CORI

PREVENTION AND MANAGEMENT OF SEA ORIGINATED RISKS TO THE COASTAL ZONE INTERREG III B ARCHIMED

PRIORITY AXIS: **3** - MEASURE: **3**

WORK PACKAGE 3 – Action 3

Development of an advection-diffusion numerical model for marine pollutant dispersion.

Responsible Partner: UNIVERSITY OF THE AEGEAN

Partners

ARISTOTLE UNIVERSITY OF THESSALONIKI

NORTH AEGEAN REGION

January 2009

1. INTRODUCTION

Hydrodynamic and pollutant transport modeling in coastal areas require a detailed knowledge of the transport processes that exist within the sea water body. In coastal areas the horizontal length scale is several order of magnitude greater than the vertical length scale. The induced circulation can be hydraulically or wind-driven. For the case of hydraulically-driven circulation, the current vertical distribution is almost uniform over the depth with sharp gradients existing only at the bottom. The current driven vertical velocity profile can be treated as if the whole water column is a boundary layer and therefore the Prandtl-von Karman logarithmic velocity profile applies. In this case, the two-dimensional horizontal depth-averaged model (2DH) can successfully simulate the depth-averaged current distribution. In the case of wind driven flows the vertical current distribution is countercurrent with the current in lower layers moving in opposite direction to wind. In order to incorporate the effects of the non-uniform velocity in the vertical plane, Koutitas (1988) developed the so-called quasi-three-dimensional (Q3D) wind driven circulation model. This model, basically, solves the depthaveraged Navier-Stokes equations assuming a parabolic velocity profile in the vertical, to be consistent with the wind driven velocity profiles. The quasi-three-dimensional model has the advantage of simplicity, requires short computational time and implicitly implements the vertical current distribution.

Advection-diffusion type equations underlie mathematical models of solute transport and water quality in coastal areas. Many numerical schemes have been proposed for solving these equations utilizing finite differences, finite elements, boundary elements and Lagrangian tracking, but with mixed success. There are two features, in particular, which any scheme should possess. Firstly, it should be accurate: unfortunately, the accuracy characteristics of most schemes are not well known and are rarely published in a detailed enough way to be of much use in practical modeling exercises. Secondly, it should be capable of application at Courant numbers in excess of one: this allows efficient computational compatibility with the numerical hydrodynamic models.

Here we develop a numerical model based on a higher order scheme. This scheme has the benefits of mass conservation, reasonably high accuracy and computational efficiency in comparison with many other higher-order-accurate schemes reported in the recent literature.

Following the introduction, the circulation model **WINDCIR** (WIND CIRculation) is presented, which describes the wind-driven velocity field. In paragraph 3 the pollutant transport model **PODIS** (POlutant DISperson) is described. FORTRAN codes and details of applications of the models are given in the Annex.

2. DESCRIPTION OF THE CIRCULATION MODEL WINDCIR

2.1. Two-dimensional horizontal depth-averaged circulation model (2DH)

Given the large horizontal dimensions relative to the vertical in relatively shallow water bodies, the vertical velocities and accelerations are small relative to the horizontal components. Therefore the vertical equation of motion may be replaced by the hydrostatic pressure approximation. The resulting depth-averaged Navier-Stokes equations become (Koutitas, 1988):

$$\frac{\partial \zeta}{\partial t} + \frac{\partial (Uh)}{\partial x} + \frac{\partial (Vh)}{\partial y} = 0$$

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} + g\frac{\partial \zeta}{\partial x} = \frac{1}{h}\frac{\partial}{\partial x}\left(\nu_h h \frac{\partial U}{\partial x}\right) + \frac{1}{h}\frac{\partial}{\partial y}\left(\nu_h h \frac{\partial U}{\partial y}\right) + fV + \frac{\tau_{sx}}{\rho h} - \frac{\tau_{bx}}{\rho h}$$

$$\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} + g\frac{\partial \zeta}{\partial y} = \frac{1}{h}\frac{\partial}{\partial x}\left(\nu_h h \frac{\partial V}{\partial x}\right) + \frac{1}{h}\frac{\partial}{\partial y}\left(\nu_h h \frac{\partial V}{\partial y}\right) - fU + \frac{\tau_{sy}}{\rho h} - \frac{\tau_{by}}{\rho h} \quad (1)$$

where ζ is the water surface elevation above the mean water level; d is the water depth; h is the total depth of water (i.e. h=d+ζ); U, V are the depth-averaged velocity components in x and y directions respectively; f is the Coriolis parameter; ν_h is the horizontal eddy viscosity coefficient, τ_{sx} and τ_{sy} are the shear stresses at the water surface in the x and y directions respectively, which represent the vertical boundary condition as follows:

$$\tau_{sx} = \rho k W_x \sqrt{W_x^2 + W_y^2}$$

$$\tau_{sy}=\rho k W_y \sqrt{W_x^2+W_y^2} \tag{2}$$

where k is the surface friction coefficient [kg/ m^3] typically of the order of 10^{-6} (here we assume k=0.000001÷0.000003); W_x and W_y are the wind speeds in x and y directions [m/s], respectively.

Similarly, the bed friction terms (τ_{bx}, τ_{by}) are expressed by quadratic forms as follows:

$$\tau_{bx}=\frac{1}{2}\rho f_c U\sqrt{U^2+V^2}$$

$$\tau_{by}=\frac{1}{2}\rho f_c V\sqrt{U^2+V^2} \tag{3}$$

where f_c is the bottom friction coefficient

$$f_c=\frac{2g}{c_c^2} \tag{4}$$

Where c_c is the Chezy coefficient (c_c =10 ~50 $m^{1/2}/s$)

The horizontal eddy viscosity coefficient is given by the well known Smagorinsky model (which is used for the representation of the damping by eddies smaller than the computational grid size):

$$\nu_h=\ell^2\left[\left(\frac{\partial U}{\partial x}\right)^2+\left(\frac{\partial V}{\partial y}\right)^2+\frac{1}{2}\left(\frac{\partial U}{\partial y}+\frac{\partial V}{\partial x}\right)^2\right]^{1/2} \tag{5}$$

Where ℓ is the mixing length , $\ell \approx 0.5\,\Delta x$ (Madsen et al., 1988)

2.2. Quasi-three dimensional circulation model (Q3D)

The advantage of this model is that it can provide the current distribution at any depth as well as the depth-averaged current structure. The equations of motion in the x and y directions and the mass continuity are simplified under the assumption that a) the water is incompressible and homogeneous, and b) the flow is quasi-hydrostatic. The second assumption is based on the fact that the horizontal dimension of the flow domain is several order of magnitude larger than the vertical dimension (depth). The assumption of nearly horizontal

flow is realistic and simplifies the model by excluding the vertical velocity component “w” from the main unknown functions and leads to a hydrostatic pressure distribution. In order to incorporate the effects of the non-uniform velocities in the vertical, especially in the wind-induced cases, a parabolic velocity profile in the vertical is assumed and the coefficients are determined from the boundary conditions at the surface, and the bottom. The velocity profile is defined as follows:

$$u(z)=\left(\frac{3}{4}\frac{\tau_{sx}h}{\rho\nu_\tau}-\frac{3}{2}U\right)\left[\left(\frac{z}{h}\right)^2-1\right]+\frac{\tau_{sx}h}{\rho\nu_\tau}\left(\frac{z}{h}+1\right)$$

$$v(z)=\left(\frac{3}{4}\frac{\tau_{sy}h}{\rho\nu_\tau}-\frac{3}{2}V\right)\left[\left(\frac{z}{h}\right)^2-1\right]+\frac{\tau_{sy}h}{\rho\nu_\tau}\left(\frac{z}{h}+1\right) \qquad (6)$$

where the z-axis has its origin coincides with the water surface; ν_τ is the vertical eddy viscosity coefficient. A constant eddy viscosity is assumed in order to be consistent with the parabolic velocity profile (Koutitas, 1988).

The eddy viscosity coefficient is given by Koutitas (1988):

$$v_T=\lambda h\left(\sqrt{\frac{\tau_s}{\rho}}+\sqrt{\frac{\tau_b}{\rho}}\right) \qquad (7)$$

where λ is a coefficient to characterise the intensity of the turbulence being of the order of O(λ)=0.1 (here a value λ=0.09 is assumed).

