Norwegian Meteorological Institute
met.no

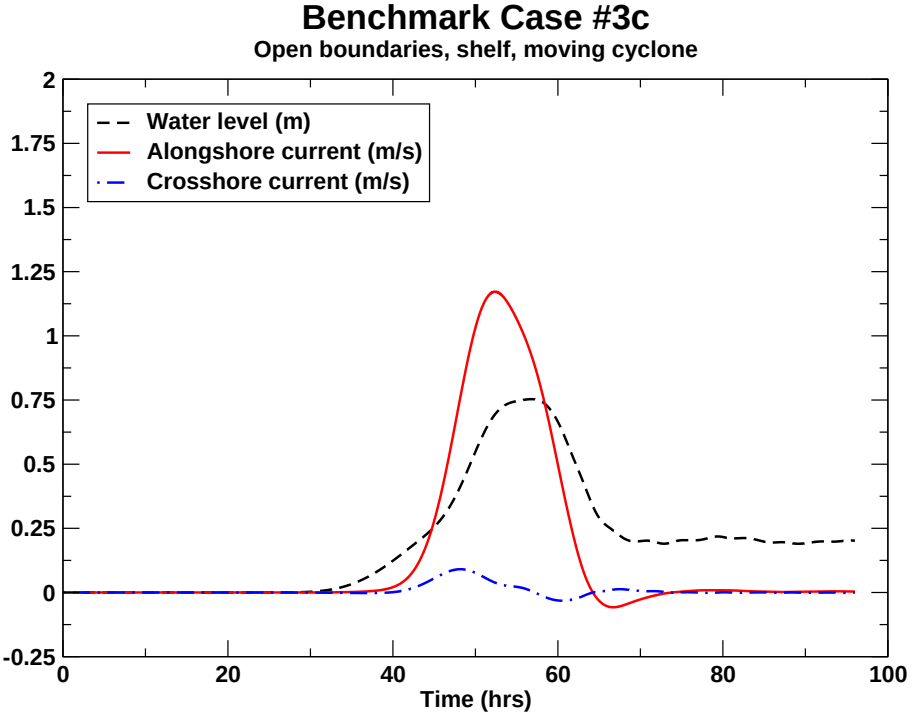
met.no report

Report no. 5/2012
Oceanography
ISSN: 1503-8025
Oslo, March 6, 2012

Documentation of simple ocean models for use in ensemble predictions

Part II: Benchmark cases

Lars Petter Røed



Norwegian Meteorological Institute
met.no

report

Number	Subject	Date	Classification	ISSN
5/2012	Oceanography	March 6, 2012	☒ Open ☐ Restricted ☐ Confidential	1503-8025

Title

Documentation of simple ocean models for use in ensemble predictions

Part II: Benchmark cases

Authors

Lars Petter Røed

Client(s)	Client reference
Statoil (Kenneth Eik Johannessen)	Contract No. xxxxxxxx

Abstract

We consider three benchmarks cases for the one-layer, linear shallow water equations. All cases are run for 96 hours and consider solutions within a rectangular domain of size 1000 km by 500 km oriented north-south with a coast to the east. The domain is either flat bottomed or features a shelf 100 km offshore. The cases differ mainly in the applied wind stress forcing. One simulates a cyclone moving diagonally across the computational domain. For each wind forced case the remaining boundaries are either closed or open. A total of nine cases are performed. Except for the moving cyclone case, all cases reaches a steady state within the first 48 hours. Moreover, a comparison with an analytic solution for one of the nine cases (uniform alongshore wind forcing) shows a perfect match.

Keywords

Physical Oceanography, Numerical Modeling, Shallow water equations, Benchmark solutions

Disiplinary signature	Responsible signature
Lars Anders Breivik, Head Ocean and Ice	Øystein Hov, Head R&D Department

Postal address PO Box 43 Blindern, N-0313 Oslo, Norway
Office Niels Henrik Abels vei 40
Telephone +47 2296 3000
Telefax +47 2296 3050
e-mail: met.inst@met.no **Web:** met.no
Bank account 7695 05 00601
Swift code DNBANOKK

Contents

List of Tables

List of Figures

1 Introduction

In Part I (*Røed*, 2012) we derived the governing equations for three simple ocean models. All of them belong to a class of models referred to as shallow water equations. We also derived a set of finite difference equations (FDEs) for one of them, namely the one-layer model. Both linear and non-linear versions of the governing equations and the associated FDEs were derived in Part I.

Here we specify some benchmark cases for the one-layer model and presents their solutions. The solutions are derived numerically based on a code written using the FORTRAN language. Whenever possible also analytic solutions are derived. These benchmark cases may be used to verify solutions based on codes written in other program languages and/or run on other computer hardware, e.g., Graphical Processor Units (GPUs).

We present a total of nine benchmark cases. They differ basically in their wind forcing, of which we specify three. All cases are solved within a rectangular domain. All cases feature a straight coast to the east, and a parallel open offshore boundary (Figure 4). The north and south boundaries may either be closed or open. In the latter case the open boundary condition used is the Flow Relaxation Scheme (FRS; *Martinsen and Engedahl*, 1987).

In the first set of wind forced cases the wind forcing consists of a uniform alongshore wind stress decaying exponentially offshore. In the second case the wind forcing consists of a bell-shaped alongshore wind stress again decaying exponentially offshore. In the third and last set of cases the wind forcing simulates a moving cyclone propagating diagonally across the basin. For each case we present three solutions. The first features a flat bottom and a closed domain. The second has a flat bottom as well, but the boundaries to the north and south as well as the offshore boundary are open. The third features open boundaries and a shelf running parallel to the coast. The benchmark cases closely follows similar benchmark cases presented by *Røed and Cooper* (1987), constructed to test various open boundary conditions for barotropic shallow water equations.

We start the presentation by recalling the continuous governing equations as well as the associated FDEs for the linear and non-linear versions of the one-layer model (Section 2). In Section 3 we give details regarding the implementation of the boundary conditions. Next we present the benchmark cases (Section 4), while Section 5 presents their solutions. We end with a brief summary and some final remarks in Section 7.

2 Equations

2.1 Governing equations

We recall that the formulation of the continuous, governing equations for the *linear* version of the one-layer ocean model is (*Røed*, 2012)

$$\partial_t U - fV = -gH\partial_x \eta + \frac{\tau_s^x - \tau_b^x}{\rho_0}, \tag{1}$$

$$\partial_t V + fU = -gH\partial_y \eta + \frac{\tau_s^y - \tau_b^y}{\rho_0}, \tag{2}$$

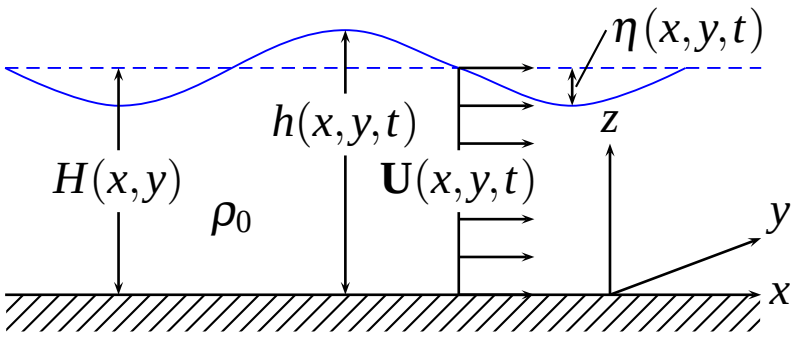


Figure 1: Sketch of a one layer, barotropic model conveniently showing some of the notation. Note that $h = h(x,y,t) = \eta(x,y,t) + H(x,y)$.

$$\partial_t \eta = -\partial_x U - \partial_y V, \tag{3}$$

while the formulation for the *non-linear* version is

$$\partial_t U - fV = -\partial_x(\frac{U^2}{H+\eta}) - \partial_y(\frac{UV}{H+\eta}) - g(H+\eta)\partial_x\eta + \frac{\tau_s^x - \tau_b^x}{\rho_0} + A\nabla_H^2 U, \tag{4}$$

$$\partial_t U + fU = -\partial_x(\frac{UV}{H+\eta}) - \partial_y(\frac{V^2}{H+\eta}) - g(H+\eta)\partial_y\eta + \frac{\tau_s^y - \tau_b^y}{\rho_0} + A\nabla_H^2 V, \tag{5}$$

$$\partial_t \eta = -\partial_x U - \partial_y V. \tag{6}$$

Here U,V are the two horizontal components of the volume transport $\mathbf{U}$ along the x- and y-axes, respectively, f is the Coriolis parameter, g is the gravitational acceleration, $H = H(x,y)$ is the equilibrium depth, η is the sea surface elevation, $\tau_s^{x,y}$ are the two horizontal components of the wind traction, $\tau_b^{x,y}$ are the two horizontal components of the bottom stress (bottom friction), and ρ_0 is the density of the sea water (Figure 1). Finally, $\nabla_H^2 = \partial_x^2 + \partial_y^2$ is the horizontal Laplace operator and A an eddy viscosity parameter tuned so as to avoid non-linear numerical instabilities. Finally we note that the thickness of the one-layer water column is $h = H + \eta$.

In all cases the computational domain is rectangular as displayed in Figure 4. We apply a linear bottom stress formulation in which the bottom stress is proportional to the the depth mean current $\mathbf{U}/H$, that is,

$$\tau_b^x = \rho_0 R\frac{U}{H}, \quad \text{and} \quad \tau_b^y = \rho_0 R\frac{V}{H}. \tag{7}$$

The wind stress and the equilibrium depth are specified for each benchmark separately (Section 4).

2.2 FDEs

We solve the above equations using numerical methods for three benchmark cases as described in Section 4. We let Δx, Δy denote the spatial increments as shown in Figure 2, while we let

Δt denote the time increment. Then the associated FDEs for the linear version are

$$U_{jk}^{n+1} = \frac{1}{B_{jk}^x}\left[U_{jk}^n + f\Delta t\overline{V}_{jk}^n + \frac{\Delta t}{\Delta x}P_{jk}^x + \Delta t X_{jk}^{n+1}\right], \tag{8}$$

$$V_{jk}^{n+1} = \frac{1}{B_{jk}^y}\left[V_{jk}^n - f\Delta t\overline{U}_{jk}^{n+1} + \frac{\Delta t}{\Delta y}P_{jk}^y + \Delta t Y_{jk}^{n+1}\right], \tag{9}$$

$$\eta_{jk}^{n+1} = \eta_{jk}^n - \frac{\Delta t}{\Delta x}\left(U_{jk}^{n+1} - U_{j-1k}^{n+1}\right) - \frac{\Delta t}{\Delta y}\left(V_{jk}^{n+1} - V_{jk-1}^{n+1}\right), \tag{10}$$

$$\tag{11}$$

where

$$B_{jk}^x = 1 + \frac{R\Delta t}{H_{jk}^x}, \quad B_{jk}^y = 1 + \frac{R\Delta t}{H_{jk}^y}, \tag{12}$$

$$H_{jk}^x = \frac{1}{2}(H_{j+1k} + H_{jk}), \quad H_{jk}^y = \frac{1}{2}(H_{jk+1} + H_{jk}), \tag{13}$$

$$\overline{U}_{jk}^n = \frac{1}{4}(U_{jk}^n + U_{j-1k}^n + U_{j-1k+1}^n + U_{jk+1}^n), \tag{14}$$

$$\overline{V}_{jk}^n = \frac{1}{4}(V_{jk}^n + V_{j+1k}^n + V_{j+1k-1}^n + V_{jk-1}^n), \tag{15}$$

$$P_{jk}^x = gH_{jk}^x\left(\eta_{j+1k}^n - \eta_{jk}^n\right), \quad P_{jk}^y = gH_{jk}^y\left(\eta_{jk+1}^n - \eta_{jk}^n\right), \tag{16}$$

$$X_{jk}^n = \frac{1}{2}\rho_0^{-1}\left([\tau_s^x]_{jk}^n + [\tau_s^x]_{jk-1}^n\right), \quad Y_{jk}^n = \frac{1}{2}\rho_0^{-1}\left([\tau_s^y]_{jk}^n + [\tau_s^y]_{j-1k}^n\right). \tag{17}$$

The associated FDEs for the non-linear version are

$$U_{jk}^{n+1} = \frac{1}{C_{jk}^x}\left[U_{jk}^{n-1} + 2\Delta t\left(f\overline{V}_{jk}^n + \frac{N_{jk}^x}{\Delta x} + \frac{P_{jk}^x + \hat{P}_{jk}^x}{\Delta x} + X_{jk}^{n+1} + AE_{jk}^x\right)\right], \tag{18}$$

$$V_{jk}^{n+1} = \frac{1}{C_{jk}^y}\left[V_{jk}^{n-1} + 2\Delta t\left(-f\overline{U}_{jk}^n + \frac{N_{jk}^y}{\Delta y} + \frac{P_{jk}^y + \hat{P}_{jk}^y}{\Delta y} + Y_{jk}^{n+1} + AE_{jk}^y\right)\right], \tag{19}$$

$$\eta_{jk}^{n+1} = \eta_{jk}^{n-1} - \frac{2\Delta t}{\Delta x}\left(U_{jk}^n - U_{j-1k}^n\right) - \frac{2\Delta t}{\Delta y}\left(V_{jk}^n - V_{jk-1}^n\right), \tag{20}$$

where

$$C_{jk}^x = 1 + \frac{2R\Delta t}{H_{jk}^x} + \frac{2A\Delta t(\Delta x^2 + \Delta y^2)}{\Delta x^2\Delta y^2}, \quad C_{jk}^y = 1 + \frac{2R\Delta t}{H_{jk}^y} + \frac{2A\Delta t(\Delta x^2 + \Delta y^2)}{\Delta x^2\Delta y^2}, \tag{21}$$

$$\tag{22}$$

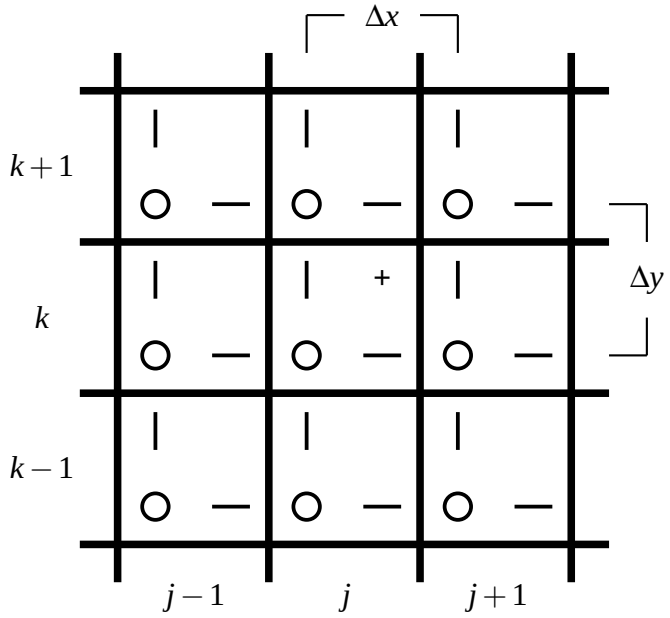


Figure 2: Displayed are the spatial grid and grid cells we use to solve (8) - (10) and (18) - (20). The spatial increments are $\Delta x, \Delta y$, respectively in the x, y directions. The total number of grid cells are $JJ \times KK$. Dummy indices j,k are used to count the number of cells along the x- and y-axes, respectively, that is, $j = 1(1)JJ$ and $k = 1(1)KK$. Circles, (O), correspond to η- and H-points, horizontal dashes, $(-)$, to U-points, and vertical lines, $(|)$, to V-points. The point marked with a + is the position of the point x_j, y_k in grid cell j,k. The coordinates of the η-points in the grid cell j,k are thus $x_j - \frac{1}{2}\Delta x, y_k - \frac{1}{2}\Delta y$.

$$N^x_{jk} = \frac{1}{4}\left\{ \frac{\left(U^n_{j+1k}+U^n_{jk}\right)^2}{H_{j+1k}+\eta^n_{j+1k}} - \frac{\left(U^n_{jk}+U^n_{j-1k}\right)^2}{H_{jk}+\eta^n_{jk}} + \frac{\Delta x}{\Delta y}\left[\frac{(U^n_{jk+1}+U^n_{jk})(V^n_{j+1k}+V^n_{jk})}{\overline{H}^n_{jk}+\overline{\eta}^n_{jk}} - \frac{(U^n_{jk}+U^n_{jk-1})(V^n_{j+1k-1}+V^n_{jk-1})}{\overline{H}^n_{jk-1}+\overline{\eta}^n_{jk-1}} \right] \right\}, \quad (23)$$

$$N^y_{jk} = \frac{1}{4}\left\{ \frac{\left(V^n_{jk+1}+V^n_{jk}\right)^2}{H^n_{jk+1}+\eta^n_{jk+1}} - \frac{\left(V^n_{jk}+V^n_{jk-1}\right)^2}{H^n_{jk}+\eta^n_{jk}} + \frac{\Delta y}{\Delta x}\left[\frac{(U^n_{jk+1}+U^n_{jk})(V^n_{j+1k}+V^n_{jk})}{\overline{H}^n_{jk}+\overline{\eta}^n_{jk}} - \frac{(U^n_{j-1k+1}+U^n_{j-1k})(V^n_{jk}+V^n_{j-1k})}{\overline{H}^n_{jk-1}+\overline{\eta}^n_{jk-1}} \right] \right\}, \quad (24)$$

$$\hat{P}^x_{jk} = \frac{1}{2}\left[(\eta^n_{j+1k})^2 - (\eta^n_{jk})^2\right], \quad \hat{P}^y_{jk} = \frac{1}{2}\left[(\eta^n_{jk+1})^2 - (\eta^n_{jk})^2\right], \quad (25)$$

$$E^x_{jk} = \frac{1}{\Delta x^2}\left(U^n_{j+1k} - U^{n-1}_{jk} + U^n_{j-1k}\right) + \frac{1}{\Delta y^2}\left(U^n_{jk+1} - U^{n-1}_{jk} + U^n_{jk-1}\right), \quad (26)$$

$$E^y_{jk} = \frac{1}{\Delta x^2}\left(V^n_{j+1k} - V^{n-1}_{jk} + V^n_{j-1k}\right) + \frac{1}{\Delta y^2}\left(V^n_{jk+1} - V^{n-1}_{jk} + V^n_{jk-1}\right). \quad (27)$$

We observe by comparing (8) - (10) and (18) - (20) that the non-linear version features the six additional terms N^x_{jk}, N^y_{jk}, $\hat{P}^x_{jk}$, $\hat{P}^x_{jk}$, E^x_{jk} and E^y_{jk}. While the first four are due to the non-linearity, the two last ones are added to avoid non-linear, numerical instabilities to appear. Moreover we notice that while the linear FDE set (8) - (10) is a forward-backward scheme, the non-linear FDE set (18) - (20) is a centered in time, centered in space scheme (leapfrog scheme). This difference is manifested in the use of $2\Delta t$ in (18) - (20), while (8) - (10) uses Δt.

