA(NM2)+QTEMA2*NCELE     !LUMPED LHS MATRIX
c         QA(NM3)=QA(NM3)+QTEMA3*NCELE     !LUMPED LHS MATRIX
CSCA...
CSCA...END OF SCALAR COMPUTER STATEMENTS
```

```
CSCA....

1057    CONTINUE

CVEC...
CVEC...UNCOMMENT THE FOLLOWING LINES TO RUN ON A VECTOR COMPUTER
CVEC...COMMENT OUT THE FOLLOWING LINES TO RUN ON A SCALAR COMPUTER
CVEC...NOTE: THESE LINES FINALIZE THE ASSEMBLY PROCESS FOR QC, QA
CVEC...       ON A VECTOR COMPUTER USING THE TEMPORARY VECTORS
CVEC...
        DO IE=1,NE
          NM1=NM(IE,1)
          NM2=NM(IE,2)
          NM3=NM(IE,3)
          QA(NM1)=QA(NM1)+QTEMA(IE,1)       !LUMPED LHS MATRIX
          QA(NM2)=QA(NM2)+QTEMA(IE,2)       !LUMPED LHS MATRIX
          QA(NM3)=QA(NM3)+QTEMA(IE,3)       !LUMPED LHS MATRIX
          QB(NM1)=QB(NM1)+QTEMB(IE,1)       !LOAD VECTOR
          QB(NM2)=QB(NM2)+QTEMB(IE,2)       !LOAD VECTOR
          QB(NM3)=QB(NM3)+QTEMB(IE,3)       !LOAD VECTOR
          END DO
CVEC...
CVEC...END OF VECTOR COMPUTER STATEMENTS
CVEC....

C....SOLVE FOR C*H NODE BY NODE

        DO I=1,NP
          NCI=ABS(NNODECODE(I))
          IF(NCI.NE.0) CH1(I)=QB(I)/QA(I)
C         IF(LBCODE(I).NE.0) CH1(I)=0.     !ESSENTIAL C=0 BOUNDARY CONDITION
          END DO

        ENDIF

C...
C...OUTPUT ELEVATION RECORDING STATION INFORMATION IF NOUTE<>0 AND THE
C....TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
C...CALCULATE ELEVATION SOLUTIONS AT STATIONS USING INTERPOLATION
C...

        IF(NOUTE.NE.0) THEN
          IF((IT.GT.NTCYSE).AND.(IT.LE.NTCYFE)) THEN
            NSCOUE=NSCOUE+1
            IF(NSCOUE.EQ.NSPOOLE) THEN
              DO I=1,NSTAE
                EE1=ETA2(NM(NNE(I),1))
                EE2=ETA2(NM(NNE(I),2))
                EE3=ETA2(NM(NNE(I),3))
                NC1=ABS(NNODECODE(NM(NNE(I),1)))
                NC2=ABS(NNODECODE(NM(NNE(I),2)))
                NC3=ABS(NNODECODE(NM(NNE(I),3)))
                NCELE=NC1*NC2*NC3
                IF(NCELE.EQ.1) ET00(I)=EE1*STAIE1(I)+EE2*STAIE2(I)
     &                                              +EE3*STAIE3(I)
                IF(NCELE.EQ.0) ET00(I)=-99999.
                END DO
              IF(ABS(NOUTE).EQ.1) THEN
                WRITE(61,2120)TIMEOUT,IT
                DO I=1,NSTAE
                  WRITE(61,2453) I,ET00(I)
                  END DO
                IESTP = IESTP+1+NSTAE
                ENDIF
              IF(ABS(NOUTE).EQ.2) THEN
                WRITE(61,REC=IESTP+1) TIMEOUT
                WRITE(61,REC=IESTP+2) IT
                IESTP = IESTP + 2
                DO I=1,NSTAE
                  WRITE(61,REC=IESTP+I) ET00(I)
```

```
                END DO
                IESTP = IESTP + NSTAE
                ENDIF
              NSCOUE=0
              ENDIF
            ENDIF
          ENDIF

C...
C...OUTPUT VELOCITY RECORDING STATION TIME SERIES INFORMATION IF NOUTV<>0
C....AND THE TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
C...CALCULATE VELOCITY SOLUTIONS AT STATIONS USING INTERPOLATION
C...
      IF(NOUTV.NE.0) THEN
        IF((IT.GT.NTCYSV).AND.(IT.LE.NTCYFV)) THEN
          NSCOUV=NSCOUV+1
          IF(NSCOUV.EQ.NSPOOLV) THEN
            DO I=1,NSTAV
              U11=UU2(NM(NNV(I),1))
              U22=UU2(NM(NNV(I),2))
              U33=UU2(NM(NNV(I),3))
              V11=VV2(NM(NNV(I),1))
              V22=VV2(NM(NNV(I),2))
              V33=VV2(NM(NNV(I),3))
              UU00(I)=U11*STAIV1(I)+U22*STAIV2(I)+U33*STAIV3(I)
              VV00(I)=V11*STAIV1(I)+V22*STAIV2(I)+V33*STAIV3(I)
              END DO
            IF(ABS(NOUTV).EQ.1) THEN
              WRITE(62,2120) TIMEOUT,IT
              DO I=1,NSTAV
                WRITE(62,2454) I,UU00(I),VV00(I)
                END DO
              IVSTP = IVSTP+1+NSTAV
              ENDIF
            IF(ABS(NOUTV).EQ.2) THEN
              WRITE(62,REC=IVSTP+1) TIMEOUT
              WRITE(62,REC=IVSTP+2) IT
              IVSTP = IVSTP + 2
              DO I=1,NSTAV
                WRITE(62,REC=IVSTP+2*I-1) UU00(I)
                WRITE(62,REC=IVSTP+2*I) VV00(I)
                END DO
              IVSTP = IVSTP + 2*NSTAV
              ENDIF
            NSCOUV=0
            ENDIF
          ENDIF
        ENDIF

C...
C...OUTPUT CONCENTRATION RECORDING STATION INFORMATION IF NOUTC<>0 AND THE
C....TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
C...CALCULATE CONCENTRATION SOLUTIONS AT STATIONS USING INTERPOLATION
C...
      IF(NOUTC.NE.0) THEN
        IF((IT.GT.NTCYSC).AND.(IT.LE.NTCYFC)) THEN
          NSCOUC=NSCOUC+1
          IF(NSCOUC.EQ.NSPOOLC) THEN
            DO I=1,NSTAC
              NM1=NM(NNC(I),1)
              NM2=NM(NNC(I),2)
              NM3=NM(NNC(I),3)
              HH2N1=DP(NM1)+IFNLFA*ETA2(NM1)
              HH2N2=DP(NM2)+IFNLFA*ETA2(NM2)
              HH2N3=DP(NM3)+IFNLFA*ETA2(NM3)
              C1=CH1(NM1)/HH2N1
              C2=CH1(NM2)/HH2N2
              C3=CH1(NM3)/HH2N3
              NC1=ABS(NNODECODE(NM1))
              NC2=ABS(NNODECODE(NM2))
```

```
                NC3=ABS(NNODECODE(NM3))
                NCELE=NC1*NC2*NC3
                IF(NCELE.EQ.1) CC00(I)=C1*STAIC1(I)+C2*STAIC2(I)
     &                                                +C3*STAIC3(I)
                IF(NCELE.EQ.0) CC00(I)=-99999.
                END DO
              IF(ABS(NOUTC).EQ.1) THEN
                WRITE(71,2120)TIMEOUT,IT
                DO I=1,NSTAC
                  WRITE(71,2453) I,CC00(I)
                  END DO
                ICSTP = ICSTP+1+NSTAC
                ENDIF
              IF(ABS(NOUTC).EQ.2) THEN
                WRITE(71,REC=ICSTP+1) TIMEOUT
                WRITE(71,REC=ICSTP+2) IT
                ICSTP = ICSTP + 2
                DO I=1,NSTAC
                  WRITE(71,REC=ICSTP+I) CC00(I)
                  END DO
                ICSTP = ICSTP + NSTAC
                ENDIF
              NSCOUC=0
              ENDIF
            ENDIF
          ENDIF

C...
C...OUTPUT GLOBAL ELEVATION DATA IF NOUTGE<>0 AND THE
C....TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
C...
        IF(NOUTGE.NE.0) THEN
          IF((IT.GT.NTCYSGE).AND.(IT.LE.NTCYFGE)) THEN
            NSCOUGE=NSCOUGE+1
            IF(NSCOUGE.EQ.NSPOOLGE) THEN
              IF(ABS(NOUTGE).EQ.1) THEN
                WRITE(63,2120) TIMEOUT,IT
2120            FORMAT(2X,E20.10,5X,I10)
                DO I=1,NP
                  IF(ABS(NNODECODE(I)).EQ.1) WRITE(63,2453) I,ETA2(I)
                  IF(NNODECODE(I).EQ.0) WRITE(63,2453) I,-99999.
2453              FORMAT(2X,I8,2X,E15.8)
                  ENDDO
                IGEP=IGEP+1+NP
                ENDIF
              IF(ABS(NOUTGE).EQ.2) THEN
                WRITE(63,REC=IGEP+1) TIMEOUT
                WRITE(63,REC=IGEP+2) IT
                IGEP = IGEP + 2
                DO I=1,NP
                  IF(ABS(NNODECODE(I)).EQ.1) WRITE(63,REC=IGEP+I)ETA2(I)
                  IF(NNODECODE(I).EQ.0) WRITE(63,REC=IGEP+I) -99999.
                  ENDDO
                IGEP = IGEP + NP
                ENDIF
              NSCOUGE=0
              ENDIF
            ENDIF
          ENDIF

C...
C...OUTPUT GLOBAL VELOCITY DATA IF NOUTGV<>0 AND THE
C....TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
C...
        IF(NOUTGV.NE.0) THEN
          IF((IT.GT.NTCYSGV).AND.(IT.LE.NTCYFGV)) THEN
            NSCOUGV=NSCOUGV+1
            IF(NSCOUGV.EQ.NSPOOLGV) THEN
              IF(ABS(NOUTGV).EQ.1) THEN
                WRITE(64,2120) TIMEOUT,IT
```

```
                DO I=1,NP
                  WRITE(64,2454) I,UU2(I),VV2(I)
2454              FORMAT(2X,I8,2(2X,E15.8))
                  ENDDO
                IGVP = IGVP+1+NP
                ENDIF
              IF(ABS(NOUTGV).EQ.2) THEN
                WRITE(64,REC=IGVP+1) TIMEOUT
                WRITE(64,REC=IGVP+2) IT
                IGVP = IGVP + 2
                DO I=1,NP
                  WRITE(64,REC=IGVP+2*I-1) UU2(I)
                  WRITE(64,REC=IGVP+2*I) VV2(I)
                  END DO
                IGVP = IGVP + 2*NP
                ENDIF
              NSCOUGV=0
              ENDIF
            ENDIF
          ENDIF

C...
C...OUTPUT GLOBAL CONCENTRATION DATA IF NOUTGC<>0 AND THE
C....TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
C...
        IF(NOUTGC.NE.0) THEN
          IF((IT.GT.NTCYSGC).AND.(IT.LE.NTCYFGC)) THEN
            NSCOUGC=NSCOUGC+1
            IF(NSCOUGC.EQ.NSPOOLGC) THEN
              IF(ABS(NOUTGC).EQ.1) THEN
                WRITE(73,2120) TIMEOUT,IT
                DO I=1,NP
                  HH2=DP(I)+IFNLFA*ETA2(I)
                  C1=CH1(I)/HH2
                  IF(ABS(NNODECODE(I)).EQ.1) WRITE(73,2453) I,C1
                  IF(NNODECODE(I).EQ.0) WRITE(73,2453) I,-99999.
                  ENDDO
                IGCP=IGCP+1+NP
                ENDIF
              IF(ABS(NOUTGC).EQ.2) THEN
                WRITE(73,REC=IGEP+1) TIMEOUT
                WRITE(73,REC=IGEP+2) IT
                IGCP = IGCP + 2
                DO I=1,NP
                  HH2=DP(I)+IFNLFA*ETA2(I)
                  C1=CH1(I)/HH2
                  IF(ABS(NNODECODE(I)).EQ.1) WRITE(73,REC=IGCP+I) C1
                  IF(NNODECODE(I).EQ.0) WRITE(73,REC=IGCP+I) -99999.
                  ENDDO
                IGCP=IGCP+NP
                ENDIF
              NSCOUGC=0
              ENDIF
            ENDIF
          ENDIF

CWIND...
CWIND...UNCOMMENT THE FOLLOWING LINES TO USE WIND AND PRESSURE FORCING
CWIND...COMMENT OUT THE FOLLOWING LINES IF NO WIND AND PRESSURE FORCING
CWIND...OUTPUT GLOBAL WIND STRESS DATA IF NOUTGW<>0 AND THE
CWIND....TIME STEP FALLS WITHIN THE SPECIFIED WINDOW
CWIND...
        IF((NWS.GT.0).AND.(NOUTGW.NE.0)) THEN
          IF((IT.GT.NTCYSGW).AND.(IT.LE.NTCYFGW)) THEN
            NSCOUGW=NSCOUGW+1
            IF(NSCOUGW.EQ.NSPOOLGW) THEN
              IF(ABS(NOUTGW).EQ.1) THEN
                WRITE(74,2120) TIMEOUT,IT
                DO I=1,NP
                  WRITE(74,2454) I,WSX2(I),WSY2(I)
```

```
                ENDDO
              IGWP = IGWP+1+NP
              ENDIF
            IF(ABS(NOUTGW).EQ.2) THEN
              WRITE(74,REC=IGWP+1) TIMEOUT
              WRITE(74,REC=IGWP+2) IT
              IGWP = IGWP + 2
              DO I=1,NP
                WRITE(74,REC=IGWP+2*I-1) WSX2(I)
                WRITE(74,REC=IGWP+2*I) WSY2(I)
                END DO
              IGWP = IGWP + 2*NP
              ENDIF
            NSCOUGW=0
            ENDIF
          ENDIF
        ENDIF
CWIND...
CWIND...END OF WIND AND PRESSURE FORCING STATEMENTS
CWIND....

C...
C...IF IHARIND=1 AND THE TIME STEP FALLS WITHIN THE SPECIFIED WINDOW AND
C...ON THE SPECIFIED INCREMENT, USE MODEL RESULTS TO UPDATE HARMONIC
C...ANALYSIS MATRIX AND LOAD VECTORS.  NOTE: AN 8 BYTE RECORD SHOULD BE
C...USED THROUGHOUT THE HARMONIC ANALYSIS SUBROUTINES, EVEN ON 32 BIT
C...WORKSTATIONS, SINCE IN THAT CASE THE HARMONIC ANALYSIS IS DONE IN
C...DOUBLE PRECISION.
C...
      IF(IHARIND.EQ.1) THEN
        IF((IT.GT.ITHAS).AND.(IT.LE.ITHAF)) THEN
          IF(ICHA.EQ.NHAINC) ICHA=0
          ICHA=ICHA+1
          IF(ICHA.EQ.NHAINC) THEN
C...
C.....UPDATE THE LHS MATRIX
C...
            CALL LSQUPDLHS(TIME,IT)
C...
C.....IF DESIRED COMPUTE ELEVATION STATION INFORMATION AND UPDATE LOAD VECTOR
C...
            IF(NHASE.EQ.1) THEN
              DO I=1,NSTAE
                EE1=ETA2(NM(NNE(I),1))
                EE2=ETA2(NM(NNE(I),2))
                EE3=ETA2(NM(NNE(I),3))
                ET00(I)=EE1*STAIE1(I)+EE2*STAIE2(I)+EE3*STAIE3(I)
                END DO
              CALL LSQUPDES(ET00,NSTAE)
              ENDIF
C...
C.....IF DESIRED COMPUTE VELOCITY STATION INFORMATION AND UPDATE LOAD VECTOR
C...
            IF(NHASV.EQ.1) THEN
              DO I=1,NSTAV
                U11=UU2(NM(NNV(I),1))
                U22=UU2(NM(NNV(I),2))
                U33=UU2(NM(NNV(I),3))
                V11=VV2(NM(NNV(I),1))
                V22=VV2(NM(NNV(I),2))
                V33=VV2(NM(NNV(I),3))
                UU00(I)=U11*STAIV1(I)+U22*STAIV2(I)+U33*STAIV3(I)
                VV00(I)=V11*STAIV1(I)+V22*STAIV2(I)+V33*STAIV3(I)
                END DO
              CALL LSQUPDVS(UU00,VV00,NSTAV)
              ENDIF
C...
C.....IF DESIRED UPDATE GLOBAL ELEVATION LOAD VECTOR
C...
            IF(NHAGE.EQ.1) CALL LSQUPDEG(ETA2,NP)
```

```
C...
C.....IF DESIRED UPDATE GLOBAL ELEVATION LOAD VECTOR
C...
              IF(NHAGV.EQ.1) CALL LSQUPDVG(UU2,VV2,NP)

              ENDIF
            ENDIF

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...ACCUMULATE THE MEAN AND VARIANCE OF THE MODEL OUTPUT GLOBALLY
CHARMV...
           IF(IT.GT.ITMV) THEN
             NTSTEPS=NTSTEPS+1
             DO I=1,NP
               ELAV(I)=ELAV(I)+ETA2(I)
               XVELAV(I)=XVELAV(I)+UU2(I)
               YVELAV(I)=YVELAV(I)+VV2(I)
               ELVA(I)=ELVA(I)+ETA2(I)*ETA2(I)
               XVELVA(I)=XVELVA(I)+UU2(I)*UU2(I)
               YVELVA(I)=YVELVA(I)+VV2(I)*VV2(I)
               END DO
             ENDIF
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

           ENDIF

C...
C...WRITE OUT HOT START INFORMATION IF NHSTAR=1 AND AT CORRECT TIME STEP
C...NOTE: THE HOT START FILES USE A RECORD LENGTH OF 8 ON BOTH 32 BIT
C....WORKSTATIONS AND THE 64 BIT CRAY.  THIS IS BECAUSE THE HARMONIC
C....ANALYSIS IS DONE IN DOUBLE PRECISION (64 BITS) ON WORKSTATIONS.
C...
         IF(NHSTAR.EQ.1) THEN
           ITEST=(IT/NHSINC)*NHSINC
           IF(ITEST.EQ.IT) THEN
             IF(IHSFIL.EQ.67) OPEN(67,FILE='fort.67',ACCESS='DIRECT',
     &                                                       RECL=8)
             IF(IHSFIL.EQ.68) OPEN(68,FILE='fort.68',ACCESS='DIRECT',
     &                                                       RECL=8)
             IHOTSTP=1
             WRITE(IHSFIL,REC=IHOTSTP) IM
             IHOTSTP=2
             WRITE(IHSFIL,REC=IHOTSTP) TIME
             IHOTSTP=3
             WRITE(IHSFIL,REC=IHOTSTP) IT
             DO I=1,NP
               WRITE(IHSFIL,REC=IHOTSTP+1) ETA1(I)
               WRITE(IHSFIL,REC=IHOTSTP+2) ETA2(I)
               WRITE(IHSFIL,REC=IHOTSTP+3) UU2(I)
               WRITE(IHSFIL,REC=IHOTSTP+4) VV2(I)
               IHOTSTP = IHOTSTP + 4
               IF(IM.EQ.10) THEN
                 WRITE(IHSFIL,REC=IHOTSTP+1) CH1(I)
                 IHOTSTP=IHOTSTP+1
                 ENDIF
               WRITE(IHSFIL,REC=IHOTSTP+1) NNODECODE(I)
               IHOTSTP=IHOTSTP+1
               END DO
             WRITE(IHSFIL,REC=IHOTSTP+1) IESTP
             WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUE
             IHOTSTP=IHOTSTP+2
             WRITE(IHSFIL,REC=IHOTSTP+1) IVSTP
             WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUV
             IHOTSTP=IHOTSTP+2
             WRITE(IHSFIL,REC=IHOTSTP+1) ICSTP
             WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUC
```

```
            IHOTSTP=IHOTSTP+2
            WRITE(IHSFIL,REC=IHOTSTP+1) IGEP
            WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUGE
            IHOTSTP=IHOTSTP+2
            WRITE(IHSFIL,REC=IHOTSTP+1) IGVP
            WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUGV
            IHOTSTP=IHOTSTP+2
            WRITE(IHSFIL,REC=IHOTSTP+1) IGCP
            WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUGC
            IHOTSTP=IHOTSTP+2
            WRITE(IHSFIL,REC=IHOTSTP+1) IGWP
            WRITE(IHSFIL,REC=IHOTSTP+2) NSCOUGW
            IHOTSTP=IHOTSTP+2
C...
C...IF APPROPRIATE ADD HARMONIC ANALYSIS INFORMATION TO HOT START FILE
C...
            IF((IHARIND.EQ.1).AND.(IT.GT.ITHAS)) THEN
              WRITE(IHSFIL,REC=IHOTSTP+1) ICHA
              IHOTSTP = IHOTSTP + 1
              CALL HAHOUT(NP,NSTAE,NSTAV,NHASE,NHASV,NHAGE,NHAGV,
     &                                                  IHSFIL,IHOTSTP)
              IF(NHASE.EQ.1) CALL HAHOUTES(NSTAE,IHSFIL,IHOTSTP)
              IF(NHASV.EQ.1) CALL HAHOUTVS(NSTAV,IHSFIL,IHOTSTP)
              IF(NHAGE.EQ.1) CALL HAHOUTEG(NP,IHSFIL,IHOTSTP)
              IF(NHAGV.EQ.1) CALL HAHOUTVG(NP,IHSFIL,IHOTSTP)
              ENDIF

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...IF APPROPRIATE ADD MEAN AND VARIANCE INFORMATION TO HOT START FILE
CHARMV...
            IF((IHARIND.EQ.1).AND.(IT.GT.ITMV)) THEN
              IHOTSTP=IHOTSTP+1
              WRITE(IHSFIL,REC=IHOTSTP) NTSTEPS
              IF(NHAGE.EQ.1) THEN
                DO I=1,NP
                  WRITE(IHSFIL,REC=IHOTSTP+1) ELAV(I)
                  WRITE(IHSFIL,REC=IHOTSTP+2) ELVA(I)
                  IHOTSTP=IHOTSTP+2
                  END DO
                ENDIF
              IF(NHAGV.EQ.1) THEN
                DO I=1,NP
                  WRITE(IHSFIL,REC=IHOTSTP+1) XVELAV(I)
                  WRITE(IHSFIL,REC=IHOTSTP+2) YVELAV(I)
                  WRITE(IHSFIL,REC=IHOTSTP+3) XVELVA(I)
                  WRITE(IHSFIL,REC=IHOTSTP+4) YVELVA(I)
                  IHOTSTP=IHOTSTP+4
                  END DO
                ENDIF
              ENDIF
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

C...
C...CLOSE THE HOT START OUTPUT FILE
C...
            CLOSE(IHSFIL)
            IF(NSCREEN.EQ.1) WRITE(6,24541) IHSFIL,IT,TIME
            WRITE(16,24541) IHSFIL,IT,TIME
24541       FORMAT(1X,'HOT START OUTPUT WRITTEN TO UNIT ',I2,
     &                ' AT TIME STEP = ',I5,' TIME = ',E15.8)
            IF(IHSFIL.EQ.67) THEN
              IHSFIL=68
              ELSE
              IHSFIL=67
              ENDIF
            ENDIF
```

```
          ENDIF

C...
C...FIND AND PRINT TO UNIT 6, THE MAXIMUM ELEVATION, THE MAXIMUM VELOCITY
C....AND THE NODE NUMBERS AT WHICH THEY OCCUR.
C...
        IF(NSCREEN.EQ.1) THEN
          ELMAX=0.0
          VELMAX=0.0
          DO I=1,NP
            IF((ABS(NNODECODE(I)).EQ.1).AND.(ABS(ETA2(I)).GT.ELMAX))THEN
              ELMAX=ABS(ETA2(I))
              KEMAX=I
              ENDIF
            VELABS=UU2(I)*UU2(I)+VV2(I)*VV2(I)
            IF (VELABS.GT.VELMAX) THEN
              VELMAX=VELABS
              KVMAX=I
              ENDIF
            END DO
          VELMAX=VELMAX**0.5
          WRITE(6,1992) IT,NUMITR,ETA2(KEMAX),KEMAX,VELMAX,KVMAX
1992      FORMAT(1X,'TIME STEP =',I8,10X,'NUMBER OF ITERATIONS =',I5,
     &      /,4X,'ELMAX = ', E10.3,' AT NODE',I7,
     &        4X,'SPEEDMAX = ',E10.3,' AT NODE',I7)
          ENDIF

C...
C...****************** TIME STEPPING LOOP ENDS HERE ********************
C...
100     CONTINUE

C...
C...IF IHARIND=1 SOLVE THE HARMONIC ANALYSIS PROBLEM AND WRITE OUTPUT
C...
      IF((IHARIND.EQ.1).AND.(IT.GT.ITHAS)) THEN

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...ACCUMULATE VARIANCE AND MEAN OF RECORD AT NODES.
CHARMV...
        IF(FMV.NE.0.) THEN
          DO I=1,NP
            ELAV(I)   = ELAV(I)/NTSTEPS
            XVELAV(I) = XVELAV(I)/NTSTEPS
            YVELAV(I) = YVELAV(I)/NTSTEPS
            ELVA(I)   = ELVA(I)/NTSTEPS   - ELAV(I)*ELAV(I)
            XVELVA(I) = XVELVA(I)/NTSTEPS - XVELAV(I)*XVELAV(I)
            YVELVA(I) = YVELVA(I)/NTSTEPS - YVELAV(I)*YVELAV(I)
            END DO
          TIMEBEG=ITMV*DTDP + (STATIM-REFTIM)*86400.D0
          OPEN(55,FILE='fort.55')
          WRITE(55,*) NP
          ENDIF
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

C...
C......Fill out and decompose the LHS harmonic analaysis matrix
C...
        CALL FULSOL(0)
C...
C......Solve the harmonic analysis problem and write the output
C...
        IF(NHAGE.EQ.1) CALL LSQSOLEG(NP)
        IF(NHAGV.EQ.1) CALL LSQSOLVG(NP)
        IF(NHASE.EQ.1) CALL LSQSOLES(NSTAE)
        IF(NHASV.EQ.1) CALL LSQSOLVS(NSTAV)
```

```
        ENDIF

      STOP
      END



C*************************************************************************
C                                                                        *
C    Transform from lon,lat (lamda,phi) coordinates into CPP coordinates.*
C    Lon,Lat must be in radians.                                         *
C                                                                        *
C*************************************************************************

      SUBROUTINE CPP(X,Y,RLAMBDA,PHI,RLAMBDA0,PHI0)
      R=6378206.4
      X=R*(RLAMBDA-RLAMBDA0)*COS(PHI0)
      Y=PHI*R
      RETURN
      END


C*************************************************************************
C                                                                        *
C    Transform from CPP coordinates to lon,lat (lamda,phi) coordinates   *
C    Lon,Lat is in radians.                                              *
C                                                                        *
C*************************************************************************

      SUBROUTINE INVCP(XXCP,YYCP,RLAMBDA,PHI,RLAMBDA0,PHI0)
      R=6378206.4
      RLAMBDA=RLAMBDA0+XXCP/(R*COS(PHI0))
      PHI=YYCP/R
      RETURN
      END



C*************************************************************************
C                                                                        *
C      Subroutine to generate a neighbor table from a connectivity table.*
c                                                                        *
c      NOTE:the node itself is listed as neighbor #1                     *
c      NOTE:all other neighbors are sorted and placed in cw order from east*
c                                                                        *
c                      R.L.       4/26/95                               *
C*************************************************************************
C                                                                        *
C     -  PARAMETERS WHICH MUST BE SET TO CONTROL THE DIMENSIONING OF ARRAYS*
C           ARE AS FOLLOWS:                                            *
C                                                                        *
C          MNP = MAXIMUM NUMBER OF NODAL POINTS                        *
C          MNE = MAXIMUM NUMBER OF ELEMENTS                            *
C          MNEI= 1+MAXIMUM NUMBER OF NODES CONNECTED TO ANY ONE NODE IN THE*
C                   FINITE ELEMENT GRID                                *
C                                                                        *
C*************************************************************************
C                                                                        *
C    VARIABLE DEFINITIONS:                                               *
C       NE - NUMBER OF ELEMENTS                                          *
C       NP - NUMBER OF NODES                                             *
C       NM(MNE,3) - NODE NUMBERS ASSOCIATED WITH EACH ELEMENT            *
C       NNEIGH(MNP) NUMBER OF NEIGHBORS FOR EACH NODE                    *
C       NEIGH(MNP,NEIMAX) 2D ARRAY OF NEIGHBORS FOR EACH NODE            *
C       NEIMIN - 1+MINIMUM NUMBER OF NEIGHBORS FOR ANY NODE              *
C       NEIMAX - 1+MAXIMUM NUMBER OF NEIGHBORS FOR ANY NODE              *
```

```
C                                                                       *
C***********************************************************************
C
      SUBROUTINE NEIGHB(NE,NP,NNEIGH,NEIGH,NEIMIN,NEIMAX,NSCREEN)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=         1082)
      PARAMETER(MNE=         1894)
      PARAMETER(MNEI=           9)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      COMMON/HGRID/ X(MNP),Y(MNP),NM(MNE,3),DP(MNP),SFAC(MNP)
      DIMENSION NEIGH(MNP,MNEI),NNEIGH(MNP),NEITEM(MNEI),ANGLE(MNEI)
      INTEGER EN1,EN2,EN3

      RAD2DEG=45./ATAN(1.)

      DO 5 N=1,NP
         NNEIGH(N) = 0
         DO 5 NN=1,MNEI
            NEIGH(N,NN) = 0
   5        CONTINUE

      DO 10 N=1,NE
         EN1 = NM(N,1)
         EN2 = NM(N,2)
         EN3 = NM(N,3)
         DO 20 J=1,NNEIGH(EN1)
  20        IF(EN2.EQ.NEIGH(EN1,J)) GOTO 25
         NNEIGH(EN1)=NNEIGH(EN1)+1
         NNEIGH(EN2)=NNEIGH(EN2)+1
         IF((NNEIGH(EN1).GT.MNEI-1).OR.(NNEIGH(EN2).GT.MNEI-1)) GOTO 999
         NEIGH(EN1,NNEIGH(EN1))=EN2
         NEIGH(EN2,NNEIGH(EN2))=EN1
  25     DO 30 J=1,NNEIGH(EN1)
  30        IF(EN3.EQ.NEIGH(EN1,J)) GOTO 35
         NNEIGH(EN1)=NNEIGH(EN1)+1
         NNEIGH(EN3)=NNEIGH(EN3)+1
         IF((NNEIGH(EN1).GT.MNEI-1).OR.(NNEIGH(EN3).GT.MNEI-1)) GOTO 999
         NEIGH(EN1,NNEIGH(EN1))=EN3
         NEIGH(EN3,NNEIGH(EN3))=EN1
  35     DO 50 J=1,NNEIGH(EN2)
  50        IF(EN3.EQ.NEIGH(EN2,J)) GOTO 10
         NNEIGH(EN2)=NNEIGH(EN2)+1
         NNEIGH(EN3)=NNEIGH(EN3)+1
         IF((NNEIGH(EN2).GT.MNEI-1).OR.(NNEIGH(EN3).GT.MNEI-1)) GOTO 999
         NEIGH(EN2,NNEIGH(EN2))=EN3
         NEIGH(EN3,NNEIGH(EN3))=EN2
  10     CONTINUE
C
C  INSERT NODE ITSELF IN PLACE #1 and SORT other NEIGHBORS by increasing cw angle from E
C
      DO I=1,NP
         DO J=1,NNEIGH(I)
            NEITEM(J)=NEIGH(I,J)
            DELX=X(NEITEM(J))-X(I)
            DELY=Y(NEITEM(J))-Y(I)
            DIST=SQRT(DELX*DELX+DELY*DELY)
            IF(DIST.EQ.0) GOTO 998
            IF(DELY.NE.0) THEN
              ANGLE(J)=RAD2DEG*ACOS(DELX/DIST)
              IF(DELY.GT.0) ANGLE(J)=360.-ANGLE(J)
              ENDIF
            IF(DELY.EQ.0) THEN
              IF(DELX.GT.0) ANGLE(J)=0.
              IF(DELX.LT.0) ANGLE(J)=180.
```

```
               ENDIF
            END DO
         ANGLEMORE=-1.
         DO JJ=1,NNEIGH(I)
            ANGLELOW=400.
            DO J=1,NNEIGH(I)
               IF((ANGLE(J).LT.ANGLELOW).AND.(ANGLE(J).GT.ANGLEMORE))
     &                                                              THEN
                  ANGLELOW=ANGLE(J)
                  JLOW=J
                  ENDIF
               END DO
            NEIGH(I,JJ+1)=NEITEM(JLOW)
            ANGLEMORE=ANGLELOW
            END DO
         NEIGH(I,1)=I
         NNEIGH(I)=NNEIGH(I)+1
         END DO

C
C   DETERMINE THE MAXIMUM AND MINIMUM NUMBER OF NEIGHBORS
C
       NEIMAX = 0
       NEIMIN = 1000
       DO 60 N=1,NP
          IF(NNEIGH(N).LT.NEIMIN) NEIMIN=NNEIGH(N)
          IF(NNEIGH(N).GT.NEIMAX) NEIMAX=NNEIGH(N)
  60      CONTINUE
C
       RETURN

C      TERMINATE PROGRAM IF MAXIMUM NUMBER OF NEIGHBORS SET TOO SMALL

 999   CONTINUE
       IF(NSCREEN.EQ.1) WRITE(6,99311)
       WRITE(16,99311)
99311  FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &       //,1X,'THE DIMENSIONING PARAMETER MNEI IS TOO SMALL'
     &      /,1X,'USER MUST RE-DIMENSION PROGRAM',
     &      //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
       STOP

 998   CONTINUE
       IF(NSCREEN.EQ.1) WRITE(6,99312) I,NEITEM(J)
       WRITE(16,99312) I,NEITEM(J)
99312  FORMAT(////,1X,'!!!!!!!!!!  WARNING - FATAL ERROR !!!!!!!!!',
     &       //,1X,'NODES ',I7,' AND ',I7,' HAVE THE SAME COORDINATES'
     &      //,1X,'!!!!!! EXECUTION WILL NOW BE TERMINATED !!!!!!',//)
       STOP
       END



C***********************************************************************
C                                                                      *
C   THE FOLLOWING SUBROUTINES READ IN AND IN SOME CASES INTERPOLATE    *
C   ONTO THE ADCIRC GRID WIND AND PRESSURE FIELDS IN VARIOUS INPUT     *
C   FORMATS.                                                           *
C                                                                      *
C   ALL WIND SPEEDS ARE CONVERTED TO M/S AND ALL PRESSURES TO M OF H20 *
C   BEFORE THEY ARE RETURNED.                                          *
C                                                                      *
C***********************************************************************

C***********************************************************************
C                                                                      *
C   Convert time from year,month,day,hour,min,sec into seconds since   *
C   the beginning of the year.                                         *
C                                                                      *
```

```
C*****************************************************************

      SUBROUTINE TIMECONV(IYR,IMO,IDAY,IHR,IMIN,SEC,TIMESEC)
      REAL*8 TIMESEC
      TIMESEC = (IDAY-1)*86400 + IHR*3600 + IMIN*60 + SEC
      IF(IMO.GE.2)  TIMESEC = TIMESEC + 31*86400
      ILEAP = (IYR/4)*4
      IF((ILEAP.EQ.IYR).AND.(IMO.GE.3)) TIMESEC = TIMESEC + 29*86400
      IF((ILEAP.NE.IYR).AND.(IMO.GE.3)) TIMESEC = TIMESEC + 28*86400
      IF(IMO.GE.4)  TIMESEC = TIMESEC + 31*86400
      IF(IMO.GE.5)  TIMESEC = TIMESEC + 30*86400
      IF(IMO.GE.6)  TIMESEC = TIMESEC + 31*86400
      IF(IMO.GE.7)  TIMESEC = TIMESEC + 30*86400
      IF(IMO.GE.8)  TIMESEC = TIMESEC + 31*86400
      IF(IMO.GE.9)  TIMESEC = TIMESEC + 31*86400
      IF(IMO.GE.10) TIMESEC = TIMESEC + 30*86400
      IF(IMO.GE.11) TIMESEC = TIMESEC + 31*86400
      IF(IMO.EQ.12) TIMESEC = TIMESEC + 30*86400
      IF(IMO.GT.12) THEN
        WRITE(6,*) 'FATAL ERROR IN SUBROUTINE TIMECONV - MONTH > 12 '
        WRITE(16,*) 'FATAL ERROR IN SUBROUTINE TIMECONV - MONTH > 12 '
        STOP
        ENDIF
      RETURN
      END