Based on the velocity component distributions given by Equations (6) the convective terms are evaluated. The two-dimensional model, improved with respect to the horizontal momentum dispersion and the bed friction for wind generated circulation, becomes (Koutitas, 1988),

$$\frac{\partial\zeta}{\partial t}+\frac{\partial(Uh)}{\partial x}+\frac{\partial(Vh)}{\partial y}=0$$

$$\frac{\partial \mathrm{U}}{\partial \mathrm{t}}+\left(1.2\,\mathrm{U}+\frac{\tau_{\mathrm{sx}}\,h}{40\,\rho\,\nu_{\tau}}\right)\frac{\partial \mathrm{U}}{\partial \mathrm{x}}+\left(1.2V+\frac{\tau_{\mathrm{sy}}\,h}{40\,\rho\,\nu_{\tau}}\right)\frac{\partial \mathrm{U}}{\partial \mathrm{y}}+\mathrm{g}\,\frac{\partial \zeta}{\partial \mathrm{x}}=$$

$$\frac{1}{\mathrm{h}}\frac{\partial}{\partial \mathrm{x}}\left(\nu_{\mathrm{h}}\mathrm{h}\,\frac{\partial \mathrm{U}}{\partial \mathrm{x}}\right)+\frac{1}{\mathrm{h}}\frac{\partial}{\partial \mathrm{y}}\left(\nu_{\mathrm{h}}\mathrm{h}\,\frac{\partial \mathrm{U}}{\partial \mathrm{y}}\right)+\mathrm{fV}+\frac{\tau_{\mathrm{sx}}}{\rho\mathrm{h}}-\frac{\tau_{\mathrm{bx}}}{\rho\mathrm{h}}-\left(3\lambda\,\frac{U}{h}\sqrt{\frac{\tau_{sx}}{\rho}}-0.5\,\frac{\tau_{sx}}{\rho h}\right)$$

$$\frac{\partial \mathrm{V}}{\partial \mathrm{t}}+\left(1.2\,\mathrm{U}+\frac{\tau_{\mathrm{sx}}\,h}{40\,\rho\,\nu_{\tau}}\right)\frac{\partial V}{\partial \mathrm{x}}+\left(1.2V+\frac{\tau_{\mathrm{sy}}\,h}{40\,\rho\,\nu_{\tau}}\right)\frac{\partial V}{\partial \mathrm{y}}+\mathrm{g}\,\frac{\partial \zeta}{\partial \mathrm{y}}=$$

$$\frac{1}{\mathrm{h}}\frac{\partial}{\partial \mathrm{x}}\left(\nu_{\mathrm{h}}\mathrm{h}\,\frac{\partial \mathrm{V}}{\partial \mathrm{x}}\right)+\frac{1}{\mathrm{h}}\frac{\partial}{\partial \mathrm{y}}\left(\nu_{\mathrm{h}}\mathrm{h}\,\frac{\partial \mathrm{V}}{\partial \mathrm{y}}\right)-\mathrm{fU}+\frac{\tau_{\mathrm{sy}}}{\rho\mathrm{h}}-\frac{\tau_{\mathrm{by}}}{\rho\mathrm{h}}-\left(3\lambda\,\frac{V}{h}\sqrt{\frac{\tau_{sy}}{\rho}}-0.5\,\frac{\tau_{sy}}{\rho h}\right)$$

(8)

The numerical solution is accomplished by a widely used simple and well documented explicit 2nd order finite difference scheme centered in space and forward in time on a staggered grid depicted in Figure 1. The scheme conserves mass and energy for non-breaking waves in a satisfactory manner. The discrete continuity equation is centered at the level points and the momentum equations at the flux points.

For the two horizontal dimensions differential equations (8) are approximated by the following finite difference equations according to the selected explicit scheme (Koutitas, 1988):

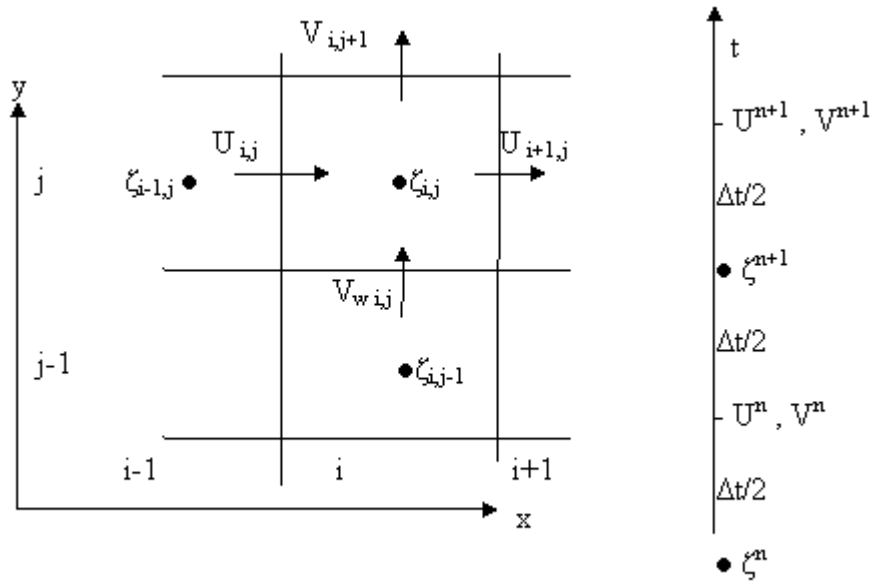

Figure 1. Spatial discretisation and indices used.

$$\frac{\zeta_i^{n+1}-\zeta_i^{n}}{\Delta t}+\frac{(U\,\overline{(d+\zeta)})_{i+1,j}^{n}-(U\,\overline{(d+\zeta)})_{i,j}^{n}}{\Delta x}+\frac{(V\,\overline{(d+\zeta)})_{i,j+1}^{n}-(V\,\overline{(d+\zeta)})_{i,j}^{n}}{\Delta y}=0$$

$$\frac{U_{i,j}^{n+1}-U_{i,j}^{n}}{\Delta t}+U_{i,j}^{n}\frac{U_{i+1,j}^{n}-U_{i-1,j}^{n}}{2\Delta x}+\bar{V}_{i,j}^{n}\frac{U_{i,j+1}^{n}-U_{i,j-1}^{n}}{2\Delta y}+g\frac{\zeta_{i,j}^{n+1}-\zeta_{i-1,j}^{n+1}}{\Delta x}$$
$$+\nu_h\frac{U_{i+1,j}^{n}-2U_{i,j}^{n}+U_{i-1,j}^{n}}{\Delta x^2}+\nu_h\frac{U_{i,j+1}^{n}-2U_{i,j}^{n}+U_{i,j-1}^{n}}{\Delta y^2}+\left[\frac{\tau_{sy}}{\rho h}-\frac{\tau_{by}}{\rho h}\right]^{n}$$

$$\frac{V_{i,j}^{n+1}-V_{i,j}^{n}}{\Delta t}+U_{i,j}^{n}\frac{U_{i+1,j}^{n}-U_{i-1,j}^{n}}{2\Delta x}+\bar{V}_{i,j}^{n}\frac{U_{i,j+1}^{n}-U_{i,j-1}^{n}}{2\Delta y}+g\frac{\zeta_{i,j}^{n+1}-\zeta_{i,j-1}^{n+1}}{\Delta y}=$$
$$\nu_h\frac{V_{i+1,j}^{n}-2V_{i,j}^{n}+V_{i-1,j}^{n}}{\Delta x^2}+\nu_h\frac{V_{i,j+1}^{n}-2V_{i,j}^{n}+V_{i,j-1}^{n}}{\Delta y^2}+\left[\frac{\tau_{sx}}{\rho h}-\frac{\tau_{bx}}{\rho h}\right]^{n}$$

(9)

where Δt is the time step, Δx and Δy are the x and y space steps, , i, j and n indices refer to the x, y and time dimensions, respectively, and

$$\overline{(d+\zeta)}^{n}_{i+1,j} = \left((d+\zeta)^{n}_{i,j} + (d+\zeta)^{n}_{i+1,j}\right)/2$$

$$U\overline{(d+\zeta)}^{n}_{i,j} = U\left((d+\zeta)^{n}_{i,j} + (d+\zeta)^{n}_{i-1,j}\right)/2$$

$$\overline{(d+\zeta)}^{n}_{i,j+1} = \left((d+\zeta)^{n}_{i,j} + (d+\zeta)^{n}_{i,j+1}\right)/2$$

$$V\overline{(d+\zeta)}^{n}_{i,j} = V\left((d+\zeta)^{n}_{i,j} + (d+\zeta)^{n}_{i,j-1}\right)/2$$

$$\overline{U}^{n}_{i,j} = \left(U^{n}_{i,j} + U^{n}_{i+1,j} + U^{n}_{i,j-1} + U^{n}_{i+1,j-1}\right)/4$$

$$\overline{V}^{n}_{i,j} = \left(V^{n}_{i,j} + V^{n}_{i,j+1} + V^{n}_{i-1,j} + V^{n}_{i-1,j+1}\right)/4$$

The total reflection boundary condition, i.e. U=0, V=0 $\partial\zeta/\partial n=0$ (where n is the unit vector normal to the boundary), is incorporated in the model in the regions where a solid boundary is present.

As calculation progresses, the kinetic energy per unit density contained in the basin is calculated at each time step by summing the square of the grid-point velocities U and V:

$$E^{n}_{\text{κιν}} = \sum_{i}\sum_{j}\left[\left(U^{n}_{ij}\right)^{2} + \left(V^{n}_{ij}\right)^{2}\right]\frac{h_{ij}}{2}\Delta x\,\Delta y \qquad (10)$$

Steady state is achieved when the ratio between the difference in kinetic energy between time steps falls between a certain accuracy (typically 1×10^{-4}).

