## Title

**Modelling of hydrocarbons dispersion. Adaptation and improvement of a state of the art model**

## Author

**Paula Martín Laguna**

## Tutor

**Agustín Sánchez-Arcilla Conejo and Marc Mestres Ridge**

## Department

**Ingeniería marítima**

## Intensification

**Modelización**

## Date

**September 2016**
Coastal waters quality is a big concern in our society. It is necessary to assure the parameters of the big ecosystem which is the sea are as invariable as possible so the life in it progresses and does not decrease. For this purpose, it is necessary to own tools in case a spill happens so the authorities can quickly design an action plan. There are several models which simulate the dispersion of hydrocarbons in the sea but few of them do it following both Lagrange and Stokes dispersion and movement theories. MEDSLIK_II is one of them but until now it was only able to process one certain type of input data. This dissertation brings a modification of the software so it is able a widely used type of currents and sea temperature forecast data: ROMS. After understanding the theory behind MEDSLIK_II, a new subroutine has been coded so the software becomes more universal.
Index

Index .................................................................................................................................... 3
Figures ................................................................................................................................ 5
Tables .................................................................................................................................. 6
1. Introduction ..................................................................................................................... 7
2. Objectives and method ................................................................................................. 8
3. MEDSLIK II. Theory background ............................................................................. 9
   3.1. Introduction ............................................................................................................... 9
   3.2. Model equations and state variables .................................................................... 10
   3.3. MEDSLIK-II tracer grid and solution methodology ........................................... 14
   3.4. Time rate of change of slick state variables ....................................................... 16
   3.5. Time rate of change of particle oil volume state variables ............................... 18
   3.6. Time rate of change of particle positions ......................................................... 18
   3.7. Numerical considerations ................................................................................... 22
4. MEDSLIK II. Code Architecture ........................................................................... 26
   4.1. Source Code .......................................................................................................... 26
   4.2. Meteo-oceanographic files .................................................................................. 27
   4.3. Input Data files ....................................................................................................... 27
   4.4. Script files and executables ................................................................................ 28
   4.5. Visualization software ........................................................................................ 28
   4.6. Output Data files ................................................................................................... 28
5. MEDSLIK II. Input files analysis ............................................................................... 30
   5.1. Introduction ........................................................................................................... 30
   5.2. MEDFF files ......................................................................................................... 30
   5.3. ROMS files .......................................................................................................... 31
   5.4. Comparison between inputs .............................................................................. 34
6. MEDSLIK II. Modifications ....................................................................................... 35
   6.1. Introduction ........................................................................................................... 35
   6.2. Extract_II.for ....................................................................................................... 35
   6.3. Bathymetry and coast line data ......................................................................... 36
   6.4. medslik_II.for ..................................................................................................... 37
7. MEDSLIK II. Case study ............................................................................................. 38
8. Conclusions and future modifications ..................................................................... 40
References ..................................................................................................................... 41
Appendix ......................................................................................................................... 43
  Input parameters in medslik_inputfile.txt ........................................................................ 44
  Input parameters in medslik5.par ...................................................................................... 47
  Subroutine added to Extract_II.for .................................................................................. 49
Figures

Figure 1: The advective and diffusive processes which are involved in the transport of contaminants in shallow seas (Noye, 1987) ................................................................. 9
Figure 2: Graphic representation of concentration classes (De Dominicis, et al., 2013) ........................................ 11
Figure 3: MEDSLIK-II model solution procedure methodology (De Dominicis, et al., 2013) ............................. 15
Figure 4: Weathering processes using Mackay's approach: evaporation (E), dispersion (D) and spreading (S) (De Dominicis, et al., 2013) .......................................................... 17
Figure 5: Ekman transport graphic .................................................................................................................. 20
Figure 6: MEDSLIK-II Reference Code Architecture (De Dominicis, 2012) ................................................... 26
Figure 7: Data storage of Medff files and reading direction ............................................................................. 31
Figure 8: Point of measurement of each variable ............................................................................................. 31
Figure 9: ROMS horizontal grid (Anon., 2016) ................................................................................................. 32
Figure 10: ROMS horizontal extracted grid (Anon., 2016) ............................................................................. 32
Figure 11: Example of ROMS vertical grid (Anon., 2015) .............................................................................. 33
Figure 12: Data storage of ROMS files and reading direction .......................................................................... 33
Figure 13: Structure of a file extracted from the main ROMS file ................................................................. 36
Figure 14: Region for the case study simulation ............................................................................................. 38
Figure 15: Partial view of the files generated by Extract_II.for ................................................................. 38
Figure 16: Structure of the hourly output file ................................................................................................. 39
Figure 17: Plot of Algeria test ......................................................................................................................... 39
Tables