C*****************************************************************
C                                                                *
C   READ IN AND INTERPOLATE ONTO THE ADCIRC GRID WIND FIELDS FROM U.S. *
C   NAVY FLEET NUMERIC WIND FILES.                               *
C                                                                *
C   NOTE: The ADCIRC grid information consists only of the Lon and Lat *
C   of the nodes.  THE LONS AND LATS MUST BE IN RADIANS!         *
C                                                                *
C                                                                *
C   MNWLAT = MAXIMUM NUMBER OF LATITUDES IN FLEET NUMERIC WIND FILE    *
C            SET = 1 IF FLEET NUMERIC WIND FILE NOT IN USE       *
C   MNWLON = MAXIMUM NUMBER OF LONGITUDES IN FLEET NUMERIC WIND FILE   *
C            SET = 1 IF FLEET NUMERIC WIND FILE NOT IN USE       *
C                                                                *
C                           R.L. 4/17/96                         *
C*****************************************************************

      SUBROUTINE NWS3GET(X,Y,SLAM,SFEA,WVNX,WVNY,IWTIME,IWYR,WTIMED,NP,
     &                   NWLON,NWLAT,WLATMAX,WLONMIN,WLATINC,WLONINC,ICS)

CUSER...
CUSER...SET PARAMETER STATEMENTS
CUSER....THE PARAMETERS HAVE TO BE RE-SET PRIOR TO EXECUTION TO INSURE
CUSER....SUFFICIENT SPACE HAS BEEN ALLOCATED FOR A SPECIFIC PROBLEM
CUSER....PARAMETER STATEMENTS MUST ALSO BE ADJUSTED IN THE SUBROUTINES
CUSER....AND FUNCTION STATEMENTS
CUSER...
      PARAMETER(MNWLAT=1)
      PARAMETER(MNWLON=1)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      REAL*8 WTIMED,RAD2DEG,PI
      DIMENSION WVXFN(MNWLAT,MNWLON),WVYFN(MNWLAT,MNWLON),
     &          PRN(MNWLAT,MNWLON)
      DIMENSION X(*),Y(*),SLAM(*),SFEA(*),WVNX(*),WVNY(*)

      COMMON /RAWMET/ WVXFN,WVYFN,PRN

      PI=3.141592653589793D0
      RAD2DEG = 180.D0/PI
      DEG2RAD = PI/180.D0
```

```
      READ(22,*) IWTIME
      IWYR = IWTIME/1000000
      IWMO = IWTIME/10000 - IWYR*100
      IWDAY = IWTIME/100 - IWYR*10000 - IWMO*100
      IWHR = IWTIME - IWYR*1000000 - IWMO*10000 - IWDAY*100
      CALL TIMECONV(IWYR,IWMO,IWDAY,IWHR,0,0.,WTIMED)

      DO I=1,NWLAT
        READ(22,*) (WVXFN(I,J),J=1,NWLON)
        END DO
      DO I=1,NWLAT
        READ(22,*) (WVYFN(I,J),J=1,NWLON)
        END DO

      DO I=1,NWLAT                                  !CONVERT TO X AND Y COMPONENTS
        DO J=1,NWLON
          WSPEED=WVXFN(I,J)
          WDIR=WVYFN(I,J)*DEG2RAD
          WVXFN(I,J)=-WSPEED*SIN(WDIR)
          WVYFN(I,J)=-WSPEED*COS(WDIR)
          END DO
        END DO

      DO I=1,NP                                     !INTERPOLATE TO ADCIRC GRID
        IF(ICS.EQ.2) THEN
          YCOOR=SFEA(I)*RAD2DEG
          XCOOR=SLAM(I)*RAD2DEG
          ENDIF
        IF(ICS.EQ.1) THEN
          YCOOR=Y(I)
          XCOOR=X(I)
          ENDIF
        LATIND2=(WLATMAX-YCOOR)/WLATINC + 1
        IF(LATIND2.EQ.NWLAT) LATIND2=LATIND2-1
        LATIND1=LATIND2 + 1
        LONIND1=(XCOOR-WLONMIN)/WLONINC + 1
        IF(LONIND1.EQ.NWLON) LONIND1=LONIND1-1
        LONIND2=LONIND1+1
        WLONM = WLONMIN + (LONIND1-1)*WLONINC
        WLATM = WLATMAX - (LATIND1-1)*WLATINC
        XWRATIO=(XCOOR-WLONM)/WLONINC
        YWRATIO=(YCOOR-WLATM)/WLATINC

        WVNX(I) = WVXFN(LATIND2,LONIND2)*XWRATIO*YWRATIO
     &            + WVXFN(LATIND2,LONIND1)*(1.-XWRATIO)*YWRATIO
     &            + WVXFN(LATIND1,LONIND2)*XWRATIO*(1.-YWRATIO)
     &            + WVXFN(LATIND1,LONIND1)*(1.-XWRATIO)*(1.-YWRATIO)
        WVNY(I) = WVYFN(LATIND2,LONIND2)*XWRATIO*YWRATIO
     &            + WVYFN(LATIND2,LONIND1)*(1.-XWRATIO)*YWRATIO
     &            + WVYFN(LATIND1,LONIND2)*XWRATIO*(1.-YWRATIO)
     &            + WVYFN(LATIND1,LONIND1)*(1.-XWRATIO)*(1.-YWRATIO)
        END DO
       RETURN
       END

C*****************************************************************
C                                                                *
C   Read onto the ADCIRC grid wind fields from the PBL-JAG model *
C                                                                *
C   Output from this subroutine is U,V (M/S) and P (M H20) on the *
C   ADCIRC grid.                                                 *
C                                                                *
C   The background pressure is assumed to be 1013 Mbars          *
C                                                                *
C                          R.L.11/06/96                          *
C*****************************************************************

      SUBROUTINE NWS4GET(WVNX,WVNY,PRN,NP,RHOWAT,G)

      DIMENSION WVNX(*),WVNY(*),PRN(*)
```

```
      CHARACTER*80 PBLJAGF

      RHOWATG=RHOWAT*G
      DO I=1,NP
        WVNX(I)=0.
        WVNY(I)=0.
        PRN(I)=101300./RHOWATG
        END DO
  170 READ(22,'(A80)') PBLJAGF
      IF(PBLJAGF(2:2).EQ.'#') GOTO 170
  171 READ(PBLJAGF,'(I8,5E13.5)') NHG,WVNX(NHG),WVNY(NHG),
     &                            PRN(NHG)
      WVNX(NHG)=WVNX(NHG)*1.04*0.5144  !CONVERT 30-MIN WINDS IN
      WVNY(NHG)=WVNY(NHG)*1.04*0.5144  !KNOTS TO 10-MIN WIND IN M/S
      PRN(NHG)=100.*PRN(NHG)/RHOWATG   !CONVERT MILLIBARS TO M OF WATER
      READ(22,'(A80)') PBLJAGF
      IF(PBLJAGF(2:2).NE.'#') GOTO 171
      RETURN
      END


C*********************************************************************
C                                                                    *
C   Read in and interpolate onto the ADCIRC grid wind fields from U.S. *
C   National Weather Service AVN model SFLUX meteorological files.     *
C                                                                    *
C   The input files are in binary and have been created by the GRIB    *
C   unpacking program unpkgrb1.f to extract only the U 10M, V 10M, and *
C   surface P fields.                                                *
C                                                                    *
C   The SFLUX files utilize a global Gaussian Lon/Lat grid which is    *
C   constructed in these subroutines.                                *
C                                                                    *
C   NOTE: The ADCIRC grid information consists only of the Lon and Lat *
C   of the nodes.  THE LONS AND LATS MUST BE IN RADIANS!             *
C                                                                    *
C   Output from this subroutine is U,V (M/S) and P (M H20) on the      *
C   ADCIRC grid.                                                     *
C                                                                    *
C   MNWLAT = LATB = 190    FOR GAUSSIAN GRID                         *
C   MNWLON = LONB = 384    FOR GAUSSIAN GRID                         *
C                                                                    *
C                           R.L. 4/18/96                             *
C*********************************************************************

      SUBROUTINE NWS10GET(NWSGGWI,FLON,FLAT,ULL,VLL,PLL,NP,RHOWAT,G,
     &                    LONB,LATB)

CUSER...
CUSER...SET PARAMETER STATEMENTS
CUSER....THE PARAMETERS HAVE TO BE RE-SET PRIOR TO EXECUTION TO INSURE
CUSER....SUFFICIENT SPACE HAS BEEN ALLOCATED FOR A SPECIFIC PROBLEM
CUSER....PARAMETER STATEMENTS MUST ALSO BE ADJUSTED IN THE SUBROUTINES
CUSER....AND FUNCTION STATEMENTS
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNWP=1082)
      PARAMETER(MNWLAT=1)
      PARAMETER(MNWLON=1)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      DIMENSION N00(MNWP),N10(MNWP),N11(MNWP),N01(MNWP),
     &          D00(MNWP),D10(MNWP),D11(MNWP),D01(MNWP)
      DIMENSION UG(MNWLAT*MNWLON),VG(MNWLAT*MNWLON),PG(MNWLAT*MNWLON)
      DIMENSION COLRAB(MNWLAT),DUMMY(MNWLAT),GCLAT(MNWLAT),GCLON(MNWLON)
      DIMENSION FLAT(*),FLON(*)
      DIMENSION ULL(*),VLL(*),PLL(*)
      INTEGER*4 KGDS(200)
```

```
      CHARACTER*1 PDS(50),FNAME2(8)
      CHARACTER*8 FNAME1
      EQUIVALENCE (FNAME1,FNAME2)
      LOGICAL FOUND

      COMMON /RAWMET/ UG,VG,PG

      PI=ACOS(-1.)
      TWOPI=PI*2.D0
      HFPI=PI/2.D0
      DEG2RAD=PI/180.D0
      RHOWATG=RHOWAT*G

C...Figure out the GRIB data file name

      FNAME1='fort.   '
      IEXT=200 + NWSGGWI*6
      IDIG1=IEXT/100
      IDIG2=(IEXT-100*IDIG1)/10
      IDIG3=(IEXT-100*IDIG1-10*IDIG2)
      FNAME2(6)=CHAR(IDIG1+48)
      FNAME2(7)=CHAR(IDIG2+48)
      FNAME2(8)=CHAR(IDIG3+48)


C...Enter, locate and open the GRIB format data file

 1010 FORMAT(' File ',A8,' WAS NOT FOUND!  FATAL ERROR',/)
 1011 FORMAT(' File ',A8,' WAS FOUND!  Opening & Processing file',/)

      WRITE(*,*) '  '
      INQUIRE(FILE=FNAME1,EXIST=FOUND)
      IF(FOUND) GOTO 32
      WRITE(*,1010) FNAME1
      WRITE(16,1010) FNAME1
      STOP
   32 WRITE(*,1011) FNAME1
      OPEN(IEXT,FILE=FNAME1,status='old',access='sequential',
     &     form='unformatted',iostat=kerr)

C...Read the GRIB data file

      READ(IEXT,END=1100) LENPDS,LENKGDS,NWORDS
      IF(LENPDS.GT.0) READ(IEXT,END=1100) (pds(j),j=1,lenpds)
      IF(LENKGDS.GT.0) READ(IEXT,END=1100) (kgds(j),j=1,lenkgds)
      IF(NWORDS.GT.0) READ(IEXT,END=1100) (UG(J),J=1,NWORDS)

      READ(IEXT,END=1100) LENPDS,LENKGDS,NWORDS
      IF(LENPDS.GT.0) READ(IEXT,END=1100) (pds(j),j=1,lenpds)
      IF(LENKGDS.GT.0) READ(IEXT,END=1100) (kgds(j),j=1,lenkgds)
      IF(NWORDS.GT.0) READ(IEXT,END=1100) (VG(J),J=1,NWORDS)

      READ(IEXT,END=1100) LENPDS,LENKGDS,NWORDS
      IF(LENPDS.GT.0) READ(IEXT,END=1100) (pds(j),j=1,lenpds)
      IF(LENKGDS.GT.0) READ(IEXT,END=1100) (kgds(j),j=1,lenkgds)
      IF(NWORDS.GT.0) READ(IEXT,END=1100) (PG(J),J=1,NWORDS)

 1100 CLOSE(IEXT)


C...The first time the subroutine is called, setup the Gaussian grid and
C...determine the interpolating factors for the ADCIRC grid.

      IF(NWSGGWI.EQ.0) THEN
        CALL GLATS(LATB/2,COLRAB,DUMMY,DUMMY,DUMMY)
        DO J=1,LATB/2
          GCLAT(J)=COLRAB(J)
          JJ=LATB-J+1
          GCLAT(JJ)=PI-COLRAB(J)
          ENDDO
```

```
        GDLON=TWOPI/LONB
        DO J=1,LONB
          GCLON(J)=GDLON*(J-1)
          END DO
        CALL G2RINI(GCLON,GCLAT,FLON,FLAT,N00,N10,N11,N01,D00,D10,D11,
     &                                                    D01,NP,LONB,LATB)
        ENDIF

C.....Do from the Gaussian grid to the ADCIRC grid
C.....Convert pressure from N/M^2 to M of H20

      DO N=1,NP
        P1=PG(N00(N))
        P2=PG(N10(N))
        P3=PG(N11(N))
        P4=PG(N01(N))
        U1=UG(N00(N))
        U2=UG(N10(N))
        U3=UG(N11(N))
        U4=UG(N01(N))
        V1=VG(N00(N))
        V2=VG(N10(N))
        V3=VG(N11(N))
        V4=VG(N01(N))
        PLL(N)=P1*D00(N)+P2*D10(N)+P3*D11(N)+P4*D01(N)
        ULL(N)=U1*D00(N)+U2*D10(N)+U3*D11(N)+U4*D01(N)
        VLL(N)=V1*D00(N)+V2*D10(N)+V3*D11(N)+V4*D01(N)
        PLL(N)=PLL(N)/RHOWATG
        END DO

      RETURN
      end


C*********************************************************************
C  Subroutine to compute the latutudes in a Global Gaussian Lat/Lon    *
C  grid with T126 resolution (GRIB Grid type 126).                     *
C                                                                      *
C       modified from the original GLATS by R.L. 4/24/96               *
C*********************************************************************

      SUBROUTINE GLATS(LGGHAF,COLRAD,WGT,WGTCS,RCS2)
CCCC       %INCLUDE DBGLATS ;
CCCC  HALF PRECISION COLRAD,WGT,WGTCS,RCS2
C     DOUBLE PRECISION COLRAD,WGT,WGTCS,RCS2
      DIMENSION COLRAD(*),WGT(*),WGTCS(*),RCS2(*)
      EPS=1.E-6
C     EPS=1.E-12
C     PRINT 101
C101  FORMAT ('0 I   COLAT   COLRAD     WGT', 12X, 'WGTCS',
CCCC 1 10X, 'ITER  RES')
      SI = 1.0
      L2=2*LGGHAF
      RL2=L2
      SCALE = 2.0/(RL2*RL2)
      K1=L2-1
      PI = ACOS(-1.)
      DRADZ = PI / 360.
      RAD = 0.0
      DO 1000 K=1,LGGHAF
      ITER=0
      DRAD=DRADZ
1     CALL POLY(L2,RAD,P2)
2     P1 =P2
      ITER=ITER+1
      RAD=RAD+DRAD
      CALL POLY(L2,RAD,P2)
      IF(SIGN(SI,P1).EQ.SIGN(SI,P2)) GO TO 2
      IF(DRAD.LT.EPS)GO TO 3
      RAD=RAD-DRAD
```

```
      DRAD = DRAD * 0.25
      GO TO 1
3     CONTINUE
      COLRAD(K)=RAD
      PHI = RAD * 180 / PI
      CALL POLY(K1,RAD,P1)
      X = COS(RAD)
      W = SCALE * (1.0 - X*X)/ (P1*P1)
      WGT(K) = W
      SN = SIN(RAD)
      W=W/(SN*SN)
      WGTCS(K) = W
      RC=1./(SN*SN)
      RCS2(K) = RC
      CALL POLY(L2,RAD,P1)
C     PRINT 102,K,PHI,COLRAD(K),WGT(K),WGTCS(K),ITER,P1
C102  FORMAT(1H ,I2,2X,F6.2,2X,F10.7,2X,E13.7,2X,E13.7,2X,I4,2X,D13.7)
1000  CONTINUE
c     PRINT 100,LGGHAF
c100  FORMAT(1H ,'SHALOM FROM 0.0 GLATS FOR ',I3)
      RETURN
      END


C*********************************************************************
C  Subroutine used by GLATS.                                         *
C*********************************************************************

CFPP$ EXPAND(POLY)
      SUBROUTINE POLY(N,RAD,P)
CCCC       %INCLUDE DBPOLY ;
      X = COS(RAD)
      Y1 = 1.0
      Y2=X
      DO 1 I=2,N
      G=X*Y2
      Y3=G-Y1+G-(G-Y1)/FLOAT(I)
      Y1=Y2
      Y2=Y3
1     CONTINUE
      P=Y3
      RETURN
      END


C*********************************************************************
C  Subroutine to compute the factors to interpolate from a global     *
C  Gaussian Lat/Lon grid with T126 resolution (GRIB Grid type 126)    *
C  onto another grid.                                                 *
C                                                                     *
C  The new grid is a series of longitude and latitude points contained *
C  in the FLON and FLAT arrays with a total number of points NP       *
C                                                                     *
C       modified from the original G2RINI by R.L. 4/17/96             *
C*********************************************************************

      SUBROUTINE G2RINI(GCLON,GCLAT,FLON,FLAT,N00,N10,N11,N01,D00,D10,
     &                                        D11,D01,NP,LONB,LATB)
      DIMENSION GCLAT(*),GCLON(*)
      DIMENSION FLAT(*),FLON(*)
      DIMENSION N00(*),N10(*),N11(*),N01(*),D00(*),D10(*),D11(*),D01(*)

      SAVE ICALL
      DATA ICALL/0/

      IF( ICALL .EQ. 0 ) THEN
        ICALL = 1
c       PRINT 1234
c1234   FORMAT(' = IN ROUTINE G2RINI FOR HORIZONTAL INTERPOLATION = ')
        PI=ACOS(-1.)
```

```
        HFPI=PI/2.0
        TWOPI=PI*2.0

C...Compute estimated DLAT, true DLON for Gaussian grid

        NLAT=LATB
        NLON=LONB
        DLAT=PI/FLOAT(NLAT-1)
        DLON=TWOPI/FLOAT(NLON)
        N=0

C...Loop through all the nodes in the grid to be interpolated onto and
C.....compute the interpolating factors.

        DO I=1,NP

C.....Compute initial guess of which lon value FLON(I) is in the Gaussian file
C.......Check that this value is reasonable.

          FLONWORK=FLON(I)
          IF(FLONWORK.LT.0.) FLONWORK=FLONWORK+TWOPI
          LON=FLONWORK/DLON + 1
          LONP1=LON+1
          IF(LON.EQ.NLON) LONP1=1         !Circle condition
          IF((LON.LT.1).OR.(LON.GT.NLON)) THEN
            PRINT *,' ***** ERROR IN LON ****'
            PRINT *,' I ',I
            PRINT *,' LON ',LON
            PRINT *,' DLON ',DLON
            PRINT *,' FLON ',FLON(I)
            STOP
            endif

C.....Compute initial guess of which lat value FLAT(I) is in the Gaussian file
C.......Check that this value is reasonable.

          COLAT=HFPI-FLAT(I)
          LAT=COLAT/DLAT + 1
          IF(LAT.EQ.NLAT) LAT=LAT-1
          LATP1=LAT+1
          IF((LAT.LT.1).OR.(LAT.GT.NLAT)) THEN
            PRINT *,' ***** ERROR IN LAT ****'
            PRINT *,' I ',I
            PRINT *,' LAT ',LAT
            PRINT *,' DLAT ',DLAT
            PRINT *,' FLAT ',FLAT(I)
            STOP
          ENDIF

 5        CONTINUE
          IF((COLAT.GE.GCLAT(LAT)).AND.(COLAT.LE.GCLAT(LATP1))) GO TO 9
          IF(COLAT.LT.GCLAT(LAT)) THEN
            LATP1=LAT
            LAT=LAT-1
            IF(LAT.LE.0) THEN
              LAT=1
              LATP1=2
              GOTO 9
              ENDIF
            GOTO 5
            ENDIF
          IF(COLAT.GT.GCLAT(LATP1)) THEN
            LAT=LAT+1
            LATP1=LAT+1
            IF(LAT.GE.NLAT ) THEN
              LAT=NLAT-1
              LATP1=NLAT
              GOTO 9
              ENDIF
            GOTO 5
```

```
            ENDIF

 9          CONTINUE
            DDLAT=GCLAT(LATP1)-GCLAT(LAT)
            XLAT=GCLAT(LAT)
            DFLAT1=(COLAT-XLAT)/DDLAT
            IF(LAT.EQ.1) DFLAT1=MAX(0.,DFLAT1)          !MODIFY THIS FOR POLAR POINTS
            IF(LATP1.EQ.NLAT) DFLAT1=MIN(1.,DFLAT1)     !MODIFY THIS FOR POLAR POINTS
            DFLAT=1.-DFLAT1
            DDLON=DLON
            XLON=GCLON(LON)
            DFLON1=(FLONWORK-XLON)/DDLON
            DFLON=1.-DFLON1
            N=N+1
            D00(N)=DFLON*DFLAT
            D10(N)=DFLON1*DFLAT
            D11(N)=DFLON1*DFLAT1
            D01(N)=DFLON*DFLAT1
            N00(N)=LON+(LAT-1)*NLON
            N10(N)=LONP1+(LAT-1)*NLON
            N11(N)=LONP1+(LATP1-1)*NLON
            N01(N)=LON+(LATP1-1)*NLON

            END DO

c         WRITE(*,*) ' D00 TO D11 SHOULD BE ALL POSITIVE.'

          ELSE
c         WRITE(*,*) ' G2RINI ALREADY CALLED '
          ENDIF

        RETURN
        END

C*******************************************************************************
C                                                                              *
C   End of subroutines to read wind and pressure fields.                       *
C                                                                              *
C*******************************************************************************



c*******************************************************************************
c                                                                              *
c   HA_SUBS.FOR     V3.01        11/9/95                                       *
c                                                                              *
c   Least Square harmonic analysis of timeseries from ADCIRC2DDI_v27           *
c                                                                              *
c    Notes:                                                                    *
c    1.)  Both the left hand side matrix and the right hand side               *
c         forcing vectors are continuously updated in time.  This              *
c         eliminates the need to store time series outputs for later           *
c         harmonic analysis.                                                   *
c    2.)  The left hand side matrix and the right hand side forcing            *
c         vectors are output in the hotstart file and can be used to           *
c         perform harmonic analysis on an incomplete run.                      *
c    3.)  Frequencies should be in rad/sec,times should be in sec.             *
c                                                                              *
c*******************************************************************************
c                                                                              *
c    Program Written by:                                                       *
c          R.A. Luettich, IMS UNC                                              *
c          J.J. Westerink, CE ND                                               *
c                                                                              *
c    Program Development History:                                              *
c    1.) lsq_stations_v004 by JJW                                              *
c    2.) LSQEX by RL used in 2D binary extr program                            *
c    3.) LSQRL by RL used in 1D test codes                                     *
c    4.) LSQ2D v1.00-v2.26 by RL real time Harmonic Analysis for ADCIRC*
c    5.) HA_SUBS v3.01 by RL real time HA for ADCIRC separate                   *
```

```
c        subroutines for elevation station, velocity station,          *
c        global elevation and global velocity harmonic analysis        *
c                                                                      *
c***********************************************************************
c                                                                      *
c SUBROUTINE LSQUPDLHS updates the LHS matrix                          *
c SUBROUTINE LSQUPDES updates the RHS load vector for elev stations    *
c SUBROUTINE LSQUPDVS updates the RHS load vector for velocity stations*
c SUBROUTINE LSQUPDEG updates the RHS load vector for elevation global *
c SUBROUTINE LSQUPDVG updates the RHS load vector for velocity global  *
c SUBROUTINE FULSOL fills out, decomposes and solves the matricies     *
c SUBROUTINE LSQSOLES solves & writes output for elevation stations    *
c SUBROUTINE LSQSOLVS solves & writes output for velocity stations     *
c SUBROUTINE LSQSOLEG solves & writes output for elevation global      *
c SUBROUTINE LSQSOLVG solves & writes output for velocity global       *
c SUBROUTINE HAHOUT writes HA parameters & LHS matrix to hotstart file *
c SUBROUTINE HAHOUTES writes elev sta RHS load vector to hotstart file *
c SUBROUTINE HAHOUTVS writes vel sta RHS load vector to hotstart file  *
c SUBROUTINE HAHOUTEG writes glob elev RHS load vector to hotstart file*
c SUBROUTINE HAHOUTVG writes glob vel RHS load vector to hotstart file *
c SUBROUTINE HACOLDS initializes HA param & LHS matrix for cold start  *
c SUBROUTINE HACOLDSES initializes elev sta RHS load vec for cold start*
c SUBROUTINE HACOLDSVS initializes vel sta RHS load vec for cold start *
c SUBROUTINE HACOLDSEG initializes glob ele RHS load vec for cold start*
c SUBROUTINE HACOLDSVG initializes glob vel RHS load vec for cold start*
c SUBROUTINE HAHOTS initializes HA params & LHS matrix for a hot start *
c SUBROUTINE HAHOTSES initializes elev sta RHS load vec for hot start  *
c SUBROUTINE HAHOTSVS initializes vel sta RHS load vec for hot start   *
c SUBROUTINE HAHOTSEG initializes glob elev RHS load vec for hot start *
c SUBROUTINE HAHOTSVG initializes glob vel RHS load vec for hot start  *
c                                                                      *
c***********************************************************************
c                                                                      *
c    INPUT FILES:                                                      *
c      - Frequency information is read in by ADCIRC from unit 15.      *
c        This information is passed in common block LSQFREQS.          *
c                                                                      *
c      - If the model is hot start, input is read from UNIT 67 or 68   *
c                                                                      *
c    OUTPUT FILES:                                                     *
C      UNIT 51 : HARMONIC CONSTITUENT ELEVATION VALUES AT SPECIFIED    *
C                ELEVATION RECORDING STATION COORDINATES (ASCII)       *
C      UNIT 52 : HARMONIC CONSTITUENT VELOCITY VALUES AT SPECIFIED     *
C                VELOCITY RECORDING STATION COORDINATES  (ASCII)       *
C      UNIT 53 : HARMONIC CONSTITUENT ELEVATIONS AT ALL NODES (ASCII)  *
C      UNIT 54 : HARMONIC CONSTITUENT VELOCITIES AT ALL NODES (ASCII)  *
C      UNIT 55 : COMPARISON BETWEEN THE MEAN AND VARIANCE OF THE TIME  *
C                SERIES GENERATED BY THE MODEL AND THE MEAN AND        *
C                VARIANCE OF A TIME SERIES RESYNTHESIZED FROM THE      *
C                COMPUTED HARMONIC CONSTITUENTS.  THIS GIVES AN        *
C                INDICATION OF HOW COMPLETE THE HARMONIC ANALYSIS      *
C                WAS. (ASCII)                                          *
C      UNIT 67 or 68 : HOT START FILES (BINARY)                        *
c                                                                      *
c***********************************************************************

c***********************************************************************
c   Subroutine to update the Left Hand Side Matrix                     *
c                                                                      *
c  TIME  - ABSOLUTE MODEL TIME (SEC)                                   *
c  IT    - MODEL TIME STEP                                             *
c  icall - number of times the subroutine has been called              *
c  a     - Left Hand Side Matrix                                       *
c                                                                      *
c                        RL 11/7/95                                    *
c***********************************************************************
c
      SUBROUTINE LSQUPDLHS(TIME,IT)

CUSER...
```

```
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/matrix/ A,P,X
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)

      icall = icall + 1

c
c***** Update the Left Hand Side Matrix
c      Note: this is a symmetric matrix and therefore only store
c            the upper triangular part.  The lower part will be
c            filled out in SUBROUTINE FULSOL prior to the matrix's decomposition

c   Take care of the steady constituent if included in the analysis

      if(nf.eq.1) then
         a(1,1)=icall
         do j=1,nfreq
            tf1=freq(j+nf)*time
            a(1,2*j)   = a(1,2*j) + cos(tf1)
            a(1,2*j+1) = a(1,2*j+1) + sin(tf1)
            end do
         endif

c   Take care of the other constituents

      do i=1,nfreq
         do j=i,nfreq
            i1=2*i-(1-nf)
            i2=i1+1
            j1=2*j-(1-nf)
            j2=j1+1
            tf1=freq(i+nf)*time
            tf2=freq(j+nf)*time
            a(i1,j1) = a(i1,j1) + cos(tf1)*cos(tf2)
            a(i1,j2) = a(i1,j2) + cos(tf1)*sin(tf2)
            a(i2,j2) = a(i2,j2) + sin(tf1)*sin(tf2)
            if(i2.le.j1) a(i2,j1) = a(i2,j1) + sin(tf1)*cos(tf2)
            end do
         end do

c   Record update time and time step

      TIMEUD = TIME
      ITUD = IT

      return
      end


c**********************************************************************
c   Subroutine to update the Right Hand Side Load Vectors for the      *
c   elevation station harmonic analysis.                               *
c                                                                      *
c  STAE  - STATION ELEVATION VALUES USED TO UPDATE LOAD VECTORS        *
c  NSTAE - NUMBER OF TIDAL ELEVATION RECORDING STATIONS                *
c                                                                      *
c  STAELV - station elevation load vector                              *
c                                                                      *
c                        RL 11/8/95                                    *
```

```
c**********************************************************************
c
      SUBROUTINE LSQUPDES(STAE,NSTAE)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAE=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveces/ STAELV
      REAL STAE(MNSTAE)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION STAELV(2*MNHARF,MNSTAE)
      CHARACTER*10 NAMEFR(MNHARF)

c
c***** Update the Right Hand Side Load Vectors
c
c   Take care of the steady constituent if included in the analysis

      if(nz.eq.0) then
         do n=1,NSTAE
            STAELV(1,N) = STAELV(1,N) + STAE(N)
            end do
         endif

c   Take care of the other constituents

      do i=1,nfreq
         i1=2*i-nz
         i2=i1+1
         tf1=freq(i+nf)*TIMEUD
         ctf1 = cos(tf1)
         stf1 = sin(tf1)
         do n=1,NSTAE
            STAELV(I1,N) = STAELV(I1,N) + STAE(N)*CTF1
            STAELV(I2,N) = STAELV(I2,N) + STAE(N)*STF1
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to update the Right Hand Side Load Vectors for the      *
c   velocity station harmonic analysis.                                *
c                                                                      *
c  STAU  - STATION U VELOCITY VALUES USED TO UPDATE LOAD VECTORS       *
c  STAV  - STATION V VELOCITY VALUES USED TO UPDATE LOAD VECTORS       *
c  NSTAV - NUMBER OF TIDAL CURRENT RECORDING STATIONS                  *
c                                                                      *
c  STAULV - station u velocity load vector                             *
c  STAVLV - station v velocity load vector                             *
c                                                                      *
c                         RL 11/8/95                                   *
c**********************************************************************
c
      SUBROUTINE LSQUPDVS(STAU,STAV,NSTAV)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
```

```
      PARAMETER(MNSTAV=           1)
      PARAMETER(MNHARF=           4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvs/ STAULV,STAVLV
      REAL STAU(MNSTAV),STAV(MNSTAV)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION STAULV(2*MNHARF,MNSTAV),STAVLV(2*MNHARF,MNSTAV)
      CHARACTER*10 NAMEFR(MNHARF)

c
c***** Update the Right Hand Side Load Vectors
c
c   Take care of the steady constituent if included in the analysis

      if(nz.eq.0) then
         do n=1,NSTAV
            STAULV(1,N) = STAULV(1,N) + STAU(N)
            STAVLV(1,N) = STAVLV(1,N) + STAV(N)
            end do
         endif

c   Take care of the other constituents

      do i=1,nfreq
         i1=2*i-nz
         i2=i1+1
         tf1=freq(i+nf)*TIMEUD
         ctf1 = cos(tf1)
         stf1 = sin(tf1)
         do n=1,NSTAV
            STAULV(I1,N) = STAULV(I1,N) + STAU(N)*CTF1
            STAVLV(I1,N) = STAVLV(I1,N) + STAV(N)*CTF1
            STAULV(I2,N) = STAULV(I2,N) + STAU(N)*STF1
            STAVLV(I2,N) = STAVLV(I2,N) + STAV(N)*STF1
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to update the Right Hand Side Load Vectors for the      *
c   global elevation harmonic analysis.                                *
c                                                                      *
c  GLOE  - GLOBAL ELEVATION VALUES USED TO UPDATE LOAD VECTORS         *
c  NP    - NUMBER OF POINTS IN GLOBAL GRID                            *
c                                                                      *
c  GLOELV - global elevation load vector                               *
c                                                                      *
c                         RL 11/8/95                                   *
c**********************************************************************
c
      SUBROUTINE LSQUPDEG(GLOE,NP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveceg/ GLOELV
      REAL GLOE(MNP)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION GLOELV(2*MNHARF,MNP)
      CHARACTER*10 NAMEFR(MNHARF)

c
c***** Update the Right Hand Side Load Vectors
c
c   Take care of the steady constituent if included in the analysis

      if(nz.eq.0) then
         do n=1,np
            GLOELV(1,N)=GLOELV(1,N)+GLOE(N)
            end do
         endif

c   Take care of the other constituents

      do i=1,nfreq
         i1=2*i-nz
         i2=i1+1
         tf1=freq(i+nf)*TIMEUD
         ctf1 = cos(tf1)
         stf1 = sin(tf1)
         do n=1,np
            GLOELV(I1,N)=GLOELV(I1,N)+GLOE(N)*CTF1
            GLOELV(I2,N)=GLOELV(I2,N)+GLOE(N)*STF1
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to update the Right Hand Side Load Vectors for the      *
c   global velocity harmonic analysis.                                 *
c                                                                      *
c  GLOU  - GLOBAL U VELOCITY VALUES USED TO UPDATE LOAD VECTORS        *
c  GLOV  - GLOBAL V VELOCITY VALUES USED TO UPDATE LOAD VECTORS        *
c  NP    - NUMBER OF POINTS IN GLOBAL GRID                            *
c                                                                      *
c  GLOULV - global u velocity load vector                              *
c  GLOVLV - global v velocity load vector                              *
c                                                                      *
c                        RL 11/8/95                                    *
c**********************************************************************
c
      SUBROUTINE LSQUPDVG(GLOU,GLOV,NP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvg/ GLOULV,GLOVLV
      REAL GLOU(MNP),GLOV(MNP)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION GLOULV(2*MNHARF,MNP),GLOVLV(2*MNHARF,MNP)
```