3. NUMERICAL SIMULATION OF THE MARINE POLLUTANT ADVECTION – DIFFUSION - MODEL PODIS

The mathematical model consists of mass balance equation for the quality variables of interest , which have the following general form

$$\frac{\partial c}{\partial t} + U\frac{\partial c}{\partial x} + V\frac{\partial c}{\partial y} = D_x\frac{\partial^2 c}{\partial x^2} + D_y\frac{\partial^2 c}{\partial y^2} \qquad (11)$$

where D_x, D_y are the dispersion coefficients and c the pollutant concentration (in ppt, ppm, mg/lit, …).

The following Boundary Conditions are also applied:

A. Solid boundaries, where mass flow is supposed to be zero:

$$\frac{\partial c}{\partial n} = 0$$

(where n is the unit vector normal to the boundary)

B. Free radiation boundary:

$$\frac{\partial^2 c}{\partial n^2} = 0$$

where the pollutant is radiated to the open sea.

The dispersion coefficients D_x, D_y are given by:

$$D_x = 5.93\, u_{*b}\, h + \nu_h$$
$$D_y = 5.93\, v_{*b}\, h + \nu_h \qquad (12)$$

The first term accounts for the longitudinal dispersion due to the depth variation of the horizontal velocity (shear dispersion coefficient, Elder, 1959,

Chikwendu, 1986). The second term simulates the dispersion inside the computational grid (Smagorinsky approximation).

The numerical solution of equation (11) is based on an explicit numerical solution by Borthwick and Joynes (1992). A forward-in-time and centered in space high order finite difference scheme is adopted.

The first computational sweep calculates the following values of the concentration:

$$c_{ij}^{*}=c_{ij}^{n}-\left\{\begin{array}{l}\frac{\alpha}{12}\left[8\left(c_{i+1j}^{n}-c_{i-1j}^{n}\right)-\left(c_{i+2j}^{n}-c_{i-2j}^{n}\right)\right]\\+\frac{\alpha^{2}}{24}\left[30c_{ij}^{n}-16\left(c_{i+1j}^{n}+c_{i-1j}^{n}\right)+\left(c_{i+2j}^{n}+c_{i-2j}^{n}\right)\right]\\+\frac{\alpha^{3}}{12}\left[-2\left(c_{i+1j}^{n}-c_{i-1j}^{n}\right)+\left(c_{i+2j}^{n}-c_{i-2j}^{n}\right)\right]\end{array}\right\} \qquad (13)$$

and the second computational sweep:

$$c_{ij}^{+}=c_{ij}^{*}-\left\{\begin{array}{l}\frac{\alpha_{1}}{12}\left[8\left(c_{ij+1}^{*}-c_{ij-1}^{*}\right)-\left(c_{ij+2}^{*}-c_{ij-2}^{*}\right)\right]\\+\frac{\alpha_{1}^{2}}{24}\left[30c_{ij}^{*}-16\left(c_{ij+1}^{*}+c_{ij-1}^{*}\right)+\left(c_{ij+2}^{*}+c_{ij-2}^{*}\right)\right]\\+\frac{\alpha_{1}^{3}}{12}\left[-2\left(c_{ij+1}^{*}-c_{ij-1}^{*}\right)+\left(c_{ij+2}^{*}-c_{ij-2}^{*}\right)\right]\end{array}\right\} \qquad (14)$$

where n where the superscript n indicates the time level in the discretization, and $\alpha=U\,\Delta t/\Delta x$ and $\alpha_1=V\,\Delta t/\Delta y$.

The final value of the concentration c, at the time level n+1, is estimated from:

$$c_{ij}^{n+1}=c_{ij}^{n}+\Delta t\left[\frac{D_{x}\left(c_{i+1j}^{+}-2c_{ij}^{+}+c_{i-1j}^{+}\right)}{\Delta x^{2}}+\frac{D_{y}\left(c_{ij+1}^{+}-2c_{ij}^{+}+c_{ij-1}^{+}\right)}{\Delta y^{2}}\right]$$

The following restriction for the time and space steps are applied:

$$U_{max}\frac{\Delta t}{\Delta x}<1, \qquad D_{max}\frac{\Delta t}{\Delta x^2}<\frac{1}{2}$$

where U_{max} and D_{max} are the maximum values of the horizontal velocities and the dispersion coefficients, respectively.

4. APPLICATION

The above model is applied near Bay of Nies in Pagasitikos Golf (Figure 2). A sudden release of a pollutant with initial concentration c_o=100 ppm is applied. In figures 3-8 the advection and the diffusion of the pollutant presented under the effects of the North wind.

Figure 2. Bay of Nies coastal area (Pagasitikos Golf).

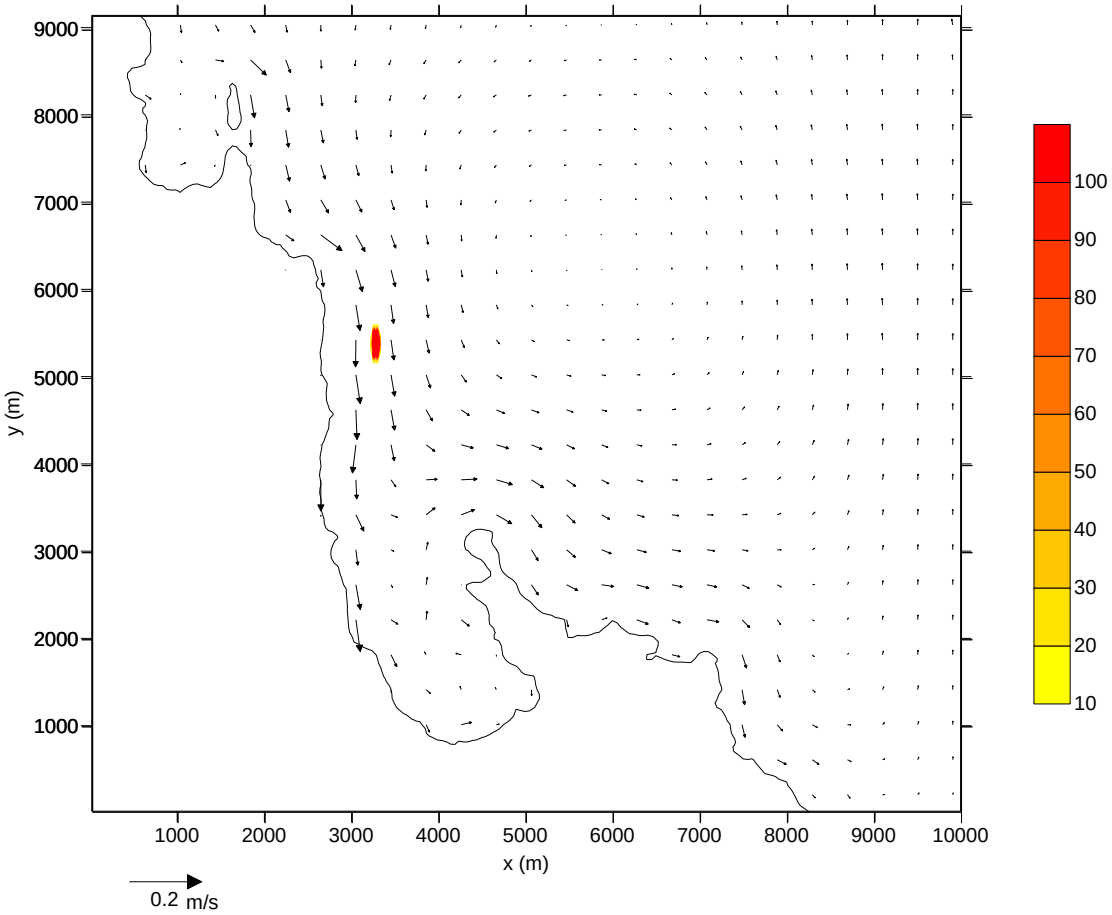

Figure 3. Pollutant advection and diffusion t=500 s.