Table 1: Oil spill model state variables (De Dominicis, et al., 2013) ................................................................. 12
Table 2: Meteo-oceanographic model outputs initially run by MEDSLIK-II .......................................................... 27
Table 3: ROMS variables’ size ............................................................................................................................... 32
1. Introduction

The quality of the sea is currently a big concern to our society. Climate change together with uncontrolled spills from factories and cities’ sewages threat everyday our coastal waters. Moreover, it is also important to take into account the oil pipes and overseas oil platforms, which need to follow high standards to ensure they do not compromise the quality of the water.

However, although governments are keeping a close eye on it and are continuously developing new strategies and policies, it is still necessary to own tools which help in case of a disaster, such as accidental leakages or spills.

Numerical modelling is a powerful tool in these cases. It allows us to work with a huge amount of data in a, relatively, small amount of time. The simulations performed by these models are an important source of information for the actions plans.

There are currently several models which have been develop in order to predict how a leak will disperse across the surface and the water column. Special attention is given to hydrocarbons leakages since they are a huge pollution source. The theory behind the simulation is usually either Lagrange, Stokes or a combination of both of them. The main inputs in every case are, on the one hand, sea variables such as currents speed, sea surface temperature and bathymetry; and, on the other, information about the leak such as type of oil and initial location. The sea data comes from different programs which extract it from boys located in the sea and process it. There are many sea data sources around the world but, unluckily, they do not generated files with the same structure. This generates problems when trying to standardize the dispersion of hydrocarbons modelling.

This dissertation has its initial point in an already existing free software called MEDSLIK II, developed originally by Robin Lardner and George Zodiatis and later modified by Michaela De Dominicis. This program can simulate oil dispersion across the entire Mediterranean Sea, allowing the user to reduce the sea section to work with by dividing it in different areas. MEDSLIK II combines both Lagrange and Stokes theories when simulating the movement of the oil particles. This document contains a chapter which goes through the theory behind the modelling process.

Even though MEDSLIK II is prepared to model any part of the Mediterranean Sea, it cannot be used in certain areas because the sea data locally obtained, ROMS data, does not match the data MEDSLIK II is prepared for. As said before, it is necessary and positive to own the more standardized tools possible so they can be used any time anywhere in the world. Therefore, this dissertation aims to upgrade MEDSLIK II so it can also process the sea data coming from ROMS, which is a free-surface, terrain-following, primitive equations ocean model widely used by the scientific community for a diverse range of applications.
2. Objectives and method

Nowadays standardization is key for the globalized society we live in. Any tool or process pursues the final goal of being able to be used anywhere in the world, providing a unique language so professionals can easily communicate and work together no matter where are they from.

Following what stated above, the main target of this dissertation is to provide an upgraded and generic tool which can process ROMS input files to model the dispersion of hydrocarbons in the sea. MEDSLIK II, a free software Fortran 77 based, is going to set the starting point.

To accomplish that, it is necessary to first understand and go through the theory that the base program MEDSLIK II uses to process and work with the data. Secondly, the structure of the code has to be analysed so it is possible to identify where it needs to be modified or added extra programing. Parallel to that, ROMS files are compared to the files that the program currently uses in order to find similarities. Once all these aspects are clear, new code lines will be added to the different sections. The structure of the code does not have to be changed in excess so the computation time does not increase. Finally, the code is compiled so it can be verified using available data for a hypothetical spill.