```
      CHARACTER*10 NAMEFR(MNHARF)

c
c***** Update the Right Hand Side Load Vectors
c
c   Take care of the steady constituent if included in the analysis

      if(nz.eq.0) then
         do n=1,np
            GLOULV(1,N) = GLOULV(1,N) + GLOU(N)
            GLOVLV(1,N) = GLOVLV(1,N) + GLOV(N)
            end do
         endif

c   Take care of the other constituents

      do i=1,nfreq
         i1=2*i-nz
         i2=i1+1
         tf1=freq(i+nf)*TIMEUD
         ctf1 = cos(tf1)
         stf1 = sin(tf1)
         do n=1,np
            GLOULV(I1,N) = GLOULV(I1,N) + GLOU(N)*CTF1
            GLOVLV(I1,N) = GLOVLV(I1,N) + GLOV(N)*CTF1
            GLOULV(I2,N) = GLOULV(I2,N) + GLOU(N)*STF1
            GLOVLV(I2,N) = GLOVLV(I2,N) + GLOV(N)*STF1
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to fill out, decompose and solve the lsq system         *
c   Solves system a*x=b by l*d*l(tr) decomp in full storage mode        *
c                                                                      *
c   NOTE: This routine has been modified so that the filling out and   *
c         decomposition (and only those operations) are done if        *
c         idecom=0.                                                    *
c                                                                      *
c   ndm -  actual dimension of a matrix                                *
c                                                                      *
c                          rl 11/7/95                                  *
c**********************************************************************
c
        subroutine fulsol(idecom)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,ndm,ITUD,ICALL
      common/matrix/ a,b,x
      dimension a(2*MNHARF,2*MNHARF),b(2*MNHARF),x(2*MNHARF)
      dimension c(2*MNHARF),y(2*MNHARF)

c
c**** If only want to fill out matrix and decomponse
c
      if(idecom.eq.0) then

c        Set up the lower triangular part of the LHS a matrix
```

```
         do j=1,ndm
            do i=j,ndm
               a(i,j)=a(j,i)
               end do
            end do

c        Decomposition of matrix a

         do 100 ir=1,ndm
            ire=ir+1
            do 20 j=ire,ndm
   20          a(ir,j)=a(ir,j)/a(ir,ir)
            if(ire.gt.ndm) goto 100
            do 40 j=ire,ndm
               do 40 k=ire,ndm
   40             a(k,j)=a(k,j)-a(k,ir)*a(ir,j)
            do 50 j=ire,ndm
   50          a(j,ir)=0.0
  100       continue
         return
         endif

c...     solve for y by forward substitution for l*y=b

         do 120 ir=1,ndm
            y(ir)=b(ir)
            do 110 jr=1,ir-1
  110       y(ir)=y(ir)-a(jr,ir)*y(jr)
  120    continue

c...     calculate c=d**(-1)*y

         do 130 ir=1,ndm
  130    c(ir)=y(ir)/a(ir,ir)

c...     solve for x by back-substituting for l(tr)*x=c

         ir=ndm
  140    continue
         x(ir)=c(ir)
         do 150 jr=ir+1,ndm
  150    x(ir)=x(ir)-a(ir,jr)*x(jr)
         ir=ir-1
         if(ir.ge.1) goto 140
         return
         end


c**********************************************************************
c   Subroutine to solve the system and write output for elevation      *
c   stations.                                                          *
c                                                                      *
c   nf=0  if no steady constituent                                     *
c   nf=1 if steady constituent                                         *
c                                                                      *
c                         R.L. 11/8/95                                 *
c**********************************************************************
c
         SUBROUTINE LSQSOLES(NSTAE)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
       PARAMETER(MNSTAE=1)
       PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

       IMPLICIT REAL*8 (A-H,O-Z)
```

```
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveces/ STAELV
      common/matrix/ A,P,X
      DIMENSION STAELV(2*MNHARF,MNSTAE)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)

      pi=3.141592653589793d0
      convrd=180.d0/pi

c
c**** Open elevation station harmonic output file and write header information
c
      open(51,file='fort.51')
      write(51,*) nfreq+nf
      do j=1,nfreq+nf
         write(51,3679) freq(j),FF(j),FACE(j),namefr(j)
         end do
3679  format(1x,e20.10,1x,f10.7,1x,f12.8,1x,a10)
      write(51,*) NSTAE

c
c**** AT each STATION TRANSFER each load vector to p and solve the system
c
      DO N=1,NSTAE
         do k=1,mm
           p(k)=STAELV(k,n)
           end do
         call fulsol(n)

c        Compute amplitude and phase for each frequency making sure that the
c        phase is between 0 and 360 deg.  Then write output.

         write(51,*) N
         do i=1,nfreq+nf
            if((nf.eq.1).and.(i.eq.1)) then
               emag=x(i)/ff(i)
               phasee=0.
               else
               i1=2*i-1-nf
               i2=i1+1
               emag=sqrt(x(i1)*x(i1)+x(i2)*x(i2))/ff(i)
               if((x(i1).eq.0.).and.(x(i2).eq.0.)) then
                  phasee=0.
                  else
                  phasee = atan2(x(i2),x(i1))
                  endif
               endif
            phasede=convrd*phasee+face(i)
            if(phasede.lt.0.) phasede=phasede+360.
            if(phasede.ge.360.) phasede=phasede-360.

            write(51,6635) emag,phasede
 6635       format(2x,e16.8,1x,f11.4)
            end do

         end do

      return
      end


c**********************************************************************
c   Subroutine to solve the system and write output for velocity       *
c   stations.                                                          *
c                                                                      *
c   nf=0  if no steady constituent                                     *
c   nf=1  if steady constituent                                        *
```

```
c                                                                       *
c                          R.L. 11/8/95                                 *
c***********************************************************************
c
        SUBROUTINE LSQSOLVS(NSTAV)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAV=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvs/ STAULV,STAVLV
      common/matrix/ A,P,X
      DIMENSION STAULV(2*MNHARF,MNSTAV),STAVLV(2*MNHARF,MNSTAV)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF),Y(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)

      pi=3.141592653589793d0
      convrd=180.d0/pi

c
c**** Open velocity station harmonic output file and write header information
c
      open(52,file='fort.52')
      write(52,*) nfreq+nf
      do j=1,nfreq+nf
         write(52,3679) freq(j),FF(j),FACE(j),namefr(j)
         end do
3679  format(1x,e20.10,1x,f10.7,1x,f12.8,1x,a10)
      write(52,*) NSTAV
c
c**** AT each STATION, transfer each load vector to p and solve the system, then write
c**** results
c
      DO N=1,NSTAV
         do k=1,mm
            p(k) = STAVLV(k,n)
            end do
         call fulsol(n)
         do k=1,mm
            y(k)=x(k)
            end do
         do k=1,mm
            p(k) = STAULV(k,n)
            end do
         call fulsol(n)

c        Compute amplitude and phase for each frequency making sure that the
c        phase is between 0 and 360 deg.  Then write output.

         write(52,*) N
         do i=1,nfreq+nf
            if((nf.eq.1).and.(i.eq.1)) then
               umag=x(i)/ff(i)
               vmag=y(i)/ff(i)
               phaseu=0.
               phasev=0.
               else
               i1=2*i-1-nf
               i2=i1+1
               umag=sqrt(x(i1)*x(i1)+x(i2)*x(i2))/ff(i)
               vmag=sqrt(y(i1)*y(i1)+y(i2)*y(i2))/ff(i)
```

```
               if((x(i1).eq.0.).and.(x(i2).eq.0.)) then
                  phaseu=0.
                  else
                  phaseu = atan2(x(i2),x(i1))
                  endif
               if((y(i1).eq.0.).and.(y(i2).eq.0.)) then
                  phasev=0.
                  else
                  phasev = atan2(y(i2),y(i1))
                  endif
               endif
            phasedu=convrd*phaseu+face(i)
            if(phasedu.lt.0.) phasedu=phasedu+360.
            if(phasedu.ge.360.) phasedu=phasedu-360.
            phasedv=convrd*phasev+face(i)
            if(phasedv.lt.0.) phasedv=phasedv+360.
            if(phasedv.ge.360.) phasedv=phasedv-360.

            write(52,6636) umag,phasedu,vmag,phasedv
 6636       format(2x,e16.8,1x,f11.4,2x,e16.8,1x,f11.4)
            end do

         end do

      return
      end


c**********************************************************************
c   Subroutine to solve the system and write output for elevation      *
c   globally.                                                          *
c                                                                      *
c   nf=0  if no steady constituent                                     *
c   nf=1  if steady constituent                                        *
c                                                                      *
c                         R.L. 11/8/95                                 *
c**********************************************************************
c
        SUBROUTINE LSQSOLEG(NP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveceg/ GLOELV
      common/matrix/ A,P,X
      DIMENSION GLOELV(2*MNHARF,MNP)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION PHASEE(MNHARF),EMAG(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      COMMON/MEANSQ/ TIMEBEG,NTSTEPS,DT,FMV,ITMV
      COMMON/MEANSQE/ ELAV,ELVA
      REAL ELAV(MNP),ELVA(MNP),DT,FMV
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...
```

```
      pi=3.141592653589793d0
      convrd=180.d0/pi

c
c**** Open velocity station harmonic output file and write header information
c
      open(53,file='fort.53')
      write(53,*) nfreq+nf
      do j=1,nfreq+nf
         write(53,3679) freq(j),FF(j),FACE(j),namefr(j)
         end do
3679  format(1x,e20.10,1x,f10.7,1x,f12.8,1x,a10)
      write(53,*) NP

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      EAVMAX=-999.
      EVAMAX=-999.
      EAVMIN= 999.
      EVAMIN= 999.
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

c
c***** AT each node transfer each load vector to p, solve and write output
c
      DO N=1,NP
         do k=1,mm
            p(k) = GLOELV(k,n)
            end do
         call fulsol(n)
c
c        Compute amplitude and phase for each frequency making sure that the
c        phase is between 0 and 360 deg.  Then write output.
c
         write(53,*) N
         do i=1,nfreq+nf
            if((nf.eq.1).and.(i.eq.1)) then
               emag(i)=x(i)
               emagt=emag(i)/ff(i)
               phasee(i)=0.
               else
               i1=2*i-1-nf
               i2=i1+1
               emag(i)=sqrt(x(i1)*x(i1)+x(i2)*x(i2))
               emagt=emag(i)/ff(i)
               if((x(i1).eq.0.).and.(x(i2).eq.0.)) then
                  phasee(i)=0.
                  else
                  phasee(i) = atan2(x(i2),x(i1))
                  endif
               endif
            phasede=convrd*phasee(i)+face(i)
            if(phasede.lt.0.) phasede=phasede+360.
            if(phasede.ge.360.) phasede=phasede-360.
            write(53,6635) emagt,phasede
 6635       format(2x,e16.8,1x,f11.4)
            end do

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...Resynthesize the time series to compute the average and variances.
CHARMV...Then compare resynthesized values with those computed during time stepping.
CHARMV...
         eav = 0.
```

```
         esq = 0.
         do it=1,ntsteps
            TIME=TIMEBEG+DT*IT
            rse=0.
            do ifr=1,nfreq+nf
               ftime=freq(ifr)*time
               rse=rse+emag(ifr)*cos(ftime-phasee(ifr))
               end do
            eav=eav+rse
            esq=esq+rse*rse
            end do

         eav=eav/ntsteps
         esq=esq/ntsteps-eav*eav
         if(elav(n).eq.0.) then
            if(eav.eq.0.) eavdif=1.0
            if(eav.ne.0.) eavdif=99e19
            else
            eavdif=eav/elav(n)
            endif
         if(elva(n).eq.0.) then
            if(esq.eq.0.) evadif=1.0
            if(esq.ne.0.) evadif=99e19
            else
            evadif=esq/elva(n)
            endif
         write(55,*) n
         write(55,7637) elav(n),eav,eavdif,elva(n),esq,evadif
 7637    format(2x,3(e16.8,1x),2x,3(e16.8,1x))

         IF(EAVDIF.GT.EAVMAX) THEN
            EAVMAX=EAVDIF
            NEAVMAX=n
            ENDIF
         IF(EAVDIF.LT.EAVMIN) THEN
            EAVMIN=EAVDIF
            NEAVMIN=n
            ENDIF
         IF(EVADIF.GT.EVAMAX) THEN
            EVAMAX=EVADIF
            NEVAMAX=n
            ENDIF
         IF(EVADIF.LT.EVAMIN) THEN
            EVAMIN=EVADIF
            NEVAMIN=n
            ENDIF
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

         end do

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      WRITE(16,7740)
 7740 FORMAT(///,5X,'THE LARGEST VALUES OF THE RATIO ',
     &              'RESYNTHESIZED ELEV TIME SERIES/RAW TIME SERIES:',/)
      WRITE(16,7741) EAVMAX,NEAVMAX
      WRITE(16,7742) EVAMAX,NEVAMAX
      WRITE(16,7747)
 7747 FORMAT(/,5X,'THE LOWEST VALUES OF THE RATIO ',
     &            'RESYNTHESIZED ELEV TIME SERIES/RAW TIME SERIES:',/)
      WRITE(16,7741) EAVMIN,NEAVMIN
      WRITE(16,7742) EVAMIN,NEVAMIN
 7741 FORMAT(9X,'  AVERAGE ELEVATION RATIO = ',E15.7,' AT NODE ',I8)
 7742 FORMAT(9X,' VARIANCE ELEVATION RATIO = ',E15.7,' AT NODE ',I8)
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
```

```
CHARMV...

      return
      end


c**********************************************************************
c   Subroutine to solve the system and write output for velocity       *
c   globally.                                                          *
c                                                                      *
c   nf=0  if no steady constituent                                     *
c   nf=1  if steady constituent                                        *
c                                                                      *
c                        R.L. 11/10/95                                 *
c**********************************************************************
c
        SUBROUTINE LSQSOLVG(NP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvg/ GLOULV,GLOVLV
      common/matrix/ A,P,X
      DIMENSION GLOULV(2*MNHARF,MNP),GLOVLV(2*MNHARF,MNP)
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION UMAG(MNHARF),VMAG(MNHARF),PHASEU(MNHARF),PHASEV(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF),Y(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      COMMON/MEANSQ/ TIMEBEG,NTSTEPS,DT,FMV,ITMV
      COMMON/MEANSQV/ xvelav,yvelav,xvelva,yvelva
      REAL DT,FMV,XVELAV(MNP),YVELAV(MNP),XVELVA(MNP),YVELVA(MNP)
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

      pi=3.141592653589793d0
      convrd=180.d0/pi

c
c**** Open velocity station harmonic output file and write header information
c
      open(54,file='fort.54')
      write(54,*) nfreq+nf
      do j=1,nfreq+nf
         write(54,3679) freq(j),FF(j),FACE(j),namefr(j)
         end do
3679  format(1x,e20.10,1x,f10.7,1x,f12.8,1x,a10)
      write(54,*) NP

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      UAVMAX=-999.
      VAVMAX=-999.
      UVAMAX=-999.
```

```
      VVAMAX=-999.
      UAVMIN= 999.
      VAVMIN= 999.
      UVAMIN= 999.
      VVAMIN= 999.
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

c
c***** AT each node transfer each load vector to p, solve and write output
c

      DO N=1,NP
         do k=1,mm
            p(k) = GLOVLV(k,n)
            end do
         call fulsol(n)
         do k=1,mm
            y(k)=x(k)
            end do
         do k=1,mm
            p(k) = GLOULV(k,n)
            end do
         call fulsol(n)
         write(54,*) n
         do i=1,nfreq+nf
            if((nf.eq.1).and.(i.eq.1)) then
               umag(i)=x(i)
               umagt=umag(i)/ff(i)
               vmag(i)=y(i)
               vmagt=vmag(i)/ff(i)
               phaseu(i)=0.
               phasev(i)=0.
               else
               i1=2*i-1-nf
               i2=i1+1
               umag(i)=sqrt(x(i1)*x(i1)+x(i2)*x(i2))
               umagt=umag(i)/ff(i)
               vmag(i)=sqrt(y(i1)*y(i1)+y(i2)*y(i2))
               vmagt=vmag(i)/ff(i)
               if((x(i1).eq.0.).and.(x(i2).eq.0.)) then
                  phaseu(i)=0.
                  else
                  phaseu(i)=atan2(x(i2),x(i1))
                  endif
               if((y(i1).eq.0.).and.(y(i2).eq.0.)) then
                  phasev(i)=0.
                  else
                  phasev(i)=atan2(y(i2),y(i1))
                  endif
               endif
            phasedu=convrd*phaseu(i)+face(i)
            if(phasedu.lt.0.) phasedu=phasedu+360.
            if(phasedu.ge.360.) phasedu=phasedu-360.
            phasedv=convrd*phasev(i)+face(i)
            if(phasedv.lt.0.) phasedv=phasedv+360.
            if(phasedv.ge.360.) phasedv=phasedv-360.

            write(54,6636) umagt,phasedu,vmagt,phasedv
 6636       format(2x,e16.8,1x,f11.4,2x,e16.8,1x,f11.4)
            end do

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...Resynthesize the time series to compute the average and variances.
CHARMV...Then compare resynthesized values with those computed during time stepping.
CHARMV...
         uav = 0.
```

```
      vav = 0.
      usq = 0.
      vsq = 0.
      do it=1,ntsteps
         TIME=TIMEBEG+DT*IT
         rsu=0.
         rsv=0.
         do ifr=1,nfreq+nf
            ftime=freq(ifr)*time
            rsu=rsu+umag(ifr)*cos(ftime-phaseu(ifr))
            rsv=rsv+vmag(ifr)*cos(ftime-phasev(ifr))
            end do
         uav=uav+rsu
         vav=vav+rsv
         usq=usq+rsu*rsu
         vsq=vsq+rsv*rsv
         end do

      uav=uav/ntsteps
      vav=vav/ntsteps
      usq=usq/ntsteps-uav*uav
      vsq=vsq/ntsteps-vav*vav
      if(xvelav(n).eq.0.) then
         if(uav.eq.0.) uavdif=1.0
         if(uav.ne.0.) uavdif=99e19
         else
         uavdif=uav/xvelav(n)
         endif
      if(yvelav(n).eq.0.) then
         if(vav.eq.0.) vavdif=1.0
         if(vav.ne.0.) vavdif=99e19
         else
         vavdif=vav/yvelav(n)
         endif
      if(xvelva(n).eq.0.) then
         if(usq.eq.0.) uvadif=1.0
         if(usq.ne.0.) uvadif=99e19
         else
         uvadif=usq/xvelva(n)
         endif
      if(yvelva(n).eq.0.) then
         if(vsq.eq.0.) vvadif=1.0
         if(vsq.ne.0.) vvadif=99e19
         else
         vvadif=vsq/yvelva(n)
         endif
      write(55,*) n
      write(55,7637) xvelav(n),uav,uavdif,xvelva(n),usq,uvadif
      write(55,7637) yvelav(n),vav,vavdif,yvelva(n),vsq,vvadif
7637  format(2x,3(e16.8,1x),2x,3(e16.8,1x))

      IF(UAVDIF.GT.UAVMAX) THEN
         UAVMAX=UAVDIF
         NUAVMAX=n
         ENDIF
      IF(UAVDIF.LT.UAVMIN) THEN
         UAVMIN=UAVDIF
         NUAVMIN=n
         ENDIF
      IF(VAVDIF.GT.VAVMAX) THEN
         VAVMAX=VAVDIF
         NVAVMAX=n
         ENDIF
      IF(VAVDIF.LT.VAVMIN) THEN
         VAVMIN=VAVDIF
         NVAVMIN=n
         ENDIF
      IF(UVADIF.GT.UVAMAX) THEN
         UVAMAX=UVADIF
         NUVAMAX=n
```

```
            ENDIF
         IF(UVADIF.LT.UVAMIN) THEN
            UVAMIN=UVADIF
            NUVAMIN=n
            ENDIF
         IF(VVADIF.GT.VVAMAX) THEN
            VVAMAX=VVADIF
            NVVAMAX=n
            ENDIF
         IF(VVADIF.LT.VVAMIN) THEN
            VVAMIN=VVADIF
            NVVAMIN=n
            ENDIF
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

         end do

CHARMV...
CHARMV...UNCOMMENT THE FOLLOWING LINES TO COMPUTE MEANS AND VARIANCES
CHARMV...FOR CHECKING THE HARMONIC ANALYSIS RESULTS.
CHARMV...
      WRITE(16,7740)
 7740 FORMAT(///,5X,'THE LARGEST VALUES OF THE RATIO ',
     &              'RESYNTHESIZED VEL TIME SERIES/RAW TIME SERIES:',/)
      WRITE(16,7743) UAVMAX,NUAVMAX
      WRITE(16,7744) UVAMAX,NUVAMAX
      WRITE(16,7745) VAVMAX,NVAVMAX
      WRITE(16,7746) VVAMAX,NVVAMAX
      WRITE(16,7747)
 7747 FORMAT(//,5X,'THE LOWEST VALUES OF THE RATIO ',
     &             'RESYNTHESIZED VEL TIME SERIES/RAW TIME SERIES:',/)
      WRITE(16,7743) UAVMIN,NUAVMIN
      WRITE(16,7744) UVAMIN,NUVAMIN
      WRITE(16,7745) VAVMIN,NVAVMIN
      WRITE(16,7746) VVAMIN,NVVAMIN
 7743 FORMAT(9X,' AVERAGE U VELOCITY RATIO = ',E15.7,' AT NODE ',I8)
 7744 FORMAT(9X,'VARIANCE U VELOCITY RATIO = ',E15.7,' AT NODE ',I8)
 7745 FORMAT(9X,' AVERAGE V VELOCITY RATIO = ',E15.7,' AT NODE ',I8)
 7746 FORMAT(9X,'VARIANCE V VELOCITY RATIO = ',E15.7,' AT NODE ',I8)
CHARMV...
CHARMV...END OF MEANS AND VARIANCES STATEMENTS (MORE FOLLOW BELOW)
CHARMV...

      return
      end


c**********************************************************************
c   Subroutine to write out to the hotstart file (UNITS 67 and 68)     *
c   header information and the LHS matrix for the harmonic analysis    *
c                                                                      *
c                        R.L.  11/8/95                                 *
c**********************************************************************
c
      SUBROUTINE HAHOUT(NP,NSTAE,NSTAV,ISTAE,ISTAV,IGLOE,IGLOV,IOUNIT,
     &                                                      IHOTSTP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
```

```
      common/matrix/ A,P,X
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)
      CHARACTER*16 FNAME
      CHARACTER*8 FNAM8(2)
      EQUIVALENCE (FNAM8(1),FNAME)

c
c***** Write Out various parameter values
c
      WRITE(IOUNIT,REC=IHOTSTP+1) NZ
      WRITE(IOUNIT,REC=IHOTSTP+2) NF
      WRITE(IOUNIT,REC=IHOTSTP+3) MM
      WRITE(IOUNIT,REC=IHOTSTP+4) NP
      WRITE(IOUNIT,REC=IHOTSTP+5) NSTAE
      WRITE(IOUNIT,REC=IHOTSTP+6) NSTAV
      WRITE(IOUNIT,REC=IHOTSTP+7) ISTAE
      WRITE(IOUNIT,REC=IHOTSTP+8) ISTAV
      WRITE(IOUNIT,REC=IHOTSTP+9) IGLOE
      WRITE(IOUNIT,REC=IHOTSTP+10) IGLOV
      WRITE(IOUNIT,REC=IHOTSTP+11) ICALL
      WRITE(IOUNIT,REC=IHOTSTP+12) NFREQ
      IHOTSTP = IHOTSTP+12

      do i=1,nfreq+nf
         FNAME=NAMEFR(I)
         WRITE(IOUNIT,REC=IHOTSTP+1) FNAM8(1)
         WRITE(IOUNIT,REC=IHOTSTP+2) FNAM8(2)
         IHOTSTP=IHOTSTP+2
         WRITE(IOUNIT,REC=IHOTSTP+1) freq(i)
         WRITE(IOUNIT,REC=IHOTSTP+2) FF(i)
         WRITE(IOUNIT,REC=IHOTSTP+3) FACE(i)
         IHOTSTP=IHOTSTP+3
         end do

c
c***** Write Out time of most recent H.A. update
c
      WRITE(IOUNIT,REC=IHOTSTP+1) TIMEUD
      WRITE(IOUNIT,REC=IHOTSTP+2) ITUD
      IHOTSTP=IHOTSTP+2

c
c***** Write Out LHS Matrix
c
      do i=1,mm
         do j=1,mm
            IHOTSTP = IHOTSTP + 1
            WRITE(IOUNIT,REC=IHOTSTP) A(I,J)
            END DO
         END DO

      return
      end


c*********************************************************************
c   Subroutine to write elevation station harmonic analysis RHS load   *
c   vector to a hot start file (UNITS 67 and 68)                       *
c                                                                      *
c                         R.L.  11/8/95                                *
c*********************************************************************
c
      SUBROUTINE HAHOUTES(NSTAE,IOUNIT,IHOTSTP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAE=1)
```

```
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveces/ STAELV
      DIMENSION STAELV(2*MNHARF,MNSTAE)

c
c***** Write Out Station Elevation RHS load vector
c
      do n=1,NSTAE
         do i=1,mm
            IHOTSTP=IHOTSTP+1
            WRITE(IOUNIT,REC=IHOTSTP) STAELV(I,N)
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to write velocity station harmonic analysis RHS load    *
c   vector to a hot start file (UNITS 67 and 68)                       *
c                                                                      *
c                         R.L.  11/8/95                                *
c**********************************************************************
c
      SUBROUTINE HAHOUTVS(NSTAV,IOUNIT,IHOTSTP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAV=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvs/ STAULV,STAVLV
      DIMENSION STAULV(2*MNHARF,MNSTAV),STAVLV(2*MNHARF,MNSTAV)

c
c***** Write Out Station Velocity LHS load vector
c
      do N=1,NSTAV
         do i=1,mm
            IHOTSTP=IHOTSTP+1
            WRITE(IOUNIT,REC=IHOTSTP) STAULV(I,N)
            IHOTSTP=IHOTSTP+1
            WRITE(IOUNIT,REC=IHOTSTP) STAVLV(I,N)
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to write global elevation harmonic analysis RHS load    *
c   vector to a hot start file (UNITS 67 and 68)                       *
c                                                                      *
c                         R.L.  11/8/95                                *
c**********************************************************************
c
```

```
      SUBROUTINE HAHOUTEG(NP,IOUNIT,IHOTSTP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveceg/ GLOELV
      DIMENSION GLOELV(2*MNHARF,MNP)

c
c***** Write Out Global Elevation RHS load vector
c
      do n=1,np
         do i=1,mm
            IHOTSTP=IHOTSTP+1
            WRITE(IOUNIT,REC=IHOTSTP) GLOELV(I,N)
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to write global velocity harmonic analysis RHS load     *
c   vector to a hot start file (UNITS 67 and 68)                       *
c                                                                      *
c                         R.L.  11/8/95                                *
c**********************************************************************
c
      SUBROUTINE HAHOUTVG(NP,IOUNIT,IHOTSTP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvg/ GLOULV,GLOVLV
      DIMENSION GLOULV(2*MNHARF,MNP),GLOVLV(2*MNHARF,MNP)

c
c***** Write Out Global Velocity RHS load vector
c
      do n=1,np
         do i=1,mm
            IHOTSTP=IHOTSTP+1
            WRITE(IOUNIT,REC=IHOTSTP) GLOULV(I,N)
            IHOTSTP=IHOTSTP+1
            WRITE(IOUNIT,REC=IHOTSTP) GLOVLV(I,N)
            end do
         end do

      return
      end


c**********************************************************************
```

```
c   Subroutine to initialize parameters for harmonic analysis with a   *
c   cold start.                                                         *
c                                                                       *
c                         R.L.  11/9/95                                 *
c***********************************************************************
c
      SUBROUTINE HACOLDS

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ NFREQ,FREQ,FF,FACE,NAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/matrix/ A,P,X
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF)

      if(freq(1).eq.0.0) then
         nz=0
         nf=1
         else
         nz=1
         nf=0
         endif
      nfreq=nfreq-nf
      mm=2*nfreq+nf
      do i=1,mm
         do j=1,mm
            a(i,j)=0.
            end do
         end do

      icall=0

      return
      end


c***********************************************************************
c   Subroutine to initialize elevation station load vectors for         *
c   harmonic analysis with a cold start.                                *
c                                                                       *
c                         R.L.  11/9/95                                 *
c***********************************************************************
c
      SUBROUTINE HACOLDSES(NSTAE)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAE=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveces/ STAELV
      DIMENSION STAELV(2*MNHARF,MNSTAE)

      do i=1,mm
         do N=1,NSTAE
```

```
            STAELV(I,N)=0.
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to initialize elevation station load vectors for        *
c   harmonic analysis with a cold start.                               *
c                                                                      *
c                         R.L.  11/9/95                                *
c**********************************************************************
c
      SUBROUTINE HACOLDSVS(NSTAV)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAV=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvs/ STAULV,STAVLV
      DIMENSION STAULV(2*MNHARF,MNSTAV),STAVLV(2*MNHARF,MNSTAV)

      do i=1,mm
         do N=1,NSTAV
            STAULV(I,N)=0.
            STAVLV(I,N)=0.
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to initialize global elevation load vectors for         *
c   harmonic analysis with a cold start.                               *
c                                                                      *
c                         R.L.  11/9/95                                *
c**********************************************************************
c
      SUBROUTINE HACOLDSEG(NP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveceg/ GLOELV
      DIMENSION GLOELV(2*MNHARF,MNP)

      do i=1,mm
         do N=1,NP
            GLOELV(I,N)=0.
            end do
         end do
```

```
      return
      end


c**********************************************************************
c   Subroutine to initialize global velocity load vectors for          *
c   harmonic analysis with a cold start.                               *
c                                                                      *
c                         R.L.  11/9/95                                *
c**********************************************************************
c
      SUBROUTINE HACOLDSVG(NP)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS (MORE FOLLOW BELOW)
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvg/ GLOULV,GLOVLV
      DIMENSION GLOULV(2*MNHARF,MNP),GLOVLV(2*MNHARF,MNP)

      do i=1,mm
         do N=1,NP
            GLOULV(I,N)=0.
            GLOVLV(I,N)=0.
            end do
         end do

      return
      end


c**********************************************************************
c   Subroutine to read in and initialize harmonic analysis for a hot   *
c   start.                                                             *
c                                                                      *
c   Checks are made to ensure agreement between values read in from    *
c   the hotstart file and values read in from the UNIT 15 file.        *
c                                                                      *
c                         R.L. 11/9/95                                 *
c**********************************************************************
c
      SUBROUTINE HAHOTS(INSTAE,INSTAV,INP,IISTAE,IISTAV,IIGLOE,IIGLOV,
     +                                          NSCREEN,IHOTSTP,IHOT)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/LSQFREQS/ INFREQ,IFREQ,IFF,IFACE,INAMEFR
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/matrix/ A,P,X
      REAL FREQ(MNHARF),FF(MNHARF),FACE(MNHARF)
      REAL IFREQ(MNHARF),IFF(MNHARF),IFACE(MNHARF)
      DIMENSION A(2*MNHARF,2*MNHARF),P(2*MNHARF),X(2*MNHARF)
      CHARACTER*10 NAMEFR(MNHARF),INAMEFR(MNHARF)
      CHARACTER*16 FNAME
      CHARACTER*8 FNAM8(2)
```

```
      EQUIVALENCE (FNAM8(1),FNAME)

c
c***** Compute parameter values for checking
c
      if(ifreq(1).eq.0.0) then
         inz=0
         inf=1
         else
         inz=1
         inf=0
         endif
      infreq=infreq-inf
      imm=2*infreq+inf

c
c***** Read in and check various parameter values
c
      READ(IHOT,REC=IHOTSTP+1) NZ
      READ(IHOT,REC=IHOTSTP+2) NF
      READ(IHOT,REC=IHOTSTP+3) MM
      READ(IHOT,REC=IHOTSTP+4) NP
      READ(IHOT,REC=IHOTSTP+5) NSTAE
      READ(IHOT,REC=IHOTSTP+6) NSTAV
      READ(IHOT,REC=IHOTSTP+7) ISTAE
      READ(IHOT,REC=IHOTSTP+8) ISTAV
      READ(IHOT,REC=IHOTSTP+9) IGLOE
      READ(IHOT,REC=IHOTSTP+10) IGLOV
      READ(IHOT,REC=IHOTSTP+11) ICALL
      READ(IHOT,REC=IHOTSTP+12) NFREQ
      IHOTSTP = IHOTSTP+12

      iflag=0
      if(nz.ne.inz) iflag=1
      if(nf.ne.inf) iflag=1
      if(mm.ne.imm) iflag=1
      if(np.ne.inp) iflag=1
      if(nstae.ne.instae) iflag=1
      if(nstav.ne.instav) iflag=1
      if(istae.ne.iistae) iflag=1
      if(istav.ne.iistav) iflag=1
      if(igloe.ne.iigloe) iflag=1
      if(iglov.ne.iiglov) iflag=1
      if(nfreq.ne.infreq) iflag=1
      do i=1,nfreq+nf
         READ(IHOT,REC=IHOTSTP+1) FNAM8(1)
         READ(IHOT,REC=IHOTSTP+2) FNAM8(2)
         IHOTSTP = IHOTSTP + 2
         NAMEFR(I) = FNAME
         read(IHOT,REC=IHOTSTP+1) freq(i)
         read(IHOT,REC=IHOTSTP+2) FF(i)
         read(IHOT,REC=IHOTSTP+3) FACE(i)
         IHOTSTP = IHOTSTP + 3

         if(namefr(i).ne.inamefr(i)) iflag=1
         if(abs(freq(i)+ifreq(i)).lt.1e-30) then
            fdiff=0.
            else
            fdiff=abs(freq(i)-ifreq(i))/abs(freq(i)+ifreq(i))
            endif
         if(fdiff.ge.1.e-6) iflag=1
         if(abs(FF(i)+iFF(i)).lt.1e-30) then
            fdiff=0.
            else
            fdiff=abs(FF(i)-iFF(i))/abs(FF(i)+iFF(i))
            endif
         if(fdiff.ge.1.e-6) iflag=1
         if(abs(FACE(i)+iFACE(i)).lt.1e-30) then
            fdiff=0.
            else
```

```
            fdiff=abs(FACE(i)-iFACE(i))/abs(FACE(i)+iFACE(i))
            endif
         if(fdiff.ge.1.e-6) iflag=1
         end do
      if(iflag.eq.1) goto 999
c
c***** Read in time of most recent H.A. update
c
      READ(IHOT,REC=IHOTSTP+1) TIMEUD
      READ(IHOT,REC=IHOTSTP+2) ITUD
      IHOTSTP = IHOTSTP + 2
c
c***** Read in RHS Matrix
c
      do i=1,mm
         do j=1,mm
            IHOTSTP = IHOTSTP + 1
            READ(IHOT,REC=IHOTSTP) A(I,J)
            end do
         end do

c
c***** FATAL Error Messages
c
 999  continue
      if(iflag.ne.0) then
         if(nscreen.eq.1) write(6,1000)
         write(16,1000)
1000     FORMAT(////,5x,'***** DISCREPANCY IN HARMONIC ANALYSIS HOT ',
     +                                          'START FILE *****',/)
         endif

      if(iflag.eq.1) then
         if(nz.ne.inz) then
            if(nscreen.eq.1) write(6,2010) inz,nz
            write(16,2010) inz,nz
2010        format(5x,'NZ COMPUTED FROM UNIT 14 INPUT = ',I2,
     +              ', NZ READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(nf.ne.inf) then
            if(nscreen.eq.1) write(6,2020) inf,nf
            write(16,2020) inf,nf
2020        format(5x,'NF COMPUTED FROM UNIT 14 INPUT = ',I2,
     +              ', NF READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(mm.ne.imm) then
            if(nscreen.eq.1) write(6,2030) imm,mm
            write(16,2030) imm,mm
2030        format(5x,'MM COMPUTED FROM UNIT 14 INPUT = ',I2,
     +              ', MM READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(np.ne.inp) then
            if(nscreen.eq.1) write(6,2040) inp,np
            write(16,2040) inp,np
2040        format(5x,'NP READ IN FROM UNIT 15 = ',I2,
     +              ', NP READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(nstae.ne.instae) then
            if(nscreen.eq.1) write(6,2050) instae,nstae
            write(16,2050) instae,nstae
2050        format(5x,'NSTAE READ IN FROM UNIT 15 = ',I2,
     +              ', NSTAE READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(nstav.ne.instav) then
            if(nscreen.eq.1) write(6,2060) instav,nstav
            write(16,2060) instav,nstav
2060        format(5x,'NSTAV READ IN FROM UNIT 15 = ',I2,
     +              ', NSTAV READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(istae.ne.iistae) then
```

```
            if(nscreen.eq.1) write(6,2070) iistae,istae
            write(16,2070) iistae,istae
2070        format(5x,'ISTAE READ IN FROM UNIT 15 = ',I2,
     +              ', ISTAE READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(istav.ne.iistav) then
            if(nscreen.eq.1) write(6,2080) iistav,istav
            write(16,2080) iistav,istav
2080        format(5x,'ISTAV READ IN FROM UNIT 15 = ',I2,
     +              ', ISTAV READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(igloe.ne.iigloe) then
            if(nscreen.eq.1) write(6,2090) iigloe,igloe
            write(16,2090) iigloe,igloe
2090        format(5x,'IGLOE READ IN FROM UNIT 15 = ',I2,
     +              ', IGLOE READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(iglov.ne.iiglov) then
            if(nscreen.eq.1) write(6,2100) iiglov,iglov
            write(16,2100) iiglov,iglov
2100        format(5x,'IGLOV READ IN FROM UNIT 15 = ',I2,
     +              ', IGLOV READ IN FROM HOT START FILE = ',I2,/)
            endif
         if(nfreq.ne.infreq) then
            if(nscreen.eq.1) write(6,2110) infreq,nfreq
            write(16,2110) infreq,nfreq
2110        format(5x,'NFREQ COMPUTED FROM UNIT 15 INPUT = ',I2,
     +              ', NFREQ READ IN FROM HOT START FILE = ',I2,/)
            endif
         do i=1,nfreq+nf
            if(namefr(i).ne.inamefr(i)) then
               if(nscreen.eq.1) write(6,2120) i,inamefr(i),namefr(i)
               write(16,2120) i,namefr(i),namefr(i)
2120           format(5x,'FOR CONSTITUENT # ',I3,
     +                 ', NAMEFR READ IN FROM UNIT 15 = ',A10,
     +                 ', NAMEFR READ IN FROM HOT START FILE = ',A10,/)
               endif
            if(freq(i).ne.ifreq(i)) then
               if(nscreen.eq.1) write(6,2130) i,ifreq(i),freq(i)
               write(16,2130) i,ifreq(i),freq(i)
2130           format(5x,'FOR CONSTITUENT # ',I3,
     +                 ', FREQ READ IN FROM UNIT 15 = ',D20.10,
     +                 ', FREQ READ IN FROM HOT START FILE = ',D20.10,/)
               endif
            if(FF(i).ne.iFF(i)) then
               if(nscreen.eq.1) write(6,2140) i,iff(i),ff(i)
               write(16,2140) i,iff(i),ff(i)
2140           format(5x,'FOR CONSTITUENT # ',I3,
     +                 ', FF READ IN FROM UNIT 15 = ',F10.5,
     +                 ', FF READ IN FROM HOT START FILE = ',F10.5,/)
               endif
            if(FACE(i).ne.iFACE(i)) then
               if(nscreen.eq.1) write(6,2150) i,iface(i),face(i)
               write(16,2150) i,iface(i),face(i)
2150           format(5x,'FOR CONSTITUENT # ',I3,
     +                 ', FACE READ IN FROM UNIT 15 = ',F10.5,
     +                 ', FACE READ IN FROM HOT START FILE = ',F10.5,/)
               endif
            end do
         if(nscreen.eq.1) write(6,1010)
         write(16,1010)
1010     FORMAT(//,5x,'********** RUN TERMINATED **********',/)
         stop
         endif

      return
      end


c*********************************************************************
```

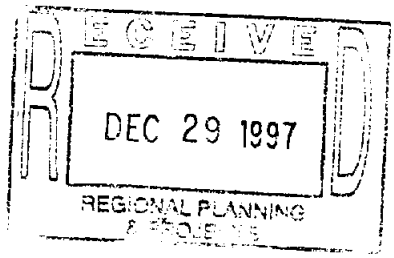
```
c   Subroutine to read in and initialize the elevation station load    *
c   vector for harmonic analysis with a hot start.                     *
c                                                                       *
c                          R.L. 11/9/95                                 *
c***********************************************************************
c
      SUBROUTINE HAHOTSES(NSTAE,IHOTSTP,IHOT)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAE=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveces/ STAELV
      DIMENSION STAELV(2*MNHARF,MNSTAE)

c
c***** Read in Station Elevation LHS load vector
c
      do n=1,NSTAE
         do i=1,mm
            IHOTSTP=IHOTSTP+1
            READ(IHOT,REC=IHOTSTP) STAELV(I,N)
            end do
         end do

      return
      end


c***********************************************************************
c   Subroutine to read in and initialize the velocity station load      *
c   vector for harmonic analysis with a hot start.                      *
c                                                                       *
c                          R.L. 11/9/95                                 *
c***********************************************************************
c
      SUBROUTINE HAHOTSVS(NSTAV,IHOTSTP,IHOT)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNSTAV=1)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvs/ STAULV,STAVLV
      DIMENSION STAULV(2*MNHARF,MNSTAV),STAVLV(2*MNHARF,MNSTAV)

c
c***** Read in Station Velocity LHS load vector
c
      do N=1,NSTAV
         do i=1,mm
            READ(IHOT,REC=IHOTSTP+1) STAULV(I,N)
            READ(IHOT,REC=IHOTSTP+2) STAVLV(I,N)
            IHOTSTP = IHOTSTP + 2
            end do
         end do
```

```
      return
      end


c**********************************************************************
c   Subroutine to read in and initialize the global elevation load     *
c   vector for harmonic analysis with a hot start.                     *
c                                                                      *
c                         R.L. 11/9/95                                 *
c**********************************************************************
c
      SUBROUTINE HAHOTSEG(NP,IHOTSTP,IHOT)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadveceg/ GLOELV
      DIMENSION GLOELV(2*MNHARF,MNP)

c
c***** Read in Global Elevation LHS load vector
c
         do n=1,np
            do i=1,mm
               IHOTSTP=IHOTSTP+1
               READ(IHOT,REC=IHOTSTP) GLOELV(I,N)
               end do
            end do

      return
      end


c**********************************************************************
c   Subroutine to read in and initialize the global velocity load      *
c   vector for harmonic analysis with a hot start.                     *
c                                                                      *
c                         R.L. 11/9/95                                 *
c**********************************************************************
c
      SUBROUTINE HAHOTSVG(NP,IHOTSTP,IHOT)

CUSER...
CUSER...SET PARAMETERS HERE
CUSER...
      PARAMETER(MNP=1082)
      PARAMETER(MNHARF=4)
CUSER...
CUSER...END OF PARAMETER STATEMENTS
CUSER....