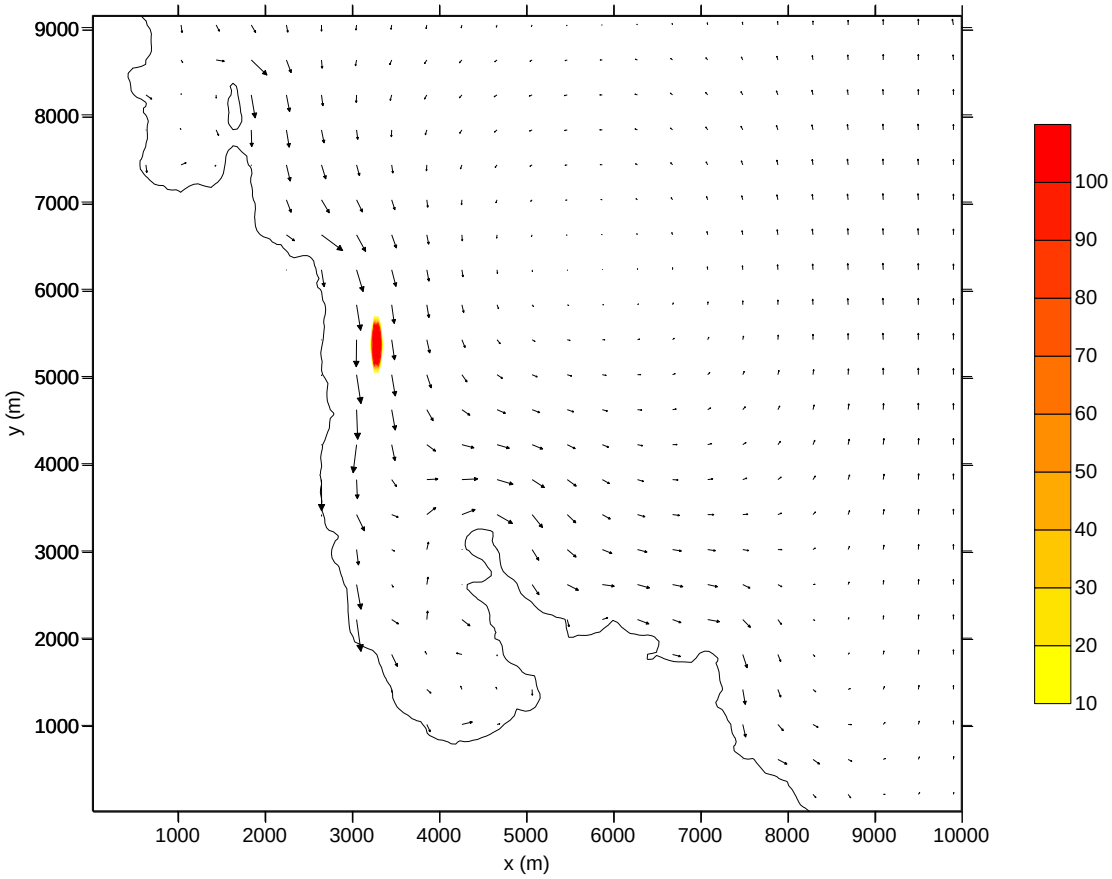

Figure 4. Pollutant advection and diffusion t=1000 s.

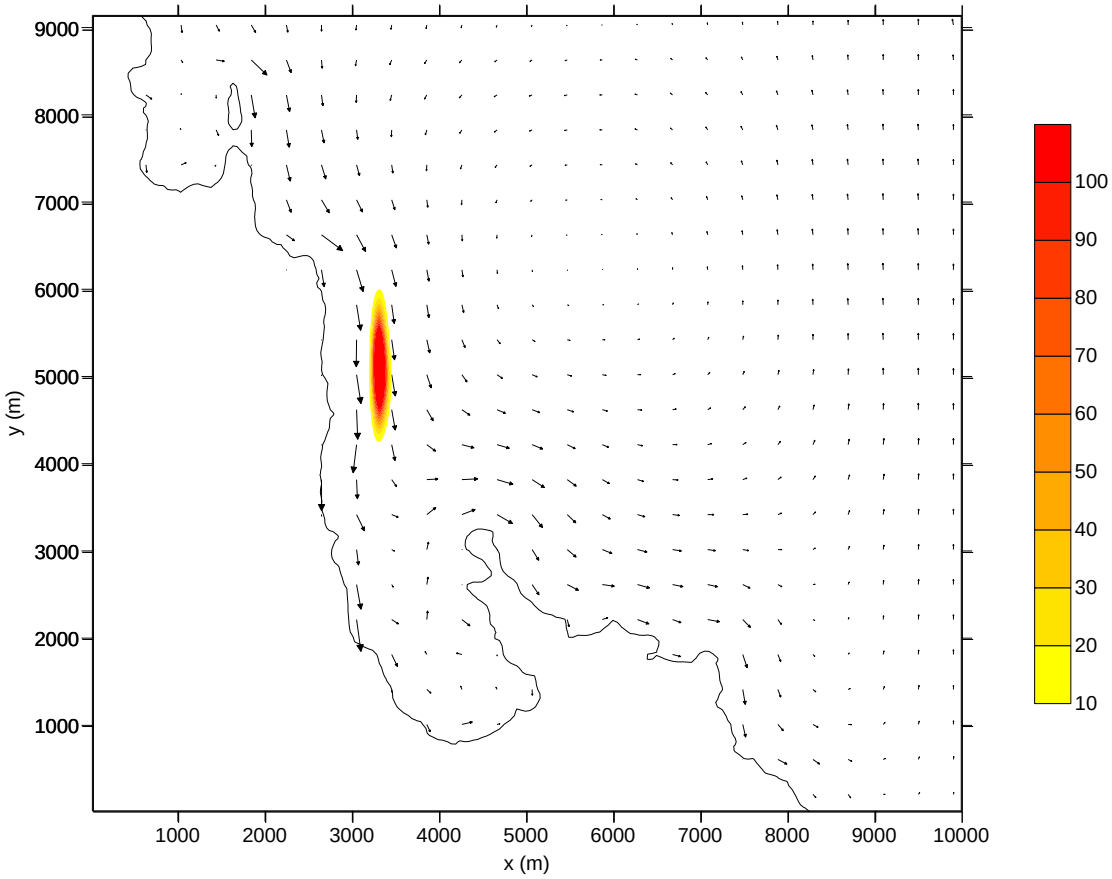

Figure 5. Pollutant advection and diffusion t=10000 s.

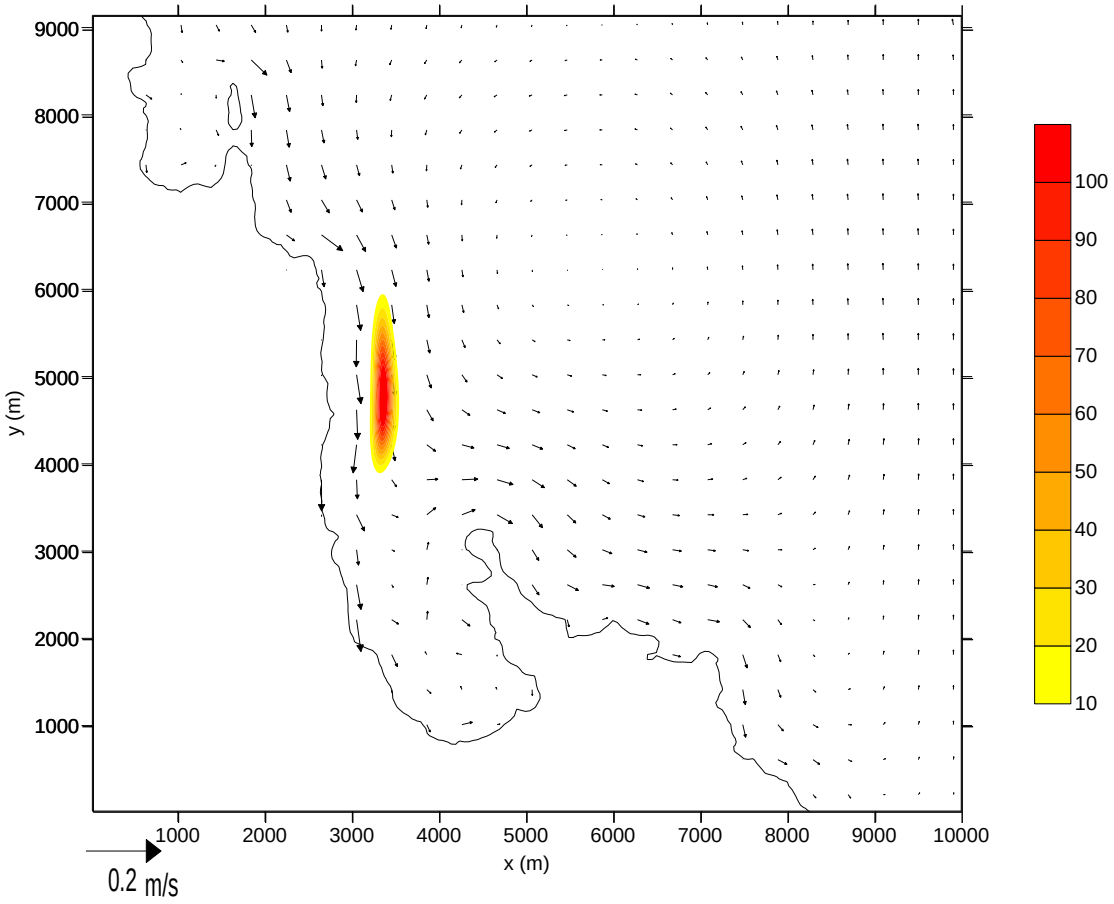

Figure 6. Pollutant advection and diffusion t=20000 s.