Special importance is given to the harmony of the code. Free software needs to be clean and clear so any user can easily read and understand it and, ultimately modify it.
3. MEDSLIK_II. Theory background

3.1. Introduction

The aim of this paper is to modify the already existent software MEDSLIK_II, so more sort of input data can be processed, specifically ROMS data. In this section the theory behind the performance of the program is presented. It is really important to understand how the software works so one can identify its needs and apply them when generating new code lines.

MEDSLIK_II simulates the evolution of an oil spill through coastal waters taking into account the meteorological and marine conditions at the air-sea interface (wind, waves and water temperature); the chemical characteristics of the oil; its initial volume and release rates; and the marine currents at different space scales and timescales. All of them are used when simulating the processes which are involved in the transport of contaminant in shallow seas, which are shown in Figure 1.

![Diagram showing the advective and diffusive processes involved in the transport of contaminants in shallow seas](image)

*Figure 1: The advective and diffusive processes which are involved in the transport of contaminants in shallow seas (Noye, 1987)*
The processes of transport, diffusion and transformation of the surface oil can be simulated with a Lagrangian model formalism coupled with Eulerian circulation models not found adequately described before and which MEDSLIK_II brings. (De Dominicis, et al., 2013)

Eulerian methods are based on the solution of the transport equation described in terms of a fixed coordinate system, while Lagrangian methods use a moving, deformable numerical coordinate system which follows the fluid flow; there is a new variable grid every time step (Noye, 1987). In Eulerian-Lagrangian methods, the advective part of the transport equation is treated in Lagrangian form and the results are usually interpolated onto a fixed (Eulerian) grid. These methods combine the convenience of a Eulerian grid for the solution of the diffusive part of the transport process with the accuracy of a Lagrangian treatment of the advective part.

Thereby, MEDSLIK_II is based on the reconstruction of an oil concentration field from the oil particles advection-diffusion processes. Also, different oil spill state variables are introduced and described. Finally, the model works with the input of recent data sets from numerical oceanographic models.

In this section, the main theory behind MEDSLIK-II is reviewed using M. De Dominicis, et al. (2013) paper as basis. One should refer to the article for the inside calculations related to oil characteristics and dispersions processes.

### 3.2. Model equations and state variables

As it has been stated in the introduction of this section, the innovation that MEDSLIK_II brings is the connection between the Lagrangian particle approach and the Eulerian oil concentration reconstruction. Equation 1 is the general equation for a tracer concentration, whose numerical solution in a Eulerian context is a well-known problem in oceanographic modelling (Noye, 1987):

\[
\frac{\partial C}{\partial t} + U \cdot \nabla C = \nabla \cdot (K \nabla C) + \sum_{j=1}^{M} r_j(x, C(x, t), t)
\]  

where \( C \) is a tracer concentration, \( U \) is the sea current mean field with components \( (U, V, W) \); \( K \) is the diffusivity tensor which parameterizes the turbulent effects and \( r_j(C) \) are the \( M \) transformation rates that modify the tracer concentration.

Different assumptions are made in order to make the connection possible. First of all, the consideration that the constituent particles do not influence water hydrodynamics and processes. Limitations exist at the water surface, because oil modifies air-sea interactions and surface wind drag. Besides, the constituent particles move with infinitesimal displacements without inertia and without interaction among them. Therefore, the volume of each particle is modified due to the chemical and physical processes acting on the entire slick instead of on every single particle.
After considering these assumptions, Equation 1 can be split in Equations 2 and 3:

\[
\frac{\partial C_1}{\partial t} = \sum_{j=1}^{M} r_j(x, C_1(x, t), t)
\]  

(2)

\[
\frac{\partial c}{\partial t} = -U \cdot \nabla C_1 + \nabla \cdot (K \nabla C_1)
\]  

(3)

where \( C_1 \) is the oil concentration solution only due the weathering processes.

The model first solves Equation 2 by mean of oil slick state variables. Then, the Lagrangian formalism is applied to solve Equation 3, discretizing the oil slick in particles with associated oil slick state variables. The concentration is obtained assembling the particles together.