As alluded to in *Røed* (2012) the indexes j,k refers to the cell numbers as shown in Figures 2 and 3. Note that the grid is staggered so that the U_{jk}-point is staggered one half grid length in the positive x-direction and the V_{jk}-point is staggered on half grid length in the positive y-direction compared to the η_{jk}-point in cell number j,k. Also note that the x_j, y_k coordinates in cell number j,k is staggered one half grid length in both the positive x-direction and positive y-direction. We let the coast run through the V^n_{j1}-points ($j = 1(1)JJ$, $k = 1$)[1], the offshore boundary through the V^n_{jKK-1}-points ($j = 1(1)JJ, k = KK-1$), the southern boundary through the U^n_{1k}-points ($j = 1$, $k = 1(1)KK$) and the northern boundary through the U^n_{JJ-1k}-points ($j = JJ-1, k = 1(1)KK$) as depicted in Figure 3. Since (1) - (3) and (4) - (6), strictly speaking, are valid only inside of the physical boundaries, that is, at "wet" points only, we emphasize that (8) and (18) are valid for $j = 2(1)JJ-2, k = 2(1)KK-1$, (9) and (19) for $j = 2(1)JJ-1$, $k = 2(1)KK-2$, while (10) and (20) are valid for $j = 2(1)JJ-1, k = 2(1)KK-1$.

As noted by *Røed* (2012) the schemes (8) - (10) and (18) - (20) are numerically stable as long as the Courant-Friedrich-Levy (CFL) condition is satisfied. Thus we require that the grid speed $\Delta s/\Delta t$ is larger than the propagation speed of gravity waves or $\sqrt{gH_{max}}$, where H_{max} is the maximum equilibrium depth and $\Delta s = \sqrt{\Delta x^2 + \Delta y^2}$. This gives an upper limit for the time increment Δt once Δx and Δy is chosen. The time increment shown in Table 2 is well within this limit.

3 Boundary conditions

3.1 Physical conditions

We note that for the linear version we may only specify two boundary conditions in the each of the two horizontal directions x,y. In the non-linear version the order of the differential equations is raised due to the presence of the eddy viscosity terms. Hence we are allowed to specify four conditions in each of the two horizontal directions for the non-linear case.

3.1.1 Closed boundaries

In this case the natural boundary conditions follows from the condition of no throughflow. Thus

$$U = 0 \quad ; \quad x = 0, L_x, \quad \forall y, \tag{28}$$

$$V = 0 \quad ; \quad y = 0, L_y, \quad \forall x. \tag{29}$$

[1]The notation $j = 1(1)JJ$ is used to imply that j runs from 1 to JJ with an increment of 1.

This is all that is allowed regarding the linear version. We note that the staggering enables us to avoid over-specifying the number of boundary conditions (*Mesinger and Arakawa*, 1976).

When the non-linear terms are added the eddy viscosity terms are included. Since these terms acts as friction terms the natural conditions to add to (28) and (29) are no-slip conditions at the solid, closed boundaries. Thus in addition to (28) and (29) we require

$$V = 0 \quad ; \quad x = 0, L_x, \quad \forall y, \tag{30}$$

$$U = 0 \quad ; \quad y = 0, L_y, \quad \forall x. \tag{31}$$

3.1.2 Open boundaries

We recall that it is only the offshore, the northern and the southern boundaries that are open (Figure 4). The coast is still an impermeably wall. Thus for $y = 0$ the condition (29) prevails.

At the offshore boundary $y = L_y$ we simply require

$$\eta = 0 \quad ; \quad y = L_y, \quad \forall n. \tag{32}$$

At the southern and northern boundaries we make use of the so called Flow Relaxation Scheme as our boundary condition as explained in *Røed* (2012). Hence we relax the solution towards a specified external solution through a buffer zone close to the boundaries. At the southern boundary we then get

$$\psi(x,y,t) = [1-\gamma(x)]\psi^*(x,y,t) + \gamma(x)\psi^e_S(y,t) \quad ; \quad x = [0, L_F], \quad \forall y, t, \tag{33}$$

while at the northern boundary we get

$$\psi(x,y,t) = [1-\gamma(x)]\psi^*(x,y,t) + \gamma(x)\psi^e_N(y,t) \quad ; \quad x = [L_x - L_F, L_x], \quad \forall y, t. \tag{34}$$

Here ψ represents any of the dependent variables η, U or V, ψ^* is the solution to the original governing equations, L_F is the width of the buffer zone, or the FRS zone, $\psi^e_{S,N}$ is the specified solution at the southern (S) and northern (N) boundary, respectively, and $\gamma(x)$ is a relaxation parameter varying smoothly from 1 at the boundaries to 0 inside of the FRS zone.

We observe that since the boundaries are open the governing equations are still valid for $x = 0$, respectively $x = L_x$. Thus the external solutions $U^e_{S,N}$, $V^e_{S,N}$ and $\eta^e_{S,N}$ may be computed from the one-dimensional version of the governing equations. Hence we get

$$\partial_t U^e_{S,N} = fV^e_{S,N} + X - R\frac{U^e_{S,N}}{H}, \tag{35}$$

$$\partial_t V^e_{S,N} = -fU^e_{S,N} - gH\partial_y \eta^e + Y - R\frac{V^e_{S,N}}{H}, \tag{36}$$

$$\partial_t \eta^e_{S,N} = -\partial_y V^e_{S,N}, \tag{37}$$

where

$$X = \rho_0^{-1}\tau_s^x, \quad Y = \rho_0^{-1}\tau_s^y. \tag{38}$$

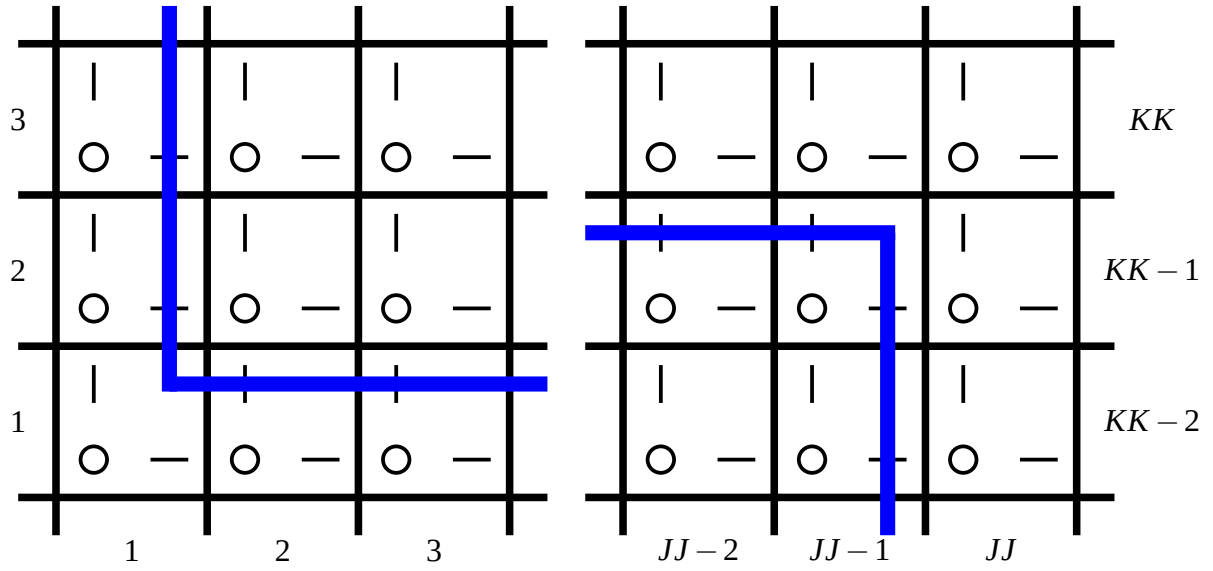


Figure 3: Displayed are the configuration of the physical boundaries in the southwest corner (a) and the northwest corner (b) of the computationals domain (Figure 4) relative to the grid structure shown in Figure 2. The boundaries are drawn as heavy, solid blue lines. Note that we let the physical boundaries go through V-points along the offshore and coastal boundaries and through U-points along the northern and southern boundaries. Note also that the cells JJ for $k = 1(1)KK$ and the cells KK for $j = 1(1)JJ$ are outside of the physical boundaries. As explained in the text their presence is necessary to account for the boundary condition associated with the non-linear version of the FDEs and when open boundaries are present.

3.2 Numerical implementation

3.2.1 Closed boundaries

In numerical terms (28) implies

$$U^n_{1k} = 0 \quad ; \qquad k = 1(1)KK, \quad \forall n, \tag{39}$$
$$U^n_{JJ-1k} = 0 \quad ; \qquad k = 1(1)KK, \quad \forall n, \tag{40}$$

while (28) requires

$$V^n_{j1} = 0 \quad ; \qquad j = 1(1)JJ, \quad \forall n, \tag{41}$$
$$V^n_{jKK-1} = 0 \quad ; \qquad j = 1(1)JJ, \quad \forall n. \tag{42}$$

For the non-linear version we have to implement the additional conditions (30) and (31). This is slightly more complicated since the U-points are not collocated with the physical coastal and offshore boundaries, and the V-points are not collocated with the physical location

of the southern and northern boundaries. We solve this problem by adding extra cells outside of the physical boundaries as displayed in the right panel of Figure 3, and performing a linear interpolation across the boundary. Thus we require

$$U^n_{j1} = -U^n_{j2} \quad ; \qquad j = 1(1)JJ, \quad \forall n, \tag{43}$$

$$U^n_{jKK} = -U^n_{jKK-1} \quad ; \qquad j = 1(1)JJ, \quad \forall n, \tag{44}$$

while at the southern and northern boundaries we require

$$V^n_{1k} = -V^n_{2k} \quad ; \qquad k = 1(1)KK, \quad \forall n, \tag{45}$$

$$V^n_{JJk} = -V^n_{JJ-1k} \quad ; \qquad k = 1(1)KK, \quad \forall n. \tag{46}$$

We furthermore emphasize that when the boundaries are closed the evaluation of the Coriolis terms at the wet points closest to the boundaries must be changed to avoid them to be too small by a factor of two. Thus along the coastal boundary $k = 2$ the terms containing $\overline{V}^n_{j2}$ in (8) and (18) must be replaced by

$$\overline{V}^n_{j2} = \frac{1}{2}\left(V^n_{j2} + V^n_{j+12}\right), \quad j = 2(1)JJ-2 \quad \forall n. \tag{47}$$

Similarly along the offshore boundary $k = KK-1$ we replace them by

$$\overline{V}^n_{jKK-1} = \frac{1}{2}\left(V^n_{jKK-2} + V^n_{j+1KK-2}\right), \quad j = 2(1)JJ-2 \quad \forall n. \tag{48}$$

Along the southern and northern boundaries the terms containing $\overline{U}^n_{jk}$ and $\overline{U}^n_{jk}$ in (9) and (19) closest to the boundaries, that is, for $j = 2$ and $j = JJ-1$, must be replaced by

$$\overline{U}^n_{2k} = \frac{1}{2}\left(U^n_{2k} + U^n_{2k+1}\right), \quad k = 2(1)KK-2 \quad \forall n. \tag{49}$$

and

$$\overline{U}^n_{JJ-1k} = \frac{1}{2}\left(U^n_{JJ-2k} + U^n_{JJ-2k+1}\right), \quad k = 2(1)KK-2 \quad \forall n. \tag{50}$$

respectively.

3.2.2 Open boundaries

Since the η-points are not collocated with the offshore boundary, but half way in between η^n_{jKK} and η^n_{jKK-1} we use the same trick of performing a linear interpolation. Thus we require

$$\eta^n_{jKK} = -\eta^n_{jKK-1} \quad ; \quad j = 1(1)JJ, \quad \forall n, \tag{51}$$

which then is the numerical implementation of (32). In addition to (51) we also let

$$U^n_{jKK} = V^n_{jKK} = 0 \quad ; \quad j = 1(1)JJ, \quad \forall n, \tag{52}$$

Table 1: Values of the relaxation parameter γ_j used in (47)

j_{FS}	γ_j	j_{FN}
1	1.00	JJ
2	0.69	$JJ-1$
3	0.44	$JJ-2$
4	0.25	$JJ-3$
5	0.11	$JJ-4$
6	0.03	$JJ-5$
7	0.00	$JJ-6$

that is, we let the transports be zero at points outside of the physical boundaries. Due to the latter we have to, as we did when the offshore boundary was closed, replace the terms containing $\overline{V}^n_{jk}$ for $k = KK-1$ by (48).

Regarding the northern and southern boundaries, we have to implement the conditions (33) and (34). Accordingly the FDEs we use to compute the external solutions at the southern boundary are the numerical version of (35) - (37) using a reduced version of (8) -(10) in which all terms representing differentiation across the boundary is set to zero, that is,

$$U^{en+1}_{Sk} = \frac{H^x_{1k}}{H^x_{1k}+R\Delta t}\left[U^{en}_{Sk}+\frac{1}{2}f\Delta t\left(V^{en}_{Sk}+V^{en}_{Sk-1}\right)+\Delta t X^{n+1}_{1k}\right], \tag{53}$$

$$V^{en+1}_{Sk} = \frac{H^y_{1k}}{H^y_{1k}+R\Delta t}\left[V^{en}_{Sk}-\frac{f\Delta t}{2}\left(U^{en+1}_{Sk+1}+U^{en+1}_{Sk}\right)+\frac{\Delta t}{\Delta y}\left(\eta^{en}_{Sk+1}-\eta^{en}_{Sk}\right)+\Delta t Y^{n+1}_{1k}\right] \tag{54}$$

$$\eta^{en+1}_{Sk} = \eta^{en}_{Sk}-\frac{\Delta t}{\Delta y}\left(V^{en+1}_{Sk}-V^{en+1}_{Sk-1}\right), \tag{55}$$

while at the northern boundary we get

$$U^{en+1}_{Nk} = \frac{H^x_{JJ-1k}}{H^x_{JJ-1k}+R\Delta t}\left[U^{en}_{Nk}+\frac{f\Delta t}{2}\left(V^{en}_{Nk}+V^{en}_{Nk-1}\right)+\Delta t X^{n+1}_{JJ-1k}\right], \tag{56}$$

$$V^{en+1}_{Nk} = \frac{H^y_{JJk}}{H^y_{JJk}+R\Delta t}\left[V^{en}_{Nk}-\frac{f\Delta t}{2}\left(U^{en+1}_{Nk+1}+U^{en+1}_{Nk}\right)+\frac{\Delta t}{\Delta y}\left(\eta^{en}_{Nk+1}-\eta^{en}_{Nk}\right)+\Delta t Y^{n+1}_{JJk}\right] \tag{57}$$

$$\eta^{en+1}_{Nk} = \eta^{en}_{Nk}-\frac{\Delta t}{\Delta y}\left(V^{en+1}_{Nk}-V^{en+1}_{Nk-1}\right). \tag{58}$$

We note that (53) and (55) and (56) and (58) are valid for $k = 2(1)KK-1$, while (54) and (57) are valid for $k = 2(1)KK-2$. For the remaining k's the appropriate boundary conditions at the coast ($k = 1$) and offshore ($k = KK$ and $k = KK-1$) takes over. Recall that the Coriolis term must be replaced by $fU^{e}_{S,N}{}^{n+1}_{jKK-1}$ in (54) and (57), respectively, to avoid reducing it at $k = KK-1$.