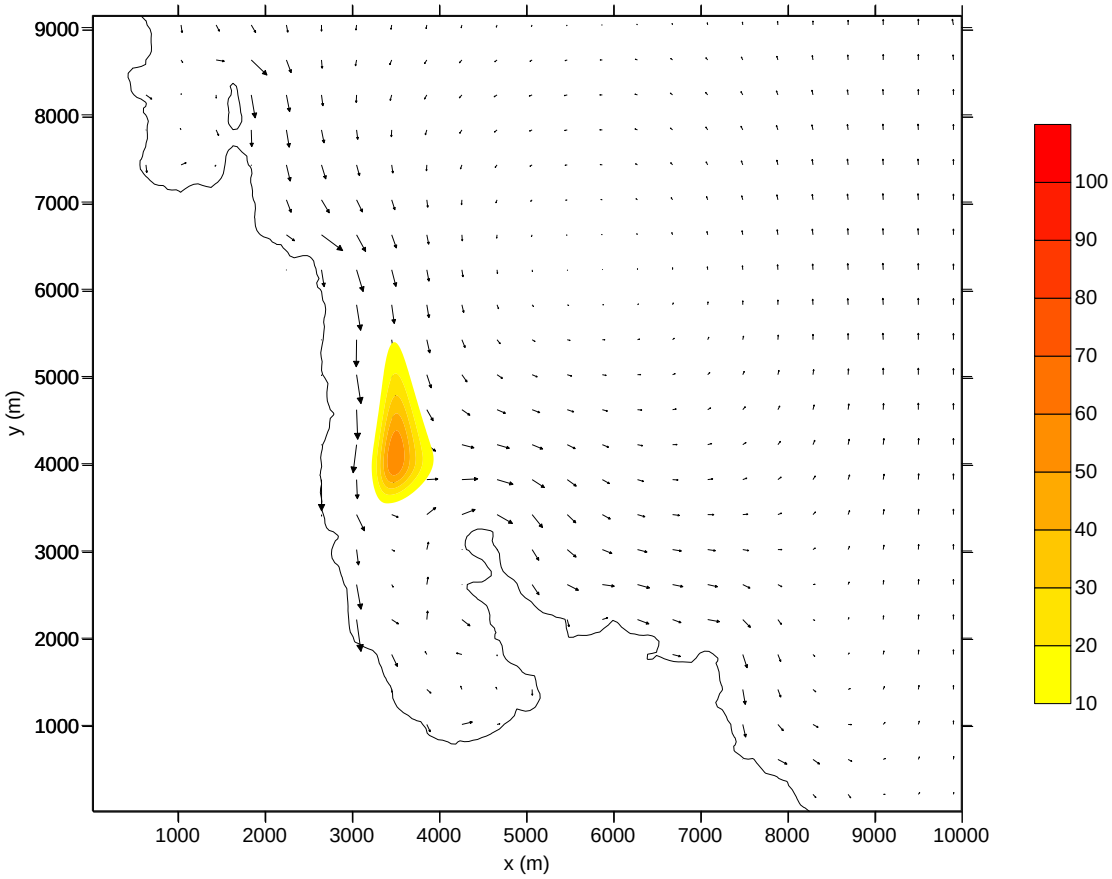

Figure 7. Pollutant advection and diffusion t=50000 s.

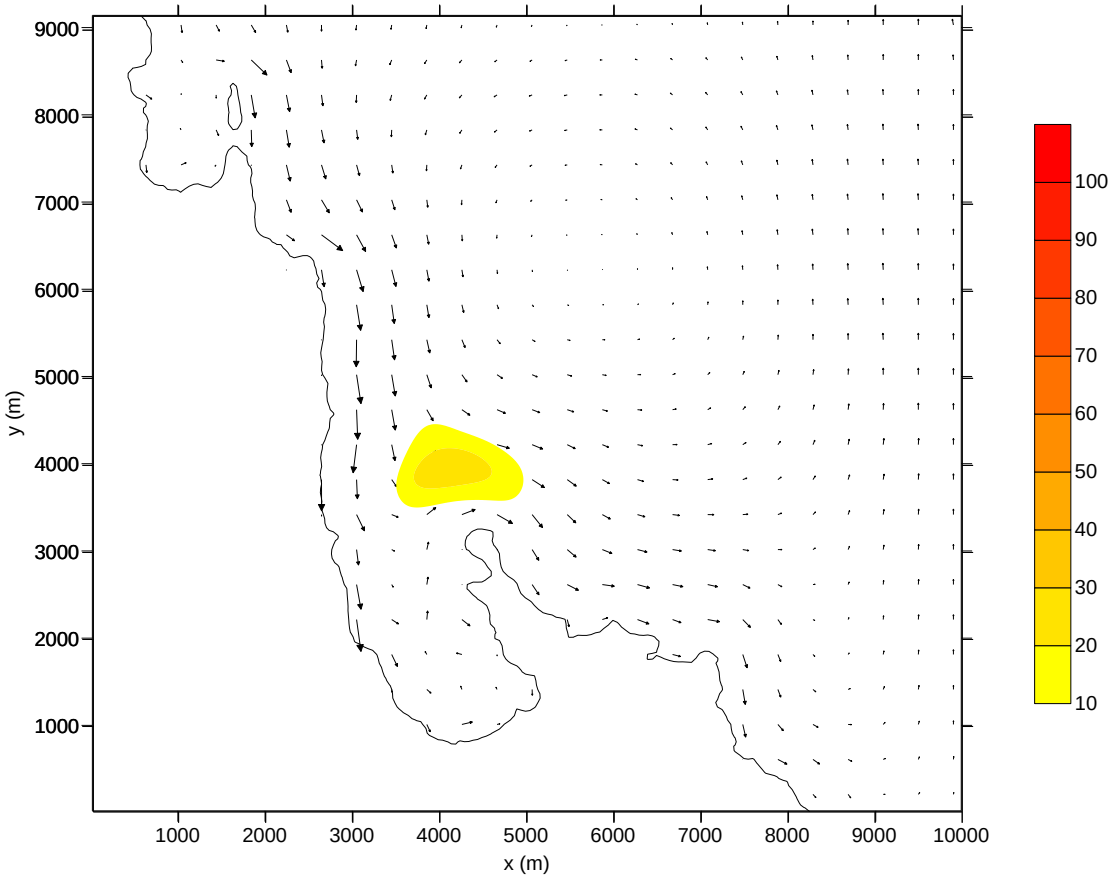

Figure 8. Pollutant advection and diffusion t=100000 s.

BIBLIOGRAPHY

Borthwick A.G.L. and D.S.A. Joynes (1992). Laboratory study of oil slick subjected to nearshore circulation, J. Environmental Eng., ASCE, Vol. 118, No 6, Nov/Dec.

Chikwendu S. C. (1986). Calculation of longitudinal shear dispersivity using an N-zone model as $N \rightarrow \infty$. Journal of Fluid Mechanics, vol. 167, pp. 19-30.

Elder J. W. (1959). The dispersion of marked fluid in turbulent shear flow. Journal of Fluid Mechanics vol. 5, pp. 554-560.

Karambas Th. V. and C. Koutitas (1998). On the dispersion process of wind generated flows in coastal waters. Protection and Restoration of the Environment IV, eds K. Katsifarakis et al., pp. 307-314.

Koutitas, C.G. (1988) Mathematical Models in Coastal Engineering, Pentech Press Limited, London.

Karambas Th. V., Koutitas Chr (2005). Mathematical models for the environmental design of coastal structures, 9th Conference on Environmental Science and Technology – 2005 Rhodes**,** Greece, 692-697.

Madsen P.A., Rugbjerg M. and Warren I.R. (1988). Subgrid Modelling in Depth Integrated Flows. ICCE-1988, pp. 505-511.

Pehlivanidis G. , Th. Karambas and M. Gousidou (2005), 'A Computational Model and An Operational Tool For the Optimal Forced Marina Flushing', J. of Marine Env. Engg., Vol. 8, 1, 21-34.

Tsanis I., U. Saied (2007). A Wind-Driven Hydrodynamic and Pollutant Transport Model, Global NEST Journal, Vol 9, no 2, 117-131.

Wu J., Tsanis i. k. (1995). A vertical/horizontal integration wind-induced circulation model (VHI3D): A method for including surface and bottom logarithmic profiles *Advances in Warer Resources,* Vol. 18, No. 2, 71-81.