MEDSLIK_II subdivides the main concentration \( C \) into four components, which are called structural state variables.

\[ \text{Figure 2: Graphic representation of concentration classes (De Dominicis, et al., 2013)} \]

Below, the different sub-concentrations are introduced:

- \( C_S \), Concentration at the surface. The oil slick is assumed as a continuous layer of material.

\[
C_S(x, y, t) = \frac{m}{A} \cdot \frac{L}{A} \cdot V_s [kg m^{-2}]
\]  

(4)

where \( m \) is the oil weight and \( A \) is the unit area.

- \( C_D \), Concentration at the subsurface. Droplets of various sizes form the oil. They can stick to the surface oil slick or sediment at the bottom.
Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna

\[ C_D(x, y, t) = \frac{\rho}{A} \cdot V_D [kgm^{-2}] \] (5)

- \( C_C \), Concentration adsorbed on the coast. A reference segment, \( L_C \), defines the coast where part of the surface oil gets adsorbed. Its volume has not any prognostic equation and it is calculated from the oil particle state variables, described below.

\[ C_C(x, y, t) = \frac{\rho}{L_C} \cdot V_C [kgm^{-2}] \] (6)

where \( L_C \) is a referent segment and \( V_C \) is the adsorbed oil

- \( C_B \), Concentration on the bottom. It is considered as a sink of oil dispersed in the water column. In the model, it is not computed but represented by a number of oil particles that reach the bottom.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Variable type</th>
<th>Variable name</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_S(x, y, t) )</td>
<td>Structural</td>
<td>Oil concentration at the surface</td>
<td>kg m(^{-2})</td>
</tr>
<tr>
<td>( C_D(x, y, t) )</td>
<td>Structural</td>
<td>Oil concentration dispersed</td>
<td>kg m(^{-2})</td>
</tr>
<tr>
<td>( C_C(x, y, t) )</td>
<td>Structural</td>
<td>Oil concentration on the coast</td>
<td>kg m(^{-1})</td>
</tr>
<tr>
<td>( C_B(x, y, t) )</td>
<td>Structural</td>
<td>Oil concentration at the bottom</td>
<td>kg m(^{-2})</td>
</tr>
<tr>
<td>( V_S(x, y, t) )</td>
<td>Slick</td>
<td>Oil slick surface volume</td>
<td>m(^3)</td>
</tr>
<tr>
<td>( V_S(x, y, t) )</td>
<td>Slick</td>
<td>Oil slick subsurface (dispersed) volume</td>
<td>m(^3)</td>
</tr>
<tr>
<td>( V_{TK}(x, y, t) )</td>
<td>Slick</td>
<td>Thick part of the surface oil slick volume</td>
<td>m(^3)</td>
</tr>
<tr>
<td>( V_{TN}(x, y, t) )</td>
<td>Slick</td>
<td>Thin part of the surface oil slick volume</td>
<td>m(^3)</td>
</tr>
<tr>
<td>( A_{TK}(t) )</td>
<td>Slick</td>
<td>Surface area of the thick part of the surface oil slick volume</td>
<td>m(^2)</td>
</tr>
<tr>
<td>( A_{TN}(t) )</td>
<td>Slick</td>
<td>Surface area of the thin part of the surface oil slick volume</td>
<td>m(^2)</td>
</tr>
<tr>
<td>( T_{TK}(x, y, t) )</td>
<td>Slick</td>
<td>Surface thickness of the thick part of the surface oil slick volume</td>
<td>m</td>
</tr>
<tr>
<td>( T_{TN}(x, y, t) )</td>
<td>Slick</td>
<td>Surface thickness of the thin part of the surface oil slick volume</td>
<td>m</td>
</tr>
<tr>
<td>( x_k(t) = (x_k(t), y_k(t), z_k(t)) )</td>
<td>Particle</td>
<td>Particle position</td>
<td>m</td>
</tr>
<tr>
<td>( u_{NE}(n_k, t) )</td>
<td>Particle</td>
<td>Non-evaporative surface oil volume particle attribute</td>
<td>m(^3)</td>
</tr>
<tr>
<td>( u_{EV}(n_k, t) )</td>
<td>Particle</td>
<td>Evaporative surface oil volume particle attribute</td>
<td>m(^3)</td>
</tr>
<tr>
<td>( \sigma(n_k, t) = 0, 1, 2, &lt; 0 )</td>
<td>Particle</td>
<td>Particle status index (on surface, dispersed, sedimented, on coast)</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 1: Oil spill model state variables (De Dominicis, et al., 2013)