      IMPLICIT REAL*8 (A-H,O-Z)
      common/lsqparms/ TIMEUD,nz,nf,mm,ITUD,ICALL
      common/loadvecvg/ GLOULV,GLOVLV
      DIMENSION GLOULV(2*MNHARF,MNP),GLOVLV(2*MNHARF,MNP)

c
c***** Read in Global Velocity LHS load vector
c
      do n=1,np
         do i=1,mm
            READ(IHOT,REC=IHOTSTP+1) GLOULV(I,N)
```

```
      READ(IHOT,REC=IHOTSTP+2) GLOVLV(I,N)
      IHOTSTP = IHOTSTP + 2
      end do
   end do

return
end
```

TIDAL PREDICTIONS FOR GALVESTON BAY, TEXAS

USING MODEL ADCIRC -2DDI



J.J. Westerink
Department of Civil Engineering and Geological Sciences
University of Notre Dame
Notre Dame, IN 46556

R.A. Luettich, Jr.
Institute of Marine Sciences
University of North Carolina at Chapel Hill
Morehead City, NC 28557

December 26, 1997

Prepared for the TEXAS WATER DEVELOPMENT BOARD
The State of Texas
Austin, TX 78711

Under Contract No. 95-483-097

CONTENTS

SUMMARY

This report describes the application of model ADCIRC-2DDI, a two dimensional depth integrated finite element based hydrodynamic circulation code, to study tidal elevation and currents in the vicinity of Galveston Bay, Texas. Issues that are emphasized include the definition of hydrodynamically accurate and simple open ocean boundaries; the use of large domains; the importance of a high degree of grid resolution in coastal regions; and the use of finite element meshes with highly varying nodal densities in order to minimize the size of the discrete problem.

Three finite element meshes are examined in this report. The first is a regional grid which encompasses Galveston Bay and vicinity only. The second grid encompasses the entire Gulf of Mexico and incorporates the first grid. The third grid includes a significant portion of the western North Atlantic in addition to the entire Gulf of Mexico and Caribbean Sea. This third grid, referred to as the Eastcoast grid, incorporates both the second and therefore first grids. All three models are forced with surface elevation forcing function on the seaward boundaries. In addition, tidal potential functions force the interior domain for the Gulf of Mexico and Eastcoast models. The tides are verified at nineteen tidal elevation stations throughout the Gulf of Mexico. Current patterns are in particular examined in the vicinity of Galveston Bay, Texas.

TIDAL PREDICTIONS FOR GALVESTON BAY, TEXAS USING MODEL ADCIRC-2DDI

PART 1: INTRODUCTION

1. Numerical models are widely used to compute sea surface elevations and currents due to mesoscale processes such as tides and hurricane storm surge (Lynch, 1983; Westerink and Gray, 1991). However, a truly predictive capability for flow in coastal regions requires that all important scales of motion be sufficiently resolved in the numerically discrete form of the governing equations. Typically an increasingly greater degree of grid refinement is required as the landward boundary is approached, in order to resolve important processes and to prevent energy from aliasing. This latter requirement stems from factors such as shallow water waves being inherently slower and shorter in shallow water as well as the importance of near coastal bathymetry and geometry which dominate both elevation and current patterns. In order to provide sufficient resolution in the near shore region without excessively increasing the size of the discrete problem, a numerical method must be used which permits a very high degree of grid flexibility.

2. The finite element (FE) method inherently offers a degree of grid flexibility which is ideally suited for coastal modeling. However, early shallow water equation models using the FE method were plagued with severe spurious mode problems and typically required the heavy handed use of non-physical dissipation, limiting their usefulness as predictive tools (Gray, 1982). The introduction of wave-continuity equation (WCE) formulations by Lynch and Gray (1979) led to the development of robust FE depth integrated coastal circulation codes. Thorough testing (Lynch and Gray, 1979; Lynch, 1981; Walters, 1983, 1984; Drolet, 1989), extensive analysis (Platzman, 1981; Foreman, 1983; Drolet and Gray, 1988) and detailed field applications (Werner and Lynch, 1987; Walters, 1987; Walters, 1988, Werner and Lynch, 1989; Walters and Werner, 1989; Gray et al., 1987; Gray, 1989; Lynch and Werner, 1990; Lynch et al., 1988; Lynch et al., 1990; Foreman, 1988; Westerink et al., 1992a) carried out during the past decade have led to a broad based fundamental understanding of WCE formulations. These studies have demonstrated

the unique advantages of these formulations for FE applications in terms of achieving a concurrent high level of computational accuracy and efficiency.

3. In this report, we examine the application of ADCIRC-2DDI, a WCE based FE shallow water equation model (Luettich et al., 1992; Westerink et al., 1992b), to study surface elevation and flow due to tides in the vicinity of Galveston Bay, Texas. It is shown that WCE based FE methods lead to highly accurate and efficient predictions with an unprecedented degree of grid flexibility.

PART II: GOVERNING EQUATIONS AND NUMERICAL DISCRETIZATION

4. The computations described in this report were performed using ADCIRC-2DDI, the depth integrated option of a system of two and three dimensional hydrodynamic codes named ADCIRC (Luettich et al., 1992). ADCIRC-2DDI uses the depth integrated equations of mass and momentum conservation, subject to the incompressibility, Boussinesq and hydrostatic pressure approximations. Using the standard quadratic parameterization for bottom stress and neglecting baroclinic terms and lateral diffusion/dispersion effects leads to the following set of conservation statements in primitive non-conservative form expressed in a spherical coordinate system (Flather, 1988; Kolar et al., 1992):

$$\frac{\partial \zeta}{\partial t} + \frac{1}{R\cos\phi}\left(\frac{\partial UH}{\partial \lambda} + \frac{\partial (VH\cos\phi)}{\partial \phi}\right) = 0 \tag{1}$$

$$\frac{\partial U}{\partial t} + \frac{1}{R\cos\phi}U\frac{\partial U}{\partial \lambda} + \frac{1}{R}V\frac{\partial U}{\partial \phi} - \left(\frac{\tan\phi}{R}U + f\right)V = -\frac{1}{R\cos\phi}\frac{\partial}{\partial \lambda}\left[\frac{p_s}{\rho_0} + g(\zeta - \eta)\right] + \frac{\tau_{s\lambda}}{\rho_0 H} + -\tau_* U \tag{2}$$

$$\frac{\partial V}{\partial t} + \frac{1}{R\cos\phi}U\frac{\partial V}{\partial \lambda} + \frac{1}{R}V\frac{\partial V}{\partial \phi} + \left(\frac{\tan\phi}{R}U + f\right)U = -\frac{1}{R}\frac{\partial}{\partial \phi}\left[\frac{p_s}{\rho_0} + g(\zeta - \eta)\right] + \frac{\tau_{s\phi}}{\rho_0 H} - \tau_* V \tag{3}$$

where

t = time

λ, ϕ = degrees longitude (east of Greenwich positive) and degrees lattitude (north of the equator positive)

ζ = free surface elevation relative to the geoid

U,V = depth averaged horizontal velocities

R = radius of the Earth

H = $\zeta + h$ = total water column

h = bathymetric depth relative to the geoid

f = $2\Omega\sin\phi$ = Coriolis parameter

Ω = angular speed of the Earth

p_s = atmospheric pressure at the free surface

g = acceleration due to gravity

η = effective Newtonian equilibrium tide potential

ρ_0 = reference density of water

τ_{sx}, τ_{sy} = applied free surface stress

τ_* = $C_f \dfrac{(U^2 + V^2)^{1/2}}{H}$

C_f = bottom friction coefficient

5. A practical expression for the effective Newtonian equilibrium tide potential is given by Reid (1990) as:

$$\eta(\lambda, \phi, t) = \sum_{n,\, j} \alpha_{jn} C_{jn} f_{jn}(t_0) L_j(\phi) \cos[2\pi(t - t_0)/T_{jn} + j\lambda + \nu_{jn}(t_0)] \qquad (4)$$

where

C_{jn} = constant characterizing the amplitude of tidal constituent n of species j

α_{jn} = effective earth elasticity factor for tidal constituent n of species j

f_{jn} = time dependent nodal factor

ν_{jn} = time dependent astronomical argument

$j = 0, 1, 2$ = tidal species ($j = 0$, declinational; $j = 1$, diurnal; $j = 2$, semi-diurnal)

L_0 = $3\sin^2\phi - 1$

L_1 = $\sin(2\phi)$

L_2 = $\cos^2(\phi)$

λ, ϕ = degrees longitude and latitude respectively

t_0 = reference time

T_{jn} = period of constituent n of species j

Values for C_{jn} are presented by Reid (1990). We note that the value for the effective earth elastic-

ity factor is typically taken as 0.69 for all tidal constituents (Schwiderski, 1980; Hendershott, 1981) although its value has been shown to be slightly constituent dependent (Wahr, 1981; Woodworth, 1990).

6. To facilitate a FE solution to equations, (1) - (3), these equations are mapped from spherical form into a rectilinear coordinate system using a Carte Parallelogrammatique (CP) projection (Pearson, 1990):

$$x' = R(\lambda - \lambda_o)\cos\phi_o \tag{5}$$

$$y' = R\phi \tag{6}$$

where

λ_0, ϕ_0 = center point of the projection

Applying the CP projection to Equations (1) - (3) gives the shallow water equations in primitive non-conservative form expressed in the CP coordinate system:

$$\frac{\partial \zeta}{\partial t} + \frac{\cos\phi_0}{\cos\phi}\frac{\partial (UH)}{\partial x'} + \frac{1}{\cos\phi}\frac{\partial (VH\cos\phi)}{\partial y'} = 0 \tag{7}$$

$$\begin{aligned} &\frac{\partial U}{\partial t} + \frac{\cos\phi_0}{\cos\phi}U\frac{\partial U}{\partial x'} + V\frac{\partial U}{\partial y'} - \left(\frac{\tan\phi}{R}U + f\right)V \\ &= -\frac{\cos\phi_0}{\cos\phi}\frac{\partial}{\partial x'}\left[\frac{p_s}{\rho_o} + g(\zeta - \eta)\right] + \frac{\tau_{s\lambda}}{\rho_o H} - \tau_* U \end{aligned} \tag{8}$$

$$\begin{aligned} &\frac{\partial V}{\partial t} + \frac{\cos\phi_o}{\cos\phi}U\frac{\partial V}{\partial x'} + V\frac{\partial V}{\partial y'} + \left(\frac{\tan\phi}{R}U + f\right)U \\ &= -\frac{\partial}{\partial y'}\left[\frac{p_s}{\rho_o} + g(\zeta - \eta)\right] + \frac{\tau_{s\phi}}{\rho_0 H} - \tau_* V \end{aligned} \tag{9}$$