ANNEX

Listing of programme WINDCIR

and

main parameters used in WINDCIR

DT	time step
DX	grid size
U,V	horizontal velocities in the x and y directions at n time level
UN,VN	horizontal velocities in the x and y directions at n+1 time level
z	sea surface elevation at n time level
zn	sea surface elevation at n+1 time level
H	water depth
tend	total time
im, jm	number of grid point in the x and y directions
WI	wind velocity
TH	wind direction in deg (TH=0 for North wind, TH=90 for East wind, …)
fcw	bottom friction factor
cs	surface friction coefficient
zob	bottom roughness
cz	Chezy coefficient
cf	Coriolis parameter

Input file: **depth.dat**
(includes the values of im and jm and the depth H(i,j) at each grid point)

```
c      PROGRAMME WINDCIR
c      Wind-induced circulation
c      Project CORI
c      PREVENTION AND MANAGEMENT OF SEA ORIGINATED RISKS TO THE
COASTAL ZONE
c      INTERREG III B ARCHIMED
c      PRIORITY AXIS: 3 - MEASURE: 3
c      WORK PACKAGE 3 - Action 3

      parameter (IMM=1000,JMM=1000)
      DIMENSION
H(imm,JMM),U(IMM,JMM),UN(IMM,JMM),V(IMM,JMM),VN(IMM,JMM)
     & ,Z(IMM,JMM),UB(IMM,JMM),VB(IMM,JMM),r(5500),VP2(5500),
     &  VP3(5500),ZP2(5500),ZP3(5500),zn(IMM,JMM),rz(5500),UP2(5500)
     & ,UP3(5500), edx(imm,jmm),edy(imm,jmm), edelx(imm,jmm)
     & ,edely(imm,jmm)



      data DT,DX/.25,20./
      data tend/25000/
c          North TH=0.
c     East  TH=90.
c     South TH=180.
          data WI,TH/20.,0./
          data fcw, cs, zob/0.001,0.000003,.04/
          data cz/40./  !Chezy coefficient



          dhr=-.5
          dhs=.25
          cf=0.0001
          aml=0.5*DX

          dif0=.1
```

```
      WX=-WI*COS((90.-TH)*3.1416/180.)
     WY=-WI*SIN((90.-TH)*3.1416/180.)
      write (*,*) WX,WY

      TSX=CS*WX*SQRT(ABS(WX)**2.+ABS(WY)**2.)
     TSY=CS*WY*SQRT(ABS(WX)**2.+ABS(WY)**2.)
     USS=SQRT(ABS(TSX))*SIGN(1.,TSX)
     VSS=SQRT(ABS(TSY))*SIGN(1.,TSY)
     USST=SQRT(ABS(USS)**2.+ABS(VSS)**2.)

      OPEN(12,FILE='depth.dat')
      read (12,*) im,jm

     DO 20 J=1,JM
     read (12,*) ij
     write (*,*) ij
     READ(12,*)(H(I,J),I=1,IM)
20    continue
     CLOSE(12)




     do i=1,im
     do j=1,jm
                r(j)=0.
      up2(j)=0
      up3(j)=0.
      u(i,j)=0.
      v(i,j)=0.
      ub(i,j)=0
      vb(i,j)=0
      zn(i,j)=0.
      un(i,j)=0.
      vn(i,j)=0.
      u(i,j)=0.
      v(i,j)=0.
```

```
        z(i,j)=0.
        zn(i,j)=0.
        un(i,j)=0.
        vn(i,j)=0.
        ub(i,j)=0.
        vb(i,j)=0.
    if (H(i,j).lt.dhs.and.H(i,j).gt.dhr) H(i,j)=dhs
    if (H(i,j).le.dhr) H(i,j)=-1.
    end do
    end do




        do j=1,jm
        write (*,*) i,H(im/2,j),H(im/2+10,j)
        end do

    N=0
    T=0
100 N=N+1
    T=T+DT


      DO 30 J=2,JM-2
        DO 30 I=2,IM-2

        if (H(i,j).gt..01.and.H(i+1,j).lt..01) then
        zn(i,j)=zn(i-1,j)
        goto 30
        else
        end if
        if (H(i,j).gt..01.and.H(i-1,j).lt..01) then
        zn(i,j)=zn(i+1,j)
        goto 30
        else
        end if
```

```
        if (H(i,j).gt..01.and.H(i,j+1).lt..01) then
        zn(i,j)=zn(i,j-1)
        goto 30
        else
        end if
        if (H(i,j).gt..01.and.H(i,j-1).lt..01) then
        zn(i,j)=zn(i,j+1)
        goto 30
        else
        end if

  HC=H(I,J)
  H2=H(I+1,J)
  H4=H(I,J+1)
  zn(I,J)=Z(I,J)-DT/DX*(H2*U(I+1,J)-HC*U(I,J)+H4*V(I,J+1)-HC*
 & V(I,J))
  GOTO 30

30 CONTINUE

         DO 150 J=2,JM-1
         DO 150 I=2,IM-1

        if (H(i,j).lt..01) then
        un(i,j)=0.0
        goto 150
        else
        end if

        if (H(i,j).gt..01.and.H(i+1,j).lt..01) then
        un(i,j)=0
        goto 150
        else
        end if
        if (H(i,j).gt..01.and.H(i-1,j).lt..01) then
        un(i,j)=0
        goto 150
```

```
      else
      end if
      if (H(i,j).gt..01.and.H(i,j+1).lt..01) then
      un(i,j)=un(i,j-1)
      goto 150
      else
      end if
      if (H(i,j).gt..01.and.H(i,j-1).lt..01) then
      un(i,j)=un(i,j+1)
      goto 150
      else
      end if



 HM=H(I,J)+zn(i,j)
 if (HM.lt..2) HM=.2
 VV=(V(I,J)+V(I-1,J)+V(I,J+1)+V(I-1,J+1))/4

      Ux=(U(I+1,J)-U(I-1,J))/2./DX
      Vy=(V(I,J+1)-V(I,J-1))/DX/2.
      Uy=(U(I,J+1)-U(I,J-1))/2./DX
      Vx=(V(I+1,J)-V(I-1,J))/DX/2.
      FRICT=.5*fcw*U(I,J)*SQRT(U(I,J)**2+VV**2)
 ustar=SQRT(ABS(FRICT))
 ELD=6.*HM*ustar+dif0
      eld=dif0

 DIF=ELD+(aml**2)*((Ux**2)+(Vy**2)+0.5*((Uy+Vx)**2))**.5
      edx(i,j)=dif

 tafs=cs*(Wx**2+Wy**2)
      tafsx=cs*Wx*sqrt(Wx**2+Wy**2)
      FRICT=.045*U(I,J)*SQRT(tafs)-0.5*tafs
      UUU=.9*U(I,J)
!     +.025*U(I+1,J)+.025*U(I-1,J)+.025*U(I,J+1)+.025*U(I,J-1)
 UN(I,J)=UUU-DT*(9.81*(zn(I,J)-zn(I-1,J))/DX-CF*VV
```

```
   & +FRICT/HM-DIF*(U(I+1,J)+U(I-1,J)+U(I,J+1)+U(I,J-1)
   & -4*U(I,J))/DX**2)+dt*tafsx/HM

150 CONTINUE

         DO 121 J=2,JM-1
         DO 121 I=2,IM-1

        if (H(i,j).lt..01) then
        vn(i,j)=0.0
        goto 121
        else
        end if

        if (H(i,j).gt..01.and.H(i,j+1).lt..01) then
        vn(i,j)=0.
        goto 121
        else
        end if
        if (H(i,j).gt..01.and.H(i,j-1).lt..01) then
        vn(i,j)=0.
        goto 121
        else
        end if
        if (H(i,j).gt..01.and.H(i+1,j).lt..01) then
        vn(i,j)=vn(i-1,j)
        goto 121
        else
        end if
        if (H(i,j).gt..01.and.H(i-1,j).lt..01) then
        vn(i,j)=vn(i+1,j)
        goto 121
        else
        end if

    HM=H(I,J)+zn(i,j)
    if (HM.lt..2) HM=.2
```

```
      UU=(U(I,J)+U(I+1,J)+U(I,J-1)+U(I+1,J-1))/4

       FRICT=.5*fcw*V(I,J)*SQRT(V(I,J)**2+UU**2)



           Ux=(U(I+1,J)-U(I-1,J))/2./DX
           Vy=(V(I,J+1)-V(I,J-1))/DX/2.
           Uy=(U(I,J+1)-U(I,J-1))/2./DX
           Vx=(V(I+1,J)-V(I-1,J))/DX/2.

      ustar=SQRT(ABS(FRICT))
      ELD=6.*HM*ustar+dif0
           eld=dif0

            DIF=ELD+(aml**2)*((Ux**2)+(Vy**2)+0.5*((Uy+Vx)**2))**.5
           edy(i,j)=dif

           tafs=cs*(Wx**2+Wy**2)
           tafsy=cs*Wy*sqrt(Wx**2+Wy**2)
           FRICT=.045*V(I,J)*SQRT(tafs)-0.5*tafs
           VVV=.9*V(I,J)
     !     +.025*V(I+1,J)+.025*V(I-1,J)+.025*V(I,J+1)+.025*V(I,J-1)
        VN(I,J)=VVV-DT*(9.81*(zn(I,J)-zn(I,J-1))/DX+CF*UU+FRICT/HM
     &  -DIF*(V(I,J+1)+V(I,J-1)+V(I+1,J)+V(I-1,J)-4*V(I,J))/
     & DX**2)+dt*tafsy/HM

   121 CONTINUE



c       Sponge layer Boundary Condition  s

c        West b. c.
        do 165 j=1,jm
        do 165 i=1,10
        ml1=i-10
```

```
      ml2=-10.0
      rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
      zn(i,j)=zn(i,j)/rm
      vn(i,j)=vn(i,j)/rm