The weathering processes showed in Equation 2 are applied to \( C_S \) and \( C_D \), particularly to the oil volumes, which are the basic oil slick variables of the problem. In Table 1 all the variables are summarized.

\[
\frac{dC_S}{dt} = \frac{\rho}{A} \frac{dV_S}{dt}
\] (7)

\[
\frac{dC_D}{dt} = \frac{\rho}{A} \frac{dV_D}{dt}
\] (8)

These two equations are the MEDSLIK_II equations for the concentration \( C_1 \) in Equation 2, the volumes change following the Mackay et al.(1980) fate algorithms which model the weathering processes. The basis of Mackay’s model is to divide the spill into a thick slick and
a thin slick. Evaporation and dispersion are considered separately for these two parts of the slick (Al-Rabeh et al., 2000).

To solve Equations 7 and 8, the surface volume is subdivided in a thin and a thick part, $V_{TN}$ and $V_{TK}$. This assumption is made in order to apply Mackay et al (1980) transformation process algorithms. The mentioned volumes are written as follows

$$V_{TN}(x,y,t) = A_{TN}(t)T_{TN}(x,y,t)$$  \hspace{1cm} (9)$$
$$V_{TK}(x,y,t) = A_{TK}(t)T_{TK}(x,y,t)$$  \hspace{1cm} (10)$$

where $ATK$ and $ATN$ are the areas occupied by the thick and thin surface and $TTK$ and $TTN$ are the thicknesses of each surface.

These are oil slick state variables which are used to solve the concentration changes due to weathering processes (see section 3.4).

The surface oil volume is then written as

$$V_S = V_{TN} + V_{TK}$$  \hspace{1cm} (11)$$

On the other hand, to solve the advection-diffusion processes in Equation 3, and compute $C_S$, $C_D$ and $C_C$, particle state variables are defined. The surface volume $V_S$ is broken into $N$ constituent particles which are characterized by a particle volume, $\delta(n_k,t)$, a particle status index, $\sigma(n_k,t)$, and a particle position vector:

$$x_k(n_k,t) = (x_k(n_k,t), y_k(n_k,t), z_k(n_k,t)) \quad k = 1, N$$  \hspace{1cm} (12)$$

where $nk$ is the particle identification number.

The evolution of this vector is given by the Langevin equation described in section 3.5.

Following Mackay’s conceptual model, the particle volume state variables are subdivided into the evaporative and non-evaporative particle volume attributes:

$$\partial(n_k,t) = \partial_E(n_k,t) + \partial_{NE}(n_k,t)$$  \hspace{1cm} (13)$$

These volumes are updated using empirical formulas that relate them to the time rate of change of oil slick volume state variables, see section 3.4.

The status index identifies the particles corresponding to the four structural state variables: for particles at the surface, $\sigma(n_k,t)=0$; for subsurface or dispersed particles $\sigma(n_k,t)=1$; for sedimented particles, $\sigma(n_k,t)=2$; and for particles on the coasts $\sigma(n_k,t)=-L_i$, where $Li$ is a coastline segment index, to be specified later.

In order to solve the complete advection-diffusion ad transformation problem in Equation 1, it is necessary to specify a numerical grid where particles can be count and compute the
concentration. Also, it allows defining a solution methodology, since there is no analytical relationship between the oil slick and the particle state variables.