7. Utilizing the FE method to resolve the spatial dependence in the shallow water equations in their primitive form gives inaccurate solutions with severe artificial near $2 \cdot \Delta x$ modes (Gray, 1982). However, reformulating the primitive equations into a Generalized Wave Continuity Equation (GWCE) form gives highly accurate, noise free, FE based solutions to the shallow water

equations (Lynch and Gray, 1979; Kinnmark, 1984). The GWCE is derived by combining a time differentiated form of the primitive continuity equation and a spatially differentiated form of the primitive momentum equations recast into conservative form, reformulating the convective terms into non-conservative form and adding the primitive form of the continuity equation multiplied by a constant in time and space, τ_o (Lynch and Gray, 1979; Kinnmark, 1984; Luettich et al., 1992). The GWCE in the CP coordinate system is:

$$\frac{\partial^2\zeta}{\partial t^2}+\tau_0\frac{\partial\zeta}{\partial t}+\frac{\cos\phi_0}{\cos\phi}\frac{\partial}{\partial x'}\frac{\partial\zeta}{\partial t}[U-\frac{\cos\phi_0}{\cos\phi}UH\frac{\partial U}{\partial x'}$$

$$-VH\frac{\partial U}{\partial y'}+\left(\frac{\tan\phi}{R}U+f\right)VH-H\frac{\cos\phi_0}{\cos\phi}\frac{\partial}{\partial x'}\left(\frac{p_s}{\rho_0}+g(\zeta-\eta)\right)$$

$$-(\tau_*-\tau_0)UH+\frac{\tau_{s\lambda}}{\rho_0}\Big]$$

$$+\frac{\partial}{\partial y'}[V\frac{\partial\zeta}{\partial t}-\frac{\cos\phi_0}{\cos\phi}UH\frac{\partial V}{\partial x'}-VH\frac{\partial V}{\partial y'}-\left(\frac{\tan\phi}{R}U+f\right)UH$$

$$-H\frac{\partial}{\partial y'}\left(\frac{p_s}{\rho_0}+g(\zeta-\eta)\right)-(\tau_*-\tau_0)VH+\frac{\tau_{s\phi}}{\rho_0}\Big]$$

$$-\frac{\partial}{\partial t}\Big[\frac{\tan\phi}{R}VH\Big]+-\tau_0\Big[\frac{\tan\phi}{R}VH\Big]=0 \qquad (10)$$

The GWCE, (10), is solved in conjunction with the primitive momentum equations in non-conservative form, (8) and (9).

8. The high accuracy of GWCE based FE solutions is a result of their excellent numerical amplitude and phase propagation characteristics. In fact, Fourier analysis indicates that in constant depth water and using linear interpolation, a linear tidal wave resolved with 25 nodes per wavelength is more than adequately resolved over the range of Courant numbers, $C = \sqrt{gh}\Delta t/\Delta x \leq 1.0$ (Luettich et al., 1992). Furthermore, the monotonic dispersion behavior of GWCE based FE solutions avoids generating artificial near $2 \cdot \Delta x$ modes which plague primitive based FE solutions (Platzman, 1981; Foreman, 1983). We note that the monotonic dispersion

behavior of GWCE based FE solutions is very similar to that associated with staggered finite difference solutions to the primitive shallow water equations (Westerink and Gray, 1991). GWCE based FE solutions to the shallow water equations allow for extremely flexible spatial discretizations which result in a highly effective minimization of the discrete size of any problem (Le Provost and Vincent, 1986; Foreman, 1988; Vincent and Le Provost, 1988; Westerink et al., 1992a).

9. The details of ADCIRC, our implementation of the GWCE based solution to the shallow water equations, are described by Luettich et al. (1992). As most GWCE based FE codes, ADCIRC applies three noded linear triangles for surface elevation, velocity and depth. Furthermore, the decoupling of the time and space discrete form of the GWCE and momentum equations, time independent and/or tri-diagonal system matrices, elimination of spatial integration procedures during time stepping, full vectorization of all major loops and pre-conditioned conjugate gradient matrix solvers results in a highly efficient code. A user manual further describes the details of using model ADCIRC-2DDI (Westerink et al., 1994b).

PART III: MODELING STRATEGY AND COMPUTATIONAL DOMAINS

10. The region of general interest consists of the vicinity of Galveston Bay, Texas. Due to the complex nature of tides and hurricanes in the Gulf of Mexico, it is very difficult to define surface elevations and/or currents which drive the boundaries of a regional model which encompasses only the immediate area of interest. The fundamental character of the tides in addition to flow features such as resonant shelf edge waves, hurricane forerunner and/or the complex wind patterns associated with a hurricane driving the flow onto the shelf, make it desirable to define larger computational domains which encompass regions well beyond the continental shelf adjacent to the area of interest. The modeling strategy has been to define basin wide or larger computational domains and to refine the region of interest to the degree required through the significant grid flexibility offered by GWCE based FE formulations. In this study three domains are examined in order to compute tidal elevations and flow in the vicinity of Galveston Bay. The first domain encompasses only Galveston Bay and the immediate vicinity as is shown in Figure 1a. A constant 1 component semi-diurnal forcing was applied on the shore parallel part of the open ocean boundary while no normal flow conditions were applied to the cross shelf portions of the open ocean boundary. The second domain encompasses the entire Gulf of Mexico as is shown in Figure 1b. This domain has the advantage that shelf to basin interactions are much better modeled than in a small regional domain. Two well defined open ocean boundaries of limited extent are used to specify the boundary forcing functions which define the interaction between the Atlantic Ocean and Caribbean Sea with the Gulf. The port boundary across the Strait of Florida runs from Cape Sable in Florida to Havana, Cuba. The second port boundary stretches across the Yucatan Channel from the vicinity of Cancun, Mexico to Cabo San Antonio, Cuba. These boundaries are forced with 5 astronomical tidal constituents (M_2, S_2, N_2, O_1 and K_1) from the *EASTCOAST_95* tidal constituent data base of Luettich and Westerink (1995). It is noted that several amphidromes or degenerate amphidromes exist in the vicinity of these ports, particularly for the diurnal constituents. Therefore the performance of the Gulf of Mexico domain with these boundaries should be carefully examined. The third domain encompasses a large portion of the

western North Atlantic ocean as well as the entire Gulf of Mexico and Caribbean Sea and is shown in Figure 1c. This domain, referred to as the Eastcoast domain, has a further advantage over the Gulf of Mexico domain in that it allows the basin to basin interactions between the Gulf of Mexico and the Atlantic ocean and Caribbean Sea to develop naturally, not requiring the details of these interactions to be specified in the ports where the response functions can be quite complex. The open ocean boundary in the Eastcoast model is driven with 5 constituents (M_2, S_2, N_2, O_1 and K_1) from LeProvost's (1995) FES95.2 data base. This data base was developed using a global tidal model and has been found to perform very well in deep ocean waters.

11. The finite element grids used for our simulations are shown in Figure 2. The Galveston and vicinity domain grid, designated as *GAL_G03_R4* grid and shown in Figure 2a, was obtained from Dr. Matsumoto at the Texas Water Development Board (1997). This grid was transformed to spherical coordinates and bathymetry was transformed to metric units. This grid contains 2213 nodes and 3397 elements. The Gulf of Mexico grid, designated as *GOMGAL_G05_R4* grid and shown in Figure 2b, was obtained from part of the *EASTCOAST_95* grid used to develop the corresponding data base. Extensive grid refinement was performed throughout the Gulf. Furthermore the *GAL_G03_R4* grid was incorporated exactly into the *GOMGAL_G05_R4* grid. It is noted that all the grid editing was done with XMGREDIT, a flexible interactive grid generation facility developed by Turner and Baptista (1991). The Gulf of Mexico grid, *GOMGAL_G05_R4* grid contains 11713 nodes and 21216 finite elements. The Eastcoast grid is designated as *EASTGAL_G03_R1* and is shown in Figure 2c. This grid is also based on the *EASTCOAST_95* grid, again with extensive grid refinement throughout the entire domain. This new eastcoast grid, *EASTGAL_G03_R1* identically incorporates the Gulf of Mexico grid, *GOMGAL_G05_R4* (and therefore also the Galveston grid, *GAL_G03_R4*). This grid contains 39812 nodes and 74246 finite elements. The level of grid refinement in this grid is based on numerous grid convergence studies as well as on the response functions obtained from previous computations. In general, the deepest waters in the Atlantic are relatively coarsely discretized

while the continental shelf waters and regions of detailed interest, in this case Galveston Bay and vicinity, are very finely resolved. The largest finite elements in this grid have a size of O(40) km while the finest elements are sized O(150) m.

12. The bathymetry in the three grids was obtained from a variety of sources. For the Galveston grid the bathymetry provided with the original grid by Dr. Matsumoto (1997) was used. For the Gulf of Mexico and Eastcoast grids, the bathymetry was obtained from the ETOPO5 data base from the National Center for Atmospheric Research and was supplemented by the National Ocean and Atmospheric Administration Digital U.S. Coastal Hydrography sounding data base (distributed by NOAA National Geophysical and Solar-Terrestrial Data Center in Boulder, CO) in all U.S. continental shelf waters with the exception of in the Galveston Bay domain, where the original bathymetry from the Galveston grid was applied. For actual simulations a minimum bathymetry of 0.5 meters was specified.

PART IV: TIDAL COMPUTATIONS