The final solution for time level $n+1$ is then found by relaxing the predicted solution to-

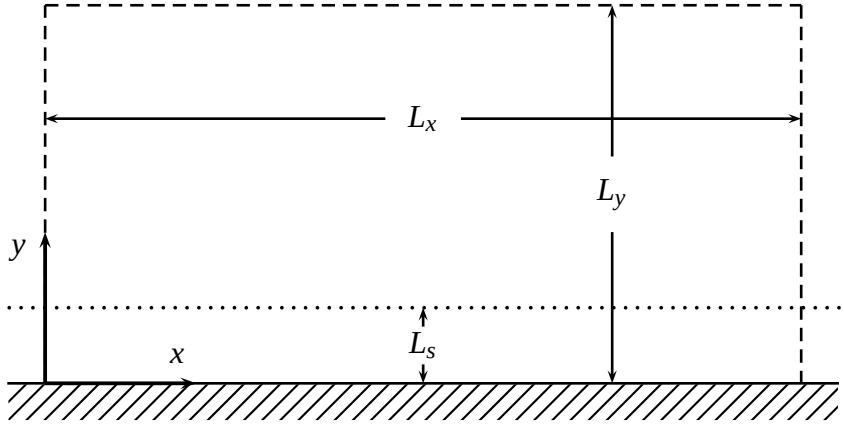


Figure 4: Sketch of the computational domain used for all the benchmark cases. The x-axis is along the straight coast and points in the northward direction. The y-axis points offshore in the westward direction. The domain is $L_x = 1000$ km long and $L_y = 500$ km wide (Table 2). The offshore boundary and the boundaries to the south and north are either open or closed. The dotted line indicates the location of the shelf break located $L_s = 100$ km offshore.

wards the external solution through the so called FRS zones. At the southern FRS we get

$$U_{jk}^{n+1} = (1-\gamma_j)U_{jk}^{n+1} + \gamma_j U_{Sk}^{en+1} \quad j = 1(1)J_{FS}, \quad k = 1(1)KK, \tag{59}$$

$$V_{jk}^{n+1} = (1-\gamma_j)V_{jk}^{n+1} + \gamma_j V_{Sk}^{en+1} \quad j = 1(1)J_{FS}, \quad k = 1(1)KK, \tag{60}$$

$$\eta_{jk}^{n+1} = (1-\gamma_j)\eta_{jk}^{n+1} + \gamma_j \eta_{Sk}^{en+1} \quad j = 1(1)J_{FS}, \quad k = 1(1)KK. \tag{61}$$

while in the northern FRS we get

$$U_{jk}^{n+1} = (1-\gamma_{j+1})U_{jk}^{n+1} + \gamma_{j+1} U_{Nk}^{en+1} \quad j = J_{FN}(1)JJ-1, \quad k = 1(1)KK, \tag{62}$$

$$V_{jk}^{n+1} = (1-\gamma_j)V_{jk}^{n+1} + \gamma_j V_{Nk}^{en+1} \quad j = J_{FN}(1)JJ, \quad k = 1(1)KK, \tag{63}$$

$$\eta_{jk}^{n+1} = (1-\gamma_j)\eta_{jk}^{n+1} + \gamma_j \eta_{Nk}^{en+1} \quad j = J_{FN}(1)JJ, \quad k = 1(1)KK, \tag{64}$$

where the counters J_{FS} and J_{FN} and the associated values of the relaxation parameter γ_j are given in Table 1.

4 Benchmark cases

The geometry of the computational domain is the same for all benchmark cases, and is chosen to simulate a shelf sea (Figure 4). The integration area is rectangular and consists of a straight coast (along the x-axis) oriented north-south, an offshore boundary running parallel to the coast, and lateral boundaries to the north and south.

Table 2: Parameters and physical constants used in the benchmark cases

Symbol	Parameter	Value	Unit
f	Coriolis parameter	$1.2{\cdot}10^{-4}$	s^{-1}
ρ_0	Density of sea water	1025.0	kg m^{-3}
g	Gravitational acceleration	9.81	m s^{-2}
R	Bottom friction coefficient	$2.4{\cdot}10^{-3}$	m s^{-1}
H_0	Equilibrium depth on shelf	50.0	m
H_1	Equilibrium depth offshelf	2500.0	m
L_x	Alongshore length of domain	1000.0	km
L_y	Width of domain	500.0	km
L_s	Shelf width	100.0	km
Δx	Alongshore spatial increment	20.0	km
Δy	Cross-shore spatial increment	20.0	km
Δt	Time increment	90.0	s
α	Offshore e-folding length ($= 1/10\Delta x$)	$5.0{\cdot}10^{-6}$	m^{-1}
τ_0	Amplitude of wind stress	0.1	Pa
τ_1	Maximum wind stress moving cyclone	3.0	Pa
R_C	Distance to maximum wind stress	.	
	from center of cyclone ($= 10\Delta x$)	200.0	km
$\lvert\mathbf{u}_C\rvert$	Translation speed of cyclone (Case 3)	15.0	m s^{-1}

There are three sets of three benchmark cases, making a total of nine cases as outlined in Table 2. The first set features a uniform wind dying exponentially offshore. The second set features an alongshore wind which is bell-shaped along the coast and dying exponentially offshore. The third and final set features a wind forcing in the form of a moving cyclone propagating diagonally across the computational domain. The various parameters and physical constants used are given in Table 2. The three specified wind forces are similar to those found in *Røed and Cooper* (1987). They also derived an analytic solution for the first case, which is copied in Section 6 below.

Of the three sub-cases the first assumes the domain to be flat bottomed, and with closed boundaries everywhere. The second still assumes the domain to be flat bottomed, but opens up the offshore, northern and southern boundaries. In the final sub-case the bottom is constructed to simulate a shelf sea in which a shallow shelf runs parallel to the coast $L_s = 100$ km offshore (Figure 4).

4.1 Case 1: Uniform alongshore wind

The aim is to test whether the model produces a proper storm surge along the coast. Thus we let the forcing consist of a positive alongshore wind stress decaying exponentially offshore,

Table 3: Overview of benchmark cases for which solutions are presented

Benchmark case	Boundary type	Shelf	Wind forcing
1a	Closed	No	Uniform
1b	Open	No	- ” -
1c	Open	Yes	- ” -
2a	Closed	No	Bell-shaped
2b	Open	No	- ” -
2c	Open	Yes	- ” -
3a	Closed	No	Moving storm
3b	Open	No	- ” -
3c	Open	Yes	- ” -

that is,

$$X = X_0 e^{-\alpha y}, \quad Y = 0, \tag{65}$$

where $X_0 = \tau_0/\rho_0$ where $\tau_0 = 0.1$ Pa, and the value chosen for α corresponds to an e-folding length of 200 km, implying that only about 8% of the wind stress is experienced at the offshore boundary. Numerically this translates to

$$X^n_{jk} = X_0 e^{\left(\frac{2k-3}{20}\right)}, \quad Y^n_{jk} = 0.0, \quad \forall j,k,n. \tag{66}$$

As shown by *Røed and Cooper* (1987) it is possible to derive an analytic solution for the steady state solution ($\partial_t = 0$) for a flat bottom case with open boundaries to the north and south and with $\eta = 0$ at the offshore boundary. The steady state solution is

$$U = \frac{HX_0}{R}, \tag{67}$$

$$\eta = \frac{fX_0}{\alpha g R}\left(e^{-\alpha y} - e^{-\alpha L_y}\right), \tag{68}$$

and is characterized by a positive alongshore depth mean current (U/H), and an elevation at the coast. With the values listed in Table 2 the depth mean current strength is 0.04 m/s, and the elevation at the coast is 0.91 m.

4.2 Case 2: Bell-shaped alongshore wind

In this case the positive alongshore wind stress is limited to a stretch along the coast by letting its amplitude vary according to a Gaussian bell. We also shut off the wind after two days (48 hours). Thus we let

$$X = X_0 \begin{cases} e^{-\alpha^2 (x-x_m)^2} e^{-\alpha y} & \text{if } 0 \le t \le 48 \text{ hrs} \\ 0 & \text{if } t > 48 \text{ hrs} \end{cases}, \quad Y = 0, \tag{69}$$

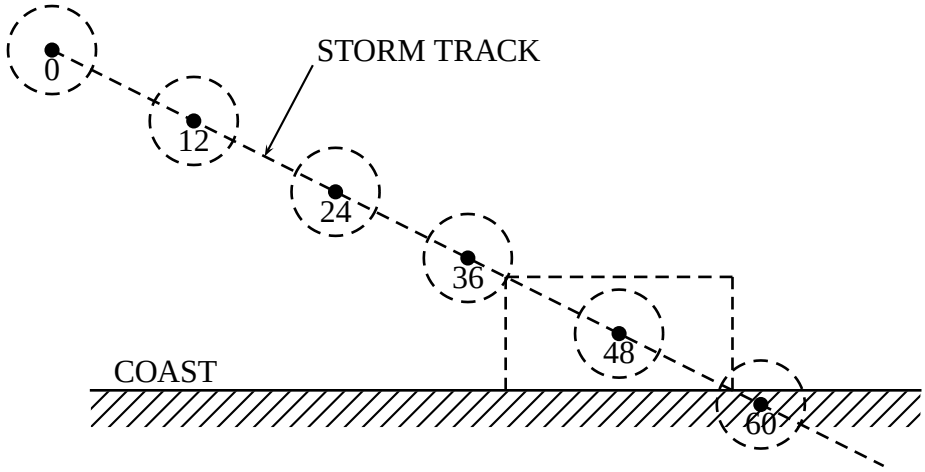


Figure 5: Sketch of the storm track relative to the computational domain shown in Figure 4 for Case 3: Moving cyclone. Numbers (in hours) along track indicates position of cyclone center after start of simulation. The dashed circles indicate the radius to maximum wind stress (200 km). Maximum wind stress is 3.0 Pa. The dashed rectangle shows the position of computational domain relative to the storm track.

where x_m is the position of maximum wind stress ($j = j_m$). Translated into numerics this implies

$$X^n_{jk} = X_0 \begin{cases} e^{-\left(\frac{j-j_m}{10}\right)^2} e^{-\frac{2k-3}{20}} & \text{if } n \le 1920 \\ 0 & \text{if } n > 1920 \end{cases}, \quad Y^n_{jk} = 0, \quad \forall j,k,n, \tag{70}$$

where we have used the parameter values listed in Table 2. This means that that both the alongshore and cross-shore e-folding lengths are the same (200 km). Thus at the offshore boundary the wind stress is again only 8 % of its value at the coast, and at the northern and southern boundaries it is only 0.2 %.

4.3 Case 3: Moving cyclone

In this case the wind stress is generated by cyclone translating at $|\mathbf{u}_C| = 15$ m/s diagonally across the domain from the southwest corner to the northeast corner as depicted in Figure 5. The translation speed is slow enough for Kelvin waves to propagate ahead of the storm. The cyclone center passes the southwest corner approximately 38 hrs into the integration, is positioned in the center of the computational domain at 48 hours and leaves the domain at about 58 hours.

The wind stress components are given by

$$X = -X_1 \left(\frac{y - y_0 - v_C t}{R_C} \right) e^{-\xi/2}, \quad Y = X_1 \left(\frac{x - x_0 - u_C t}{R_C} \right) e^{-\xi/2}, \tag{71}$$

where the argument of the exponential function, ξ, is given by

$$\xi = \left(1 - \frac{r}{R_C}\right)^2, \quad r = \sqrt{(x - x_0 - u_C t)^2 + (y - y_0 - v_C t)^2}. \tag{72}$$

Here $X_1 = \rho_0^{-1}\tau_1$ where $\tau_1 = 3$ Pa is the maximum wind stress and $R_C = 200$ km is the radius from the center of the cyclone to the position of maximum wind stress (Table 2). Moreover, u_C and v_C are the x,y components of the translation velocity $\mathbf{u}_C$ of the storm center, where $u_C = 2v_C$, while x_0,y_0 is the initial position of the storm center at start of the simulation ($t = 0$). The translation speed is $|\mathbf{u}_C| = \sqrt{u_0^2 + v_0^2} = 15$ m/s (Table 2).

5 Solutions

Below we present solutions derived using the three wind forcing cases. For each wind forcing case, as outlined in Table 3, we have run three sub-cases . The first of these has closed boundaries everywhere and a flat bottom (sub-case a). The second has open boundaries to the south, north and offshore, and a flat bottom (sub-case b). The final has open boundaries to the south, north and offshore and a shelf running parallel to the coast (sub-case c).

For each wind forcing case we present the solutions in form of time series (Figures 6, 7 and 8) and snapshots of sea surface elevation (Figures 9, 10 and 11). The time series are extracted half a grid distance from the coast (10 km) and midway between the southern and northern boundaries ($j = 26, k = 2$). The time series reveal the temporal evolution of the sea surface elevation, the alongshore and cross-shore mean depth currents. The snapshots of the sea surface elevation are extracted at 96 hours for Case 1 and at 48 hours for Cases 2 and 3.

We note that for Case 1b (uniform alongshore wind forcing, flat bottom, open boundaries) the steady state solution shown in the middle panel in Figure 6 matches perfectly the steady state solution given in Section 4.1.

To plot the results we have used free software packages. For the time series we used Grace (http://plasma-gate.weizmann.ac.il/Grace/) under the GNU General Public License, while we used Ncview: a netCDF visual browser by David W. Pierce, Scripps Institution of Oceanography (http://meteora.ucsd.edu/ pierce/ncview_home_page.html) to create the snapshots.

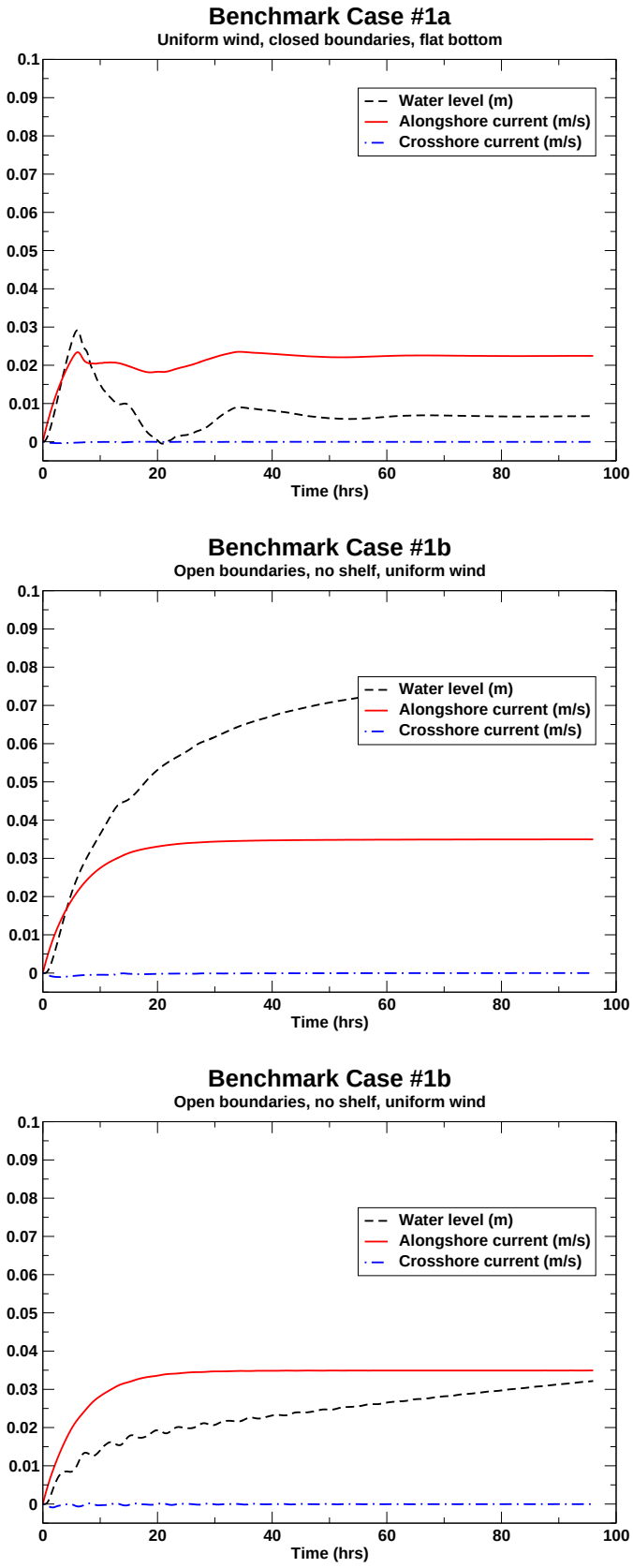


Figure 6: For caption see List of Figures

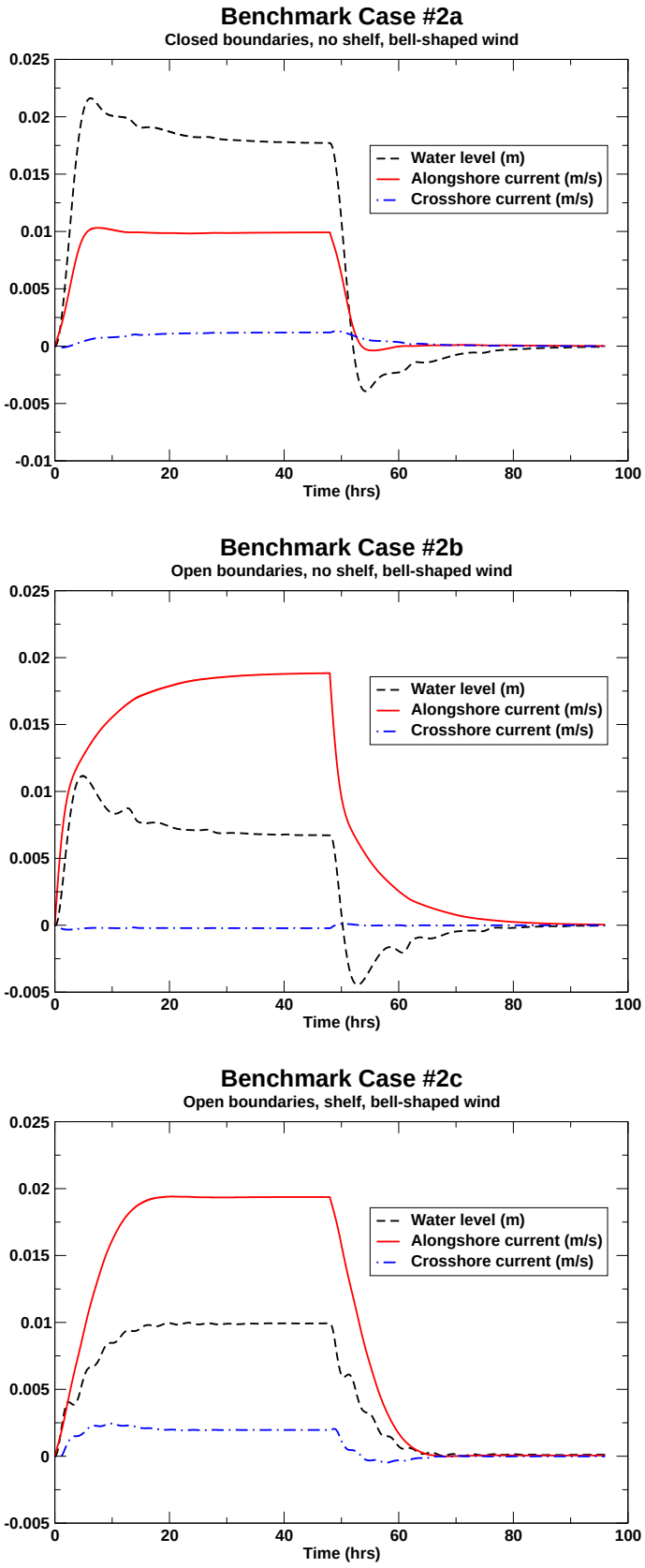


Figure 7: As Figure 6, but for Case 2: Bell-shaped wind forcing (Table 3).