### 3.3. MEDSLIK-II tracer grid and solution methodology

To connect Equation 2 and Equation 3, a discrete oil tracer grid system is defined, $x_T=(x_T, y_T)$, with a uniform but different grid spacing in the zonal and meridional directions, $(\delta x_T, \delta y_T)$. Therefore, the spatially discretized time evolution equations for the structural state variables are

$$\frac{d C_S}{dt}(x_T, y_T, t) = \frac{\rho}{\delta x_T \delta y_T} \frac{d V_S}{dt}(x_T, y_T, t)$$  \hspace{1cm} (14)$$

$$\frac{d C_D}{dt}(x_T, y_T, t) = \frac{\rho}{\delta x_T \delta y_T} \frac{d V_D}{dt}(x_T, y_T, t)$$  \hspace{1cm} (15)$$

The coastline is designed as a polygonal chain of points connected by segments of different lengths $\delta L_i$, which finally constitute the coastline segment $L_i$. Following this, Equation 6 becomes

$$C_L(L_i, t) = \frac{\rho}{\delta L_i} \cdot V_C(L_i, t)$$  \hspace{1cm} (16)$$

By relating the particle state variables to the oil tracer grid, it is possible to write the relationship between structural and particle state variables. That means being able to, for instance, follow the evolution of the different concentrations. The countable groups, $I_S$, $I_D$, of surface and subsurface particles contained in an oil tracer grid cell are defined as

$$I_S(x_T, y_T, t) = \left\{ n_k; \begin{array}{c} x_T - \frac{\delta x_T}{2} = x_k(t) = x_T + \frac{\delta x_T}{2} \\ y_T - \frac{\delta y_T}{2} = y_k(t) = y_T + \frac{\delta y_T}{2} \\ \sigma(n_k, t) = 0 \end{array} \right\}$$  \hspace{1cm} (17)$$

$$I_D(x_T, y_T, t) = \left\{ n_k; \begin{array}{c} x_T - \frac{\delta x_T}{2} = x_k(t) = x_T + \frac{\delta x_T}{2} \\ y_T - \frac{\delta y_T}{2} = y_k(t) = y_T + \frac{\delta y_T}{2} \\ \sigma(n_k, t) = 1 \end{array} \right\}$$  \hspace{1cm} (18)$$

Taking into account that, the surface and dispersed concentration can be reconstructed as

$$\left\{ \begin{array}{c} C_S(x_T, y_T, t) = \frac{\rho}{\delta x_T \delta y_T} \sum_{n_k \in I_S} \rho(n_k, t) \\ C_D(x_T, y_T, t) = \frac{\rho}{\delta x_T \delta y_T} \sum_{n_k \in I_D} \rho(n_k, t) \end{array} \right.$$  \hspace{1cm} (19)$$

To calculate the oil concentration on the coast, the set of particles attached to the coastal segment $L_i$, $I_C$, is used:

$$I_C(L_i, t) = \left\{ n_k; \sigma(n_k, t) = -L_i \right\}$$  \hspace{1cm} (20)$$

Then, the concentration of oil on each coastal segment is calculated by
In order to solve the different concentrations using the oil slick and particle state variable equation, it is necessary to create a sequential solution. As it is represented in Figure 3 MEDSLIK_II sets the initial conditions and solves the transformation processes (evaporation, dispersion, spreading). After, the thin and thick parts are updated, together with the particles volumes. After, the calculations to relocate the particle positions and update the particle status index are performed. Finally, MEDSLIK_II calculates the oil concentrations as described in Equations 19 and 21.

![Figure 3: MEDSLIK-II model solution procedure methodology (De Dominicis, et al., 2013)](image)

One of the most significant approximations is that the oil slick state variables depend only on the slick’s centre geographical position, which is updated after each advection-diffusion time step. The oil spill centre position is defined as follows:

\[ x_C(t) = \frac{\sum_{k=1}^{N} x_k(t)}{N} \]  \hspace{1cm} (22)

\[ y_C(t) = \frac{\sum_{k=1}^{N} y_k(t)}{N} \]  \hspace{1cm} (23)

### 3.3.1. Initial conditions

The oil release can be instantaneous or continuous. When the leakage lasts several hours or even months, it can happen that the initial volumes spilled have been transported away from the release site when the last ones are released (Liu, et al., 2011). To model that,
MEDSLIK II divides the total spill in sub-spills, consisting of a part of the oil released during an interval of time:

$$N_s = \frac{D_C}{T_C}$$  \hspace{1cm} (24)

where DC(s) is the release duration.