13. Tidal simulations for all three domains/grids were performed with the ADCIRC code run in depth integrated mode (2DDI option). The Gulf Mexico and Eastcoast computations were compared to long term field data obtained from the International Hydrographic Office (IHO), NOAA and Reid and Whitaker (1981) at nineteen elevation stations throughout the Gulf shown in Figure 3 and listed in Table 1. An additional 7 stations were defined to obtain detailed elevation and current data within the immediate vicinity of Galveston Bay. All simulations are entirely predictive and no calibration procedures were performed.

14. The open ocean boundary for the *GAL_G03_R4* simulation was forced with the M_2 constituent only with an amplitude of 0.50 m on the shore parallel portion of the open ocean boundary while weak no normal flow conditions were specified on the cross shelf portions of the open ocean boundary. No tidal potential forcing function was specified for this simulation. The *GOMGAL_G05_R4* simulation was forced with on the Strait of Florida and the Yucantan Channel using the K_1, O_1, M_2, N_2 and S_2 astronomical tidal constituents from the *EASTCOAST_95* tidal constituent data base of Luettich and Westerink (1995). The open ocean boundary in the *EASTGAL_G03_R1* model is driven using the K_1, O_1, M_2, N_2 and S_2 astronomical tidal constituents from LeProvost's (1995) FES95.2 data base. Both the *GOMGAL_G05_R4* and *EASTGAL_G03_R1* models applied tidal potential forcing functions due to the size of each domain as well as the resonant features of the Gulf. An effective tidal potential forcing within the interior domain was applied for the same five constituents as are forced on the boundaries. Amplitudes for the interior domain forcing function are listed in Table 2. The values for the effective earth elasticity factor, α, for the various constituents were obtained from Wahr (1981) and are also listed in Table 2. No nodal factors or astronomical arguments were applied to either the boundary or the interior domain forcing functions.

15. A constant value for the bottom friction coefficient, equal to c_f = 0.003, and a constant value for lateral eddy diffusion/dispersion equal to 10 m^2/sec were used in the simulations and were applied throughout all three domains. The bottom friction value is based on natural

river channels which are plain, clean and straight (Chow, 1959). The lateral eddy viscosity value is a physically realistic value in ocean and estuarine applications. It is noted that it is not necessary to apply any eddy viscosity for numerical reasons but it is physically necessitated in order to develop eddying with estuarine inlets. All nonlinearities including finite amplitude effects, nonlinear friction and convective acceleration were taken into consideration in the computations. The flooding/drying feature of the model was also activated.

16. The model was spun up from homogeneous initial conditions and a time ramp was used in order to avoid problems with short period gravity modes and vortex modes in the subinertial frequency range in addition to a free Helmholtz mode (Reid and Whitaker, 1981). A very smooth hyperbolic tangent time ramp function which acts over essentially fifteen days was applied to both boundary conditions and direct forcing functions. We tested both significantly underdamped and correctly damped Gulf of Mexico systems in order to establish spin up requirements. Although a free Helmholtz mode was excited, its amplitude relative to the forced modes was always small due to the smoothness of the hyperbolic tangent time ramp function. In fact, our estimates indicate that even for the underdamped systems, the free mode was less than 0.1% of the representative forced mode after about thirty days. Therefore, a fifteen day spin up is more than adequate for all signals of interest. The actual simulations were run for 120 days of which the first 30 days were discarded. A time step of 15 seconds was used such that the maximum Courant number based on wave celerity is approximately equal to unity. The requirement on Courant number is related to the explicit treatment of the nonlinear terms. Time weighting factors of 0.35, 0.30 and 0.35 were used for the future, current and past time levels in the GWCE equation and a Crank-Nicholson scheme was used for the momentum equations. Finally, the parameter τ_0 was set equal to 0.005 which represents a balance between the primitive continuity and wave equation portions of the GWCE equation and results in excellent Fourier propagation properties as well as excellent mass balance characteristics (Kolar et al, 1994a).

17. The results of the three simulation are qualitatively compared for elevation and currents in a number of movie files for the three simulations: For the *GAL_G03_R4* simulation,

Table 1: Data Stations for elevation and currents used in the Galveston Bay Simulation

Station No.	Name	λ (degree)	ϕ (degree)	type	measured data
1	Key West, FL	-81.1800003	24.716000	elevation	yes
2	Naples, FL	-81.1800003	26.129997	elevation	yes
3	Cedar Key, FL	-83.1031602	29.130004	elevation	yes
4	St. Marks Light, FL	-84.1182998	30.065997	elevation	yes
5	Alligator Bayou, FL	-85.1750000	30.132773	elevation	yes
6	Bay St. Louis, MS	-89.1319555	30.270683	elevation	yes
7	Cat Island, MS	-89.1166000	30.233002	elevation	yes
8	Southwest Pass, LA	-89.1428001	28.929999	elevation	yes
9	Point au Fer, LA	-91.1449997	29.286004	elevation	yes
10	Galveston, TX	-94.1780646	29.296207	elevation	yes
11	Port Aransas, TX	-97.1057999	27.825004	elevation	yes
12	South Padre Island, TX	-97.1150001	26.066002	elevation	yes
13	Madero, Mexico	-97.1711313	22.216007	elevation	yes
14	Coatzacoalcos, Mexico	-94.1412003	18.233697	elevation	yes
15	Campeche, Mexico	-90.1642026	19.850312	elevation	yes
16	Progreso, Mexico	-89.1650002	21.317480	elevation	yes
17	IAPSO #30-1.2	-89.1650002	24.760004	elevation	yes
18	IAPSO #30-1.2.13	-84.1249999	26.700001	elevation	yes
19	Havana, Cuba	-82.1366600	23.165764	elevation	yes
20	Galveston Bay - E01	-95.088247	29.068228	current	no
21	Galveston Bay - E02	-94.670157	29.331112	current	no
22	Galveston Bay - E03	-94.50731	29.494837	current	no
23	Galveston Bay - E04	-95.001626	29.229648	current	no
24	Galveston Bay - E05	-94.835314	29.441799	current	no
25	Galveston Bay - E06	-94.652833	29.525968	current	no
26	Galveston Bay - E07	-94.809905	29.665481	current	no

Table 2: C_{jn}, the Tidal Potential Constants (in meters), and α_{jn}, the Effective Earth Elasticity Factor, for Tidal Potential Forcing Functions (from Reid, 1990 and Wahr, 1981).

Constituent	C_{jn}	α_{jn}
K_1	0.141565	0.736
O_1	0.100514	0.695
N_2	0.046398	0.693
M_2	0.242334	0.693
S_2	0.112841	0.693

gal_entire.flc and gal_z1.flc; for the *GOMGAL_G05_R4* simulation, gomgal_entire.flc, gomgal_z1.flc and gomgal_z2.flc; for *EASTGAL_G03_R1* simulation, eastgal_entire.flc, eastgal_z1.flc and eastgal_z2.flc. These comparisons indicate that the wave propagation patterns and the associated currents are entirely in the cross shelf direction and are clearly not correct in the *GAL_G03_R4* simulation. The results from the *GOMGAL_G05_R4* and *EASTGAL_G03_R1* simulations are in fact very similar. These results indicate that the tides within the Gulf are predominantly diurnal or mixed with the exception of the tides on the west Florida Shelf, where they are strongly semi-diurnal. Reid and Whitaker (1981) note that the semi-diurnal tides off of Florida are associated with the near resonant response to the direct tidal potential forcing and they hypothesize that this response is related to the excitation of shelf edge waves.

18. Tidal elevations amplitudes and phases for the 5 forcing constituents (K_1, O_1, M_2, N_2 and S_2) are compared to data at the 19 measurement stations for the *GOMGAL_G05_R4* and *EASTGAL_G05_R4* simulations in Figure 4. Overall the quality of the comparisons is good, in particular when considering that no tuning was performed in our simulation. It is noted that the diurnal constituents, which in fact dominate the tidal signal in most of the Gulf with the exception of on the Florida shelf, are somewhat over-predicted throughout the Gulf. Furthermore the *GOMGAL_G05_R4* and *EASTGAL_G03_R1* simulations result in very similar response functions throughout the domain. It is clear that for the current discretizations, bathymetric definition and parameter specifications (including bottom friction and tidal potential reduction factors), there is

no justification for including the additional regions defined in the *EASTGAL_G03_R1* domain in order to study tides in the vicinity of Galveston Bay. Therefore for purposes of tidal calculations, the *GOMGAL_G05_R4* domain should be more than sufficient to define the character of tidal wave propagation and the associated current patterns in the vicinity of Galveston Bay. A fourteen day time history of tidal predicted tidal elevations (based on both a 5 constituent resynthesis of ADCIRC harmonically decomposed data and the raw time history produced by ADCIRC) at all nineteen data stations are shown in Figure 5.

19. It is noted that the model performance was excellent. The *GAL_G03_R4* grid ran at 0.045 CPU seconds per time step, the *GOMGAL_G05_R4* grid ran at 0.62 CPU seconds per time step and the *EASTGAL_G03_R1* ran at 2.21 CPU seconds on a SUN Ultra 140e.

PART V: DISCUSSION AND CONCLUSIONS