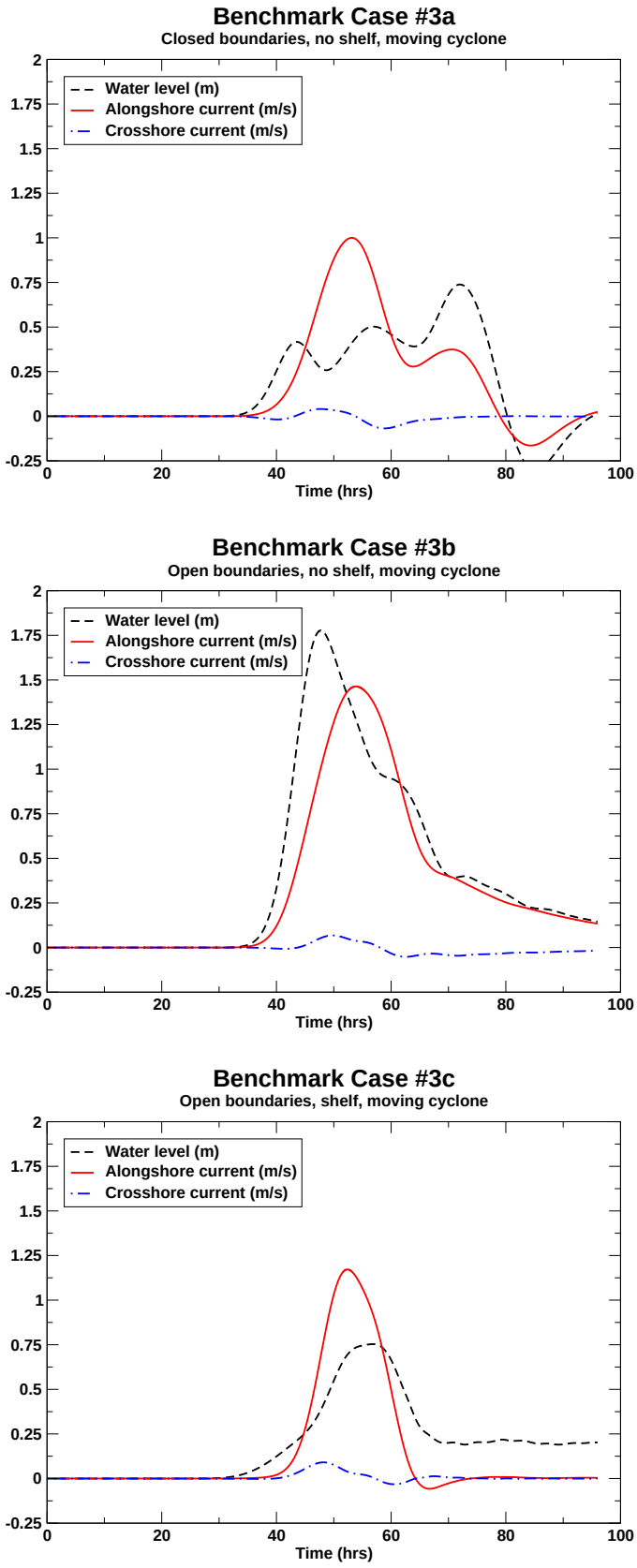


Figure 8: As Figure 6, but for Case 3: Moving cyclone case (Table 3).

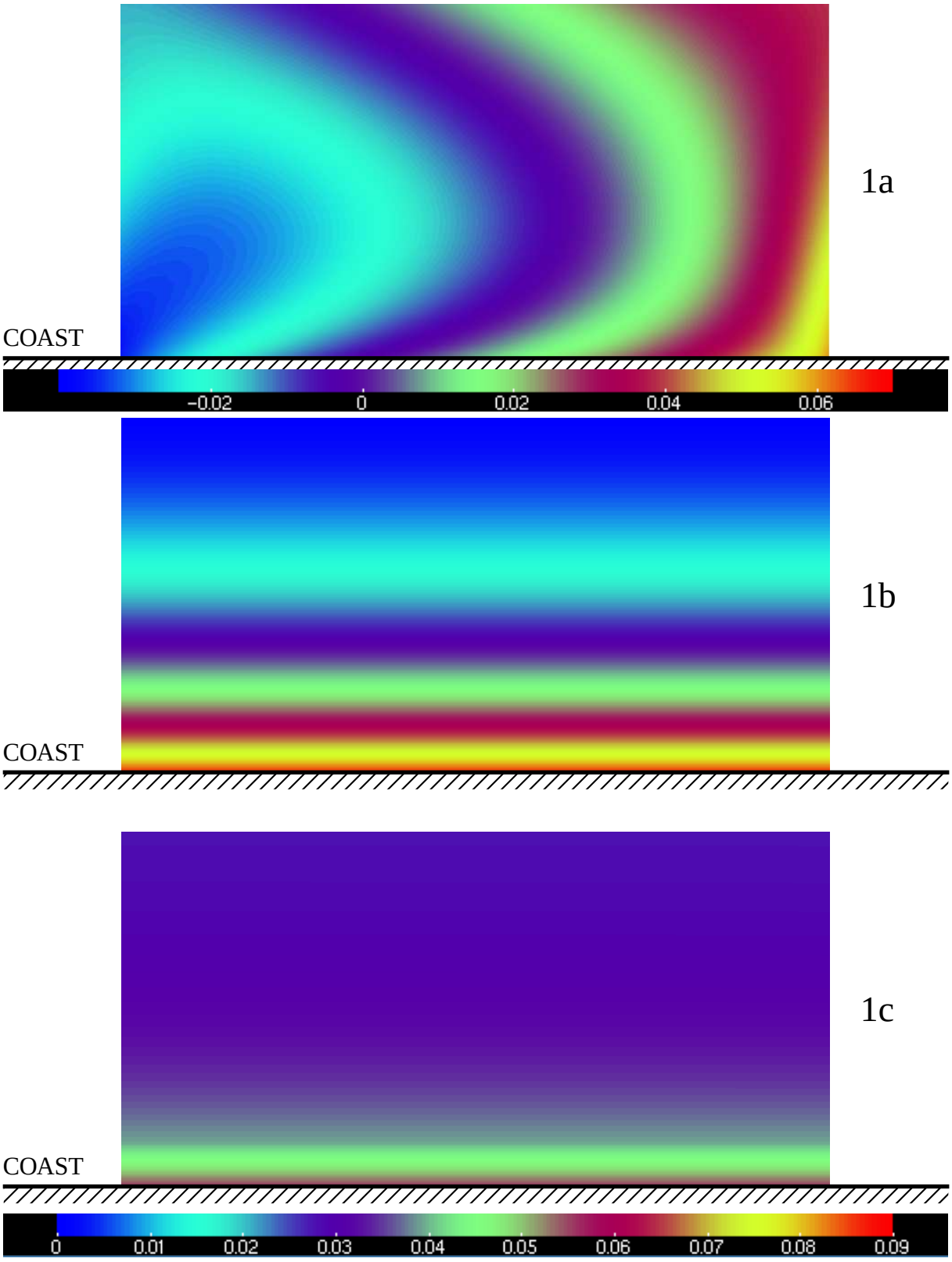


Figure 9: Sea surface elevation after 96 hours for Cases 1a (top), 1b (middle) and 1c (bottom) (Table 3). Color scale at bottom indicates elevation (cm) for Cases 1b and 1c, while color scale below top panel indicates scale for Case 1a (cm).

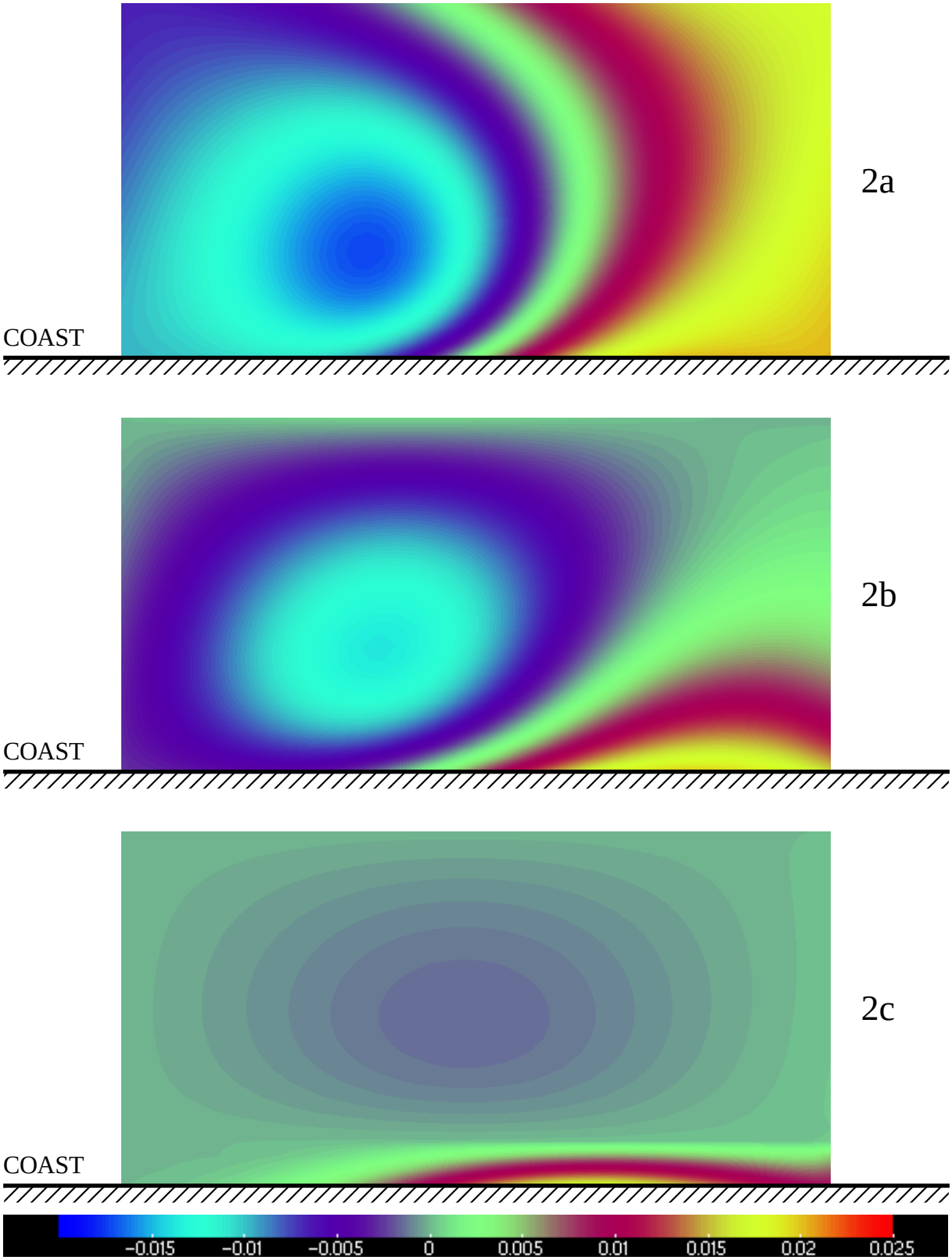


Figure 10: As Figure 9, but for Cases 2a (top), 2b (middle) and 2c (bottom) and after 48 hours. Color scale at bottom is the same for all three panels and indicates elevation in cm.

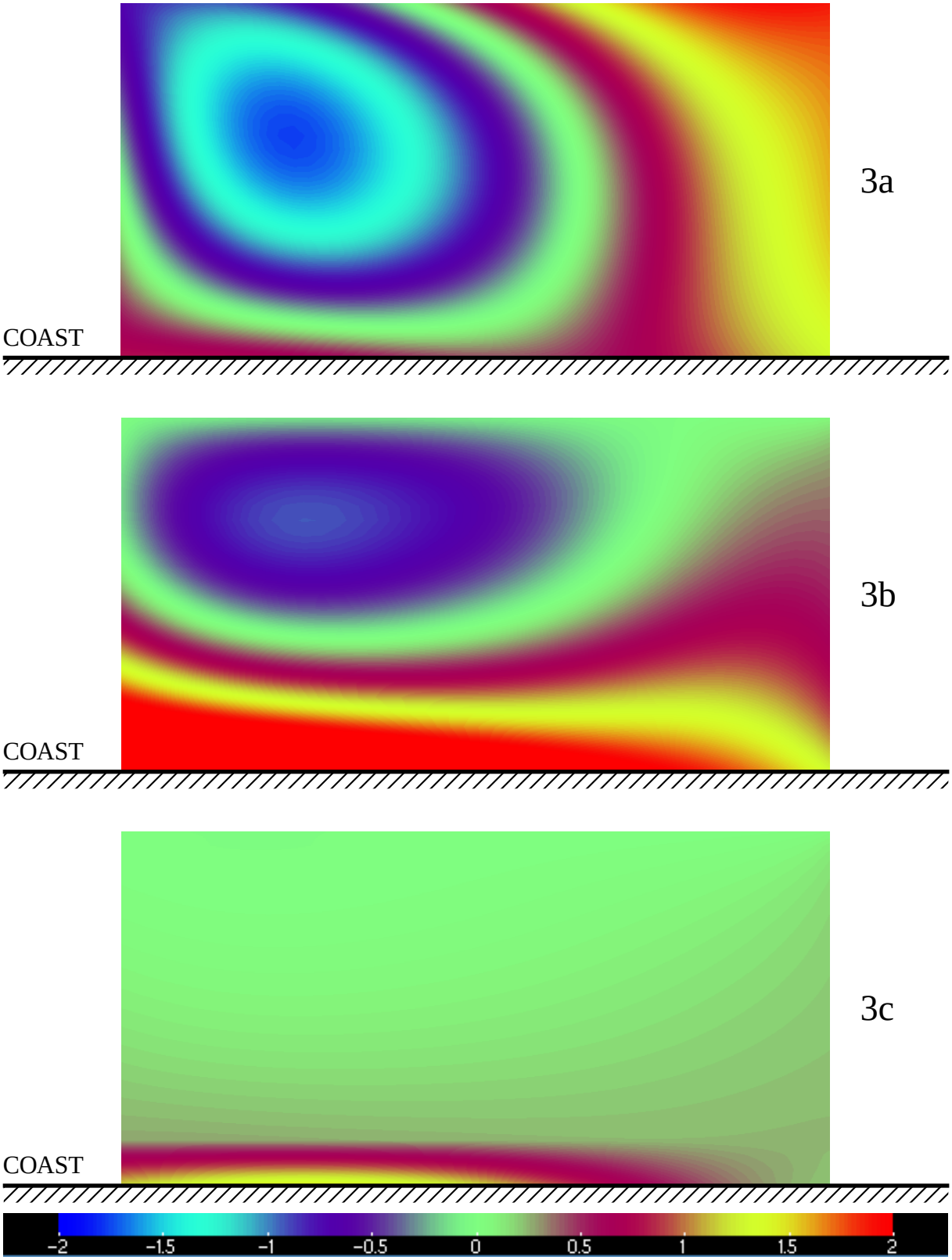


Figure 11: As Figure 10, but for Cases 3a (top), 3b (middle) and 3c (bottom).