For instantaneous releases, the initial oil released is equal to the total oil released \( V_S(x_C, t_0) \). However, for a continuous oil spill release, for each interval the volume released is

$$V_S(x_C, t_0) = R_C T_C$$  \hspace{1cm} (25)

where Rc is the oil spill rate.

During an instantaneous release, \( N \) particles are released at the beginning, while in a continuous one, \( N_C \) particles are released every \( T_C \):

$$N_C = \frac{N}{N_s}$$  \hspace{1cm} (26)

The initial particle volume can be written as

$$\partial(n_k, t_0) = \frac{N_s V_S(x_C, t_0)}{N}$$  \hspace{1cm} (27)

And also the evaporative and non-evaporative oil volume components:

$$\partial_E(n_k, t_0) = \left(1 - \frac{\partial_{NE}}{100}\right) \partial(n_k, t_0)$$  \hspace{1cm} (28)

$$\partial_{NE}(n_k, t_0) = \frac{\partial_{NE}}{100} \partial(n_k, t_0)$$  \hspace{1cm} (29)

Where \( \partial_{NE} \) is the percentage of the non-evaporative component of the oil that depends on the oil type.

The first thin and thick area values are taken from the initial surface amount of oil released using relative thicknesses and \( F \), which is the area ratio. These three values are defined as input.

$$A^{TN}(t_0) = F A^{TK}$$  \hspace{1cm} (30)

$$A^{TK}(t_0) = \frac{V_S(x_C, t_0)}{T^{TK}(x_C, t_0) + F T^{TN}(x_C, t_0)}$$  \hspace{1cm} (31)

### 3.4. Time rate of change of slick state variables

Following Equation 11, the time rate of change of oil volume is written as

$$\frac{\partial V_S}{\partial t} = \frac{\partial V^{TK}}{\partial t} + \frac{\partial V^{TN}}{\partial t}$$  \hspace{1cm} (32)
Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna

Figure 4 shows the three main processes that induce changes of the surface oil volume. They are known as weathering processes: evaporation, dispersion and spreading. The first one acting on the oil slick is evaporation, since initially the main volume is on the surface. In addition, for the first hours, the spill spreads mechanically due to gravitational forces.

Figure 4: Weathering processes using Mackay’s approach: evaporation (E), dispersion (D) and spreading (S) (De Dominicis, et al., 2013)

The three processes are considered separately for the thick and thin slick and only at the slick centre. Therefore, the forecast equations are

\[
\frac{dV^\text{T}_\text{K}}{dt} = \frac{dV^\text{T}_\text{K}}{dt}_\text{(E)} + \frac{dV^\text{T}_\text{K}}{dt}_\text{(D)} + \frac{dV^\text{T}_\text{K}}{dt}_\text{(S)} \tag{33}
\]

\[
\frac{dV^\text{T}_\text{N}}{dt} = \frac{dV^\text{T}_\text{N}}{dt}_\text{(E)} + \frac{dV^\text{T}_\text{N}}{dt}_\text{(D)} + \frac{dV^\text{T}_\text{N}}{dt}_\text{(S)} \tag{34}
\]

Each term of these equations is described in detail in Appendices B1, B2 and B4 in De Dominicis, et al (2013) paper, given in terms of modified Mackay fate algorithms for evaporation, dispersion and spreading.

Following Mackay’s assumptions, \(T^\text{T}_\text{N}\) does not change. Hence, \(A^\text{T}_\text{N}\) is calculated as

\[
\frac{dA^\text{T}_\text{N}}{dt} = \frac{1}{T^\text{T}_\text{N}} \frac{dV^\text{T}_\text{N}}{dt} \tag{35}
\]

Where

On the other hand, for the thick slick

\[
\frac{dV^\text{T}_\text{K}}{dt} = T^\text{T}_\text{K} \frac{dA^\text{T}_\text{K}}{dt} + A^\text{T}_\text{K} \frac{dT^\text{T}_\text{K}}{dt} \tag{36}
\]