20. Large domains are extremely important in developing a truly predictive flow computation capability. Large domains reduce or even eliminate the tedious and very difficult task of boundary condition calibration and allow for a much more accurate interaction between the coastal region of immediate interest and the adjacent waters. Application of the *GOMGAL_G05_R4* domain allowed for the tides to propagate along the shelf in the vicinity of Galveston Bay and allowed for the development of realistic current patterns in the area. It was noted that the *EASTCOAST_95* tidal constituent data base did a realistic job of forcing the *GOMGAL_G05_R4* domain and that application of the *EASTGAL_G04_R1* domain was not necessary to obtain good tidal predictions in the Gulf and Mexico and in the vicinity of Galveston Bay. However it is noted that for hurricane storm surge predictions, the larger Eastcoast domain may be necessary for certain applications (Blaine et al, 1995). While the Gulf domain does an excellent job at representing peak hurricane storm surge, the Eastcoast domain in necessary to represent hurricane storm surge forerunner.

21. It is critical to resolve coastal areas for both tide and storm surge simulations. In order to correctly predict tidal elevations at open water locations, we must resolve the tidal waves of interest with a sufficient number of grid points. Our experience indicates that about thirty km resolution in deep waters such as in the Gulf of Mexico and Atlantic ocean is generally sufficient to accurately represent tidal flows with model ADCIRC. However, correct predictions of tidal velocities in a near coastal region may require a much greater level of grid detail. The tides propagating onto the shelf requires significantly increased level of grid resolution. Furthermore, near coastal bathymetry and geometry force the wavenumber content of the flow to be much higher than in the deep ocean. In fact, the flow adjacent to the land boundary may have the same or a higher wavenumber content than the land boundary itself. Not providing sufficient coastal resolution can drastically alter the flow results such that the entire character of the flow is misrepresented. In order to capture the details of the flow in the vicinity of Galveston Bay, required that we increase the grid resolution to a O(100m) near the breakwater tips. The numerical discretization

can be localized to account for the regional discretization requirements such that there is a high density of nodes in coastal regions and a much lower nodal density in more remote deep sea regions. In this way, accuracy can be achieved at a minimum expense in terms of the total number of grid points.

22. The FE method can be realistically implemented within the framework of GWCE formulations and applied to grids with highly variable nodal densities. The computations presented are unprecedented in their scope, level of localized detail and degree of grid size variability. They demonstrate that the accuracy performance of GWCE based FE formulations is excellent over a very wide range of grid and flow conditions. Furthermore, the flow results are inherently smooth without the use of numerical or nonphysical damping. Finally, the algorithms applied lead the very good CPU performance.

23. Currently, we are developing an updated data base for the Eastcoast domain which through increased resolution and parameter calibration strives to optimally match measured tidal elevation and flow data within the entire domain.

REFERENCES

Blain, C.A., J.J. Westerink and R.A. Luettich, "The Influence of Domain Size on the Response Characteristics of a Hurricane Storm Surge Model," *Journal of Geophysical Research*, **99, C9**, 18467-18479, 1994.

Blain, C.A., J.J. Westerink and R.A. Luettich, "Grid Convergence Studies for the Prediction of Hurricane Storm Surge," *International Journal for Numerical Methods in Fluids*, In Press, 1997.

Chow, V.T., *Open Channel Hydraulics*, McGraw-Hill, N.Y. N.Y., 1959.

Cialone, M.A., *The Coastal Modeling System: User's Manual*, CERC Miscellaneous Publication, U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS., 1991.

Drolet, J., and W.G. Gray, "On the Well Posedness of Some Wave Formulations of the Shallow Water Equations", *Advances in Water Resources*, 11, 84-91, 1988.

Drolet, J., "Application and Analysis of Wave Formulations of the Shallow Water Equations", Ph.D. thesis, Department of Civil Engineering, Princeton University, 1989.

Flather, R.A., Estimates of Extreme Conditions of Tide and Surge using a Numerical Model of the North-west European Continental Shelf, *Estuarine, Coastal and Shelf Science*, 24, 69-73, 1987.

Flather, R.A., A Numerical Model Investigation of Tides and Diurnal-Period Continental Shelf Waves along Vancouver Island", *J. of Physical Oceanography*, 18, 115-139, 1988.

Foreman, M.G.G., An Analysis of the Wave Equation Model for Finite Element Tidal Comparisons, *Journal of Computational Physics*, 52, 290-312, 1983.

Foreman, M.G.G., A Comparison of Tidal Models for the Southwest Coast of Vancouver Island, *Proceedings of the VII International Conference on Computational Methods in Water Resources*, held in Cambridge, MA, Elsevier, 1988.

Foreman, M.G.G., *Manual for Tidal Heights Analysis and Prediction*, Pacific Marine Science Report 77-10, Institute of Ocean Sciences, Patricia Bay, Victoria, B.C., 1977.

Foreman, M.G.G., "An Analysis of the 'Wave Equation' Model for Finite Element Tidal Computations", *Journal of Computational Physics*, 52, 290-312, 1983.

Foreman, M.G.G., "A Comparison of Tidal Models for the Southwest Coast of Vancouver Island", *Proceedings of the VII International Conference on Computational Methods in Water Resources*, held in Cambridge, MA, Elsevier, 1988.

Gray, W.G., Some Inadequacies of Finite Element Models as Simulators of Two-Dimensional Circulation, *Advances in Water Resources*, 5, 171-177, 1982.

Gray, W.G., A Finite Element Study of Tidal Flow Data for the North Sea and English Channel, *Advances in Water Resources*, 12, 143-154, 1989.

Gray, W.G., J. Drolet, and I.P.E. Kinnmark. "A Simulation of Tidal Flow in the Southern Part of the North Sea and the English Channel", *Advances in Water Resources*, 10, 131-137, 1987.

Gray, W.G., "A Finite Element Study of Tidal Flow Data for the North Sea and English Channel", *Advances in Water Resources*, 12, 143-154, 1989.

Grenier, R.R., R.A. Luettich and J.J. Westerink, "A Comparison of the Nonlinear Frictional Characteristics of Two-Dimensional and Three-Dimensional Models of a Shallow Tidal Embayment," *Journal of Geophysical Research*, **100, C7**, 13719-13735, 1995.

Hendershott, M.C., Long Waves and Ocean Tides, in *Evolution of Physical Oceanography*, 292-341, B.A. Warren and C. Wunsch, Eds., MIT Press, Cambridge, MA, 1981.

Kinnmark, I.P.E., The Shallow Water Wave Equations: Formulation, Analysis and Application, Ph.D. Dissertation, Department of Civil Engineering, Princeton University, 1984.

Kolar, R.L., and W.G. Gray., "Shallow Water Modeling in Small Water Bodies", *Proceedings of the Eighth International Conference on Computational Methods in Water Resources*, 149-155, 1990.

Kolar, R.L., J.J. Westerink, M.E. Cantekin and C.A. Blain, "Aspects of Nonlinear Simulations Using Shallow Water Models Based on the Wave Continuity Equation," *Computers and Fluids*, **23**, 3, 523-538, 1994a.

Kolar, R.L., W.G. Gray, J.J. Westerink and R.A. Luettich, "Shallow Water Modeling in Spherical Coordinates: Equation Formulation, Numerical Implementation and Application," *Journal of Hydraulic Research*, **32**, 1, 3-24, 1994b.

Kolar, R.L., W.G. Gray and J.J. Westerink, "Boundary Conditions in Shallow Water Models - An Alternative Implementation for Finite Element Codes," *International Journal for Numerical Methods in Fluids*, **22**, 603-618, 1996.

Le Provost, C. and P. Vincent, "Some Tests of Precision for a Finite Element Model of Ocean Tides," *Journal of Computational Physics*, 65, 273-291, 1986.

Le Provost, C., Personal communication, 1995.

Luettich, R.A. and J.J. Westerink, "A Solution for the Vertical Variation of Stress, Rather than Velocity, in a Three-Dimensional Circulation Model," *International Journal for Numerical Methods in Fluids*, **12**, 911-928, 1991.

Luettich, R.A., J.J. Westerink, and N.W. Scheffner, ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts and Estuaries, Report 1: Theory and Methodology of ADCIRC-2DDI and ADCIRC-3DL, *Coastal Engineering Research Center*, U.S. Army Engineers, 1992.

Luettich, R.A., S. Hu and J.J. Westerink, "Development of the Direct Stress Solution Technique for Three Dimensional Hydrodynamic Models Using Finite Elements," *International Journal for Numerical Methods in Fluids*, **19**, 295-319, 1994.

Lynch, D.R., Progress in Hydrodynamic Modeling, Review of U.S. Contributions, 1979-1982, *Reviews of Geophysics and Space Physics*, 21(3), 741-754, 1983.

Lynch, D.R., and W.G. Gray, A Wave Equation Model for Finite Element Tidal Computations, *Computers and Fluids*, 7, 207-228, 1979.

Lynch, D.R., "Comparison of Spectral and Time-Stepping Approaches for Finite Element Modeling of Tidal Circulation", *Oceans 81 Conference Record*, Vol. 2; Boston MA., September 16-18, IEEE Publ. No. 81CH1685-7, 1981.

Lynch, D.R., and F.E. Werner, "Three-Dimensional Hydrodynamics on Finite Elements, Part II: Nonlinear Time-Stepping Model", *International Journal on Numerical Methods in Fluids*, 1990.

Lynch, D.R., F.E. Werner, A. Cantos-Figuerola, and G. Parilla, "Finite Element Modeling of Reduced-Gravity Flow in the Alboran Sea: Sensitivity Studies", in *Seminario Sobre Oceanografia Fisica del Estrecho de Gibraltar*, Madrid, 283-295, 1988.

Lynch, D.R., F.E. Werner, J.M. Molines, and M. Fornerino, "Tidal Dynamics in a Coupled Ocean/ Lake System", *Estuarine, Coastal and Shelf Science*, 31, 319-343, 1990.

Matsumoto, J., Personal communication, 1997.

Pearson F., Map Projections: Theory and Applications, CRC Press, Inc., Boca Raton, Florida, 1990.

Platzman, G.W., Some Response Characteristics of Finite Element Tidal Models, *Journal of Computational Physics*, 40, 36-63, 1981.

Reid, R.O., and R.E. Whitaker, Numerical Model for Astronomical Tides in the Gulf of Mexico, Technical Report for the U.S. Army Engineers, Department of Oceanography, Texas A&M University, 1981.