6 Discussions

We observe from Figure 6 that due to presence of a bottom stress all Case 1 solutions (application of a uniform wind stress decaying offshore) reach a steady state within 20 - 40 hours. Note that the latter numbers depends on the frictional scale given by H_0/R. Thus the deeper the ocean the longer it takes to reach the steady state. The same is true for Case 2, but after the wind is shut off after 48 hours a spin-down is experienced due to the application of the bottom stress.

When the boundaries are closed Kelvin waves, and also shelf edge waves in sub-case c, starts to propagate cyclonically (anticlockwise) in the basin, and this causes the transient response to be quite different in sub-case a from sub-cases b and c having open boundaries. In the latter cases the Kelvin waves and shelf edge waves are allowed to propagate out of the domain through the open boundary to the north. In Case 1a with a persistent wind stress a steady state is reached with generally higher water levels at northern boundary (Figure on front page) and a circulation with northward currents along the coast and southwards along the offshore coast (Figure 9). In Case 2a we also observe how Kelvin waves are generated when the wind is shut off. These waves are damped due to the application of bottom friction.

As outlined in *Røed and Cooper* (1987) it is possible to obtain an exact, analytic steady state solution for Case 1b, that is, in the open boundary, flat bottom case. The solution is found by solving the time independent and one-dimensional version of (1) -(3), that is, with $\partial_t = 0$ and $\partial_x = 0$. The solution is

$$U_s(y) = \frac{H_0 X_0}{R} e^{-\alpha y}, \quad V_s = 0, \tag{73}$$

$$\eta_s(y) = \frac{f X_0}{\alpha g R} \left(e^{-\alpha y} - e^{-\alpha L_y} \right). \tag{74}$$

where the subscript s is used to denote the steady state solution. We note that the steady state solution for the mean depth current at the point (26,2) then is

$$\frac{U_{26,2}}{H_0} = \frac{\tau_s^x}{\rho_0 R} e^{-\frac{1}{2}\alpha \Delta y} = 3.9\text{cm/s}, \tag{75}$$

where we have used the values listed in Table 2 to arrive at the number after the last equal sign. We observe that this matches exactly the value shown in the middle and bottom panels of Figure 6 (Cases 1b and 1c).

7 Summary

In this Part II we derive the solutions to three benchmark cases (Table 3). In all cases the computational domain is rectangular with a 1000 km long coast along the x-axis. The latter is oriented in the north-south direction. The offshore boundary is located 500 km offshore. The domain is either flat bottomed or features a shelf running parallel to the coast 100 km offshore. The coastal boundary is always closed (impermeable), while the remaining boundaries are either open or closed. All cases are run for 96 hours.

Case 1 treats the response to a constant positive alongshore wind stress, decaying exponentially offshore from its maximum value of 1 Pa at the coast. Case 2 is similar to Case 1, but the alongshore wind stress is now limited to a Gaussian bell in the middle of the computational domain and is shut off after 48 hours. Case 3 treats the transient response to a moving, circular cyclone moving diagonally across the computational domain from the southwest corner to the northeast corner. The maximum wind stress of the cyclone is 3 Pa.

The solutions are displayed as time series of sea surface elevation, and alongshore and cross-shore mean depth currents half a grid distance from the coast midway between the southern and northern boundaries. In addition snapshots of the sea surface elevation after 96 hours is shown for Case 1, while snapshots of the sea surface elevation after 48 hours are displayed for Cases 2 and 3.

Appendix: FORTRAN programs for one-layer model

Below follows the source code for the FORTRAN program written on basis of the FDEs for the one-layer barotropic model. Only the linear version is presented.