165     un(i,j)=un(i,j)/rm
      ims1=im-10
      do 175 j=1,jm
      do 175 i=20,10,-1
      ml1=10-i
      ml2=-10.0
      rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
      zn(i,j)=zn(i,j)/rm
      vn(i,j)=vn(i,j)/rm
175     un(i,j)=un(i,j)/rm

c       South b.c.
c         do 665 i=1,im
c         do 665 j=1,20
c         ml1=j-20
c         ml2=-20.0
c         rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
c         zn(i,j)=zn(i,j)/rm
c         vn(i,j)=vn(i,j)/rm
c665     un(i,j)=un(i,j)/rm
c         do 675 i=1,im
c         do 675 j=40,20,-1
c         ml1=20-j
c         ml2=-20
c         rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
c         zn(i,j)=zn(i,j)/rm
c         vn(i,j)=vn(i,j)/rm
c675     un(i,j)=un(i,j)/rm


c       North b.c.
c      jms=jm-20
c      do 667 i=1,im
```

```
c       do 667 j=jms,jm-10
c       ml1=j-jm+10
c       ml2=-10.0
c       rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
c       zn(i,j)=zn(i,j)/rm
c       vn(i,j)=vn(i,j)/rm
c667     un(i,j)=un(i,j)/rm
c       jms1=jm-10
c       do 677 i=1,im
c       do 677 j=jm,jms1,-1
c       ml1=j-jm
c       ml2=-10
c       rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
c       zn(i,j)=zn(i,j)/rm
c       vn(i,j)=vn(i,j)/rm
c677     un(i,j)=un(i,j)/rm

cc       East b.c.
c       ims=im-20
c       do 668 j=1,jm
c       do 668 i=ims,im-10
c       ml1=i-im+10
c       ml2=-10.0
c       rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
c       zn(i,j)=zn(i,j)/rm
c       vn(i,j)=vn(i,j)/rm
c668     un(i,j)=un(i,j)/rm
c       ims1=im-10
c       do 679 j=1,jm
c       do 679 i=im,ims1,-1
c       ml1=i-im
c       ml2=-10
c       rm=exp((1.8**(ml1)-1.8**(ml2))*1.6)
c       zn(i,j)=zn(i,j)/rm
c       vn(i,j)=vn(i,j)/rm
c679     un(i,j)=un(i,j)/rm
```

```
      do i=2,IM-2
      VN(i,2)=((1.-r(i))*VP2(i)+2.*r(i)*V(i,3))/(1.+r(i))
      end do
      do i=2,IM-2
      UN(i,1)=UN(i,2)
      end do
      do i=2,IM-2
      den=VN(i,3)+VP3(i)-2.*V(i,4)
      if (ABS(den).lt..00001) goto 207
      r(i)=(VP3(i)-VN(i,3))/den
      if (r(i).gt..99) r(i)=.99
      if (r(i).lt.0.) r(i)=0.0
      goto 60
207   r(i)=0.0
60    end do

      do i=2,IM-2
      ZN(i,2)=((1.-r(i))*ZP2(i)+2.*r(i)*Z(i,3))/(1.+r(i))
      end do
      do i=2,IM-2
      den=ZN(i,3)+ZP3(i)-2.*Z(i,4)
      if (ABS(den).lt..00001) goto 203
      rz(i)=(ZP3(i)-ZN(i,3))/den
      if (rz(i).gt..99) rz(i)=.99
      if (rz(i).lt.0.) rz(i)=0.0
      goto 603
203   rz(i)=0.0
603   end do

      do i=1,IM
      VP2(i)=V(i,2)
      VP3(i)=V(i,3)
      ZP2(i)=Z(i,2)
      ZP3(i)=Z(i,3)
      end do
```

```
      do j=2,jM-2
      UN(IM-1,j)=((1.-r(j))*UP2(j)+2.*r(j)*U(im-2,j))/(1.+r(j))
      end do
      do j=2,jM-2
      VN(im,j)=VN(im-1,j)
      end do
      do j=2,jM-2
      den=UN(im-2,j)+UP3(j)-2.*U(im-3,j)
      if (ABS(den).lt..00001) goto 217
      r(j)=(UP3(j)-UN(j,3))/den
      if (r(j).gt..99) r(j)=.99
    if (r(j).lt.0.) r(j)=0.0
      goto 61
217   r(j)=0.0
 61   end do

      do J=1,JM
      UP2(J)=U(IM-1,J)
      UP3(J)=U(IM-2,J)
      end do

      do J=2,JM-2
      UN(2,J)=((1.-r(j))*UP2(j)+2.*r(j)*U(3,J))/(1.+r(j))
      end do
      do J=2,JM-2
      VN(1,J)=VN(2,J)
      end do
      do J=2,JM-2
      den=UN(3,J)+UP3(j)-2.*U(4,J)
      if (ABS(den).lt..00001) goto 227
      r(j)=(UP3(j)-UN(3,J))/den
      if (r(j).gt..99) r(j)=.99
      if (r(j).lt.0.) r(j)=0.0
      goto 62
227   r(j)=0.0
```

```
 62   end do

          do j=1,JM
          UP2(j)=U(2,J)
          UP3(j)=U(3,J)
          end do



          DO 400 J=1,JM
          DO 400 I=1,IM
          z(I,J)=zn(I,J)
          V(I,J)=VN(I,J)
400       U(I,J)=UN(I,J)
          EK=0
          DO 410 I=2,IM-2
          DO 410 J=2,JM-2
410 EK=EK+H(I,J)*((V(I,J)+V(I,J+1))**2+(U(I,J)+U(I+1,J))**2)/8
          WRITE(*,*)t,EK
      IF(t.LT.tend) GOTO 100


          Jnst=2
      OPEN(13,FILE='vel.dat')
          OPEN(14,FILE='vel_bottom.dat')
          OPEN(15,FILE='vel_z.dat')
          OPEN(16,FILE='eddy.dat')
          OPEN(17,FILE='elev.dat')
          OPEN(18,FILE='eddy_eld.dat')


          do j=1,JM
          do i=1,IM
          write (13,250) i*dx,j*dx,U(i,j),V(i,j)

c         z_h=z/h (z negative)
          z_h=-.8
```

```
      ed=.035*H(i,j)*USST
      ed=(cs/.05)*H(i,j)*SQRT(ABS(WX)**2.+ABS(WY)**2.)
      ed=.1*h(i,j)*(sqrt(abs(ts))+sqrt(abs(tb)))
      if (ed.lt..02) ed=.02

      ts=cs*WI**2
      tb=(9.81/cz**2)*(V(I,J)**2+U(i,j)**2)

      thx=TSX*H(i,j)/ed*(z_h+1.)
      uzh=(0.75*TSX*H(i,j)/ed-1.5*U(i,j))*(z_h**2-1.)+thx
      thy=TSY*H(i,j)/ed*(z_h+1.)
      vzh=(0.75*TSY*H(i,j)/ed-1.5*V(i,j))*(z_h**2-1.)+thy

      IF(H(I,J).lt..01) then
      uzh=0.
      vzh=0.
      end if

      IF(H(I,J).gt..01) then
UBB=.4*((U(I,J)+U(I+1,J))/2-USS/.4)/(ALOG(H(I,J)/ZOB)-1)
VBB=.4*((V(I,J)+V(I,J+1))/2-VSS/.4)/(ALOG(H(I,J)/ZOB)-1)
      else
      UBB=0.
      VBB=0.
      END IF


      IF(H(I,J).gt..01) then
      ed=.2*h(i,j)*(sqrt(abs(ts))+sqrt(abs(tb)))
      if (ed.lt..05) ed=.05

      b=tsx/ed
      a=(tsx-tbx)/ed/h(i,j)/2.
      gu=U(i,j)-a*(h(i,j)**2)/3.+b*h(i,j)/2.
```

```
      h7=h(i,j)**7
      h6=h(i,j)**6
      h5=h(i,j)**5
      h4=h(i,j)**4
      h3=h(i,j)**3
      par=(a**2)*h7/84-2*a*b*h6/48.-2*a*gu*h5/15.+b**2*h5/30.+
!  2*b*gu*h4/12.-(gu**2)*h3/3.


      edx(i,j)=dif0+abs(par)/(h(i,j)*ed)


      b=tsy/ed
      a=(tsy-tby)/ed/h(i,j)/2.
      gu=V(i,j)-a*(h(i,j)**2)/3.+b*h(i,j)/2.




      h7=h(i,j)**7
      h6=h(i,j)**6
      h5=h(i,j)**5
      h4=h(i,j)**4
      h3=h(i,j)**3
      par=(a**2)*h7/84-2*a*b*h6/48.-2*a*gu*h5/15.+b**2*h5/30.+
!  2*b*gu*h4/12.-(gu**2)*h3/3.

      edy(i,j)=dif0+abs(par)/(h(i,j)*ed)

      else
      edx(i,j)=0.
      edy(i,j)=0.
      ukar=0.
      vkar=0.
      end if
```

```
IF(H(I,J).gt..01) then
eldmin=0.23*H(i,j)*sqrt(abs(tb))
tbx=(9.81/cz**2)*U(i,j)*sqrt(V(I,J)**2+U(i,j)**2)
tby=(9.81/cz**2)*V(i,j)*sqrt(V(I,J)**2+U(i,j)**2)
edelx(i,j)=6.*H(i,j)*sqrt(abs(tbx))
edely(i,j)=6.*H(i,j)*sqrt(abs(tby))
if (edelx(i,j).lt.eldmin) edelx(i,j)=eldmin
if (edely(i,j).lt.eldmin) edely(i,j)=eldmin
else
edelx(i,j)=0.
edely(i,j)=0.
end if

write (14,250) i*dx,j*dx,UBB,VBB
write (15,250) i*dx,j*dx,uzh,vzh
write (16,250) i*dx,j*dx,edx(i,j),edy(i,j)
write (18,250) i*dx,j*dx,edelx(i,j),edely(i,j)

end do
end do


do j=1,JM
do i=1,IM
if (H(I,J).GT..01) THEN
zzz=z(i,j)
ELSE
zzz=-1.
END IF

write (17,250) i*dx,j*dx,zzz
end do
end do
```

```
      IF(t.LT.tend) GOTO 100
      CLOSE(1)
           CLOSE(2)
250   format(4f18.3)
           END
```