Reid, R.O., Waterlevel Changes, *Handbook of Coastal and Ocean Engineering*, J. Herbich, Ed., Gulf Publishing, Houston, Texas, 1990.

Schureman, P. (1940). *Manual of Harmonic Analysis and Prediction of Tides*, Special Publication No. 98, U.S. Department of Commerce.

Schwiderski, E.W., On Charting Global Ocean Tides, *Reviews in Geophysics and Space Physics*, 18, 243-268, 1980.

Turner, P.J. and A.M. Baptista, ACE/Gredit Users Manual: Software for Semi-automatic Generation of Two-Dimensional Finite Element Grids", CCALMR Software Report SDS2(91-2), Oregon Graduate Institute, Beaverton, OR, 1991.

U.S. Geological Survey, Atlas of Tidal Elevation and Current Observations on the Northeast American Continental Shelf and Slope, U.S. Geological Survey Bulletin #1611, 1984.

Vincent, P. and C. Le Provost, Semidiurnal Tides in the Northeast Atlantic From a Finite Element Numerical Model, *Journal of Geophysical Research*, 93, No. C1, 543-555, 1988.

Wahr, J.M., Body Tides on an Elliptical, Rotating, Elastic and Oceanless Earth, *Geophys. J.R. astr. Soc.* 64, 677-703, 1981.

Walters, R.A., A Finite Element Model For Tides and Currents with Field Applications, *Comm. Applied Numerical Methods*, 4, 401-411, 1988.

Walters, R.A. and F.E. Werner, A Comparison of Two Finite Element Models of Tidal Hydrodynamics Using a North Sea Data Set, *Advances in Water Resources*, 12(4), 184-193, 1989.

Walters, R.A., "Numerically Induced Oscillations in Finite Element Approximations to the Shallow Water Equations", *International Journal for Numerical Methods in Fluids*, 3, 591-604, 1983.

Walters, R.A., "Finite Element Solution Methods for Circulation in Estuaries", *Finite Elements in Water Resources; Proceedings of the 5th International Conference*, J.P. Laible et al. (eds.) Burlington, VT., 1984.

Walters, R.A., "A Model for Tides and Currents in the English Channel and Southern North Sea", *Advances in Water Resources*, 10, 138-148, 1987

Werner, F.E., and D.R. Lynch, Harmonic Structure of English Channel/Southern Bight Tides from a Wave Equation in Simulation, *Advances in Water Resources*, 12, 121-142., 1989.

Westerink, J.J., K.D. Stolzenbach and J.J. Connor, General Spectral Computations of the Nonlinear Shallow Water Tidal Interactions within the Bight of Abaco, *Journal of Physical Oceanography*, 19, 1348-1371, 1989.

Westerink, J.J., and W.G. Gray, Progress in Surface Water Modeling, *Reviews of Geophysics*, April Supplement, 210-217, 1991.

Westerink, J.J., R.A. Luettich, A.M. Baptista, N.W. Scheffner and P. Farrar, Tide and Storm Surge Predictions Using a Finite Element Model, *Journal of Hydraulic Engineering*, 118, 1373-1390, 1992a.

Westerink, J.J., J.C. Muccino and R.A. Luettich, Resolution Requirements for a Tidal Model of the Western North Atlantic and Gulf of Mexico, in Proceedings of the IX International Conference on Computational Methods in Water Resources, Denver, CO, T.F. Russell et al. eds., Computational Mechanics Publications, Southampton, UK, 1992b.

Westerink, J.J., R.A. Luettich and J.C. Muccino, "Modeling Tides in the Western North Atlantic Using Unstructured Graded Grids," *Tellus*, **46A**, 178-199, 1994a.

Westerink, J.J., R.A. Luettich, C.A. Blain and N.W. Scheffner, ADCIRC: An Advanced Three-Dimensional Circulation Model for Shelves, Coasts and Estuaries, Report 2: Users Manual for ADCIRC-2DDI, *Coastal Engineering Research Center*, U.S. Army Engineers, 1994b.

Westerink, J.J., R.A. Luettich, J.K. Wu and R.L. Kolar, "The Influence of Normal Flow Boundary Conditions on Spurious Modes in Finite Element Solutions to the Shallow Water Equations," *International Journal for Numerical Methods in Fluids*, **18**, 1021-1060, 1994c.

Woodworth, P.L., Summary of Recommendations to the UK Earth Observation Data Centre (UK-EODC) by the Proudman Oceanographic Laboratory (POL) for Tide Model Corrections on ERS-1 Geophysical Data Records, Proudman Oceanographic Laboratory Comm., 1990.

LIST OF FIGURES

ADDITIONAL PRODUCTS DELIVERED WITH THIS REPORT

1. Grid file for Galveston Bay and vicinity: gal_g03_r4.grd
2. Input file for Galveston Bay and vicinity: gal_g03_r4.inp
3. Grid file for the Gulf of Mexico including Galveston Bay and vicinity: gomgal_g05_r4.grd
4. Input file for the Gulf of Mexico including Galveston Bay and vicinity: gomgal_g05_r4.inp
5. Grid file for the Eastcoast including Galveston Bay and vicinity: eastgal_g03_r1.grd
6. Input file for the Eastcoast including Galveston Bay and vicinity: eastgal_g03_r1.inp
7. ADCIRC source codes: adc31_06.f and srcv2d.f
8. ADCIRC setup program: adcsetup31_06.f
9. Movie files with results of gal_g03_r4 simulation
   (a) gal_entire.flc
   (b) gal_z1.flc
10. Movie files with results of gomgal_g05_r4 simulation
    (a) gomgal_entire1.flc
    (b) gomgal_z1.flc
    (c) gomgal_z2.flc
11. Movie files with results of eastgal_g03_r1 simulation
    (a) eastgal_entire1.flc
    (b) eastgal_z1.flc
    (c) eastgal_z2.flc

Figure 1. Maps of pertinent domains indicating bathymetry in meters.
(a) Galveston Bay, Texas and vicinity

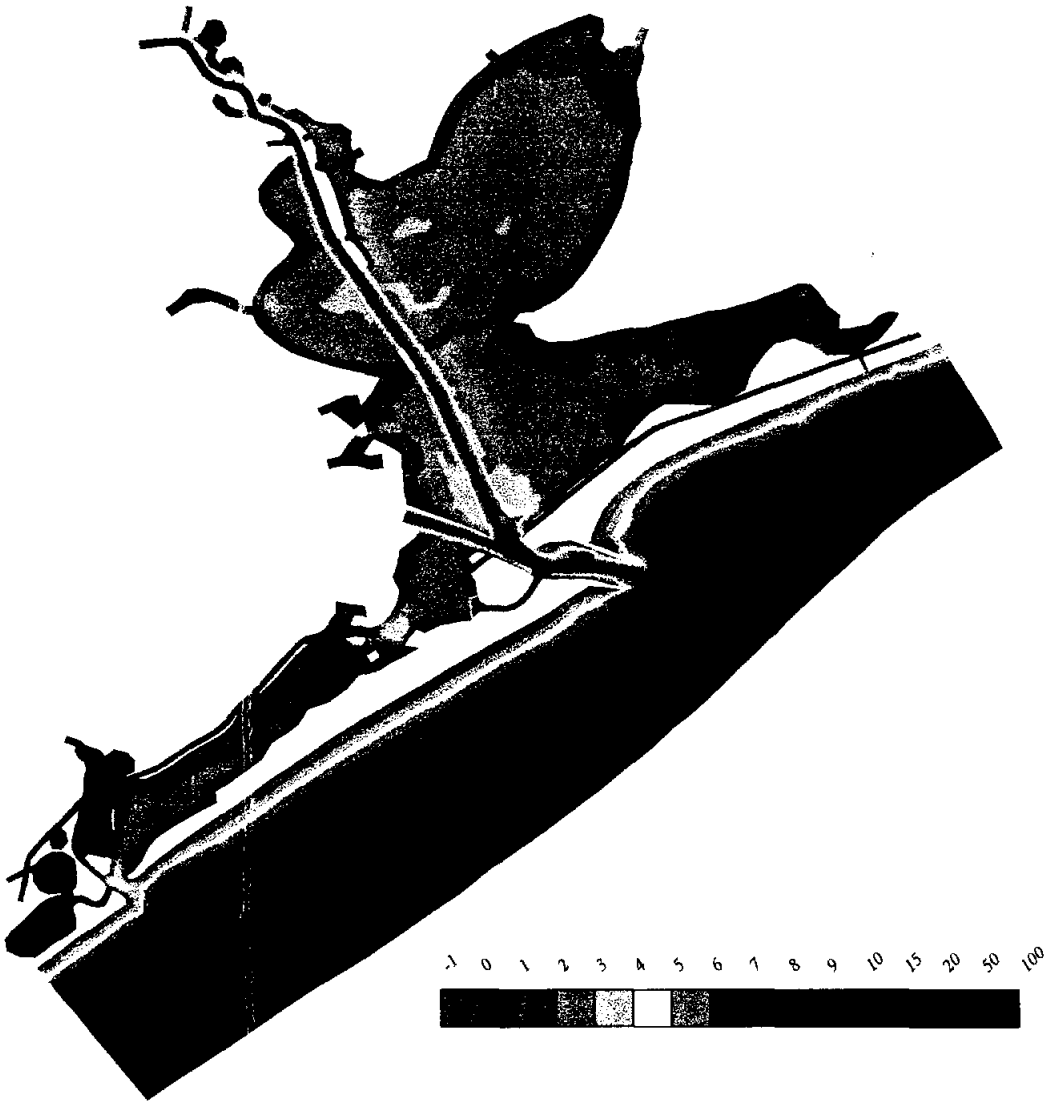


Figure 1. Maps of pertinent domains indicating bathymetry in meters.
(b) Gulf of Mexico

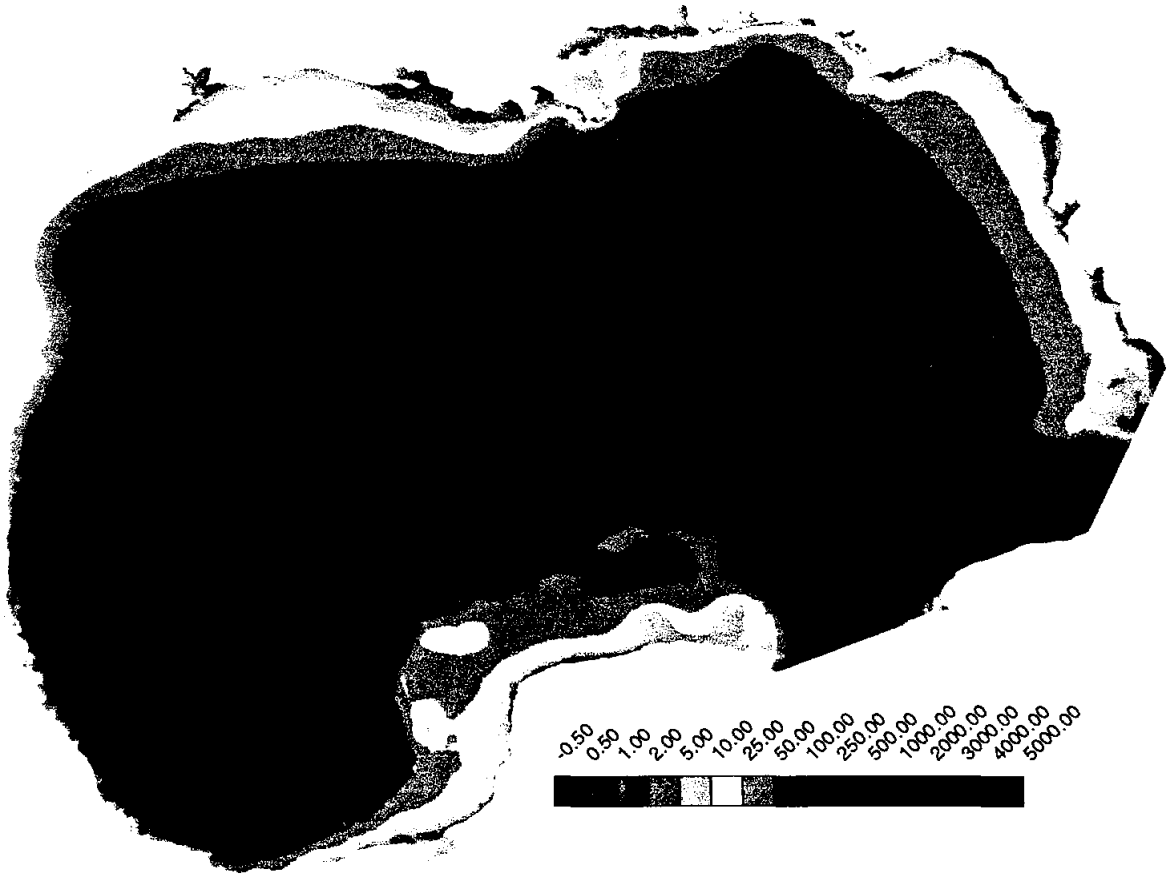


Figure 1. Maps of pertinent domains indicating bathymetry in meters.
(c) Eastcoast including the Gulf of Mexico and Caribbean Sea

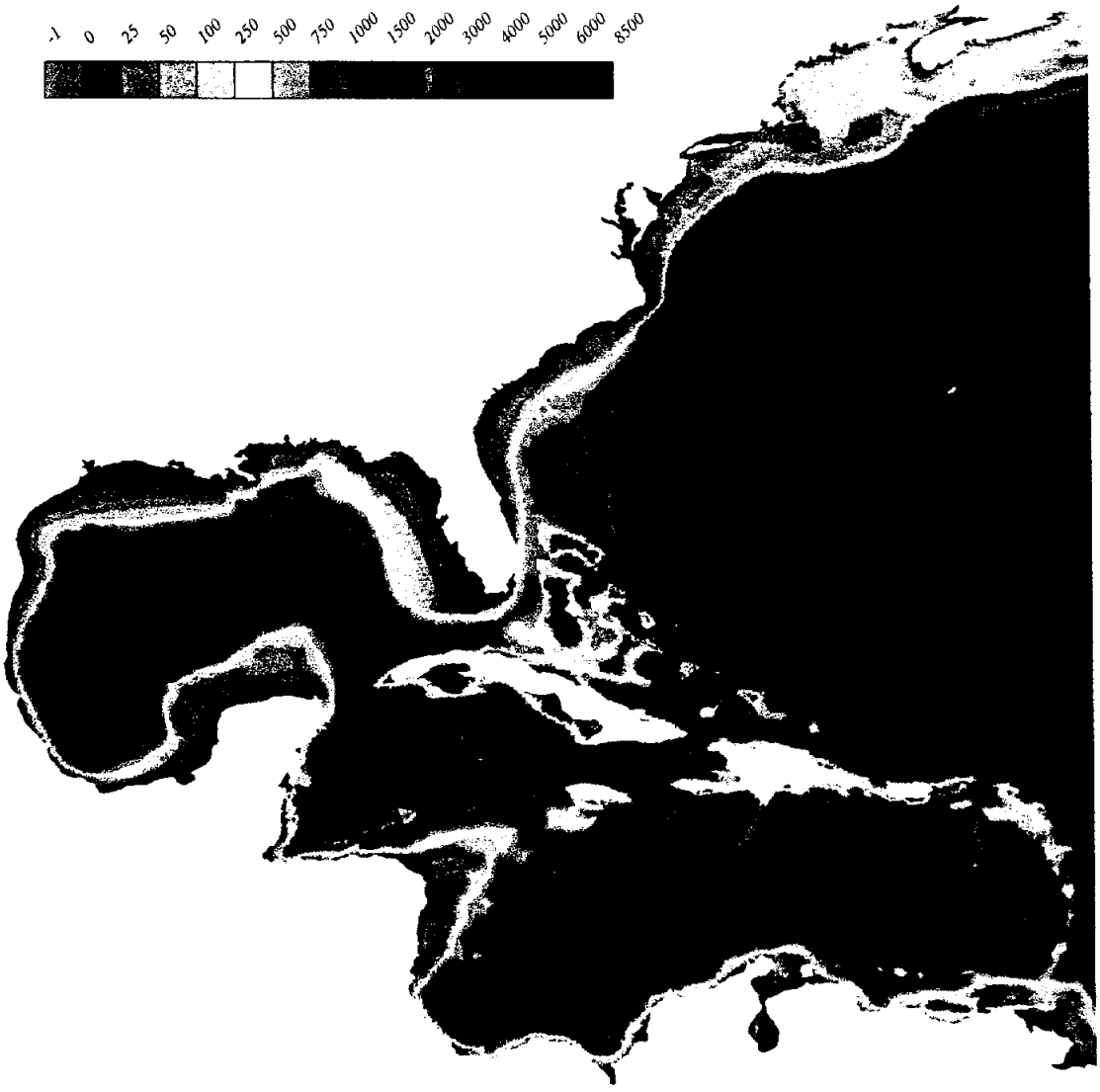


Figure 2. Finite element discretizations
(a) Galveston Bay, Texas and vicinity: *GAL_G03_R4*

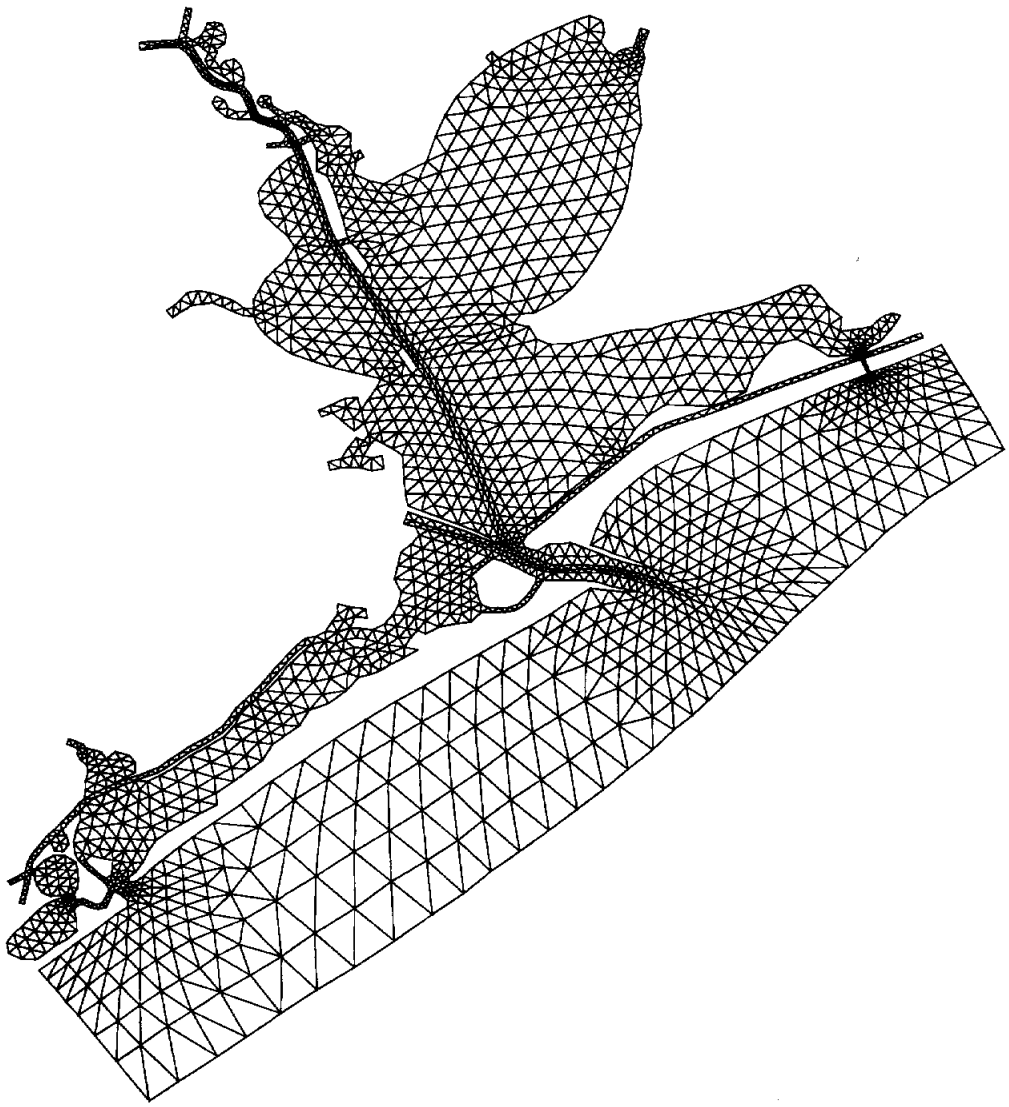


Figure 2. Finite element discretizations
(b) Gulf of Mexico: *GOMGAL_G05_R4*

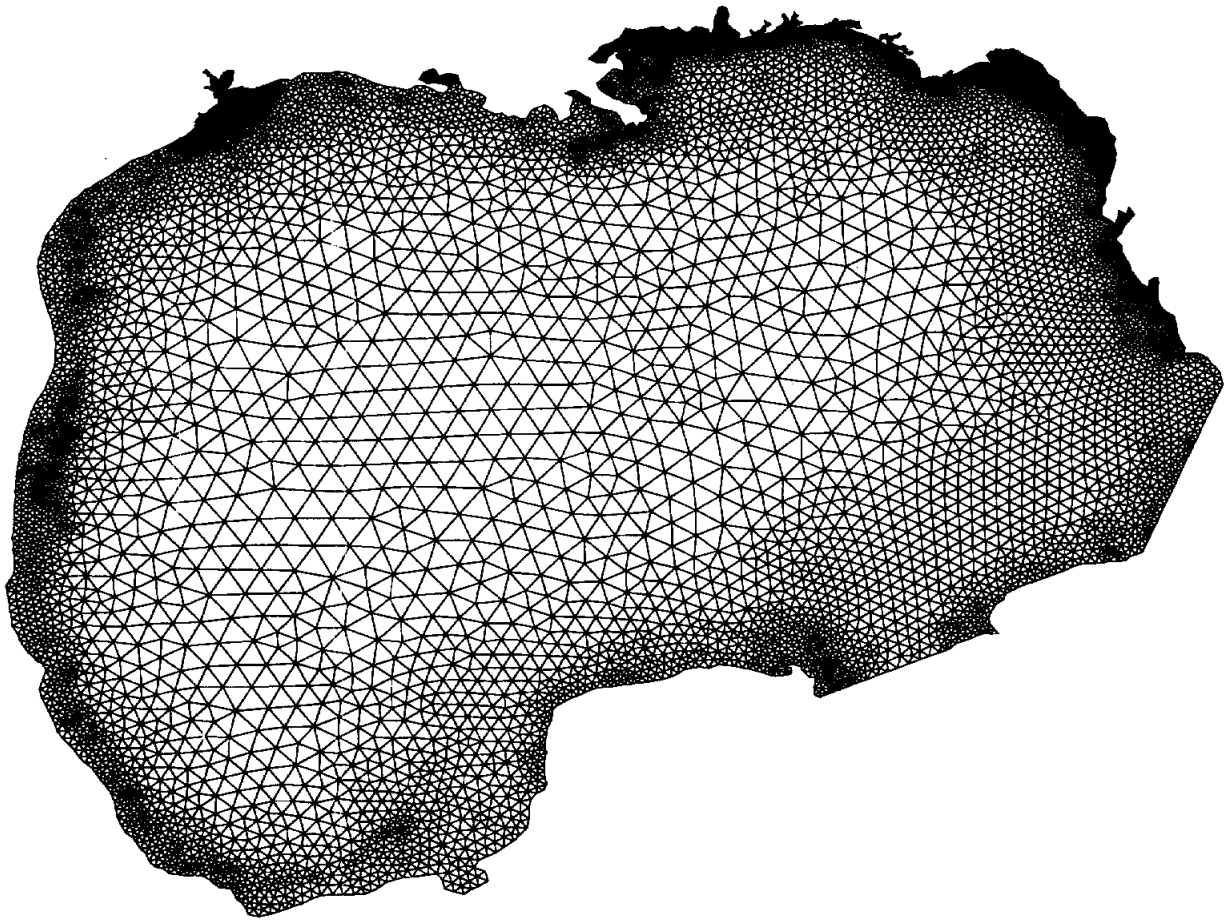


Figure 2. Finite element discretizations
(c) Eastcoast: *EASTGAL_G03_R1*

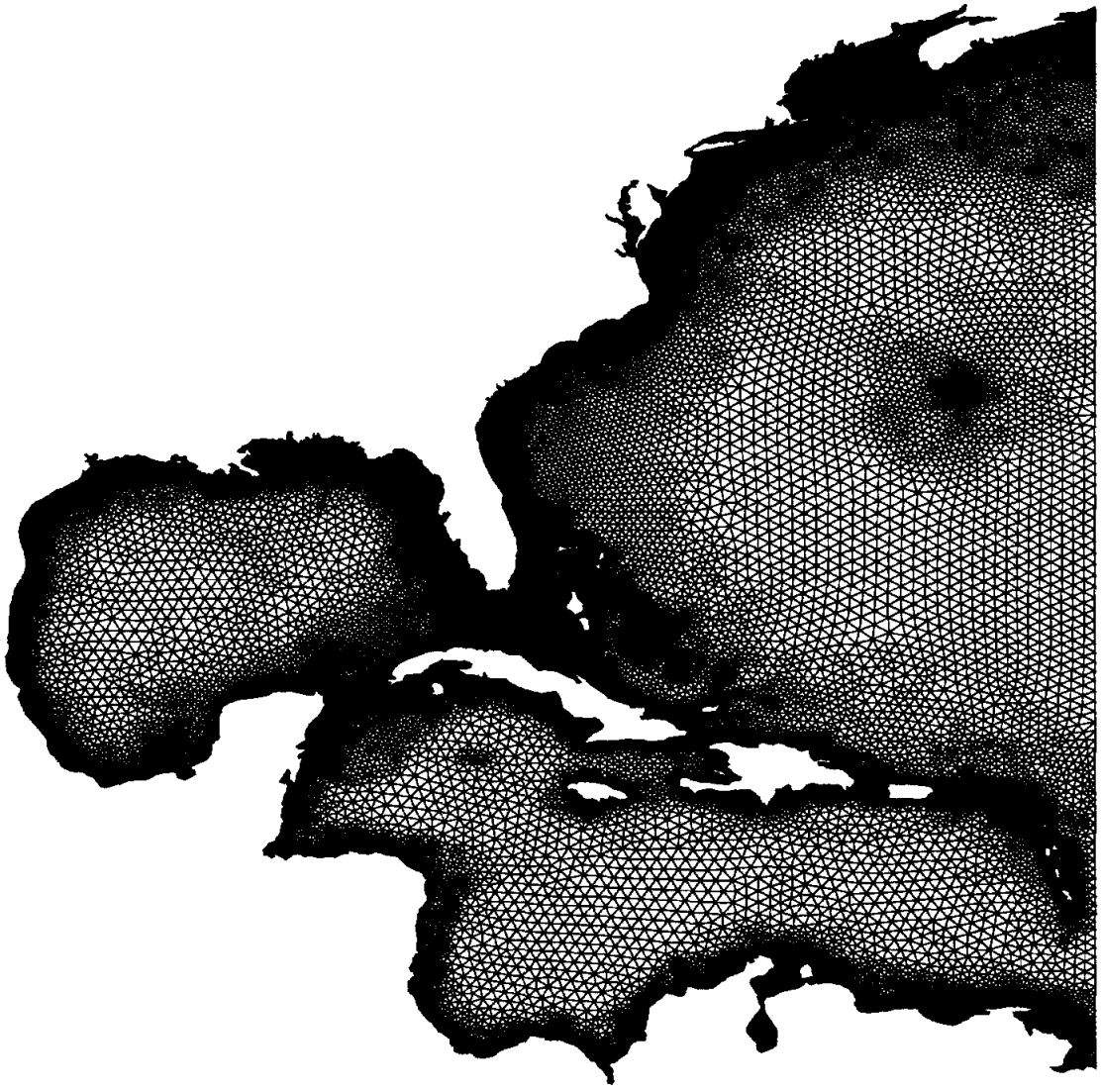


Figure 3. Locations of tidal elevation data (stations 1-19) and additional data (stations 20-26) recording stations.
(a) Gulf of Mexico

Figure 3. Locations of tidal elevation data (stations 1-19) and additional data (stations 20-26) recording stations.
(b) Detail of Galveston Bay, Texas

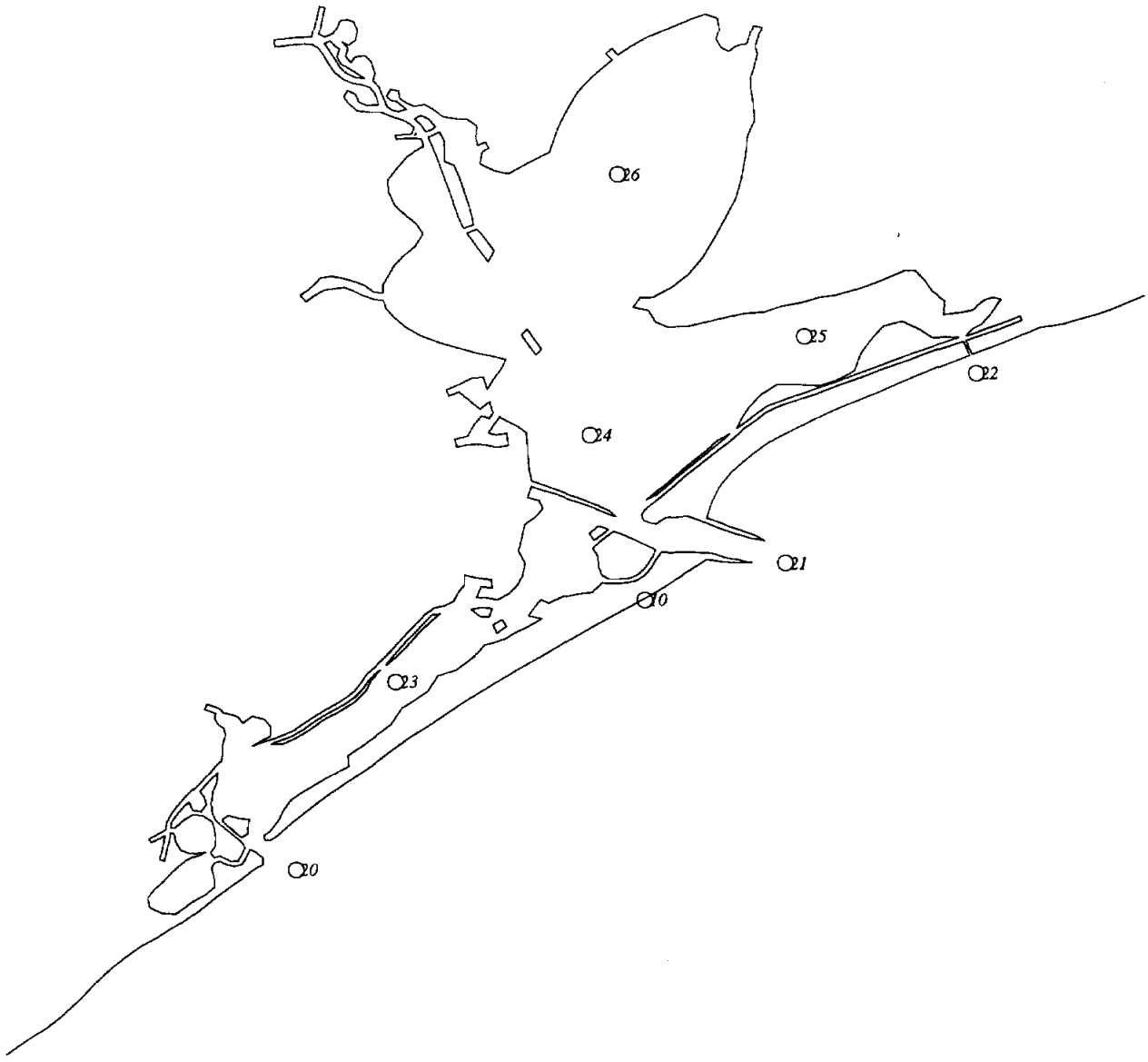


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (a) K_1 amplitude

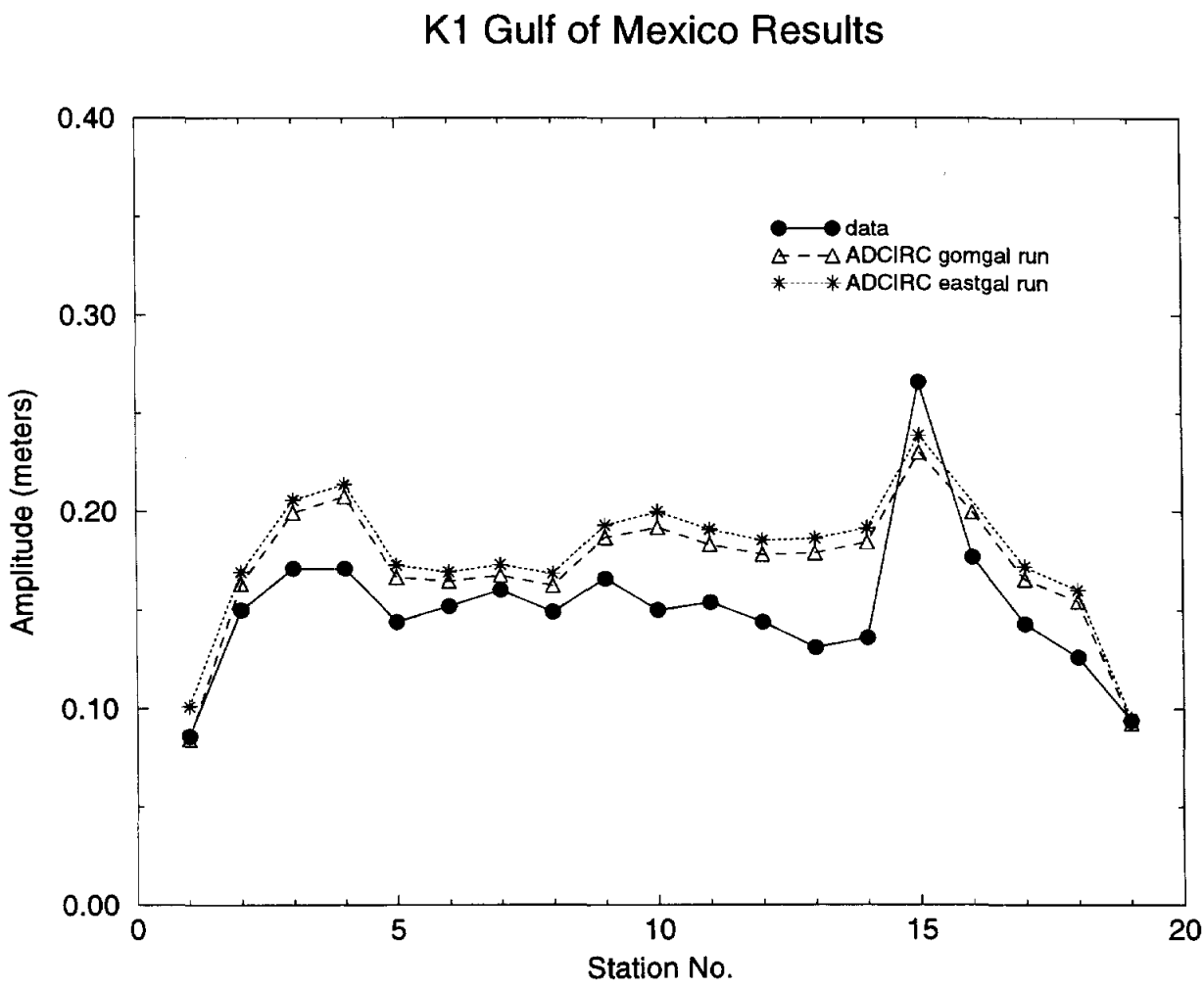


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (b) K_1 phase

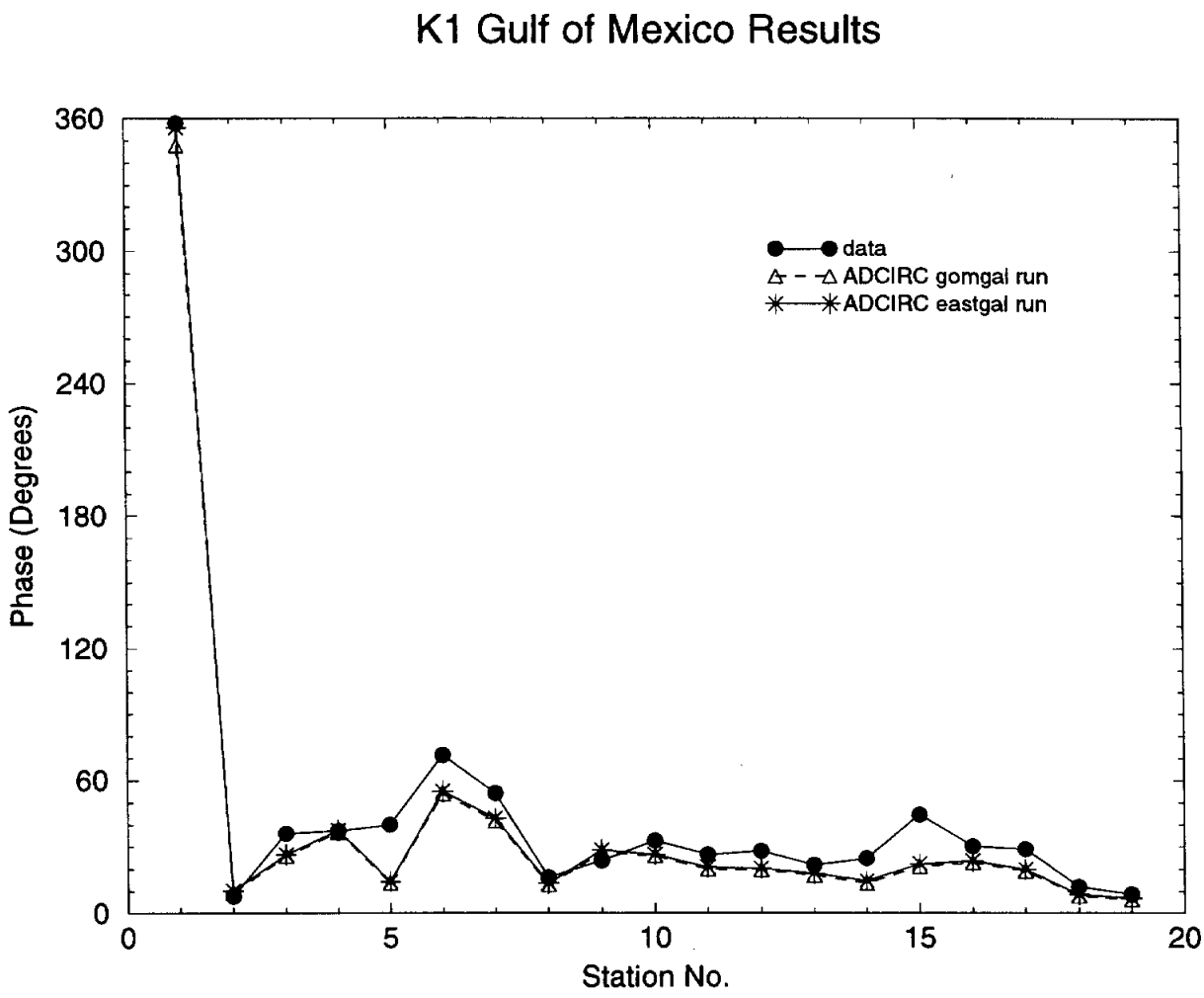


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(c) M_2 amplitude

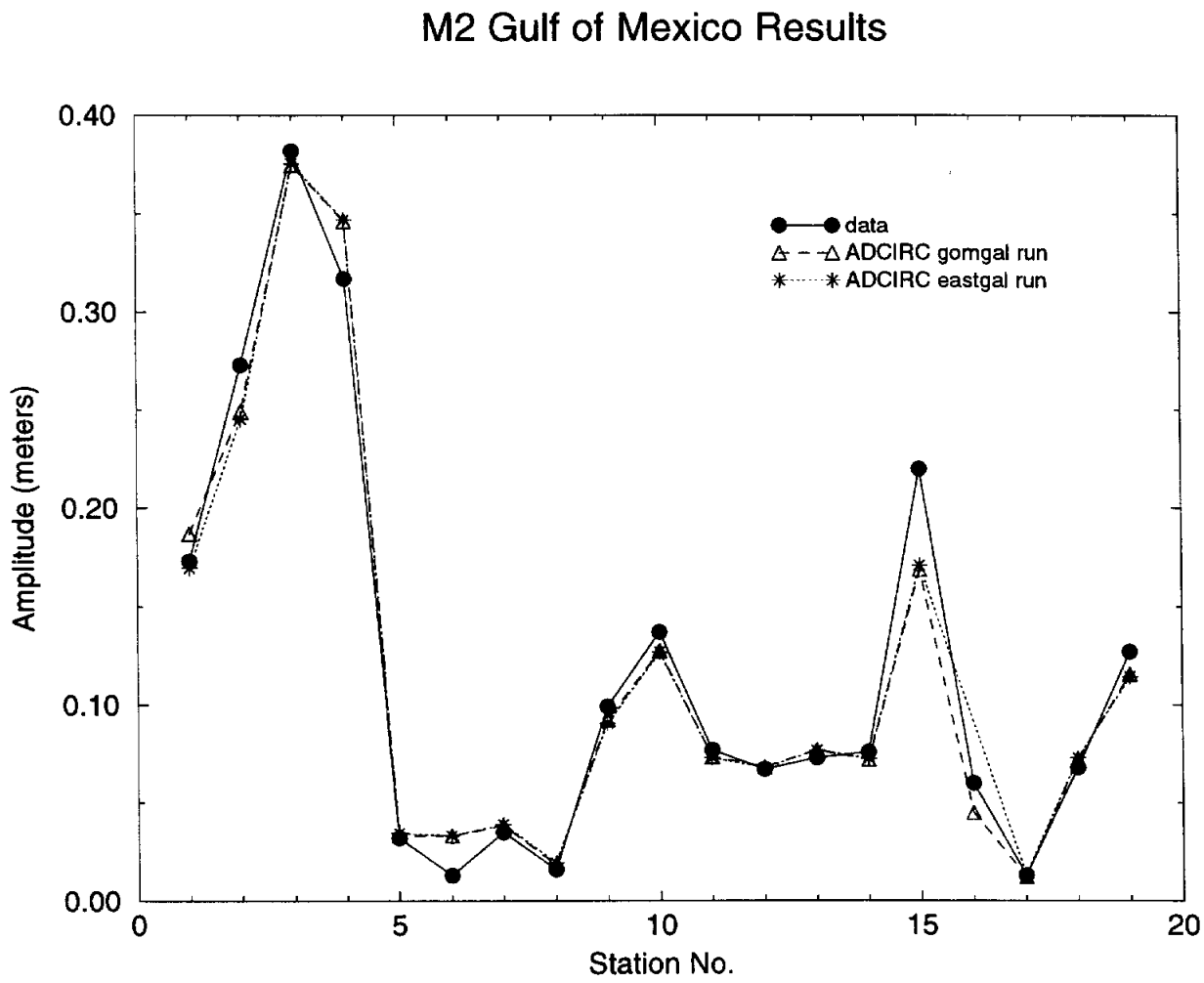


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (d) M_2 phase

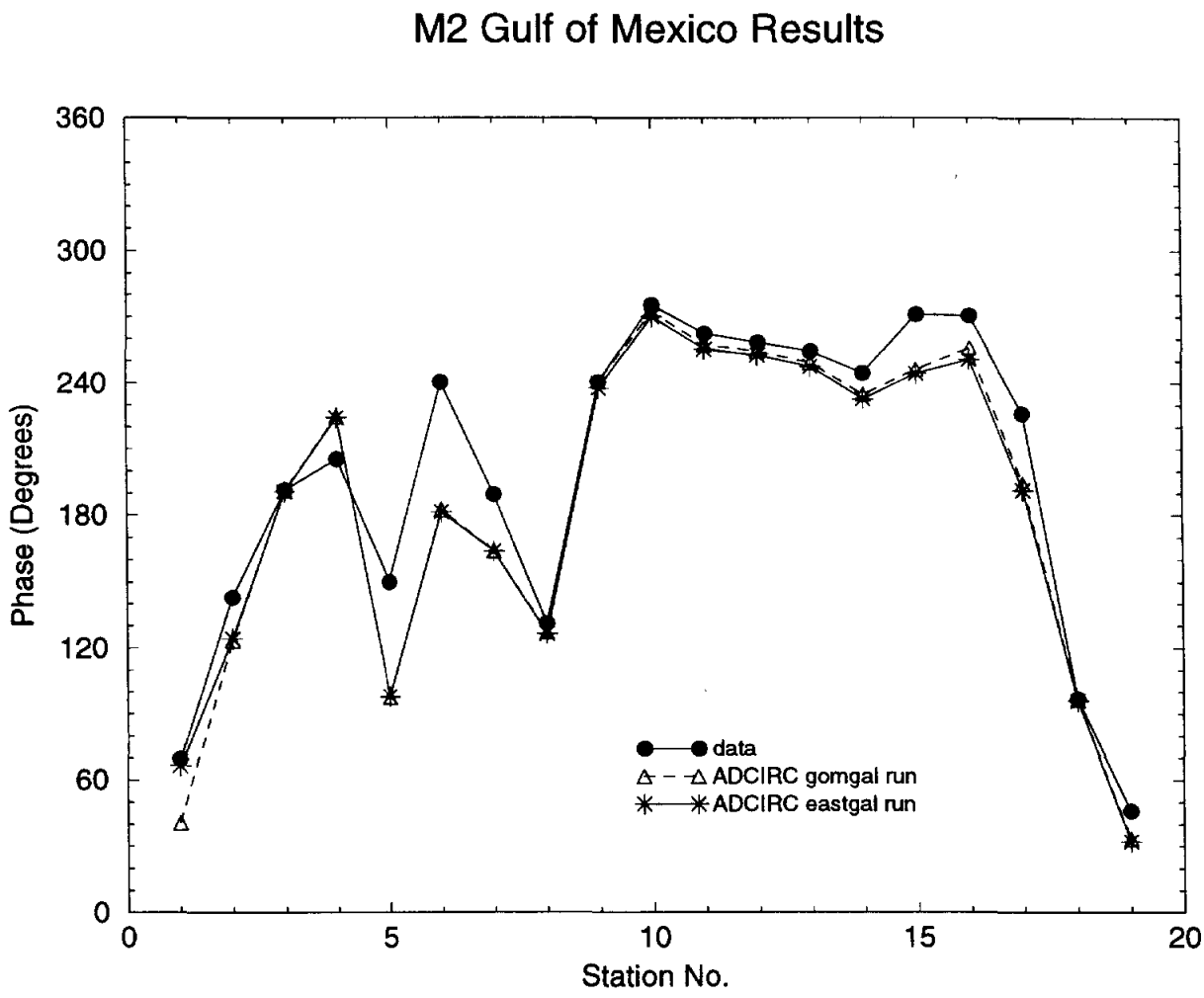


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (e) N_2 amplitude

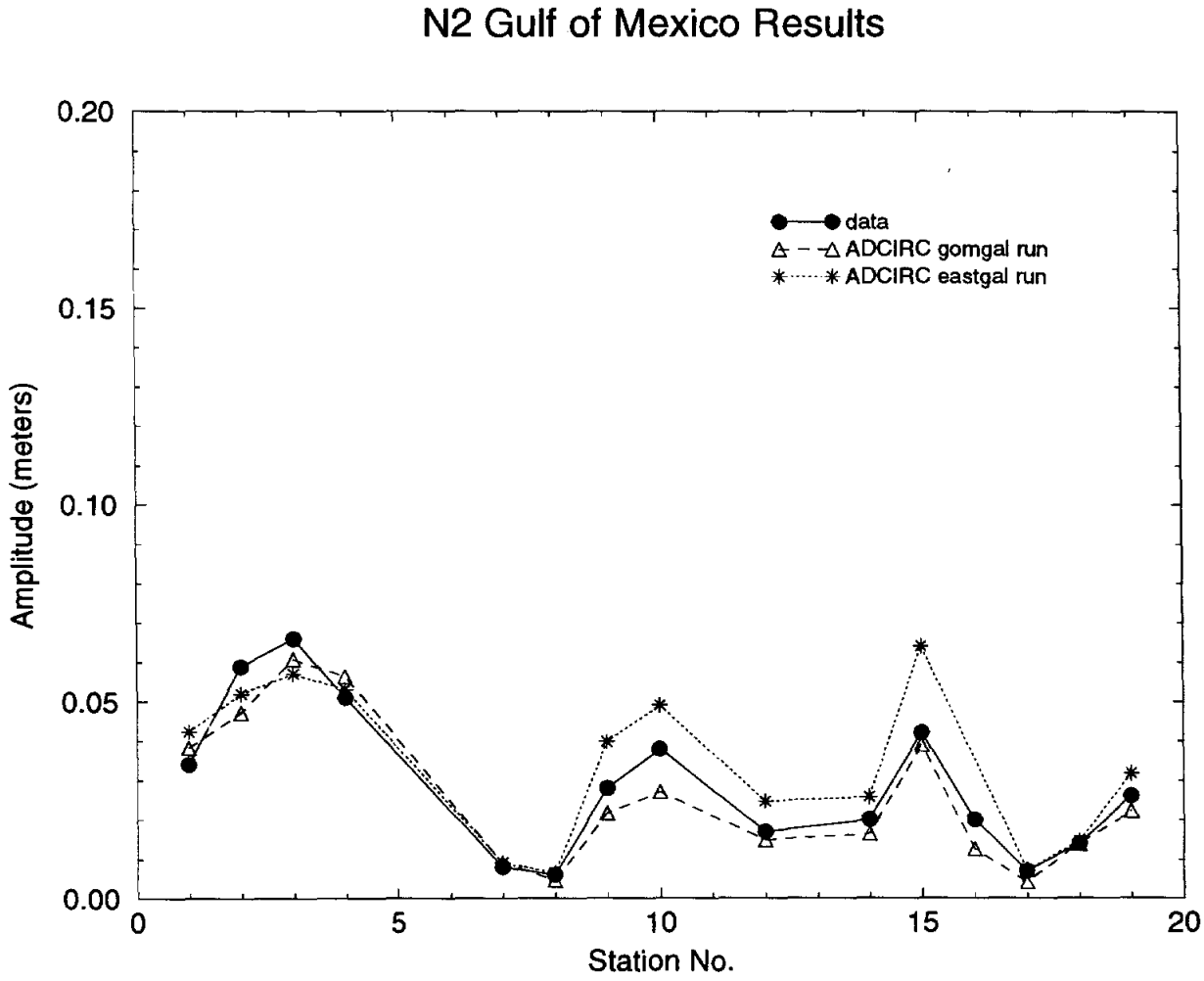


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (f) N_2 phase

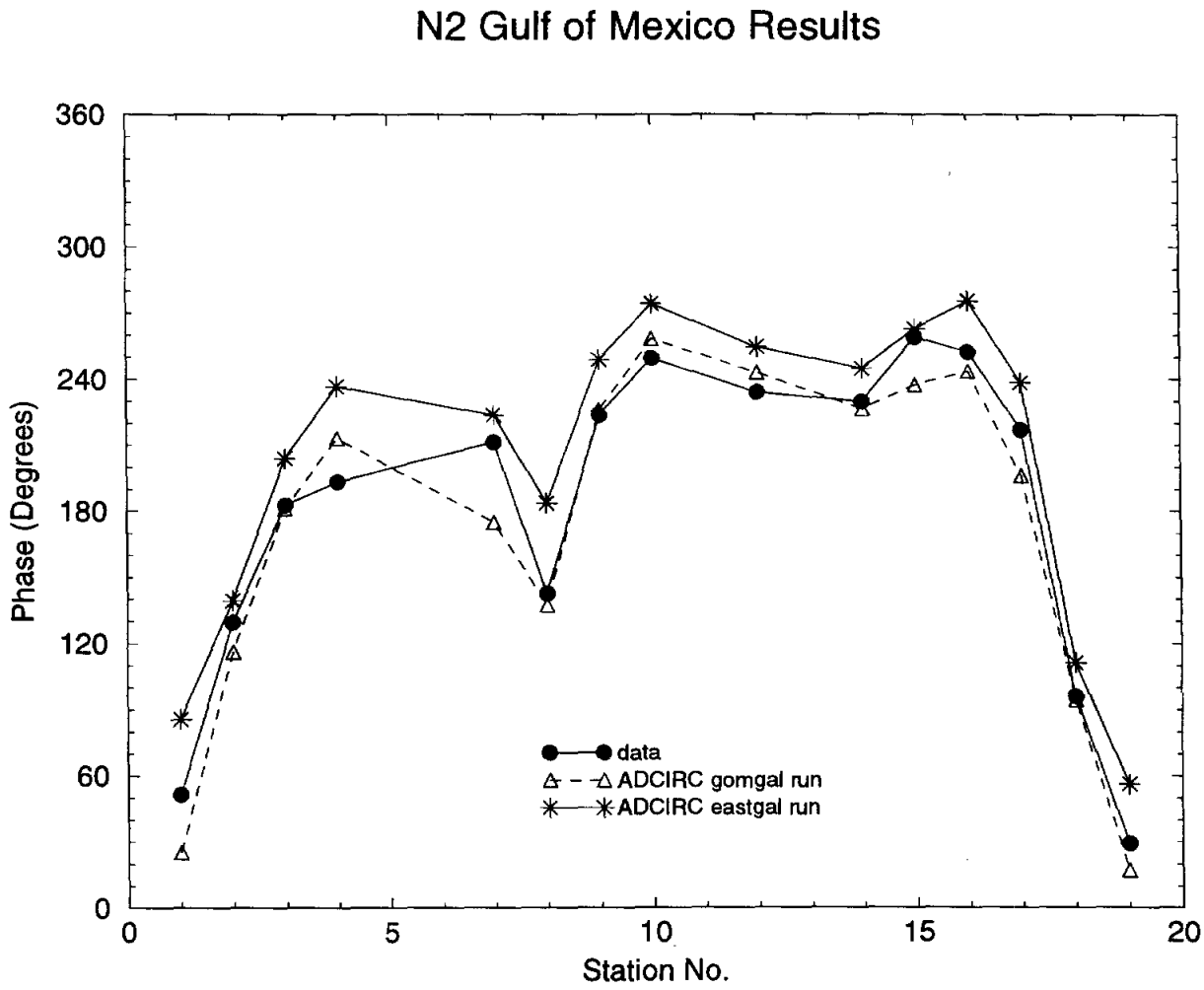


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values. (g) O_1 amplitude

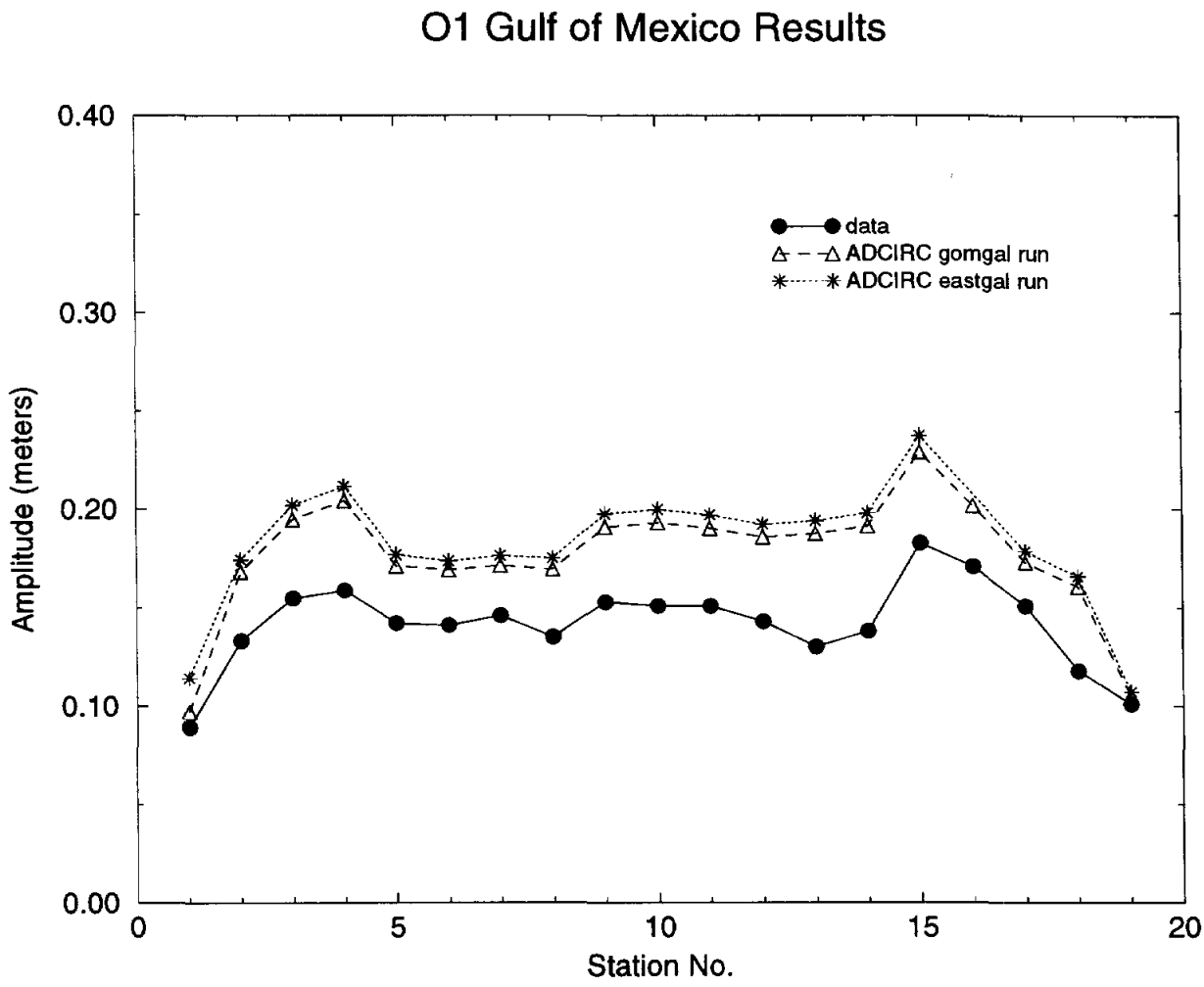


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(h) O_1 phase

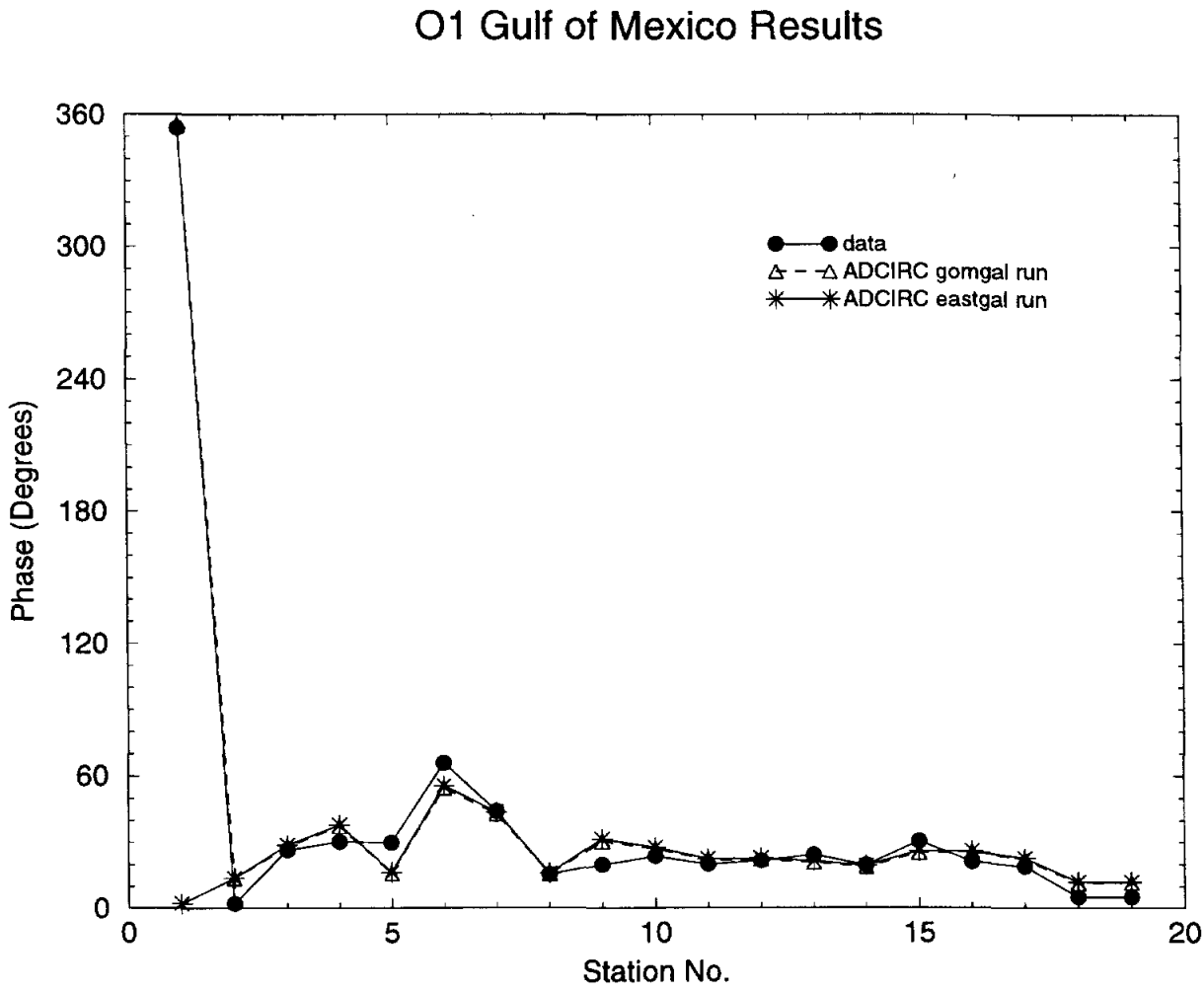


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(i) S_2 amplitude

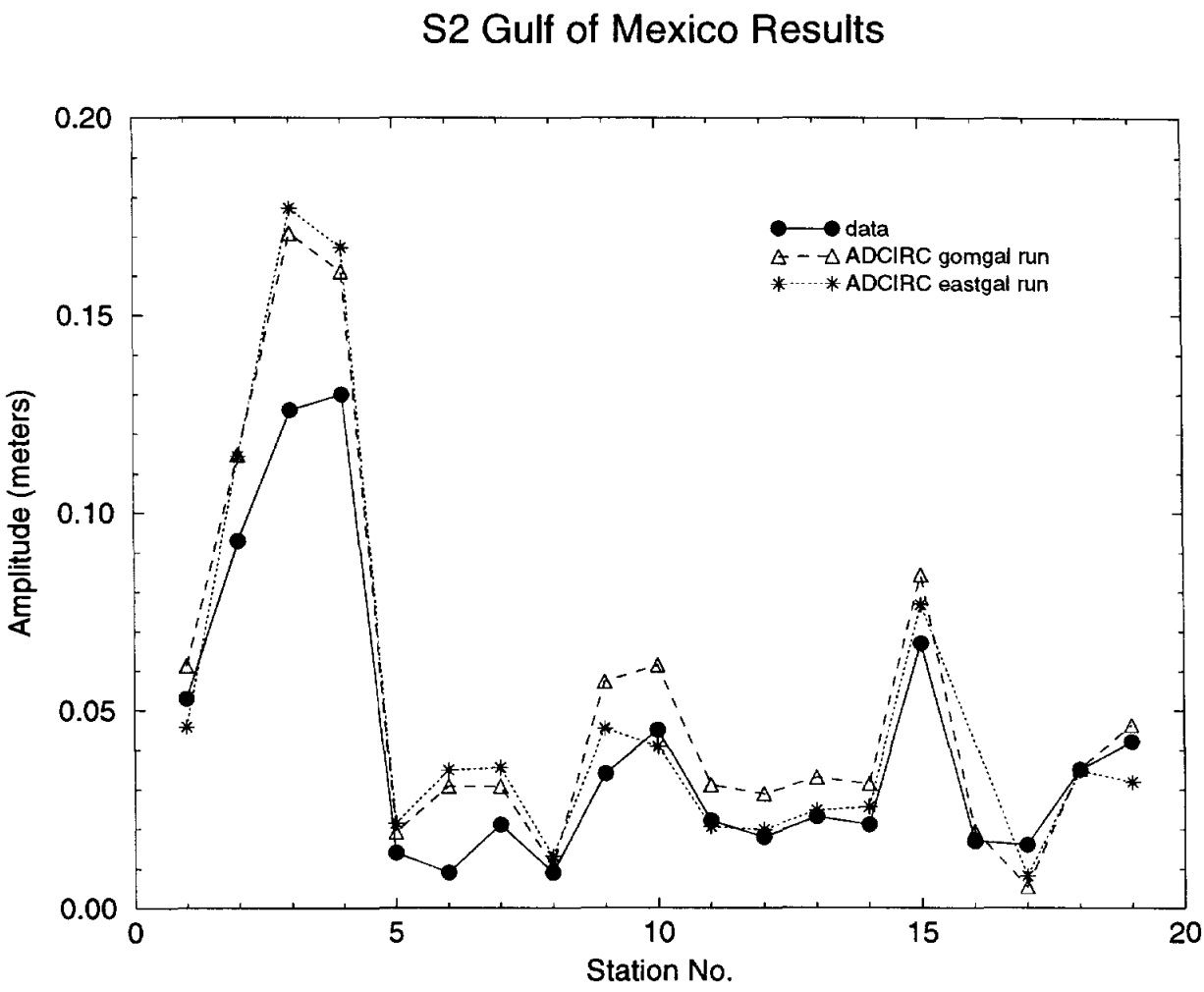


Figure 4. Comparison of ADCIRC computed tidal elevation amplitudes and phases using grids *GOMGAL_G05_R4* and *EASTGAL_G03_R1* with measured data values.
(j) S_2 phase

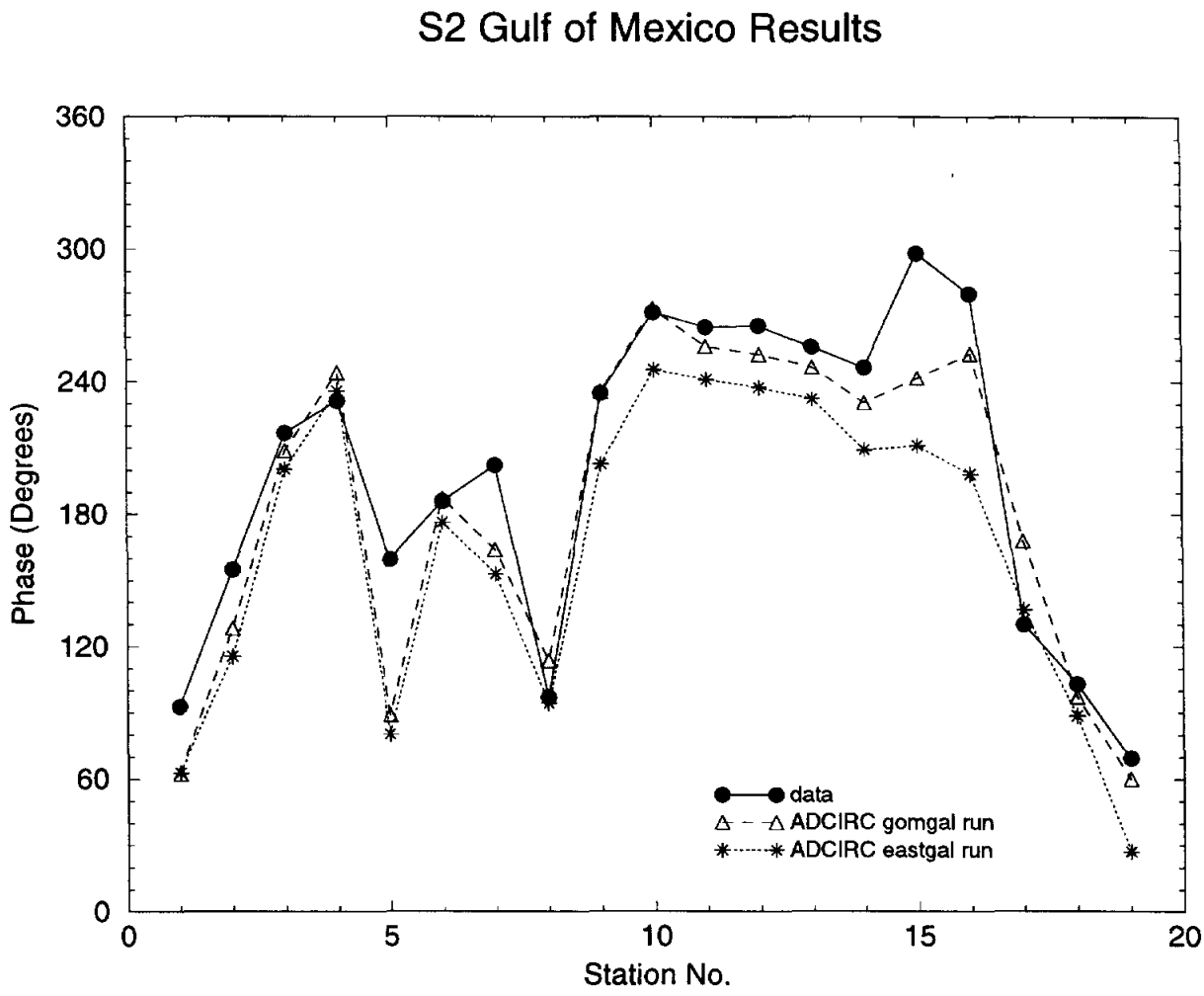


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(a) Key West, FL

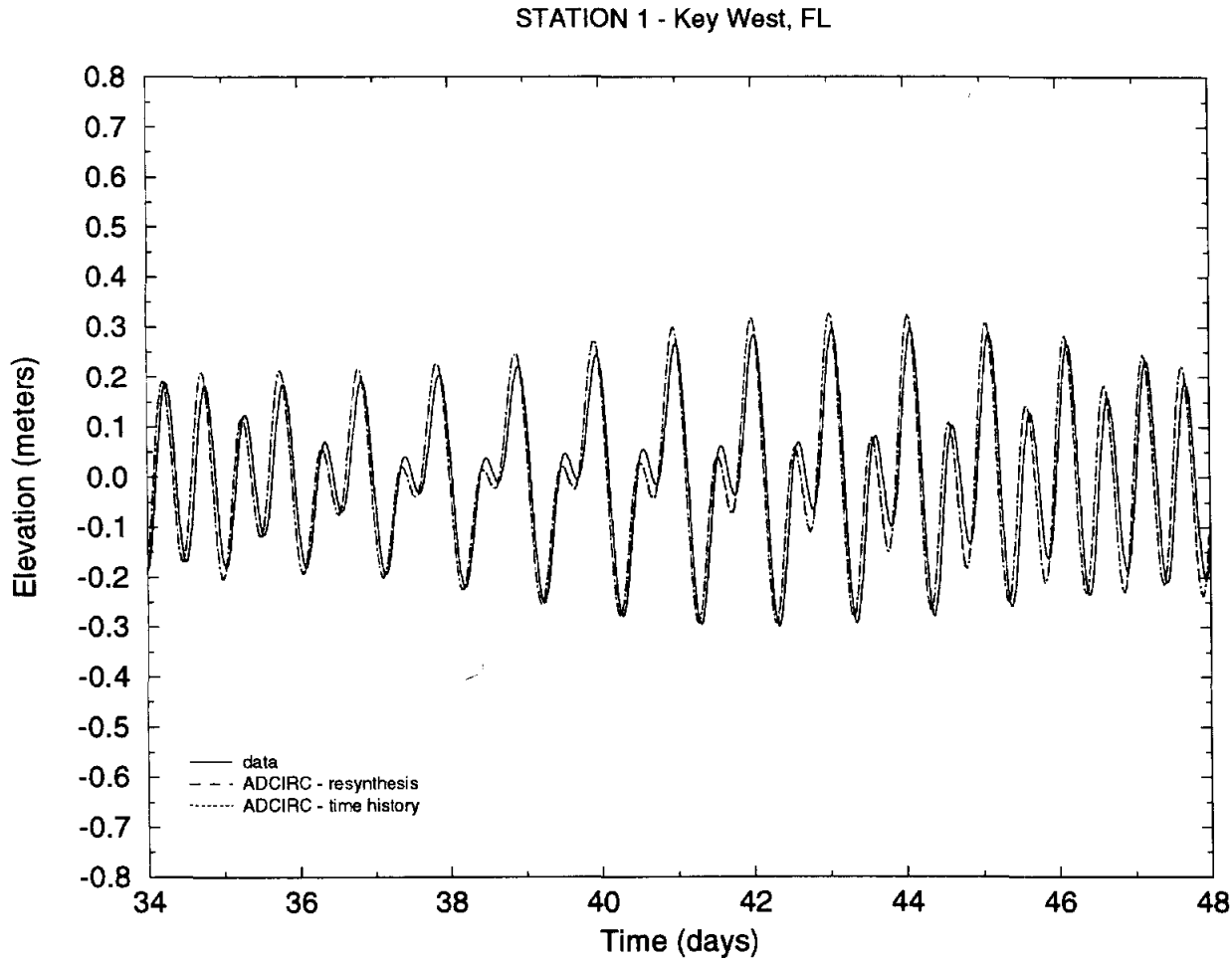


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(b) Naples, FL

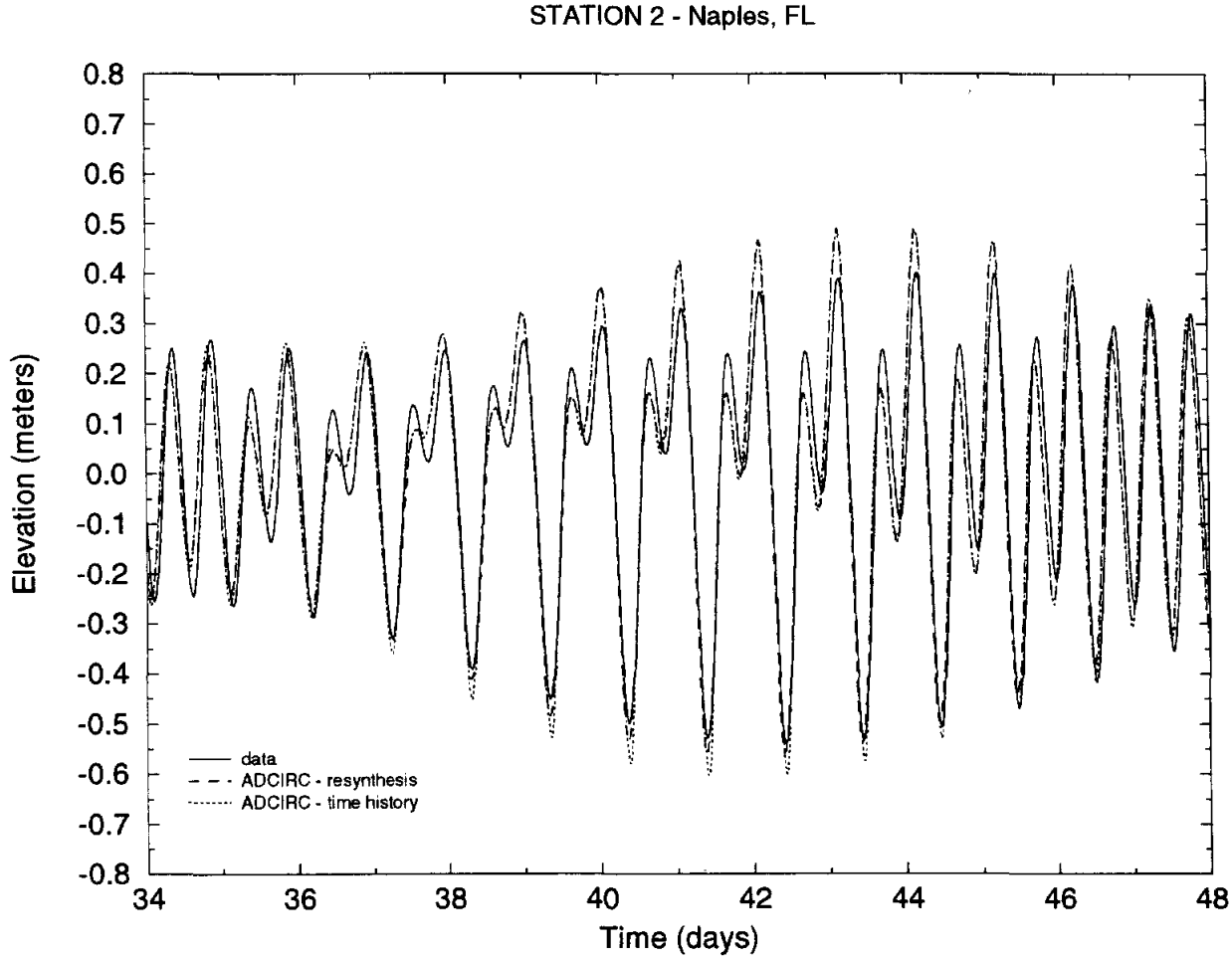


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(c) Cedar Key, FL

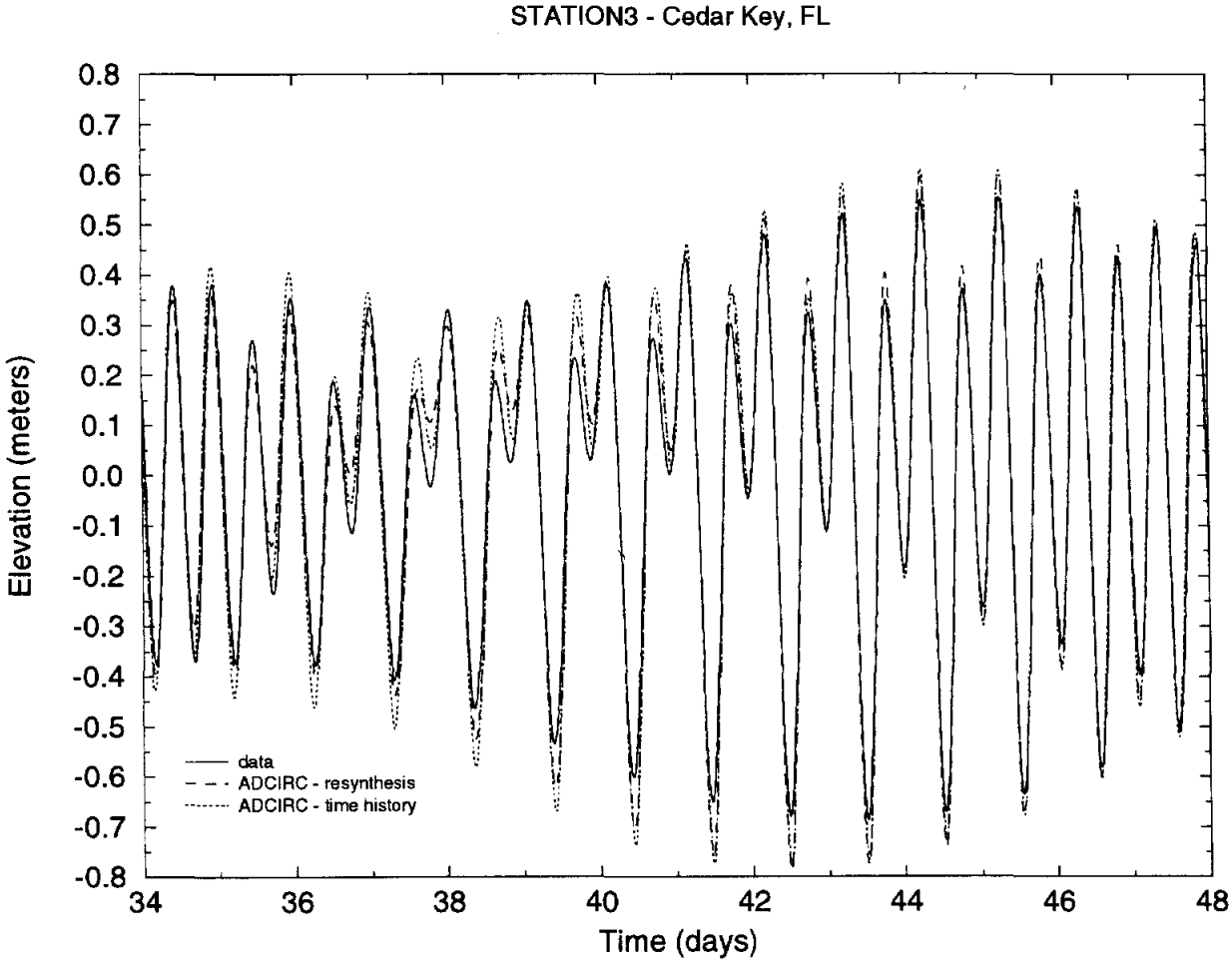


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(d) St. Marks Light, FL

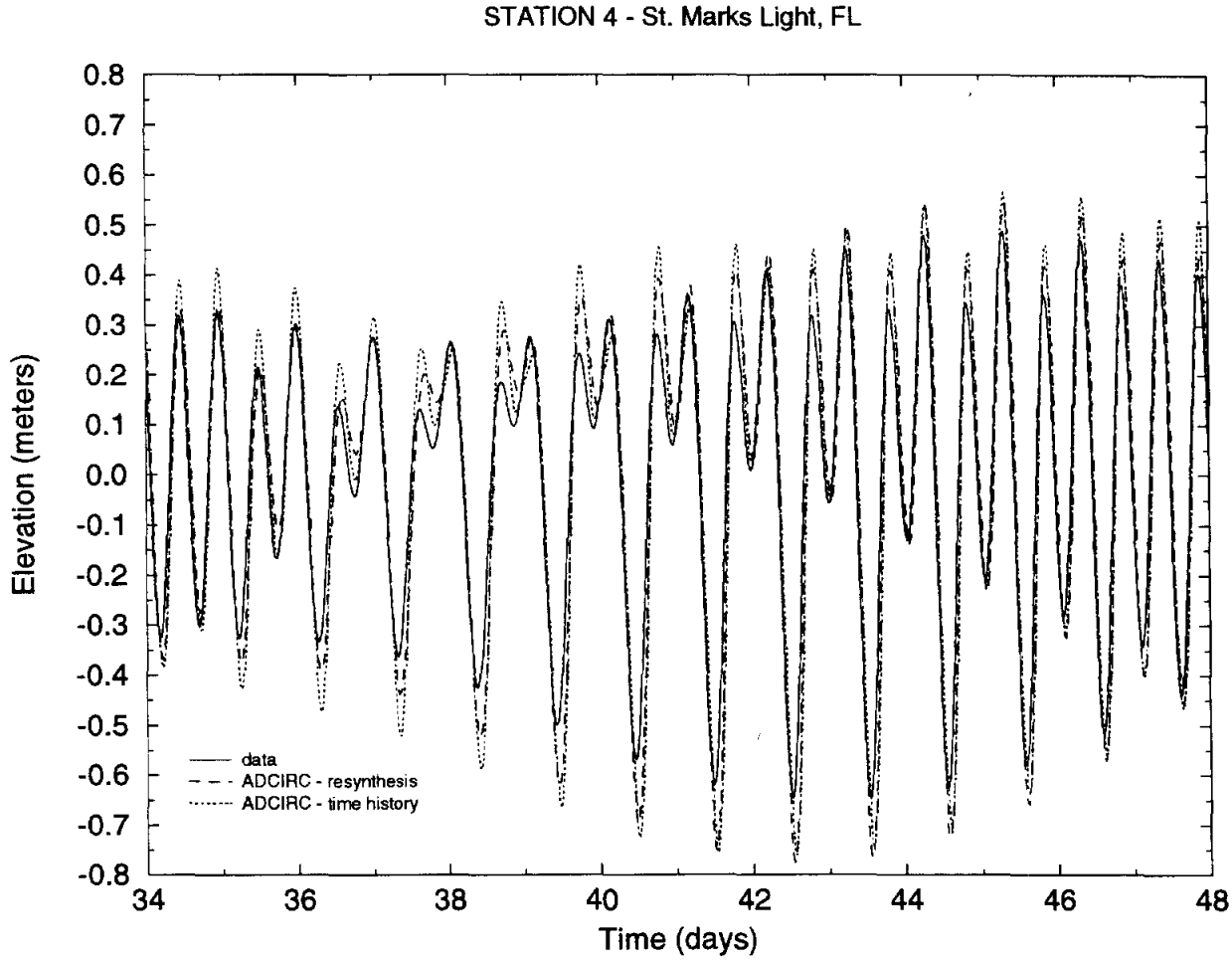


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(e)Alligator Bayou, FL

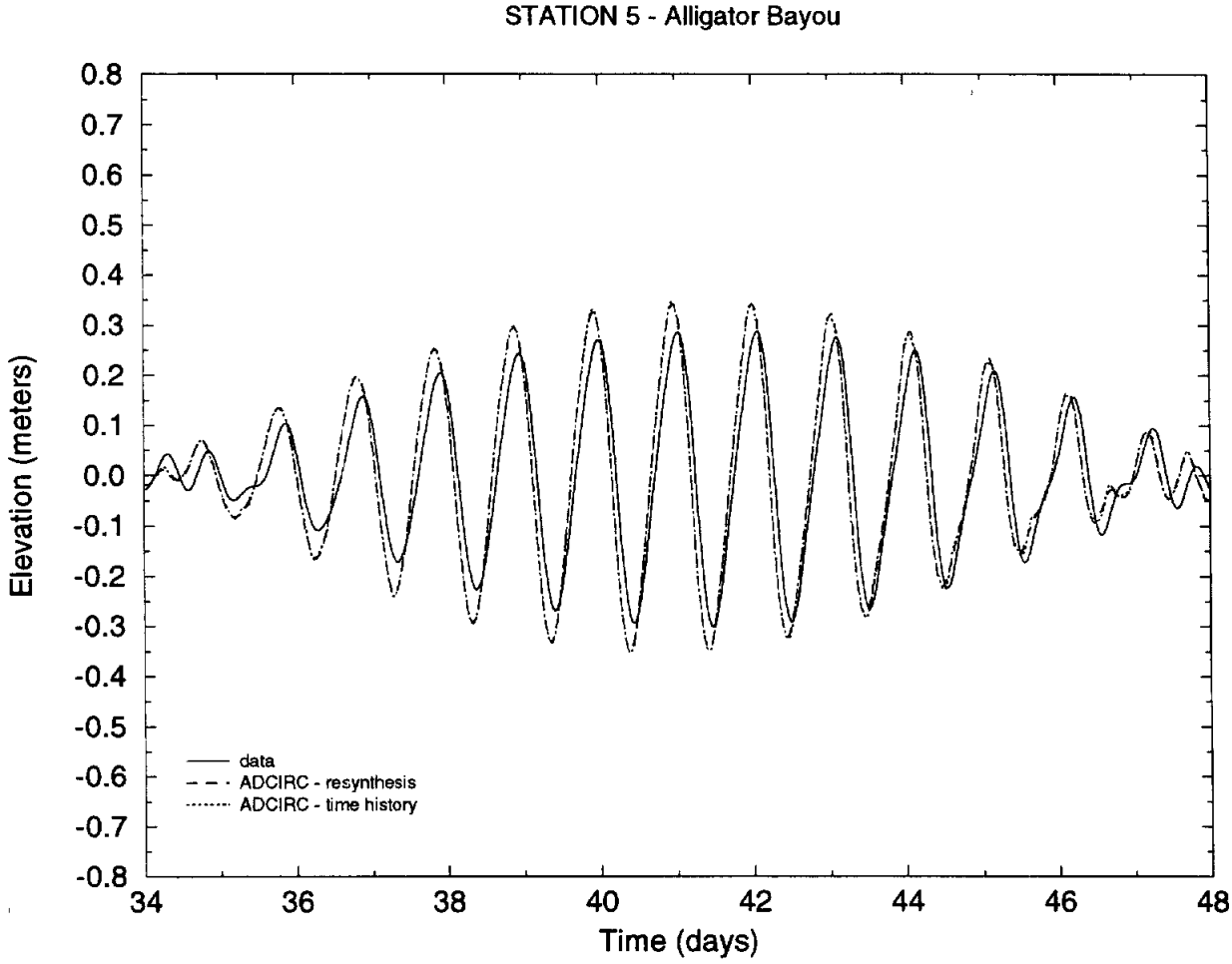


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(f) Bay St. Louis, MS

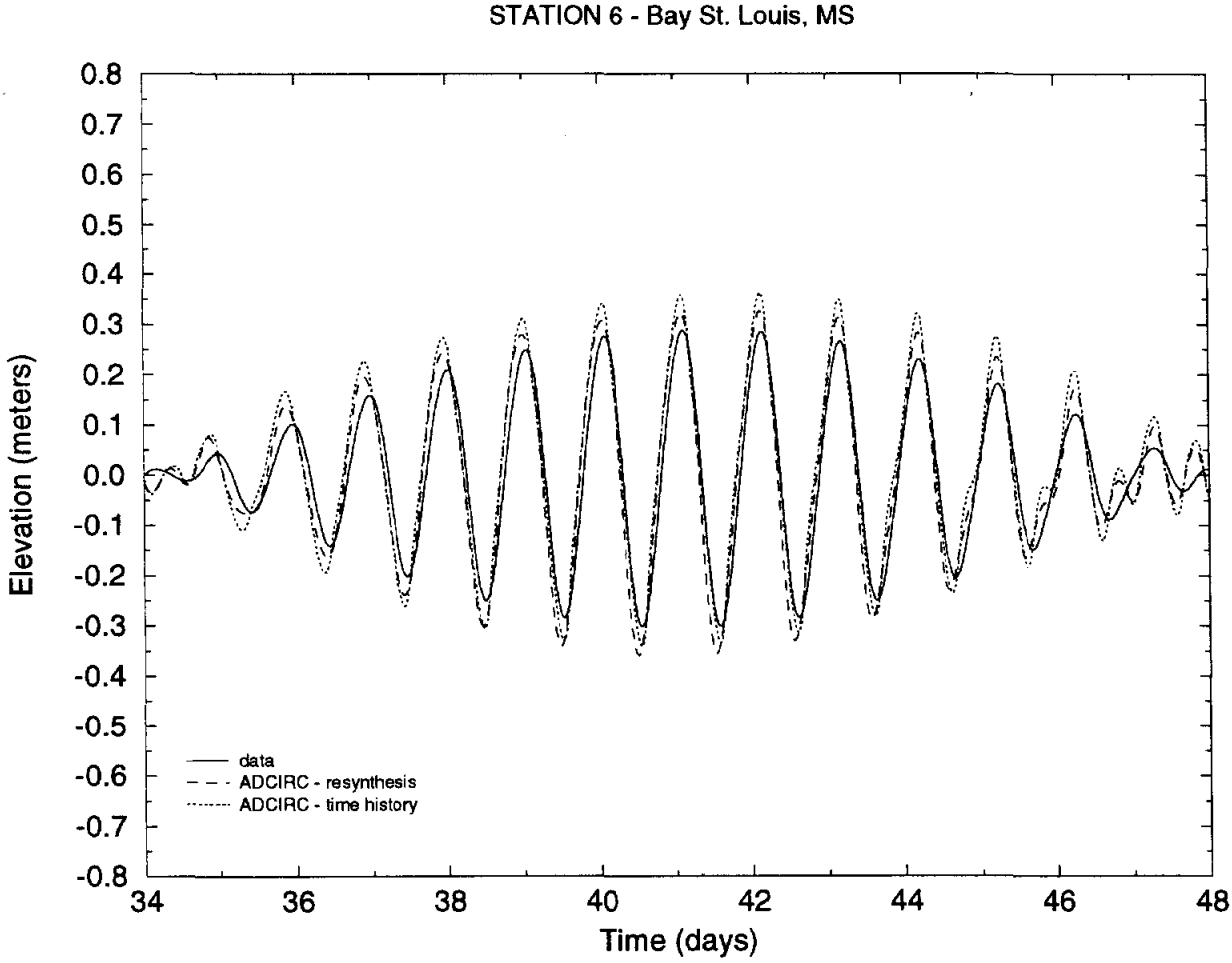


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(g) Cat Island, MS

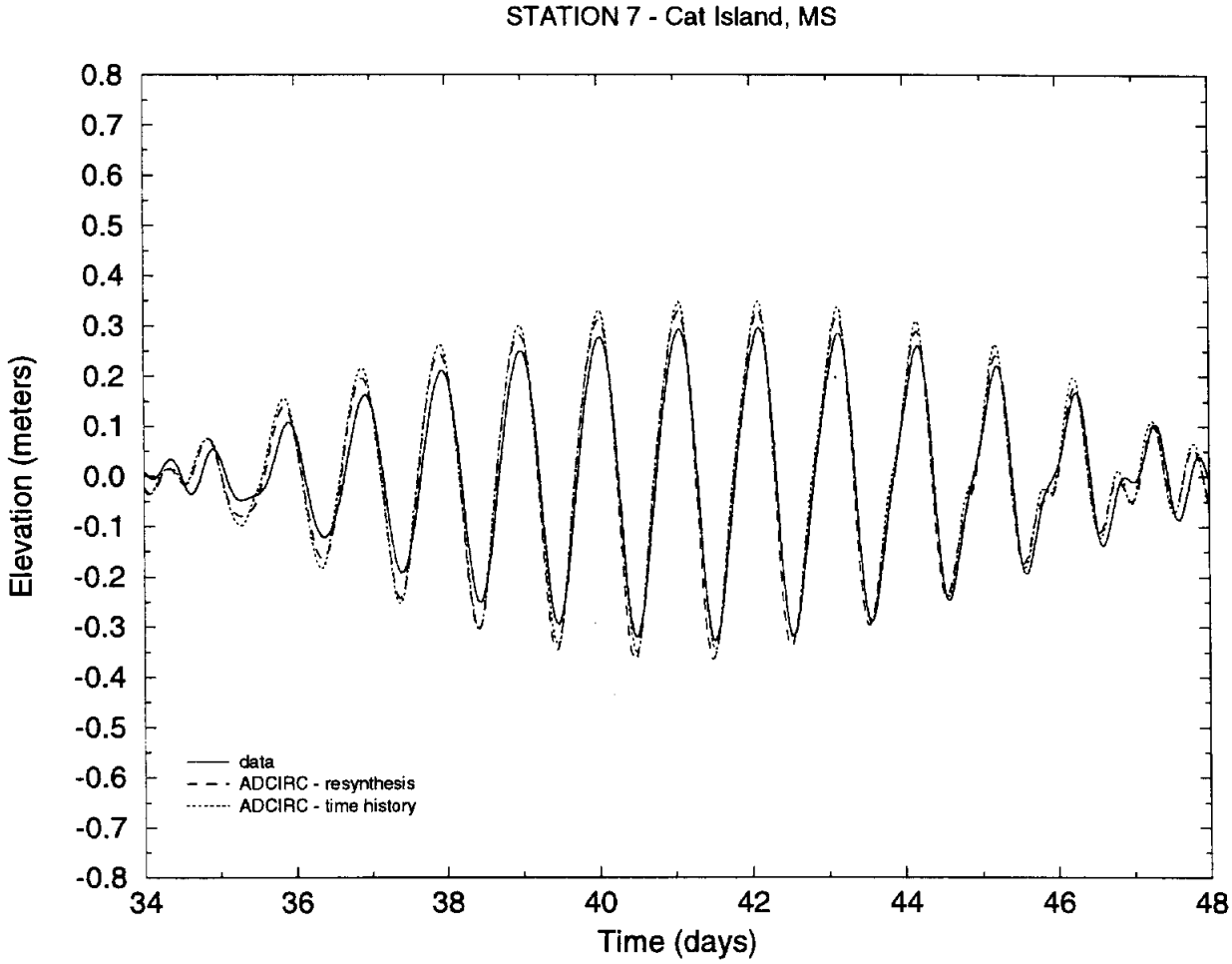


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(h) Southwest Pass, LA

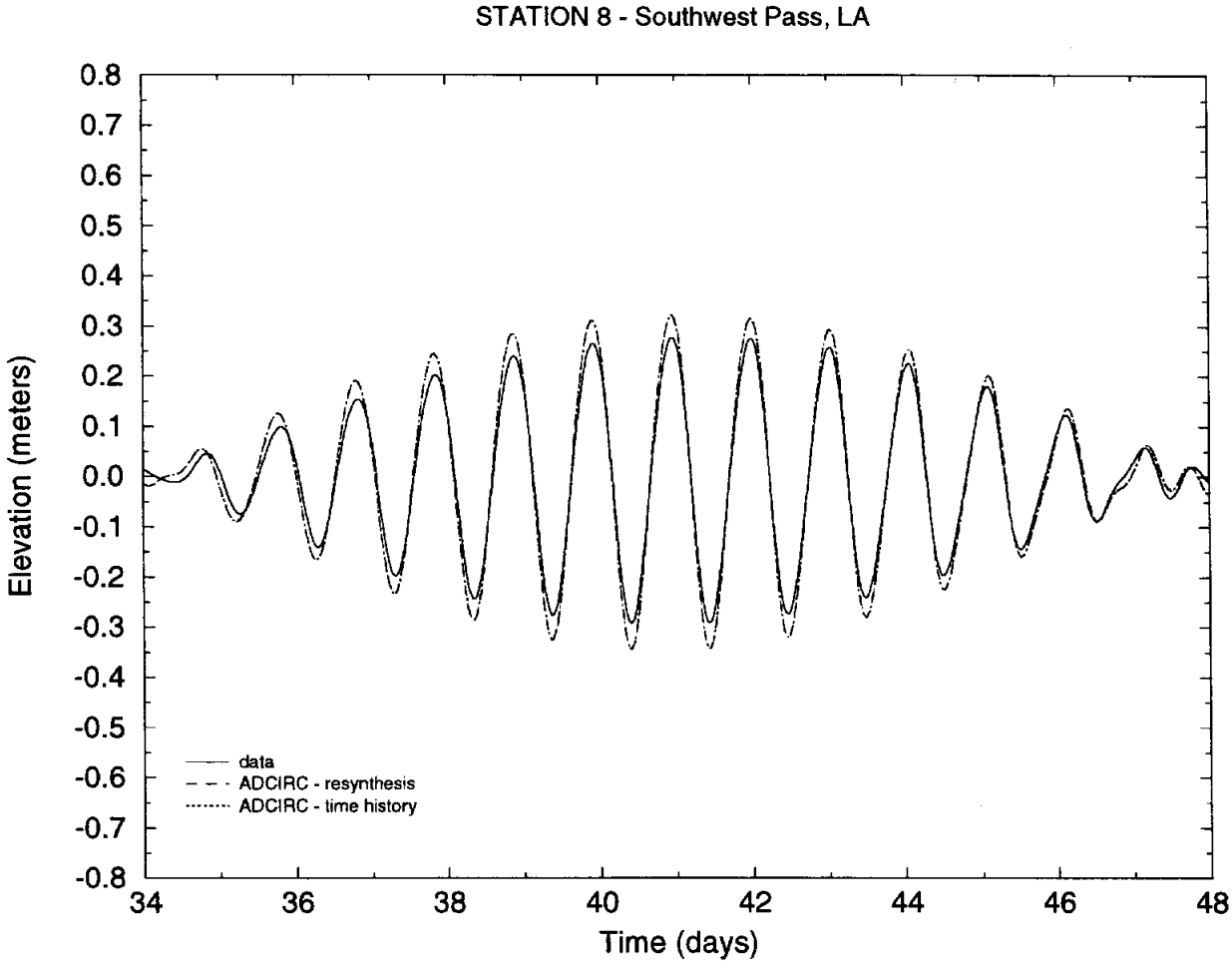


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(i) Point au Fer, LA

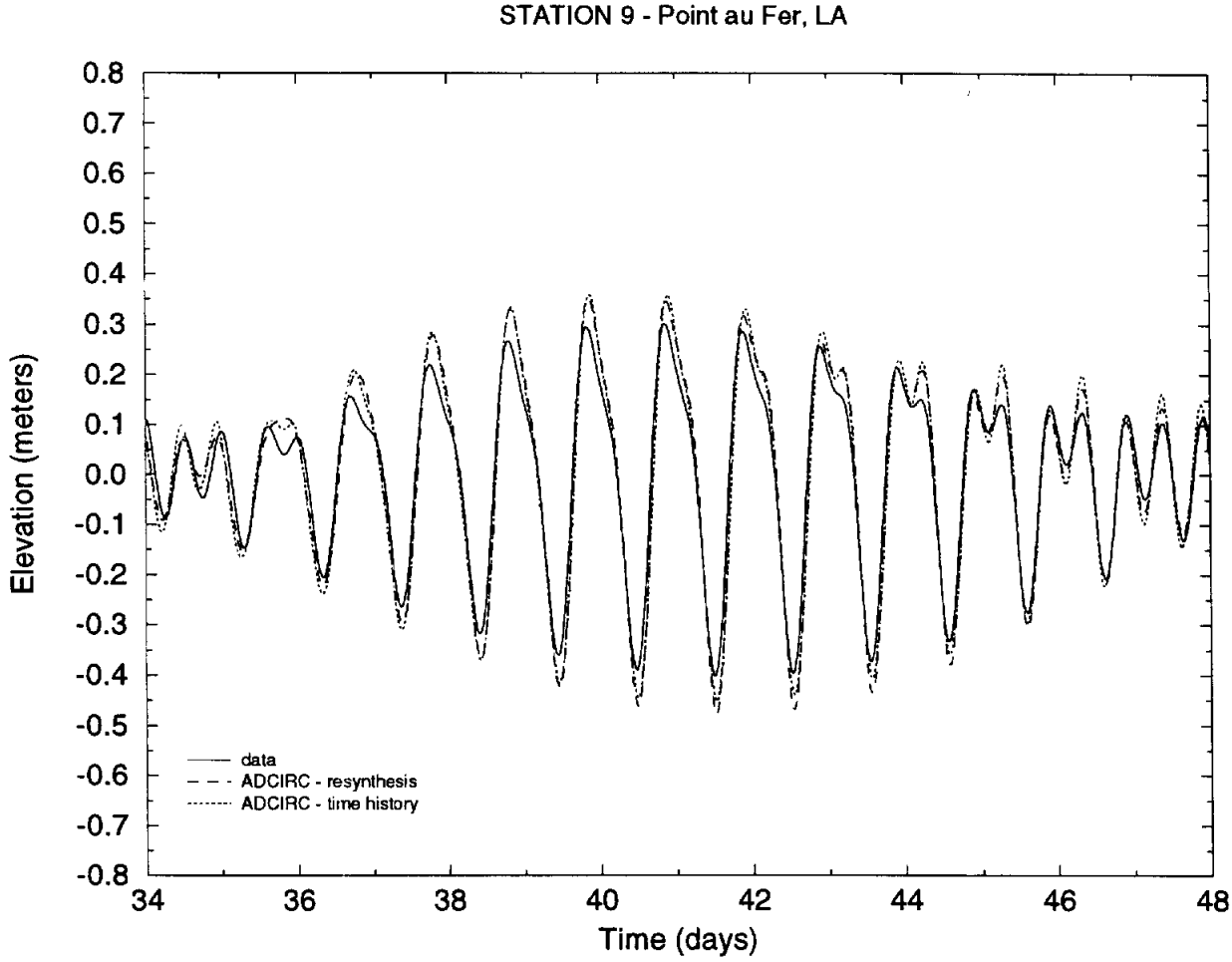


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(j) Galveston, TX

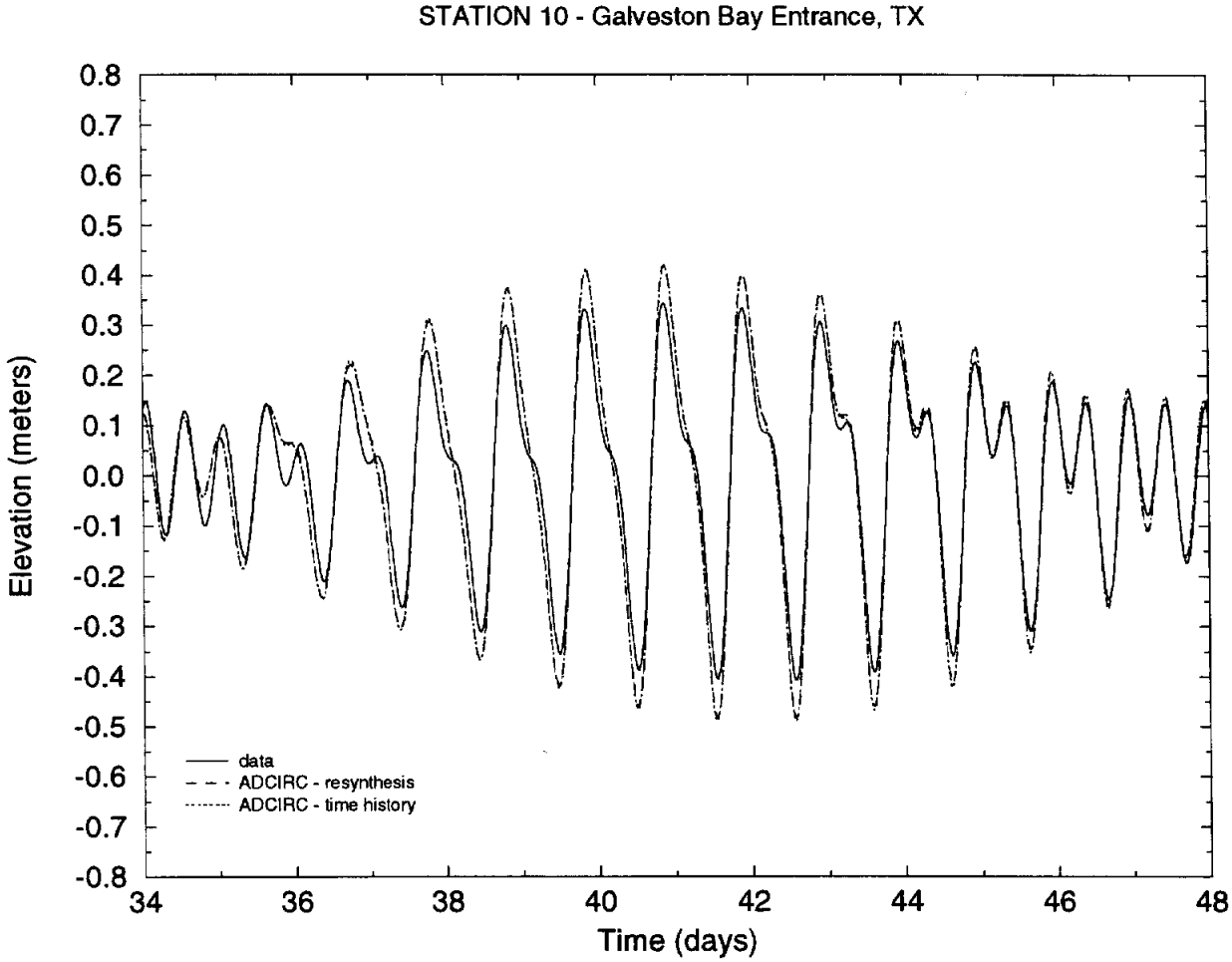


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(k) Port Arkansas, TX

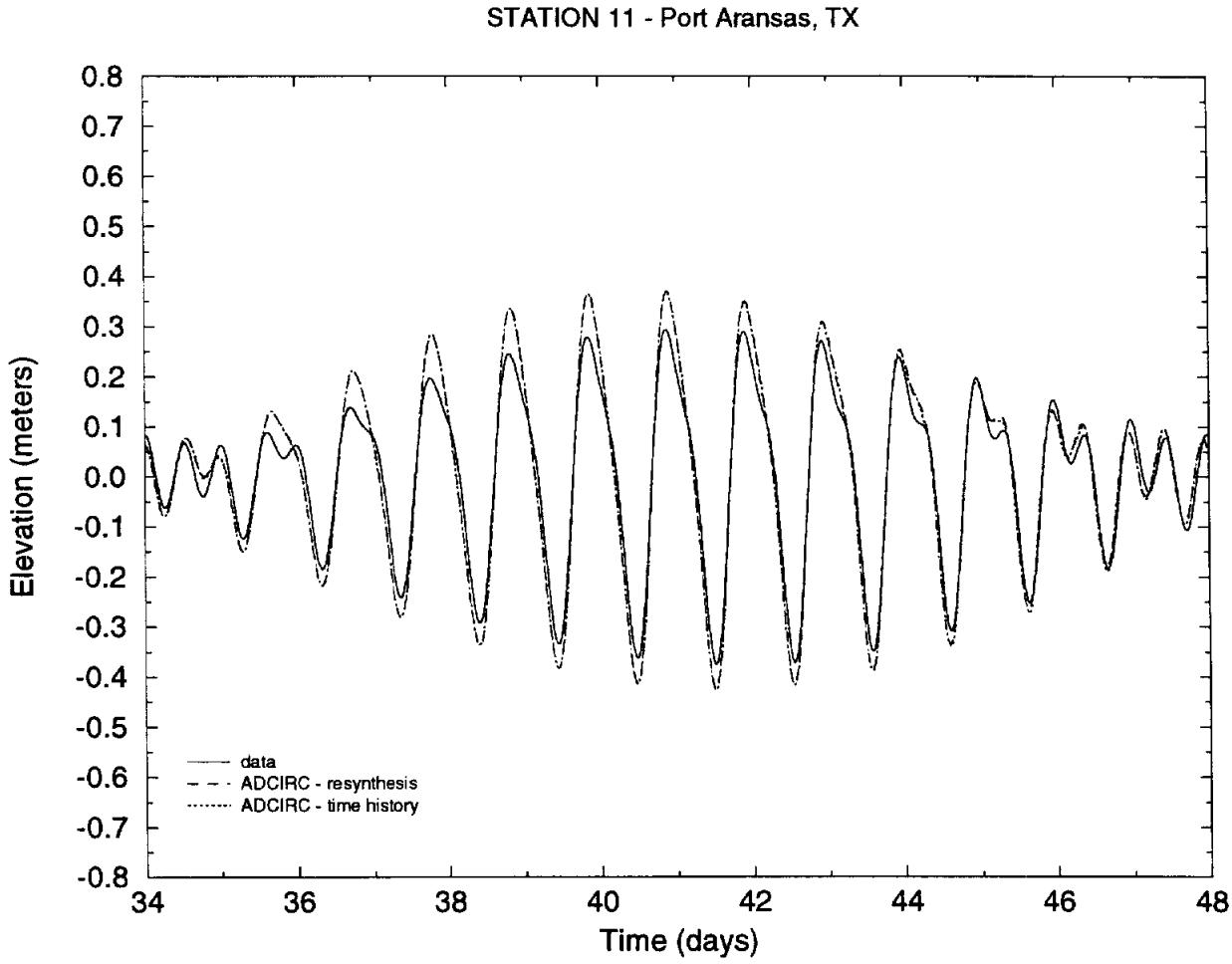


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(l) South Padre Island, TX

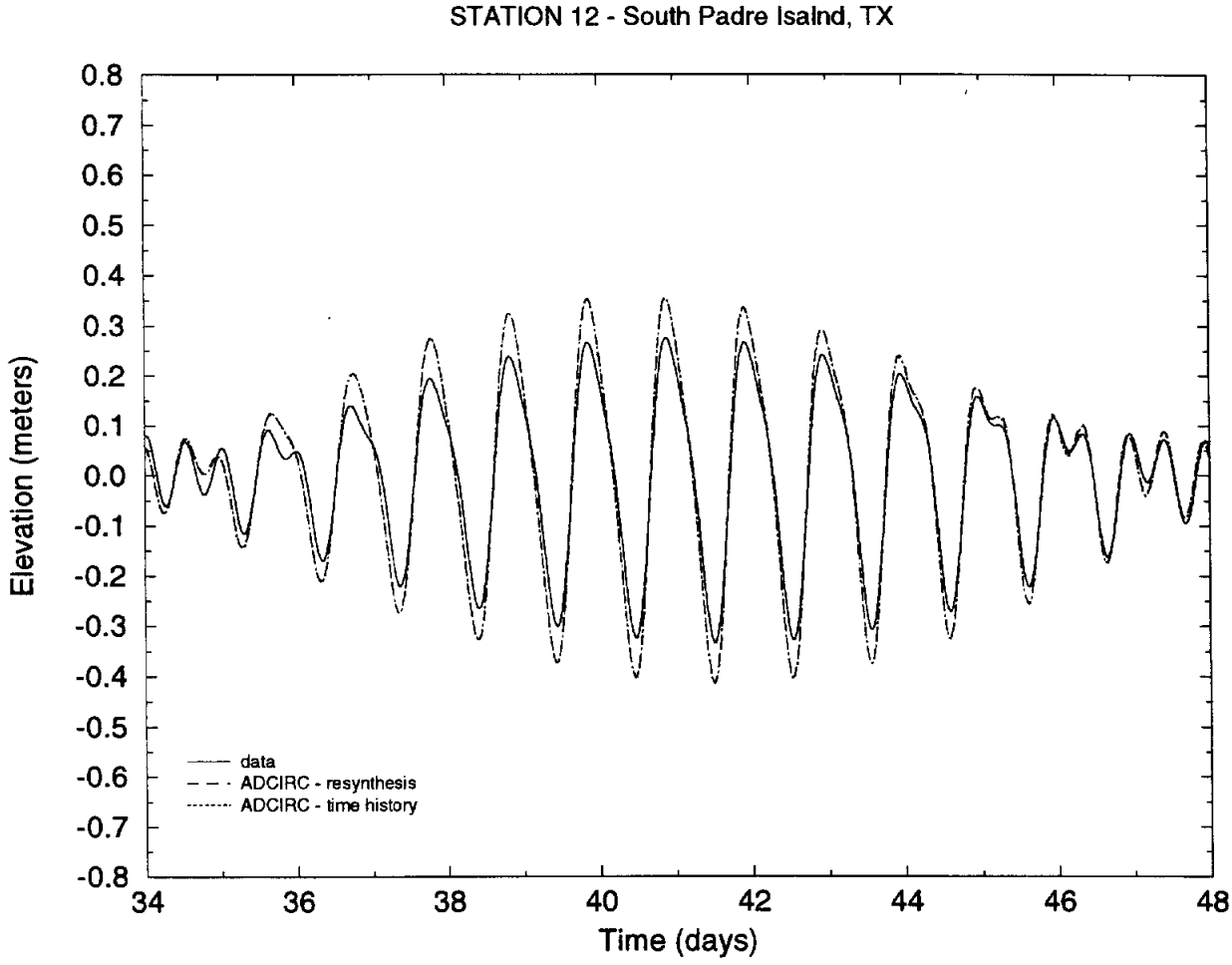


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(m) Madero, Mexico

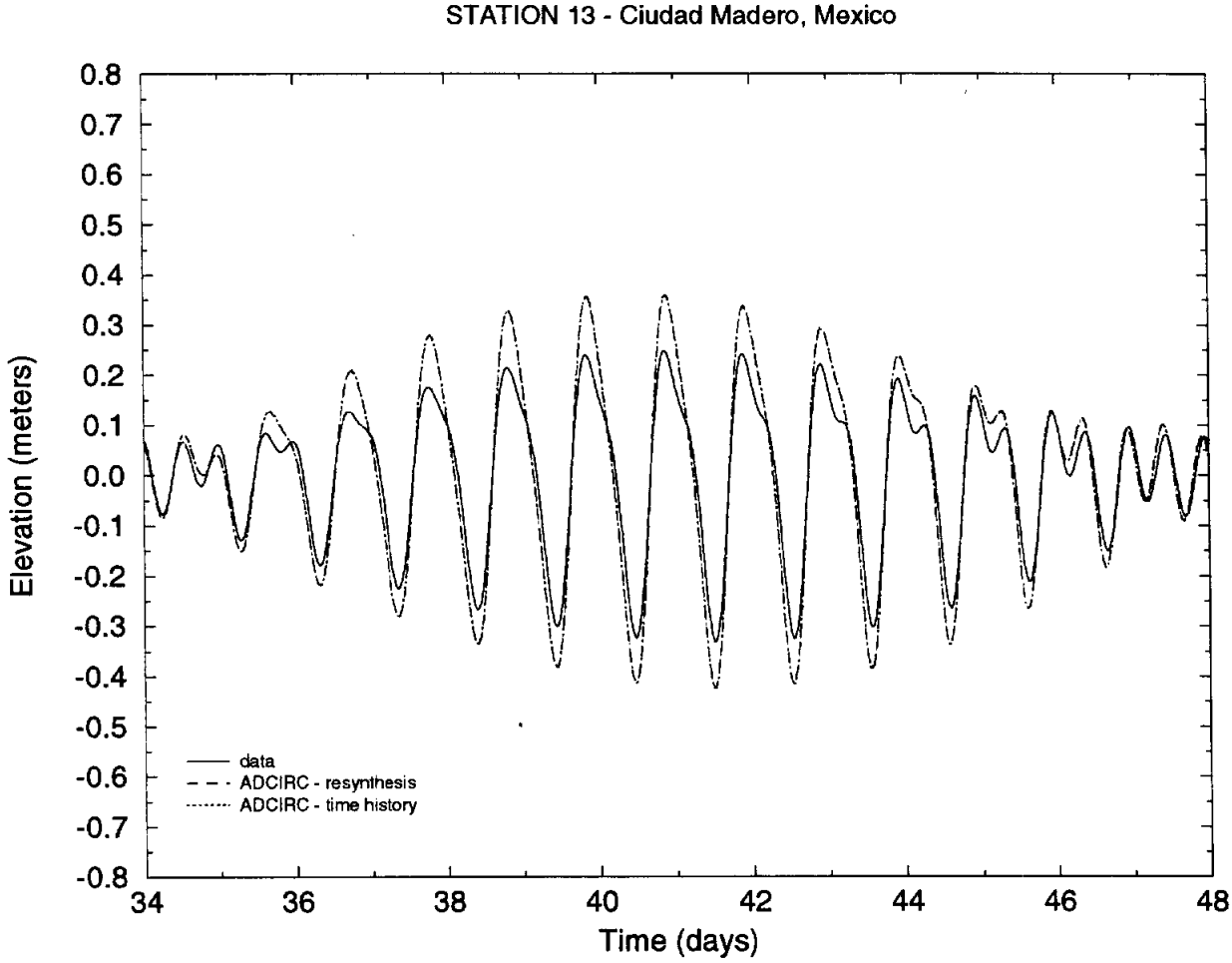


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(n) Coatzacoalcos, Mexico

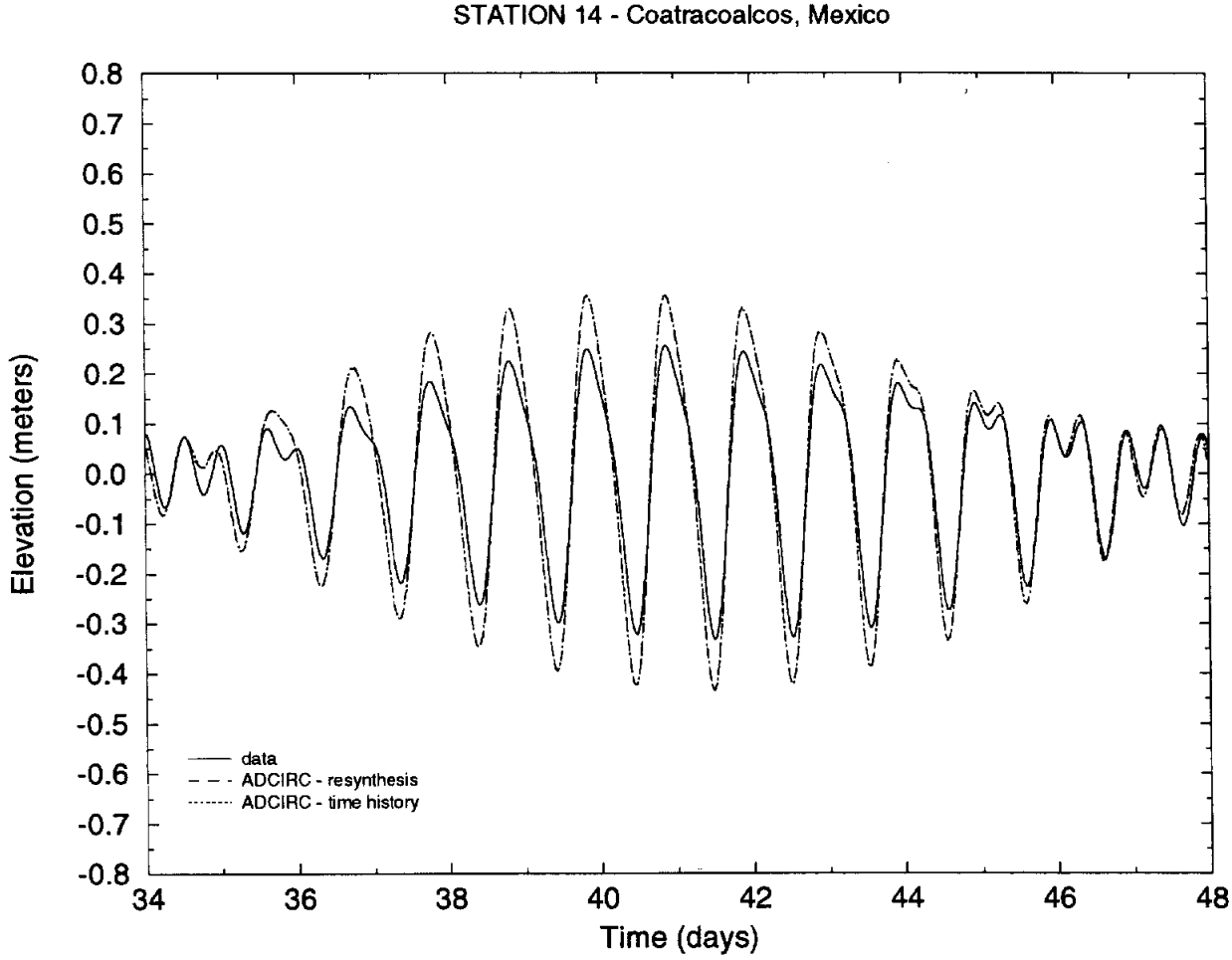


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(o) Campeche, Mexico

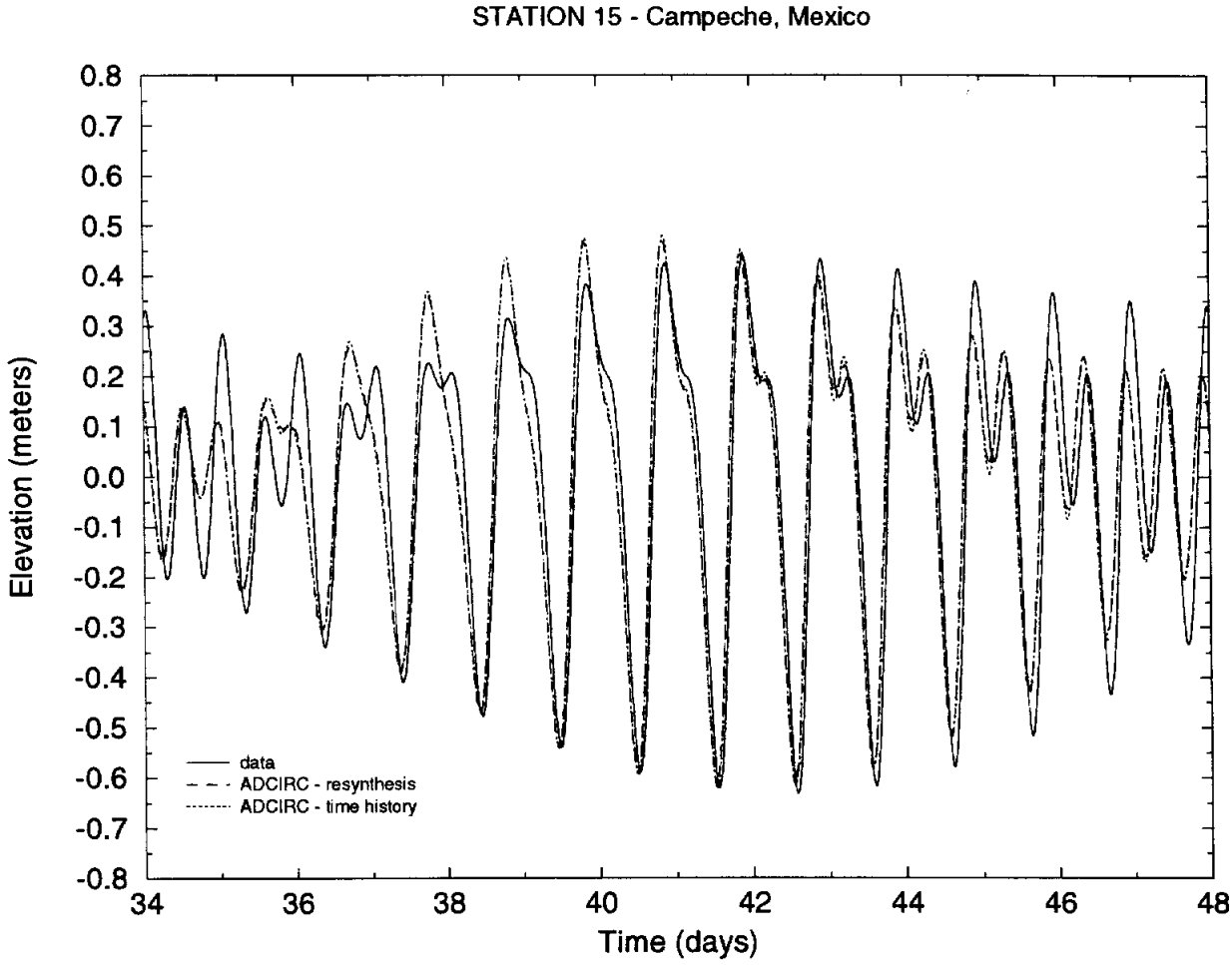


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(p) Progreso, Mexico

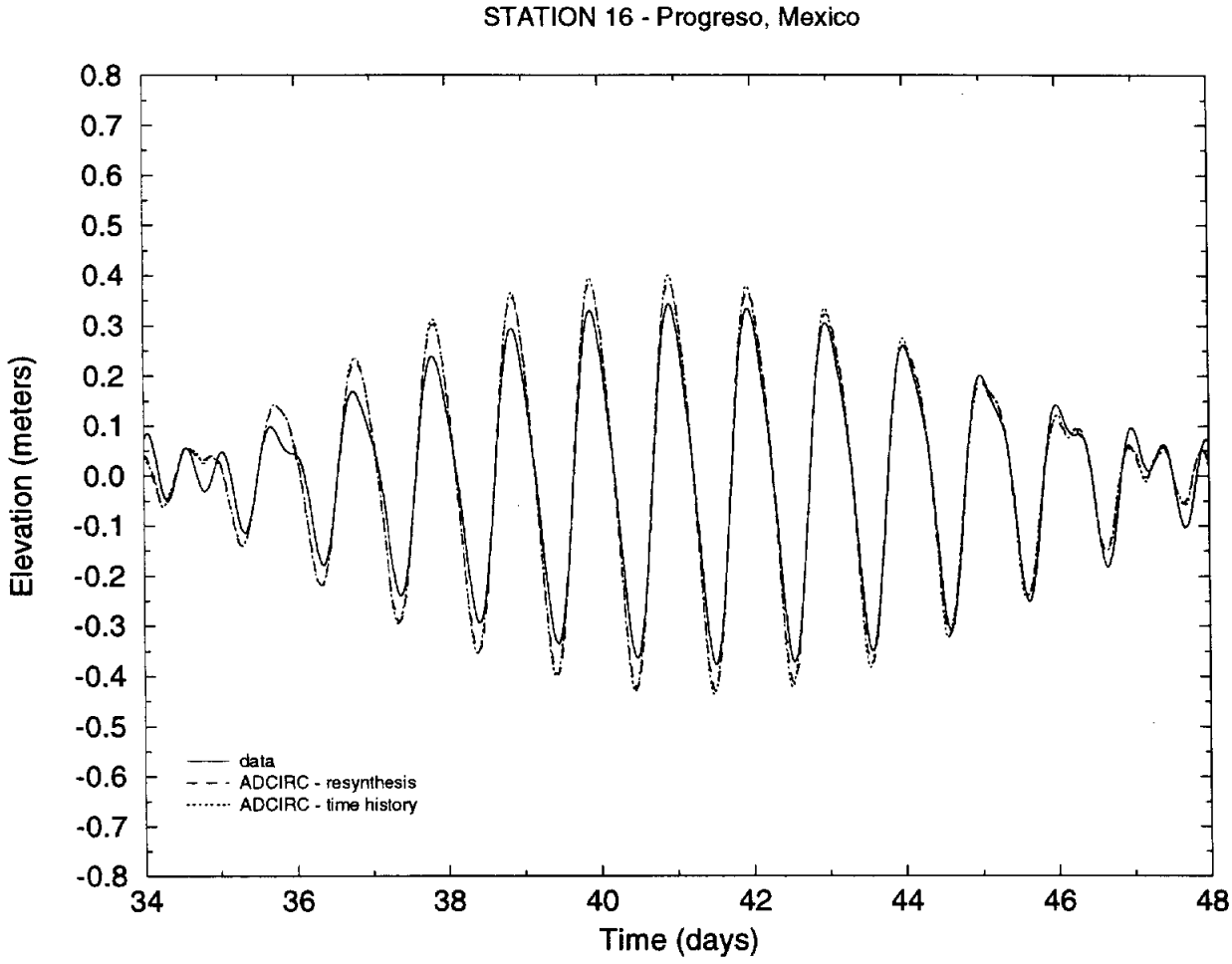


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(q) GOM Pelagic - IAPSO #30-1.2

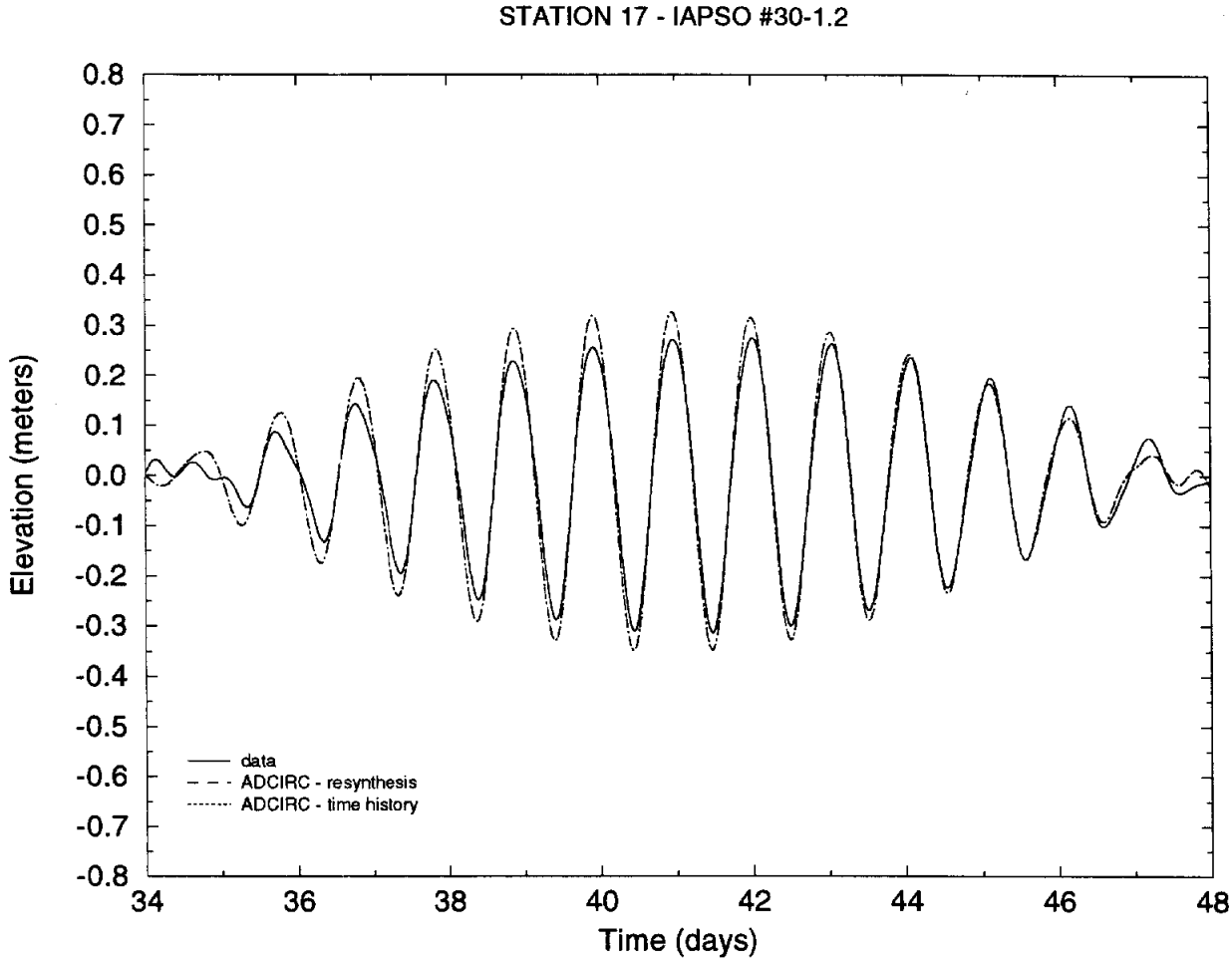


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(r) Florida Shelf - IAPSO #30-1.2.13

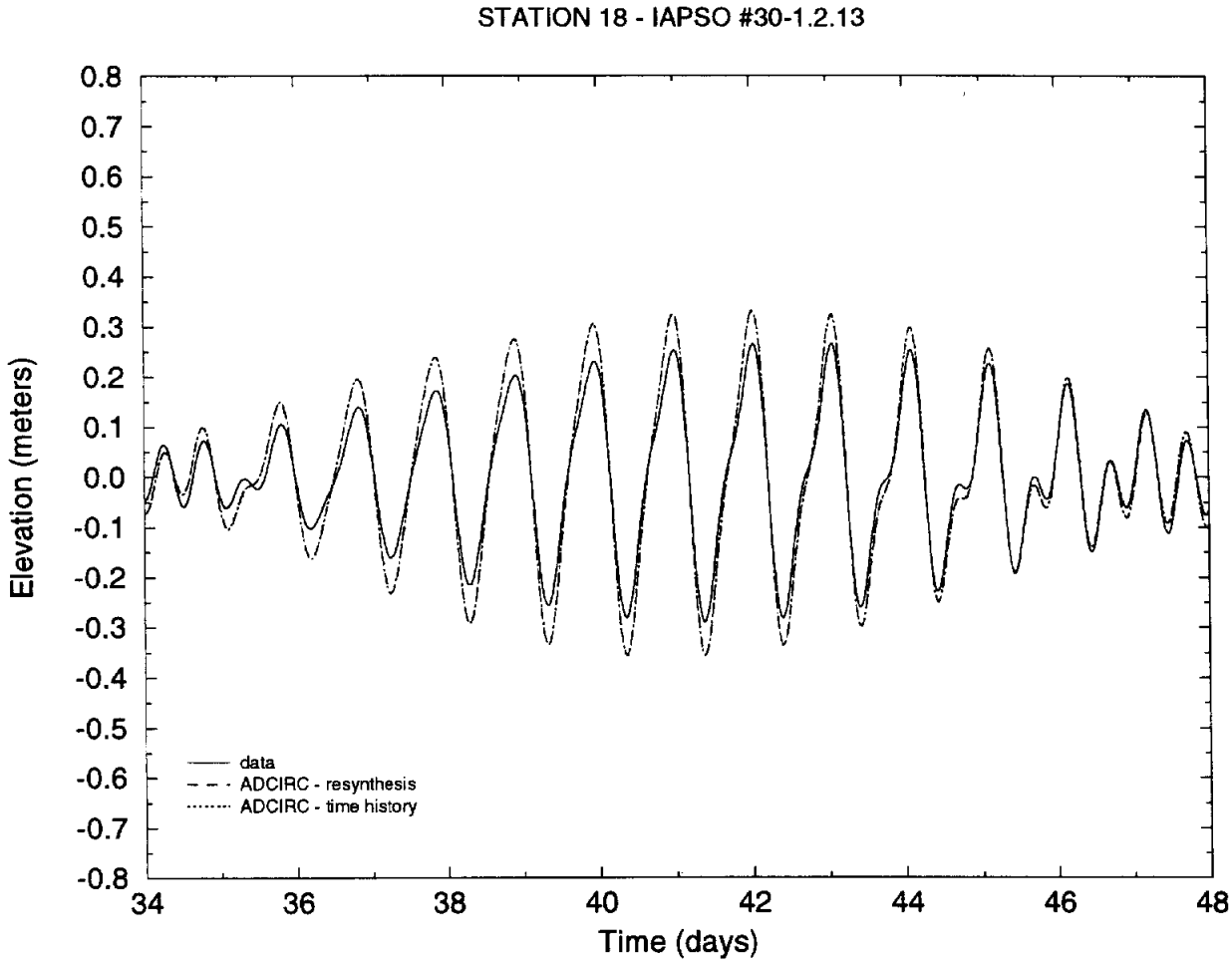


Figure 5. Comparisons of ADCIRC *EASTGAL_G03_R1* computed raw and 5 constituent resynthesized time histories of surface elevations with 5 constituents synthesized measured tidal data at representative tidal stations within the Gulf of Mexico.
(s) Havana, Cuba