Appendix A: FORTRAN program, closed boundaries

```
PROGRAM SHALLW
  !-------------------------------------------------------------
  !  PURPOSE:
  !
  !   - To produce results for the benchmark cases
  !     with closed boundaries everywhere, that is,
  !     Cases 1a, 2a, and 3a
  !
  !   - Closed boundaries everywhere and no shelf.
  !     Case 1a: A positive uniform alongshore wind stress of
  !              strength 0.1 Pa decaying offshore.
  !     Case 2a: Bell-shaped wind along the coast, of maximum
  !              strenght 0.1 Pa decaying offshore.
  !     Case 3a: Moving cyclone of maximum strength 3.0 Pa
  !              moving diagonally across the basin from
  !              southwest to northeast. Distance from center
  !              to max strength is 200 km. The cyclone is in
  !              the middle of the domain at 48hrs.
  !
  !-------------------------------------------------------------
  !  METHOD:
  !
  !   - The linear, shallow water equations to solve are
  !
  !       dU/dt = + F*V - G*H*dE/dx + (TSX - TBX)/RHO,
  !       dV/dt = - F*U - G*H*dE/dy + (TSY - TBY)/RHO.
  !       dE/dt = - dU/dx - dV/dy,
  !-90
  !     Here x is positive northward, y is positive westward
  !     and t is time. U(x,y,t) and V(x,y,t) are the volume
  !     transport components along the x,y axes, while
  !     E(x,y,t) is the sea surface deviation away from its
  !     equilibrium position, F is the Coriliolis parameter,
  !     G is thegravitational acceleration, H(x,y) is the
  !     equilibrium depth, TSXand TSY are the two horizontal
  !     components of the wind stress, and TBX and TBY the
  !     similar components of the bottom stress. RHO is a
```

```
!     reference water density.
!
!   - All boundaries are closed. Thus U = 0 at the northern
!     and southern boundaries while V=0 at the coast and
!     offshore boundary.
!
!   - Integration is performed using finite difference
!     methods:
!      1. Forward-Backward scheme (Martinsen et al., 1979),
!      2. Wind and bottom stress are treated backwards
!
!-----------------------------------------------------------
!   FINITE DIFFERENCE EQUATIONS:
!
!   - The grid is staggered, corresponding to lattice C of
!     Mesinger and Arakawa (two dimensions) to avoid over-
!     specifying the variables at the boundaries. The
!     finite difference equations are:
!
!      U(J,K,INEW) = BU*(U(J,K,ICUR) + CU + PX + XX),
!      V(J,K,INEW) = BV*(V(J,K,ICUR) + CV + PY + XY),
!      E(J,K,INEW) = E(J,K,ICUR) - DU - DV,
!
!     where
!
!      DU = (DT/DX)*(U(J,K,INEW) - U(J-1,K,INEW)),
!      DV = (DT/DY)*(V(J,K,INEW) - V(J,K-1,INEW)),
!      VC = V(J,K,ICUR) + V(J+1,K,ICUR)
!                       + V(J+1,K-1,ICUR) + V(J,K-1,ICUR),
!      UC = U(J,K,INEW) + U(J-1,K,INEW)
!                       + U(J-1,K+1,INEW) + U(J,K+1,INEW),
!      CU = F*DT*VC,
!      CV = - F*DT*UC,
!      PX = - G*(DT/DX)*HU*(E(J+1,K,ICUR) - E(J,K,ICUR)),
!      PY = - G*(DT/DY)*HV*(E(J,K+1,ICUR) - E(J,K,ICUR)),
!      XX = (TBX(J,K,INEW)/RHO),
!      YY = (TBY(J,K,INEW)/RHO),
!      HU = 0.5*(H(J+1,K) + H(J,K)),
!      HV = 0.5*(H(J,K+1) + H(J,K)),
!      BU = HU/(HU + RDT),
!      BV = HV/(HV + RDT).
!
!     The various physical constants are
!
```

```
!      DT          : Time step in seconds
!      DX          : Space increment in meters along x-axis
!      DY = DX     : Space increment in meters along y-axis
!      F           : Coriolis parameter in 1/s
!      RHO         : Reference water density in kg/m**3
!      G           : Gravitational acceleration in m/s**2
!      RR          : Friction coefficent in m/s
!      T0          : Amplitude of wind stress (N/m**2)
!      X0 = T0/RHO : Maximum wind stress divided by water
!                    density
!      H0          : Equilibrium depth on shelf (m)
!      H1          : Equilibrium depth off shelf (m)
!
!     In addition we define
!      J                   : Counter J=1(1)JJ starting at
!                            J=1, X(1)=0.
!      K                   : Counter K=1(1)KK starting at
!                            K=1, Y(1)=0.
!      X(J)   = (J-1)*DX   : Position along x-axis of
!                            U-points in m
!      Y(K)   = (K-1)*DY   : Position along y-axis of E-
!                            and V-points in m
!      XKM(J) = 0.001*X(J) : As X(J), but in km
!      YKM(K) = 0.001*Y(K) : As Y(K), but in km
!
!-------------------------------------------------------------
!  COMMUNICATION:
!
!   - Synoptic values of sea surface elevation, velocity
!     components are saved on the specified netCDF file
!     every hour.
!
!   - A string of values are printed to terminal while
!     running the program.
!
!   - Time series (ASCII) are saved on file unit 52.
!
!-------------------------------------------------------------
!  STATUS:
!
!  - CODED BY       : LARS PETTER ROED, met.no, Oslo
!  - DATE           : 27 February 2012
!
!  REVISION HISTORY :
```

```
!  - DATE           : 13 March, 2012
!  - Description    : Merged all closed boundary cases into
!                     one program
!-----------------------------------------------------------
!
use netcdf
!
implicit none
!
!-----------------------------------------------------------
!---------------------- Specify dimensions -----------------
!                                                 Grid and time
INTEGER, PARAMETER :: JJ  = 52   ! # cells in x direction
INTEGER, PARAMETER :: KK  = 27   ! # cells in y-direction
INTEGER, parameter :: JP  = 50   ! # cells in x for saving
INTEGER, parameter :: KP  = 25   ! # cells in y for saving
INTEGER, PARAMETER :: NT  = 2    ! # of time levels scheme
INTEGER, PARAMETER :: NTM = 3840 ! Max # of time steps
INTEGER, parameter :: KS  = 5    ! Location of shelf break
INTEGER, parameter :: KM  = 13   ! Counter mid basin in y
INTEGER, parameter :: JM  = 26   ! Counter mid basin in x
!
!---------------------- Dimensionalize everything ----------
!                           1. Dependent variables (arrays)
REAL E(JJ,KK,NT)   ! Sea surface anomaly (m)
REAL U(JJ,KK,NT)   ! Volume transport along x-axis (m/s)
REAL V(JJ,KK,NT)   ! Volume transport along y-axis (m/s)
!                          2. Independent variables (arrays)
REAL XX(JJ,KK,NT)  ! Wind stress along x-axis
!                    divided by water density (m**2/s**2)
REAL YY(JJ,KK,NT)  ! Wind stress along y-axis
!                    divided by water density (m**2/s**2)
!
!                                    3. Time series variables
REAL EP(JP,KP) ! Saved depth anomaly (m)
REAL UP(JP,KP) ! Saved velocity along x at E-point (m/s)
REAL VP(JP,KP) ! Saved velocity along y at E-point (m/s)
!
!                                4. Other variables and flags
REAL X(JJ),XKM(JJ) ! Distance along x-axis (m and km)
REAL Y(KK),YKM(KK) ! Distance along y-axis (m and km
REAL H(JJ,KK)      ! Equilibrium depth (m)
INTEGER J,K        ! Counter indexes along x,y
INTEGER IT         ! Index for time level number
```

```
INTEGER ICUR       ! Counter for current time level (n)
INTEGER INEW       ! Counter for new time level (n+1)
INTEGER ISAVE      ! Auxiliary time level counter
INTEGER IWIND      ! Flag to specify wind forcing:
!                    1: Uniform wind
!                    2: Bell-shaped wind
!                    3: Moving cylone
!
!---------------------- Specify physical constants ---------
INTEGER, parameter :: DT  = 90 ! Time step (s). 3600
!                                has to be multiplyer of DT
REAL, parameter :: DX  = 20000.  ! Space increment -> x (m)
REAL, parameter :: DY  = 20000.  ! Space increment -> y (m)
REAL, parameter :: F   = 0.00012 ! Coriolis parameter
REAL, parameter :: RHO = 1025.   ! Water density (kg/m**3)
REAL, parameter :: G   = 9.81    ! Grav. accel. (m/s**2)
REAL, parameter :: RR  = 0.0024  ! Drag coefficents bottom
!                                  stress (m/s)
REAL, parameter :: H0  = 50.     ! Flat bottom equilibrium
!                                  depth (m)
REAL, parameter :: H1  = 2500.   ! Equilibrium depth off-
!                                  shelf (m)
REAL, parameter :: U0C = 15.     ! Translation speed center
!                                  of moving cyclone (m/s)
REAL, parameter :: RC  = 200000. ! Distance to maximum wind
!                                  stress from center of
!                                  cyclone (m)
!
!---------------------- Specify auxiliary variables --------
REAL DU,DV,VC,UC,CU,CV,PX,PY,PXI,HU,HV,BU,BV
REAL ARG1,ARG2,ARG3,ARG4
!
!---------------------- Specify auxiliary help constants ---
REAL, parameter    :: DTX = DT/DX   ! Coefficient (s/m)
REAL, parameter    :: DTY = DT/DY   ! Coefficient (s/m)
REAL, parameter    :: GX  = G*DTX   ! Coefficient (1/s)
REAL, parameter    :: GY  = G*DTY   ! Coefficient (1/s)
REAL, parameter    :: FT  = F*DT    ! Coefficient (-)
REAL, parameter    :: RDT = RR*DT   ! Coefficient (m)
!
INTEGER, parameter :: NSAVE=3600/DT ! Saving data at NSAVE
INTEGER, parameter :: NPRINT=NSAVE  ! Printing data on
!                                     terminal at NPRINT
REAL               :: HNTM  ! Maximum integr. time (hrs)
```

```
REAL               :: DTIME ! Time in days
REAL               :: HTIME ! Time in hours
REAL               :: TIME  ! Time in seconds
REAL               :: H48   ! 48 hours measured in seconds
REAL               :: XM    ! x at middle of domain (m)
REAL               :: YM    ! y at middle of domain (m)
REAL               :: T0    ! Maximum wind strength (Pa)
REAL               :: X0    ! Maximum wind stress diveded
!                             by water density (m**2/s**2)
REAL               :: XDT   ! Coefficient = X0*DT (ms)
REAL               :: X0C   ! Initial x-position of center
!                             of moving cyclone (m)
REAL               :: Y0C   ! Initial y-position of center
!                             of moving cyclone (m)
REAL               :: U0    ! x-component translation speed
!                             of moving cyclone
REAL               :: V0    ! y-component translation speed
!                             of moving cyclone
!
INTEGER            :: NTIME  ! Time counter
INTEGER            :: NTIME2 ! Time step counter = NTIME+1
INTEGER            :: NTMAX  ! Maximum number of time steps
!
!---------------------- Specify auxiliary help constants ---
!-----------------------------------------------------------
!---------------------- NetCDF stuff -----------------------
! nmk:
!                    1. Variables needed for netCDF output:
INTEGER            :: statuso
INTEGER            :: ncido
INTEGER            :: dim_xo
INTEGER            :: dim_yo
INTEGER            :: dim_timeo
INTEGER            :: HVarIdo
INTEGER            :: UVarIdo
INTEGER            :: VVarIdo
INTEGER            :: TimeVarIdo
INTEGER            :: HTIME2
character(len=13)  :: outfile   ! # characters in filename
!
!                                           2. Make NC-file
outfile="closed_swe.nc"
statuso = nf90_create(trim(outfile),IOR(nf90_clobber,     &
     nf90_64BIT_OFFSET),ncido)
```

```
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
!                                        3. Define dimensions
statuso = nf90_def_dim(ncido,'x',JP,dim_xo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_def_dim(ncido,'y',KP,dim_yo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_def_dim(ncido,'time',                        &
     nf90_unlimited,dim_timeo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
!                                        4. Define variables
statuso = nf90_def_var(ncido,'h',nf90_float,                &
     (/dim_xo, dim_yo, dim_timeo/),HVarIdo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_def_var(ncido,'time',nf90_double,            &
     dim_timeo,TimeVarIdo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
statuso = nf90_def_var(ncido,'u',nf90_float,                &
     (/dim_xo, dim_yo, dim_timeo/),UVarIdo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
statuso = nf90_def_var(ncido,'v',nf90_float,                &
     (/dim_xo, dim_yo, dim_timeo/),VVarIdo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
!                                     5. Variable attributes
statuso = nf90_put_att(ncido,HVarIdo,'long_name',           &
     'sea level')
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_put_att(ncido,HVarIdo,'units','metres')
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_put_att(ncido,HVarIdo,'time','time')
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_put_att(ncido,HVarIdo,'field',               &
     'sea level, scalar, series')
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
statuso = nf90_put_att(ncido,TimeVarIdo,'long_name',        &
     'time since start')
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_put_att(ncido,TimeVarIdo,'field',            &
     'time, scalar, series')
```

```
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_put_att(ncido,nf90_global,'type',          &
       'Guinea program - output')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_enddef(ncido)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  !---------------------- End netCDF stuff -------------------
  !-----------------------------------------------------------
  !
  !-------------- Read in maximum time and benchmark case ---
  WRITE(6,*) ' '
25 WRITE(6,*) 'Specify maximum simulation time in hours: '
  READ(5,*) NTMAX
  WRITE(6,*) 'Specify wind case:'
  WRITE(6,*) '1: Uniform wind, 2: Bell-shaped,              &
              3: Moving cyclone'
  READ(5,*) IWIND
  IF((IWIND.EQ.1).OR.(IWIND.EQ.2)) THEN
   X0 = 0.1/RHO
  ELSE
   X0 = 3.0/RHO
  ENDIF
  !
  WRITE(6,*) 'Maximum simulation time is:', NTMAX, 'hrs'
  WRITE(6,*) 'Results are saved each:', NSAVE, 'hrs'
  WRITE(6,*) 'Results are written to terminal each:',       &
      NPRINT, 'hrs'
  !
  !------------ Compute auxiliary constants -----------------
  NTMAX=NTMAX*3600/DT  ! Convert max hrs into max time steps
  HNTM=NTM*DT/3600     ! Convert max # of time steps into hrs
  WRITE(6,*) NTMAX,NSAVE,NPRINT
  IF(NTMAX.GT.NTM) THEN
     WRITE(6,9005) NTMAX,HNTM
9005 FORMAT(/,1X,'NTMAX=',1X,I5,1X,                         &
     'Sorry, maximum simulation',                           &
     /,'time has to be less than',1X,F6.1,' hrs',/,         &
     'Please try again')
     GO TO 25
  ENDIF
  !
  WRITE(6,*) 'Maximum # of time steps for this run is:',    &
              NTMAX
```

```
!
!------------ Set all variables to zero everywhere --------
!                                    1. Independent variables
DO J=1,JJ
 X(J)   = 0.
 XKM(J) = 0.
ENDDO
DO K=1,KK
 Y(K)   = 0.
 YKM(K) = 0.
ENDDO
!                             2. All the dependent variables
DO IT=1,NT
 DO J=1,JJ
  DO K=1,KK
   E(J,K,IT)  = 0.
   U(J,K,IT)  = 0.
   V(J,K,IT)  = 0.
   XX(J,K,IT) = 0.
   YY(J,K,IT) = 0.
  ENDDO
 ENDDO
ENDDO
!                                3. Variables to be printed
DO J=1,JP
 DO K=1,KP
  EP(J,K) = 0.
  UP(J,K) = 0.
  VP(J,K) = 0.
 ENDDO
ENDDO
!                                       4. Equlibrium depth
DO J=1,JJ
  DO K=1,KK
    H(J,K)  = 0.
  ENDDO
ENDDO
!
!------------ Initialize everything ----------------------
!                                   1. Time counters and time
NTIME  = 0
NTIME2 = NTIME + 1
TIME   = 0.
HTIME  = 0.
```

```
DTIME  = 0.
ISAVE=0
ICUR=1
INEW=2
!                                      2. Independent variables
DO J=1,JJ
   X(J) = (J-1)*DX
   XKM(J) = 0.001*X(J)
ENDDO
DO K=1,KK
   Y(J) = (K-1)*DY
   YKM(J) = 0.001*Y(J)
ENDDO
!                                         3. Equilibrium depth
DO J=1,JJ
 DO K=1,KK
   H(J,K)  = H0
 ENDDO
ENDDO
!                                    4. All dependent variables
DO J=1,JJ
  DO K=1,KK
    E(J,K,ICUR)  = 0.
    U(J,K,ICUR)  = 0.
    V(J,K,ICUR)  = 0.
    E(J,K,INEW)  = 0.
    U(J,K,INEW)  = 0.
    V(J,K,INEW)  = 0.
    XX(J,K,ICUR) = 0.
    YY(J,K,ICUR) = 0.
  ENDDO
ENDDO
!
!------------ End initialize ------------------------------
!----------------------------------------------------------
!------------ Save the initial values ---------------------
!                      1. Time series mid domain along coast
K=2
J=JM
WRITE(52,8001) HTIME,XX(J,K,INEW),EP(J,K),UP(J,K),VP(J,K)
!                                      2. Write to netCDF file
!write(*,*) "Wrote NetCDF.. Time: ", HTIME
!HTIME2=int(HTIME)
!statuso = nf90_put_var(ncido,HVarIdo,EP,                &
```

```
  !     start=(/1,1,NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !statuso = nf90_put_var(ncido,UVarIdo,UP,                &
  !     start=(/1,1,NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !statuso = nf90_put_var(ncido,VVarIdo,VP,                &
  !     start=(/1,1,NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !statuso = nf90_put_var(ncido,TimeVarIdo,HTIME2,          &
  !     start=(/NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  !-------------------------------------------------------
  !------------ START TIME LOOP ----------------------------
  !                                   1. Forward the time step
30 NTIME=NTIME+1
  IF(NTIME.GT.NTMAX) GO TO 160    ! Time to quit?
  NTIME2 = NTIME + 1
  TIME   = TIME+DT
  HTIME  = TIME/3600.
  DTIME  = HTIME/24.
  !             2. Write a comforting message to the terminal
  WRITE(6,*) 'Working on time step no. ',NTIME,' or ',      &
      HTIME,' hrs '
  !------------ Update wind forcing ------------------------
  XDT = X0*DT
  IF(IWIND.EQ.1) THEN
  !                                      Case 1: Uniform wind
    DO J=1,JJ
      DO K=1,KK
        ARG1 = (2.*K - 3)/20.
        XX(J,K,INEW) = XDT*EXP(-ARG1)
        YY(J,K,INEW) = 0.
      ENDDO
    ENDDO
  ELSEIF(IWIND.EQ.2) THEN
  !                                   Case 2: Bell-shaped wind
    IF(HTIME.LT.48.) THEN
      DO J=1,JJ
        ARG2 = ((J-JM)/10.)**2
        ARG3 = EXP(-ARG2)
        DO K=1,KK
          ARG1 = (2.*K - 3)/20.
          XX(J,K,INEW) = XDT*EXP(-ARG1)*ARG3
```

```
        YY(J,K,INEW) = XDT*0.
      ENDDO
    ENDDO
  ELSE
    DO J=1,JJ
      DO K=1,KK
        XX(J,K,INEW) = XDT*0.
        YY(J,K,INEW) = XDT*0.
      ENDDO
    ENDDO
  ENDIF
ELSE
!                                    Case 3: Moving cyclone
  V0 = - U0C/sqrt(5.)
  U0 = - 2.*V0
  H48 = 48.*3600.
  XM  = (JM - 1)*DX
  YM  = (KM + 0.5)*DY
  X0C = XM - U0*H48
  Y0C = YM - V0*H48
  DO J=1,JJ
    ARG1 = (J-1)*DX - X0C - U0*TIME
    DO K=1,KK
      ARG2 = (K-1)*DY - Y0C - V0*TIME
      ARG3 = SQRT(ARG1**2 + ARG2**2)
      ARG4 = EXP(- 0.5*(1.- (ARG3/RC))**2)
      XX(J,K,INEW) = - XDT*(ARG2/RC)*ARG4
      YY(J,K,INEW) = XDT*(ARG1/RC)*ARG4
    ENDDO
  ENDDO
ENDIF
!
!-------------------------------------------------------------
!-------- Update north-south volume flux component (U) ----
!                    1. All points inside of the boundaries
!                                    J=2(1)JJ-2 and K=2(1)KK-1
DO J=2,JJ-2
 DO K=2,KK-1
  HU = 0.5*(H(J+1,K) + H(J,K))
  BU = HU/(HU + RDT)
  IF(K.EQ.2) THEN
   VC = V(J,K,ICUR) + V(J+1,K,ICUR)
   CU = 0.5*FT*VC
  ELSEIF(K.EQ.KK-1) THEN
```

```
    VC = V(J,K-1,ICUR) + V(J+1,K-1,ICUR)
    CU = 0.5*FT*VC
   ELSE
    VC = V(J,K,ICUR) + V(J+1,K,ICUR)                          &
                     + V(J+1,K-1,ICUR) + V(J,K-1,ICUR)
    CU = 0.25*FT*VC
   ENDIF
   PX = - GX*HU*(E(J+1,K,ICUR) - E(J,K,ICUR))
   U(J,K,INEW) = BU*(U(J,K,ICUR) + CU + PX + XX(J,K,INEW))
  ENDDO
 ENDDO
 !    2. Ensure boundaries to the north and south are closed
 DO K=1,KK
  U(1,   K,INEW) = 0.
  U(JJ-1,K,INEW) = 0.
 ENDDO
 !   3. Put zeros in points outside of the offshore boundary
 !                                                    K=1, K=KK
 DO J=1,JJ
  U(J,1,INEW) = 0.
  U(J,KK,INEW) = 0.
 ENDDO
 !                 and outside of the northern boundary J=JJ
 DO K=1,KK
  U(JJ,K,INEW) = 0.
 ENDDO
 !-- End update the north-south volume flux component (U) --
 !-------------------------------------------------------------
 !----- Update east-west volume flux component (V) ---------
 !                    1. All points inside of the boundaries
 !                                  J=2(1)JJ-1 and K=2(1)KK-2
 DO J=2,JJ-1
  DO K=2,KK-2
   HV = 0.5*(H(J,K+1) + H(J,K))
   BV = HV/(HV + RDT)
   IF(J.EQ.2) THEN
    UC = U(J,K,INEW) + U(J,K+1,INEW)
    CV = - 0.5*FT*UC
   ELSEIF(J.EQ.JJ-1) THEN
    UC = U(J-1,K,INEW) + U(J-1,K+1,INEW)
    CV = - 0.5*FT*UC
   ELSE
    UC = U(J,K,INEW) + U(J-1,K,INEW)                          &
                     + U(J-1,K+1,INEW) + U(J,K+1,INEW)
```

```
   CV = - 0.25*FT*UC
  ENDIF
  PY = - GY*HV*(E(J,K+1,ICUR) - E(J,K,ICUR))
  V(J,K,INEW) = BV*(V(J,K,ICUR) + CV + PY + YY(J,K,INEW))
 ENDDO
ENDDO
!           2. Ensure coast and offshore boundary is closed
!                                              K=1 and K=KK-1
DO J=1,JJ
  V(J,   1,INEW) = 0.
  V(J,KK-1,INEW) = 0.
ENDDO
!            3. Put zeros in points outside of the offshore
!                       and outside of the northern boundary
!                                                 J=JJ, K=KK
DO J=1,JJ
 V(J,KK,INEW) = 0.
ENDDO
DO K=1,KK
 V(JJ,K,INEW) = 0.
ENDDO
!--- End update the east-west volume flux component (V) ---
!-----------------------------------------------------------
!----- Update sea surface anomaly (E) ---------------------
!                     1. All points inside of the boundaries
!                                    J=2(1)JJ-1 and K=2(1)KK-1
DO J=2,JJ-1
 DO K=2,KK-1
  DU = DTX*(U(J,K,INEW) - U(J-1,K,INEW))
  DV = DTY*(V(J,K,INEW) - V(J,K-1,INEW))
  E(J,K,INEW) = E(J,K,ICUR) - DU - DV
 ENDDO
ENDDO
!          2. Put zeros in points outside of the boundaries
!                                                  J=1, J=JJ
DO K=1,KK
  E( 1,K,INEW) = 0.
  E(JJ,K,INEW) = 0.
ENDDO
DO J=2,JJ-1
  E(J, 1,INEW) = 0.
  E(J,KK,INEW) = 0.
ENDDO
!--- End update sea surface anomaly (E) -------------------
```

```
  !-------------------------------------------------------
  !--- Save results -------------------------------------
  !---------------------- Prepare fields to be printed ------
  !                   Convert transports to mean depth current
  !                       at E-points inside of the boundaries
  !                                    J=2(1)JJ-1, K=2(1)KK-1
  DO J=1,JP
    DO K=1,KP
      EP(J,K) = E(J+1,K+1,INEW)
      VP(J,K) = 0.5*(V(J+1,K+1,INEW)                          &
                        + V(J+1,K,INEW))/H(J+1,K)
      UP(J,K) = 0.5*(U(J+1,K+1,INEW)                          &
                        + U(J,K+1,INEW))/H(J+1,K)
    ENDDO
  ENDDO
  !                               2. Write to time series file
  K=2
  J=JM+1
  WRITE(52,8001) HTIME,XX(J,K,INEW),EP(J,K),UP(J,K),VP(J,K)
8001 FORMAT(1X,F10.4,1X,10E12.4)
  !
  !                    3. Save the results for this time step?
  IF(MOD(NTIME,NSAVE).EQ.0) THEN
     write(*,*) "Wrote NetCDF.. Time: ", HTIME
     HTIME2=int(HTIME)
     statuso = nf90_put_var(ncido,HVarIdo,EP,                 &
          start=(/1,1,HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
     statuso = nf90_put_var(ncido,UVarIdo,UP,                 &
          start=(/1,1,HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
     statuso = nf90_put_var(ncido,VVarIdo,VP,                 &
          start=(/1,1,HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
     statuso = nf90_put_var(ncido,TimeVarIdo,HTIME2,          &
          start=(/HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
  ENDIF
  !                           3. Write some results to terminal
  IF(MOD(NTIME,NPRINT).EQ.0) THEN
     K=1
     DO J=1,JP
        WRITE(6,*) HTIME,XKM(J+1),XX(J+1,K+1,INEW),EP(J,K),       &
                   UP(J,K),VP(J,K)
```

```
     ENDDO
  ENDIF
  !-------------------------------------------------------------
  !                                      Go to next time level
  !                               remember to swap indexes first
  ISAVE = ICUR
  ICUR  = INEW
  INEW  = ISAVE
  GO TO 30
  !---------------------- END TIME LOOP ----------------------
  !-------------------------------------------------------------
160 CONTINUE
  !C
  statuso = nf90_sync(ncido)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_close(ncido)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  !-------------------------------------------------------------
  !   If run succesful write this comforting note to terminal
  STOP 'AFTER GOOD RUN'
  !
END PROGRAM SHALLW
```