Listing of programme PODIS

and

main parameters used in PODIS

dt	time step
dx	grid size
tend	total time
im, jm	number of grid point in the x and y directions
xo,yo	coordinates of the initial pollutant position
U,V	horizontal velocities in the x and y directions
cs	pollutant concentration at the first computational sweep
cp	pollutant concentration at the second computational sweep
c	pollutant concentration at the final computational sweep
coo	initial pollutant concentration at the point (xo,yo)
H	water depth
disx , disy	dispersion coefficients in the x and y directions

Input files: **depth.dat**
(includes the values of im and jm and the depth H(i,j) at each grid point)

vel.dat (result file of the programme WINDCIR)
(includes the values of the horizontal velocities U(i,j) and V(i,j) at each grid point)

eddy.dat (result file of the programme WINDCIR)
(includes the values of the dispersion coefficients disx(i,j) and disy(i,j) at each grid point)

```
c      PROGRAMME PODIS
c      Advection-diffusion model for marine pollutant dispersion
c      Project CORI
c      PREVENTION AND MANAGEMENT OF SEA ORIGINATED RISKS TO THE
COASTAL ZONE
c      INTERREG III B ARCHIMED
c      PRIORITY AXIS: 3 - MEASURE: 3
c      WORK PACKAGE 3 - Action 3

c
       parameter (ijm=2500,iim=2500)
c

      dimension u(ijm,iim),v(ijm,iim),cs(ijm,iim),Disy(ijm,iim)
      dimension cp(ijm,iim),c(ijm,iim),Disx(ijm,iim), H(ijm,iim)



       t=0
       NN=0
       pi=3.141593
        data dx,dt,tend/20.,2., 1000/
          data xo,yo/3290,5410/
          data coo/100.0/

          ioo=xo/dx
          joo=yo/dx

          OPEN(12,FILE='depth.dat')
          read (12,*) im,jm

     DO 200 J=1,JM
     read (12,*) ij
     write (*,*) ij
     READ(12,*)(H(I,J),I=1,IM)
c      write(*,*)(H(I,J),I=1,IM)
200    continue
```

```
      CLOSE(12)



          OPEN(13,FILE='vel.dat')



          do j=1,JM
          do i=1,IM
          read (13,*) aii,ajj,U(i,j),V(i,j)
          end do
          end do
          close (13)



          OPEN(14,FILE='eddy.dat')
          do j=1,JM
          do i=1,IM
          read (14,*) aii,ajj,disx(i,j), disy(i,j)
          end do
          end do
          close (14)



55     t=t+dt
       NN=NN+1
       write (*,*) t

       if (NN/2..eq.int(NN/2)) then
       write (*,*) '**'
      do 20 j=3,jm-2
      do 20 i=3,im-2

      afx=u(i,j)*dt/dx
```

```
      afy=v(i,j)*dt/dx

      cx=8.*(c(i+1,j)-c(i-1,j))-(c(i+2,j)-c(i-2,j))
      cxx=30.*c(i,j)-16.*(c(i+1,j)+c(i-1,j))+c(i+2,j)+c(i-2,j)
      cs(i,j)=c(i,j)-((afx/12.)*cx+((afx**2)/24.)*cxx)

20    continue


      do 30 i=3,im-2
      do 30 j=3,jm-2
      afx=u(i,j)*dt/dx
      afy=v(i,j)*dt/dx

      cx=8.*(cs(i,j+1)-cs(i,j-1))-(cs(i,j+2)-cs(i,j-2))
      cxx=30*cs(i,j)-16*(cs(i,j+1)+cs(i,j-1))+cs(i,j+2)+cs(i,j-2)
      cp(i,j)=cs(i,j)-((afy/12.)*cx+((afy**2)/24.)*cxx)

30    continue

      do 40 j=3,jm-2
      do 40 i=3,im-2

      cxx=(cp(i+1,j)-2.*cp(i,j)+cp(i-1,j))/dx**2
      cyy=(cp(i,j+1)-2.*cp(i,j)+cp(i,j-1))/dx**2
      c(i,j)=cp(i,j)+dt*Disx(i,j)*cxx+dt*Disy(i,j)*cyy

40    continue
      else

      write (*,*) '*'

      do 930 i=3,im-2
      do 930 j=3,jm-2
      afx=u(i,j)*dt/dx
      afy=v(i,j)*dt/dx
```

```
      cx=8.*(cs(i,j+1)-cs(i,j-1))-(cs(i,j+2)-cs(i,j-2))
      cxx=30*cs(i,j)-16*(cs(i,j+1)+cs(i,j-1))+cs(i,j+2)+cs(i,j-2)
      cp(i,j)=cs(i,j)-((afy/12.)*cx+((afy**2)/24.)*cxx)

930   continue

      do 920 j=3,jm-2
      do 920 i=3,im-2

      afx=u(i,j)*dt/dx
      afy=v(i,j)*dt/dx

      cx=8.*(c(i+1,j)-c(i-1,j))-(c(i+2,j)-c(i-2,j))
      cxx=30.*c(i,j)-16.*(c(i+1,j)+c(i-1,j))+c(i+2,j)+c(i-2,j)
      cs(i,j)=c(i,j)-((afx/12.)*cx+((afx**2)/24.)*cxx)

920   continue



      do 940 j=3,jm-2
      do 940 i=3,im-2

      cxx=(cp(i+1,j)-2.*cp(i,j)+cp(i-1,j))/dx**2
      cyy=(cp(i,j+1)-2.*cp(i,j)+cp(i,j-1))/dx**2
      c(i,j)=cp(i,j)+dt*Disx(i,j)*cxx+dt*Disy(i,j)*cyy

940   continue

       end if

      if (t.lt.500) then
      c(ioo,joo)=c(ioo,joo)+100.
          c(ioo+1,joo+1)=c(ioo+1,joo+1)+coo
          c(ioo-1,joo+1)=c(ioo-1,joo+1)+coo
          c(ioo-1,joo-1)=c(ioo-1,joo-1)+coo
          c(ioo+1,joo-1)=c(ioo+1,joo-1)+coo
```

```
      else
           end if


           do i=1,im
           do j=1,jm
           if (H(i,j).lt.0.01) c(i,j)=0.
           end do
           end do

           do i=1,im
           do j=1,jm
           if (H(i-1,j).lt..01.and.H(i,j).gt..01) then
           c(i,j)=c(i+2,j)
           c(i+1,j)=c(i+2,j)
           end if
           if (H(i+1,j).lt..01.and.H(i,j).gt..01) then
           c(i,j)=c(i-2,j)
           c(i-1,j)=c(i-2,j)
           end if
           if (H(i,j-1).lt..01.and.H(i,j).gt..01) then
           c(i,j)=c(i,j+2)
           c(i,j+1)=c(i,j+2)
           end if
           if (H(i,j+1).lt..01.and.H(i,j).gt..01) then
           c(i,j)=c(i,j-2)
           c(i,j-1)=c(i,j-2)
           end if
           end do
           end do

          do 22 i=1,im
          c(i,jm-1)=c(i,jm-2)
22         c(i,jm)=c(i,jm-2)
          do 23 j=1,jm
          c(2,j)=c(3,j)
23         c(1,j)=c(3,j)
```

```
      do 24 j=1,jm
      c(im-1,j)=c(im-2,j)
24     c(im,j)=c(im-2,j)
        do i=1,im
        c(i,1)=c(i,3)
        c(i,2)=c(i,3)
        end do



      if (t.lt.tend) then
      goto 55
      else
      open (10,file='poll1000.dat')
      do 15 i=1,im
      do 15 j=1,jm
15     write (10,*) i*dx,j*dx,c(i,j)
      end if

      stop
      end
```