Appendix B: FORTRAN program, open boundaries

```
PROGRAM SHALLW
  !-------------------------------------------------------------
  !  PURPOSE:
  !
  !   - To produce results for the benchmark cases
  !     with open boundaries to the south, north and
  !     offshore , that is, Cases 1b, 1c, 2b, 2c, 3b,
  !     and 3c
  !
  !   - Closed boundaries everywhere and no shelf.
  !     Case 1b: A positive uniform alongshore wind
  !              stress of strength 0.1 Pa decaying
  !              offshore.
  !     Case 1c: As 1b, but including a shelf
  !     Case 2b: Bell-shaped wind along the coast,
  !              of maximum strenght 0.1 Pa decaying
  !              offshore
  !     Case 2c: As 2b, but including a shelf
```

```
!     Case 3b: Moving cyclone of maximum strength
!              3.0 Pa moving diagonally across the
!              basin from southwest to northeast.
!              Distance from center to max strength
!              is 200 km. The cyclone is in the
!              middle of the domain at 48hrs.
!     Case 3c: As 3b, but including a shelf
!
!-------------------------------------------------------
!  METHOD:
!
!   - The linear, shallow water equations to solve are
!
!       dU/dt = + F*V - G*H*dE/dx + (TSX - TBX)/RHO,
!       dV/dt = - F*U - G*H*dE/dy + (TSY - TBY)/RHO.
!       dE/dt = - dU/dx - dV/dy,
!
!     Here x is positive northward, y is positive westward
!     and t is time. U(x,y,t) and V(x,y,t) are the volume
!     transport components along the x,y axes, while
!     E(x,y,t) is the sea surface deviation away from its
!     equilibrium position, F is the Coriliolis parameter,
!     G is the gravitational acceleration, H(x,y) is the
!     equilibrium depth, TSX and TSY are the two horizontal
!     components of the wind stress, and TBX and TBY the
!     similar components of the bottom stress. RHO is
!     a reference water density.
!
!   - The boundaries to the north and south are open.
!     Here we use the FRS as our boundary condition with
!     the external solution computed using the one-
!     dimensional version of above equations (d/dx = 0).
!     The boundary condition at the coast is V=0, and at
!     the open offshore boundary we let E=0
!
!   - Integration is performed using finite difference
!     methods:
!      1. Forward-Backward scheme (Martinsen et al., 1979)
!      2. Wind and bottom stress treated backwards
!
!-------------------------------------------------------
!   FINITE DIFFERENCE EQUATIONS:
!
!   - The grid is staggered, corresponding to lattice C of
```

```
!     Mesinger and Arakawa (two dimensions) to avoid
!     overspecifying the variables at the boundaries. The
!     finite difference equations are:
!
!      U(J,K,INEW) = BU*(U(J,K,ICUR) + CU + PX + XX),
!      V(J,K,INEW) = BV*(V(J,K,ICUR) + CV + PY + XY),
!      E(J,K,INEW) = E(J,K,ICUR) - DU - DV,
!
!     where
!
!      DU = (DT/DX)*(U(J,K,INEW) - U(J-1,K,INEW)),
!      DV = (DT/DY)*(V(J,K,INEW) - V(J,K-1,INEW)),
!      VC = V(J,K,ICUR) + V(J+1,K,ICUR)
!                       + V(J+1,K-1,ICUR) + V(J,K-1,ICUR),
!      UC = U(J,K,INEW) + U(J-1,K,INEW)
!                       + U(J-1,K+1,INEW) + U(J,K+1,INEW),
!      CU = F*DT*VC,
!      CV = - F*DT*UC,
!      PX = - G*(DT/DX)*HU*(E(J+1,K,ICUR) - E(J,K,ICUR)),
!      PY = - G*(DT/DY)*HV*(E(J,K+1,ICUR) - E(J,K,ICUR)),
!      XX = (TBX(J,K,INEW)/RHO),
!      YY = (TBY(J,K,INEW)/RHO),
!      HU = 0.5*(H(J+1,K) + H(J,K)),
!      HV = 0.5*(H(J,K+1) + H(J,K)),
!      BU = HU/(HU + RDT),
!      BV = HV/(HV + RDT).
!
!     The various physical constants are
!
!      DT          : Time step in seconds
!      DX          : Increment along x (m)
!      DY = DX     : Increment along y (m)
!      F           : Coriolis parameter in 1/s
!      RHO         : Reference water density in kg/m**3
!      G           : Gravitational acceleration in m/s**2
!      RR          : Friction coefficent in m/s
!      T0          : Amplitude of wind stress (N/m**2)
!      X0 = T0/RHO : Maximum wind stress divided by
!                    water density (m**2/s**2)
!      H0          : Equilibrium depth on shelf (m)
!      H1          : Equilibrium depth off shelf (m)
!
!     In addition we define
!      J                  : Counter J=1(1)JJ, X(1)=0.
```

```
!      K                   : Counter K=1(1)KK, Y(1)=0.
!      X(J)   = (J-1)*DX   : Position U-points along x (m)
!      Y(K)   = (K-1)*DY   : Position V-points along y (m)
!      XKM(J) = 0.001*X(J) : As X(J), but in km
!      YKM(K) = 0.001*Y(K) : As Y(K), but in km
!
!     Finite difference equation for external solution
!     The 1-D version in a staggered grid is:
!
!      UES(K,INEW) = BU*(UES(K,ICUR) + CU + XX),
!      VES(K,INEW) = BV*(VES(K,ICUR) + CV + PY + XY),
!      EES(K,INEW) = EES(K,ICUR) - DV,
!
!      UEN(K,INEW) = BU*(UEN(K,ICUR) + CU + XX),
!      VEN(K,INEW) = BV*(VEN(K,ICUR) + CV + PY + XY),
!      EEN(K,INEW) = EEN(K,ICUR) - DV,
!
!     where
!
!
!-------------------------------------------------------------
!  COMMUNICATION:
!
!   - Synoptic values of sea surface elevation, velocity
!     components are saved on the specified netCDF file
!     every hour
!
!   - A string of values are printed to terminal
!     while running the program
!
!   - Time series are saved on file unit 52.
!
!
!-------------------------------------------------------------
!  STATUS:
!
!  - CODED BY       : LARS PETTER ROED, met.no, Oslo
!  - DATE           : 27 February 2012
!
!  REVISION HISTORY :
!  - DATE           : 14 March, 2012
!  - Description    : Merged all open boundary cases
!                     into one program
!-------------------------------------------------------------
```

```
!
use netcdf
!
implicit none
!
!-------------------------------------------------------------
!---------------------- Specify dimensions -----------------
INTEGER, PARAMETER :: JJ  = 52      ! # x cells
INTEGER, PARAMETER :: KK  = 27      ! # y cells
INTEGER, parameter :: JF  = 1       ! # x cells in FRS zone
INTEGER, parameter :: KF  = 1       ! # x cells in FRS zone
INTEGER, parameter :: JP  = JJ-2*JF ! # x cells to save
INTEGER, parameter :: KP  = KK-2*KF ! # y cells to save
INTEGER, PARAMETER :: NT  = 2       ! # of time levels
INTEGER, PARAMETER :: NTM = 3840    ! Max # of time steps
INTEGER, parameter :: KS  = 5       ! Location shelf break
INTEGER, parameter :: KM  = 13      ! Counter mid basin y
INTEGER, parameter :: JM  = 26      ! Counter mid basin x
!
!---------------------- Dimensionalize everything ----------
!                                    1. Dependent variables
REAL E(JJ,KK,NT)    ! Sea surface anomaly (m)
REAL U(JJ,KK,NT)    ! Volume transport along x-axis (m/s)
REAL V(JJ,KK,NT)    ! Volume transport along y-axis (m/s)
REAL XX(JJ,KK,NT)   ! Wind stress along x-axis divided
!                     by water density (m**2/s**2)
REAL YY(JJ,KK,NT)   ! Wind stress along y-axis divided
!                     by water density (m**2/s**2)
!
!                  2. External solution dependent variables
REAL EES(KK,NT)     ! External sea surface elevation along
!                     southern boundary (m)
REAL UES(KK,NT)     ! Northward component of volume
!                     transport at southern boundary (m/s)
REAL VES(KK,NT)     ! Westward component of volume
!                     transport at southern boundary (m/s)
REAL EEN(KK,NT)     ! External sea surface elevation along
!                     northern boundary (m)
REAL UEN(KK,NT)     ! Northward component of volume
!                     transport at northern boundary (m/s)
REAL VEN(KK,NT)     ! Westward component of volume
!                     transport at northern boundary (m/s)
!
!                                  3. Time series variables
```

```
REAL EP(JP,KP)      ! Saved depth anomaly (m)
REAL UP(JP,KP)      ! Saved velocity along x-axis at
!                     E-point (m/s)
REAL VP(JP,KP)      ! Saved velocity along y-axis at
!                     E-point (m/s)
!
!                              4. Other variables and flags
REAL H(JJ,KK)       ! Equilibrium depth (m)
REAL X(JJ),XKM(JJ)  ! Distance along x (m and km)
REAL Y(KK),YKM(KK)  ! Distance along y (m and km
REAL GAM(JJ)        ! Relaxation parameter in the FRS zones
INTEGER J,K         ! Counter indexes along x and y
INTEGER IT          ! Index for time level #
INTEGER ICUR        ! Counter for current time level (n)
INTEGER INEW        ! Counter for new time level (n+1)
INTEGER ISAVE       ! Auxiliary time level counter
INTEGER IWIND       ! Flag to specify wind forcing:
!                     1: Uniform wind
!                     2: Bell-shaped wind
!                     3: Moving cylone
INTEGER ISHELF      ! Flag to specify shelf/no shelf:
!                     1: Shelf
!                     2: No shelf (flat bottom)
!
!-------------------------------------------------------------
!--------------------- Specify physical constants ---------
INTEGER, parameter  :: DT  = 90       ! Time step (s).
!                 !!!! NOTE: 3600 has to be multiplyer of DT
REAL, parameter     :: DX  = 20000.   ! Increment in x (m)
REAL, parameter     :: DY  = 20000.   ! Increment in y (m)
REAL, parameter     :: F   = 0.00012  ! Coriolis parameter
REAL, parameter     :: RHO = 1025.    ! Reference water
!                                       density (kg/m**3)
REAL, parameter     :: G   = 9.81     ! Grav. acc. (m/s**2)
REAL, parameter     :: RR  = 0.0024   ! Drag coeff. (m/s)
REAL, parameter     :: H0  = 50.      ! Equilibrium depth
!                                       flat bottom depth
!                                       and shelf (m)
REAL, parameter     :: H1  = 2500.    ! Equilibrium depth
!                                       off-shelf (m)
REAL, parameter     :: U0C = 15.      ! Translation speed
!                                       of moving cyclone
!                                       center (m/s)
REAL, parameter     :: RC  = 200000.  ! Distance to max
```

```
!                                        wind stress from
!                                        cyclone center(m)
!
!-------------------------------------------------------------
!---------------------- Specify auxiliary variables --------
REAL DU,DV,VC,UC,CU,CV,PX,PY,PXI,HU,HV,BU,BV,ARG1
REAL ARG2,ARG3,ARG4
!
!-------------------------------------------------------------
!---------------------- Specify auxiliary constants --------
REAL, parameter    :: DTX = DT/DX   ! Coefficient (s/m)
REAL, parameter    :: DTY = DT/DY   ! Coefficient (s/m)
REAL, parameter    :: GX  = G*DTX   ! Coefficient (1/s)
REAL, parameter    :: GY  = G*DTY   ! Coefficient (1/s)
REAL, parameter    :: FT  = F*DT    ! Coefficient (-)
REAL, parameter    :: RDT = RR*DT   ! Coefficient (m)
!
INTEGER, parameter :: NSAVE=3600/DT ! Saving every NSAVE
INTEGER, parameter :: NPRINT=NSAVE  ! Printing every NPRINT
!
REAL               :: HNTM   ! Maximum sim. time in hrs
REAL               :: DTIME  ! Time in days
REAL               :: HTIME  ! Time in hrs
REAL               :: TIME   ! Time in sec
REAL               :: H48    ! 48 hours measured in sec
!
REAL               :: XM     ! x in middle of domain (m)
REAL               :: YM     ! y in middle of domain (m)
!
REAL               :: T0     ! Maximum wind strength (Pa)
REAL               :: X0     ! Maximum wind stress diveded
!                              by water density (m**2/s**2)
REAL               :: XDT    ! Coefficient = X0*DT (ms)
REAL               :: X0C    ! Initial x-position of center
!                              of moving cyclone (m)
REAL               :: Y0C    ! Initial y-position of center
!                              of moving cyclone (m)
REAL               :: U0     ! x-component of translation
!                              speed of moving cyclone
REAL               :: V0     ! y-component of translation
!                              speed of moving cyclone
!
INTEGER            :: NTIME  ! Time counter
INTEGER            :: NTIME2 ! Time step counter = NTIME+1
```

```
INTEGER            :: NTMAX  ! Maximum number of time steps
!
!-------------------------------------------------------------
!--------------------- NetCDF stuff ----------------------
! nmk:
!-------------------------------------------------------------
!                        Variables needed for netCDF output:
INTEGER           :: statuso
INTEGER           :: ncido
INTEGER           :: dim_xo
INTEGER           :: dim_yo
INTEGER           :: dim_timeo
INTEGER           :: HVarIdo
INTEGER           :: UVarIdo
INTEGER           :: VVarIdo
INTEGER           :: TimeVarIdo
INTEGER           :: HTIME2
character(len=11) :: outfile   ! # of characters in outfile
!-------------------------------------------------------------
!                                              1. Make NC-file
outfile="open_swe.nc"
statuso = nf90_create(trim(outfile),IOR(nf90_clobber,     &
     nf90_64BIT_OFFSET),ncido)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!-------------------------------------------------------------
!                                         2. Define dimensions
statuso = nf90_def_dim(ncido,'x',JP,dim_xo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_def_dim(ncido,'y',KP,dim_yo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_def_dim(ncido,'time',                      &
     nf90_unlimited,dim_timeo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!-------------------------------------------------------------
!                                          3. Define variables
statuso = nf90_def_var(ncido,'h',nf90_float,              &
     (/dim_xo, dim_yo, dim_timeo/),HVarIdo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
statuso = nf90_def_var(ncido,'time',nf90_double,dim_timeo,&
     TimeVarIdo)
if(statuso /= nf90_NoErr) call handle_err(statuso)
!
statuso = nf90_def_var(ncido,'u',nf90_float,              &
     (/dim_xo, dim_yo, dim_timeo/),UVarIdo)
```

```
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  statuso = nf90_def_var(ncido,'v',nf90_float,              &
       (/dim_xo, dim_yo, dim_timeo/),VVarIdo)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !-------------------------------------------------------
  !                                    4. Variable attributes
  statuso = nf90_put_att(ncido,HVarIdo,'long_name',         &
       'sea level')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_put_att(ncido,HVarIdo,'units','metres')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_put_att(ncido,HVarIdo,'time','time')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_put_att(ncido,HVarIdo,'field',             &
       'sea level, scalar, series')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  statuso = nf90_put_att(ncido,TimeVarIdo,'long_name',      &
       'time since start')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_put_att(ncido,TimeVarIdo,                  &
       'field','time, scalar, series')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_put_att(ncido,nf90_global,                 &
       'type','Guinea program - output')
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_enddef(ncido)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  !---------------------- End netCDF stuff -------------------
  !-------------------------------------------------------
  !
  !-------------- Read in maximum time and benchmark case ---
  WRITE(6,*) ' '
25 WRITE(6,*) 'Specify maximum simulation time in hours: '
  READ(5,*) NTMAX
  WRITE(6,*) 'Specify wind case:'
  WRITE(6,*) '1: Uniform wind, 2: Bell-shaped,              &
              3: Moving cyclone'
  READ(5,*) IWIND
  IF((IWIND.EQ.1).OR.(IWIND.EQ.2)) THEN
   X0 = 0.1/RHO  ! Max wind stress set to 0.1 Pa
  ELSE
```

```
   X0 = 3.0/RHO  ! Max wind stress set to 3.0 Pa
  ENDIF
  WRITE(6,*) 'Specify shelf/no shelf:'
  WRITE(6,*) '1: Shelf, 2: Flat bottom'
  READ(5,*) ISHELF
  !
  WRITE(6,*) 'Maximum simulation time is          :',       &
              NTMAX, 'hrs'
  WRITE(6,*) 'Results are saved each              :',       &
              NSAVE, 'hrs'
  WRITE(6,*) 'Results are written to terminal each:',       &
              NPRINT, 'hrs'
  !
  !
  !---------------------- Compute auxiliary constants --------
  NTMAX=NTMAX*3600/DT  ! Convert max hrs into max time steps
  HNTM=NTM*DT/3600     ! Convert max # of time steps into hrs
  WRITE(6,*) NTMAX,NSAVE,NPRINT
  IF(NTMAX.GT.NTM) THEN
     WRITE(6,9005) NTMAX,HNTM
9005 FORMAT(/,1X,'NTMAX=',1X,I5,1X,                         &
     'Sorry, maximum simulation',/,                         &
     'time has to be less than',1X,F6.1,' hrs',/,           &
     'Please try again')
     GO TO 25
  ENDIF
  !
  WRITE(6,*) 'Maximum # of time steps for this run is:',    &
              NTMAX
  !
  !-------------------------------------------------------------
  !------- Set all variables to zero everywhere -------------
  !                                  1. Independent variables
  DO J=1,JJ
    X(J)   = 0.
    XKM(J) = 0.
  ENDDO
  DO K=1,KK
    Y(K)   = 0.
    YKM(K) = 0.
  ENDDO
  !                             2. All the dependent variables
  DO IT=1,NT
    DO J=1,JJ
```

```
    DO K=1,KK
      E(J,K,IT)  = 0.
      U(J,K,IT)  = 0.
      V(J,K,IT)  = 0.
      XX(J,K,IT) = 0.
      YY(J,K,IT) = 0.
    ENDDO
  ENDDO
ENDDO
!                                  3. Variables to be printed
DO J=1,JP
  DO K=1,KP
    EP(J,K) = 0.
    UP(J,K) = 0.
    VP(J,K) = 0.
    H(J,K)  = 0.
  ENDDO
ENDDO
!                              4. External solution variables
DO IT=1,NT
  DO K=1,KK
    EES(K,IT) = 0.
    UES(K,IT) = 0.
    VES(K,IT) = 0.
    EEN(K,IT) = 0.
    UEN(K,IT) = 0.
    VEN(K,IT) = 0.
  ENDDO
ENDDO
!                                       5. Equlibrium depth
DO J=1,JJ
  DO K=1,KK
    H(J,K)  = 0.
  ENDDO
ENDDO
!                                     6. Relaxation parameter
DO J=1,JJ
  GAM(J)  = 0.
ENDDO
!
!-------------------------------------------------------------
!---------------------- Initialize everything --------------
!                                    1. Time counters and time
NTIME  = 0
```

```
NTIME2 = NTIME + 1
TIME   = 0.
HTIME  = 0.
DTIME  = 0.
ISAVE=0
ICUR=1
INEW=2
!                                    2. Independent variables
DO J=1,JJ
   X(J) = (J-1)*DX
   XKM(J) = 0.001*X(J)
ENDDO
DO K=1,KK
   Y(J) = (K-1)*DY
   YKM(J) = 0.001*Y(J)
ENDDO
!                                       3. Equilibrium depth
IF(ISHELF.EQ.2) THEN  ! Flat bottom, no shelf
  DO J=1,JJ
    DO K=1,KK
      H(J,K)  = H0
    ENDDO
  ENDDO
ELSE                  ! Shelf
  DO J=1,JJ
    DO K=1,KK
      IF(K.LE.KS) THEN
        H(J,K) = H0
      ELSE
        H(J,K) = H1
      ENDIF
    ENDDO
  ENDDO
ENDIF
!                                     4. Dependent variables
DO J=1,JJ
  DO K=1,KK
    E(J,K,ICUR)  = 0.
    U(J,K,ICUR)  = 0.
    V(J,K,ICUR)  = 0.
    XX(J,K,ICUR) = 0.
    YY(J,K,ICUR) = 0.
!
    E(J,K,INEW)  = 0.
```

```
    U(J,K,INEW)  = 0.
    V(J,K,INEW)  = 0.
    XX(J,K,INEW) = 0.
    YY(J,K,INEW) = 0.
  ENDDO
ENDDO
!                              5. External solution variables
DO K=1,KK
  EES(K,ICUR) = 0.
  UES(K,ICUR) = 0.
  VES(K,ICUR) = 0.
  EES(K,INEW) = 0.
  UES(K,INEW) = 0.
  VES(K,INEW) = 0.
!
  EEN(K,ICUR) = 0.
  UEN(K,ICUR) = 0.
  VEN(K,ICUR) = 0.
  EEN(K,INEW) = 0.
  UEN(K,INEW) = 0.
  VEN(K,INEW) = 0.
ENDDO
!                                     6. Relaxation parameter
GAM(1)    = 1.0
GAM(2)    = 0.69
GAM(3)    = 0.44
GAM(4)    = 0.25
GAM(5)    = 0.11
GAM(6)    = 0.03
GAM(JJ-5) = GAM(6)
GAM(JJ-4) = GAM(5)
GAM(JJ-3) = GAM(4)
GAM(JJ-2) = GAM(3)
GAM(JJ-1) = GAM(2)
GAM(JJ)   = GAM(1)
!
!-------------------------------------------------------------
!--------------------- Save the initial values ------------
!                                                1. Time series
K=2
J=JM
WRITE(52,8001) HTIME,XX(J,K,INEW),EP(J,K),UP(J,K),VP(J,K)
!                                     2. Write to netCDF file
!write(*,*) "Wrote NetCDF.. Time: ", HTIME
```

```
  !HTIME2=int(HTIME)
  !statuso = nf90_put_var(ncido,HVarIdo,EP,                  &
  !     start=(/1,1,NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !statuso = nf90_put_var(ncido,UVarIdo,UP,                  &
  !     start=(/1,1,NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !statuso = nf90_put_var(ncido,VVarIdo,VP,                  &
  !     start=(/1,1,NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !statuso = nf90_put_var(ncido,TimeVarIdo,HTIME2,           &
  !     start=(/NTIME2/))
  !if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  !---------------------- End initialize everything ----------
  !-----------------------------------------------------------
  !
  !-----------------------------------------------------------
  !---------------------- Time loop --------------------------
  !                                      Forward the time step
30 NTIME=NTIME+1
  IF(NTIME.GT.NTMAX) GO TO 160    ! Time to quit?
  !                                 If not continue
  NTIME2 = NTIME + 1
  TIME=TIME+DT
  HTIME=TIME/3600.
  DTIME=HTIME/24.
  !                Write a comforting message to the terminal
  WRITE(6,*) 'WORKING ON TIME STEP NO. ',                   &
              NTIME,' or ',HTIME,' hrs '
  !
  !-----------------------------------------------------------
  !------------------------ Update wind forcing -------------
  XDT = X0*DT
  IF(IWIND.EQ.1) THEN
  !                                       Case 1: Uniform wind
    DO J=1,JJ
      DO K=1,KK
        ARG1 = (2.*K - 3)/20.
        XX(J,K,INEW) = XDT*EXP(-ARG1)
        YY(J,K,INEW) = 0.
      ENDDO
    ENDDO
  ELSEIF(IWIND.EQ.2) THEN
```

```
!                                        Case 2: Bell-shaped wind
  IF(HTIME.LT.48.) THEN
    DO J=1,JJ
      ARG2 = ((J-JM)/10.)**2
      ARG3 = EXP(-ARG2)
      DO K=1,KK
        ARG1 = (2.*K - 3)/20.
        XX(J,K,INEW) = XDT*EXP(-ARG1)*ARG3
        YY(J,K,INEW) = XDT*0.
      ENDDO
    ENDDO
  ELSE
    DO J=1,JJ
      DO K=1,KK
        XX(J,K,INEW) = XDT*0.
        YY(J,K,INEW) = XDT*0.
      ENDDO
    ENDDO
  ENDIF
ELSE
!                                         Case 3: Moving cyclone
  V0 = - U0C/sqrt(5.)
  U0 = - 2.*V0
  H48 = 48.*3600.
  XM  = (JM - 1)*DX
  YM  = (KM + 0.5)*DY
  X0C = XM - U0*H48
  Y0C = YM - V0*H48
  DO J=1,JJ
    ARG1 = (J-1)*DX - X0C - U0*TIME
    DO K=1,KK
      ARG2 = (K-1)*DY - Y0C - V0*TIME
      ARG3 = SQRT(ARG1**2 + ARG2**2)
      ARG4 = EXP(- 0.5*(1.- (ARG3/RC))**2)
      XX(J,K,INEW) = - XDT*(ARG2/RC)*ARG4
      YY(J,K,INEW) = XDT*(ARG1/RC)*ARG4
    ENDDO
  ENDDO
ENDIF
!  Distribute wind forcing onto appropriate U- and V-points
DO J=1,JJ
  DO K=2,KK
    XX(J,K,INEW) = 0.5*(XX(J,K,INEW) + XX(J,K-1,INEW))
  ENDDO
```

```
ENDDO
DO J=2,JJ
  DO K=1,KK
    YY(J,K,INEW) = 0.5*(YY(J,K,INEW) + YY(J-1,K,INEW))
  ENDDO
ENDDO
!-------------------------------------------------------------
!-------- Update the north-south  volume flux component (U)
!             1. All points inside of the boundaries except
!                                      along the coast (K=2)
!                                  J=2(1)JJ-2 and K=2(1)KK-1
DO J=2,JJ-2
  DO K=2,KK-1
    HU = 0.5*(H(J+1,K) + H(J,K))
    BU = HU/(HU + RDT)
    IF(K.EQ.2) THEN
      VC = V(J,K,ICUR) + V(J+1,K,ICUR)
      CU = 0.5*FT*VC
    ELSE
      VC = V(J,K  ,ICUR) + V(J+1,K  ,ICUR)                &
         + V(J,K-1,ICUR) + V(J+1,K-1,ICUR)
      CU = 0.25*FT*VC
    ENDIF
    PX = - GX*HU*(E(J+1,K,ICUR) - E(J,K,ICUR))
    U(J,K,INEW) = BU*(U(J,K,ICUR) + CU + PX + XX(J,K,INEW))
  ENDDO
ENDDO
!          2. Update external solution at southern boundary
!                                             J=1, K=2(1)KK-1
J=1
DO K=2,KK-1
  HU = 0.5*(H(J+1,K) + H(J,K))
  BU = HU/(HU + RDT)
  IF(K.EQ.2) THEN
    VC = VES(K,ICUR)
    CU = FT*VC
  ELSE
    VC = VES(K,ICUR) + VES(K-1,ICUR)
    CU = 0.5*FT*VC
  ENDIF
  PX = 0.
  UES(K,INEW) = BU*(UES(K,ICUR) + CU + PX + XX(J,K,INEW))
ENDDO
UES(1,INEW) = 0.
```

```
UES(KK,INEW) = 0.
!          3. Update external solution at northern boundary
!                                          J=JJ-1, K=2(1)KK-1
J=JJ-1
DO K=2,KK-1
  HU = 0.5*(H(J-1,K) + H(J,K))
  BU = HU/(HU + RDT)
  IF(K.EQ.2) THEN
    VC = VEN(K,ICUR)
    CU = FT*VC
  ELSE
    VC = VEN(K,ICUR) + VEN(K-1,ICUR)
    CU = 0.5*FT*VC
  ENDIF
  PX = 0.
  UEN(K,INEW) = BU*(UEN(K,ICUR) + CU + PX + XX(J,K,INEW))
ENDDO
UEN(1,INEW) = 0.
UEN(KK,INEW) = 0.
!        4. Put zeros in U-points outside of the boundaries
!                    J=1(1)JJ, K=1, K=KK; and K=1(1)KK, J=JJ
DO J=1,JJ
  U(J,1,INEW) = 0.
  U(J,KK,INEW) = 0.
ENDDO
DO K=1,KK
  U(JJ,K,INEW) = 0.
ENDDO
!                             5. Relax solution in FRS zones
DO J=1,JJ-1
  DO K=2,KK-1
    IF(J.LE.JM) THEN
      U(J,K,INEW) = (1.-GAM(J))*U(J,K,INEW)                &
                             + GAM(J)*UES(K,INEW)
    ELSE
      U(J,K,INEW) = (1.-GAM(J+1))*U(J,K,INEW)              &
                             + GAM(J+1)*UEN(K,INEW)
    ENDIF
  ENDDO
ENDDO
!----- End update the north-south volume flux component (U)
!-------------------------------------------------------------
!----------- Update the east-west volume flux component (V)
!                    1. All points inside of the boundaries
```

```
!                                    J=2(1)JJ-1 and K=2(1)KK-2
DO J=2,JJ-1
  DO K=2,KK-1
    HV = 0.5*(H(J,K+1) + H(J,K))
    BV = HV/(HV + RDT)
    IF(K.EQ.KK-1) THEN
      UC = U(J,K,INEW) + U(J-1,K,INEW)
      CV = - 0.5*FT*UC
    ELSE
      UC = U(J,K  ,INEW) + U(J-1,  K,INEW)                &
         + U(J,K+1,INEW) + U(J-1,K+1,INEW)
      CV = - 0.25*FT*UC
    ENDIF
    PY = - GY*HV*(E(J,K+1,ICUR) - E(J,K,ICUR))
    V(J,K,INEW) = BV*(V(J,K,ICUR) + CV + PY + YY(J,K,INEW))
  ENDDO
ENDDO
!                 2. External solution at southern boundary
!                                          J=1, K=2(1)KK-1
J=1
DO K=2,KK-1
  HV = 0.5*(H(J,K+1) + H(J,K))
  BV = HV/(HV + RDT)
  IF(K.EQ.KK-1) THEN
    UC = UES(K,INEW)
    CV = - FT*UC
  ELSE
    UC = UES(K,INEW) + UES(K+1,INEW)
    CV = - 0.5*FT*UC
  ENDIF
  PY = - GY*HV*(EES(K+1,ICUR) - EES(K,ICUR))
  VES(K,INEW) = BV*(VES(K,ICUR) + CV + PY + YY(J,K,INEW))
ENDDO
VES(1,INEW) = 0.  ! Boundary condition at coast
VES(KK,INEW) = 0. ! Never used, but ensures that it is zero
!                 3. External solution at northern boundary
!                                          J=JJ, K=2(1)KK-1
J=JJ
DO K=2,KK-1
  HV = 0.5*(H(J,K+1) + H(J,K))
  BV = HV/(HV + RDT)
  IF(K.EQ.KK-1) THEN
    UC = UEN(K,INEW)
    CV = - FT*UC
```

```
  ELSE
    UC = UEN(K,INEW) + UEN(K+1,INEW)
    CV = - 0.5*FT*UC
  ENDIF
  PY = - GY*HV*(EEN(K+1,ICUR) - EEN(K,ICUR))
  VEN(K,INEW) = BV*(VEN(K,ICUR) + CV + PY + YY(J,K,INEW))
ENDDO
VEN(1,INEW) = 0.  ! Boundary condition at coast
VEN(KK,INEW) = 0. ! Never used, but ensures that it is zero
!
!    4. Implement costal b.c., set to zero outside offshore
!                                                  K=1, K=KK
DO J=1,JJ
  V(J,1,INEW) = 0.
  V(J,KK,INEW) = 0.
ENDDO
!                              5. Relax solution in FRS zones
DO J=1,JJ
  DO K=2,KK-1
    IF(J.LE.JM) THEN
      V(J,K,INEW) = (1.-GAM(J))*V(J,K,INEW)               &
                              + GAM(J)*VES(K,INEW)
    ELSE
      V(J,K,INEW) = (1.-GAM(J))*V(J,K,INEW)               &
                              + GAM(J)*VEN(K,INEW)
    ENDIF
  ENDDO
ENDDO
!------- End update the east-west volume flux component (V)
!-------------------------------------------------------------
!--------------------------- Update sea surface anomaly (E)
!                     1. All points inside of the boundaries
!                                     J=2(1)JJ-1 and K=2(1)KK-1
DO J=2,JJ-1
  DO K=2,KK-1
    DU = DTX*(U(J,K,INEW) - U(J-1,K,INEW))
    DV = DTY*(V(J,K,INEW) - V(J,K-1,INEW))
    E(J,K,INEW) = E(J,K,ICUR) - DU - DV
  ENDDO
ENDDO
!           2. Outside of the coastal and offshore boundary
!                                     J=2(1)JJ-1, K= 1 and K=KK
DO J=1,JJ
  E(J,KK,INEW) = - E(J,KK-1,INEW) ! Offshore b.c.
```

```
  E(J,1 ,INEW) = 0. ! Outside of domain
ENDDO
!
!          3. Update external solution at southern boundary
!                                          J=1, K=2(1)KK-1
J=1
DO K=2,KK-1
  DU = 0.
  DV = DTY*(VES(K,INEW) - VES(K-1,INEW))
  EES(K,INEW) = EES(K,ICUR) - DU - DV
ENDDO
EES(1,INEW) = 0.
EES(KK,INEW) = - EES(KK-1,INEW)
!
!          4. Update external solution at northern boundary
!                                          J=JJ, K=2(1)KK-1
J=JJ
DO K=2,KK-1
  DU = 0.
  DV = DTY*(VEN(K,INEW) - VEN(K-1,INEW))
  EEN(K,INEW) = EEN(K,ICUR) - DU - DV
ENDDO
EEN(1,INEW) = 0.
EEN(KK,INEW) = - EEN(KK-1,INEW) ! Offshore b.c.
!
!                             5. Relax solution in FRS zones
DO J=1,JJ-1
  DO K=2,KK-1
    IF(J.LE.JM) THEN
      E(J,K,INEW) = (1.-GAM(J))*E(J,K,INEW)               &
                             + GAM(J)*EES(K,INEW)
    ELSE
      E(J,K,INEW) = (1.-GAM(J))*E(J,K,INEW)               &
                             + GAM(J)*EEN(K,INEW)
    ENDIF
  ENDDO
ENDDO
!-------------------------------------------------------------
!---------------------- Prepare fields to be printed ------
!                 Convert transports to mean depth currents
!            at E-points inside of FRS zones and boundaries
!                                    J=7(1)JJ-7, K=2(1)KK-1
DO J=1,JP
  DO K=1,KP
```

```
      EP(J,K) = E(J+JF,K+KF,INEW)
      VP(J,K) = 0.5*(V(J+JF,K+KF  ,INEW)                    &
                   + V(J+JF,K-1+KF,INEW))/H(J+JF,K+KF)
      UP(J,K) = 0.5*(U(J+JF  ,K+KF,INEW)                    &
                   + U(J-1+JF,K+KF,INEW))/H(J+JF,K+KF)
    ENDDO
  ENDDO
  !-------------------------------------------------------------
  !                          Write results to time series file
  K=2
  J=JM+6
     WRITE(52,8001) HTIME,XX(J,K,INEW),EP(J,K),             &
                                       UP(J,K),VP(J,K)
8001 FORMAT(1X,F10.4,1X,10E12.4)
  !
  !-------------------------------------------------------------
  !                       Save the results for this time step?
  IF(MOD(NTIME,NSAVE).EQ.0) THEN
     write(*,*) "Wrote NetCDF.. Time: ", HTIME
     HTIME2=int(HTIME)
     statuso = nf90_put_var(ncido,HVarIdo,EP,               &
          start=(/1,1,HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
     statuso = nf90_put_var(ncido,UVarIdo,UP,               &
          start=(/1,1,HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
     statuso = nf90_put_var(ncido,VVarIdo,VP,               &
          start=(/1,1,HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
     statuso = nf90_put_var(ncido,TimeVarIdo,HTIME2,        &
          start=(/HTIME2/))
     if(statuso /= nf90_NoErr) call handle_err(statuso)
  ENDIF
  !-------------------------------------------------------------
  !                            Write some results to terminal?
  IF(MOD(NTIME,NPRINT).EQ.0) THEN
     K=2
     DO J=2,JJ-1
        WRITE(6,*) HTIME,XKM(J),XX(J,K,INEW),EP(J,K),       &
                                             UP(J,K),VP(J,K)
     ENDDO
  ENDIF
  !-------------------------------------------------------------
  ! Go to next time level, but remember to swap indexes first
```

```
  ISAVE = ICUR
  ICUR  = INEW
  INEW  = ISAVE
  GO TO 30
  !--------------------- End time loop ----------------------
  !----------------------------------------------------------
160 CONTINUE
  !C
  statuso = nf90_sync(ncido)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  statuso = nf90_close(ncido)
  if(statuso /= nf90_NoErr) call handle_err(statuso)
  !
  !----------------------------------------------------------
  !   If run succesful write this comforting note to terminal
  STOP 'AFTER GOOD RUN'
  !
END PROGRAM SHALLW
```

References

Martinsen, E. A., and H. Engedahl (1987), Implementation and testing of a lateral boundary scheme as an open boundary condition in a barotropic ocean model., *Coast. Eng.*, *11*, 603–627.

Mesinger, F., and A. Arakawa (1976), Numerical methods used in atmospheric models, GARP Publication Series No. 17, 64 p., World Meteorological Organization, Geneva, Switzerland.

Røed, L. P. (2012), Documentation of simple ocean models for use in ensemble predictions. Part I: Equations, *met.no Report 03/2012*, Norwegian Meteorological Institute, Box 43 Blindern, N-0313 Oslo, Norway.

Røed, L. P., and C. K. Cooper (1987), A study of various open boundary conditions for wind-forced barotropic numerical ocean models, in *Three-dimensional models of marine and estuarine dynamics*, *Elsevier Oceanography Series*, vol. 45, edited by J. C. J. Nihoul and B. M. Jamart, pp. 305–3, Elsevier Science Publishers B.V.