The area of the thick slick only changes due to spreading, so

\[
\frac{dA^\text{T}_\text{K}}{dt} = \frac{dA^\text{T}_\text{K}}{dt}_\text{(S)} \tag{37}
\]

\(V^\text{T}_\text{K}\) is updated using Equation 34 and the thickness changes are calculated by

\[
T^\text{T}_\text{K} = \frac{V^\text{T}_\text{K}}{A^\text{T}_\text{K}} \tag{38}
\]
3.5. Time rate of change of particle oil volume state variables

After the transformation processes have acted on the oil slick, the particle oil volumes are changed. For all particle status index, the evaporative oil particle volume changes following the empirical relationship

\[ \frac{\partial E(n_k, t)}{\partial t} = \left[ 1 - \frac{\partial NE}{100} - f^{(E)}(x_c, t) \right] \partial (n_k, t) \]  

(39)

Where \( f(E) \) is the fraction of oil evaporated defined as:

\[ f^{(E)}(x_c, t) = \frac{V_{TK}(x_c, t) + V_{TN}(x_c, t)}{V_{TK}(t_0) + V_{TN}(t_0)} \]  

(40)

For both surface and dispersed particles, the non-evaporative oil component does not change, while a certain fraction of the non-evaporative oil component of a beached particle can be modified due to adsorption processes occurring on a particular coastal segment, seeping into the sand or forming a layer on rocky shore. For beached particles, the non-evaporative oil component is then

\[ \frac{\partial NE(n_k, t)}{\partial t} = \frac{t-t_0}{T_S(L_i)} \sigma(n_k, t) = -i \]  

(41)

where \( t_0 \) is the instant when the particle passes from surface to beached status and vice versa, \( T_S(L_i) \) is a half-life for seepage or any other mode of permanent attachment to the coasts. It describes the “absorbency” of the shoreline by describing the rate of entrainment of the oil after it has landed at a given shoreline (Shen, et al., 1987). The half-life depends on the coastal type.

3.6. Time rate of change of particle positions

Immediately upon entering into a water body, the spilled oil spreads and forms a surface slick which covers a large area of the water surface and can be moved about by the action of winds, waves or currents (Shen, et al., 1987).

The evolution of the particle positions is given by \( n_k \) uncouple Langevin equations:

\[ \frac{dx_k(t)}{dt} = A(x_k, t) + B(x_k, t) \partial \xi(t) \]  

(42)

where the tensor \( A \) represents the deterministic part of the flow field and \( B \) the stochastic term together with \( \xi \) which is a random factor.

If we define the Wiener process

\[ W(t) = \int_0^t \partial(\sigma) \delta \sigma \]  

(43)
and apply the Itô assumption (Tompson & Gelhar, 1990), Equation 42 becomes equivalent to the Itô stochastic differential equation:

\[ dx_k(t) = A(x_k, t)dt + B(x_k, t)dW(t) \]  

where \( dW \) is a random increment.

The Wiener process describes the path of a particle due to Brownian motion modelled by independent random increments \( dW(t) \) sampled from a normal distribution with zero mean and second order moment \( dt \). Hence, \( dW(t) \) in Equation 44 can be replaced by a vector \( Z \) of independent random numbers, normally distributed, i.e. \( Z \sim \mathcal{N}(0,1) \), and multiplied by \( \sqrt{dt} \):

\[ dx_k(t) = A(x_k, t)dt + B(x_k, t)Z\sqrt{dt} \]  

The unknown tensors \( A(x_k, t) \) and \( B(x_k, t) \) are written as (Risken, 1984):

\[ dx_k(t) = \begin{bmatrix} U(x_k, t) \\ V(x_k, t) \\ W(x_k, t) \end{bmatrix} dt + \begin{bmatrix} \sqrt{2K_x} & 0 & 0 \\ 0 & \sqrt{2K_y} & 0 \\ 0 & 0 & \sqrt{2K_z} \end{bmatrix} \begin{bmatrix} Z_1 \\ Z_2 \\ Z_3 \end{bmatrix} \sqrt{dt} \]  

where \( K \) are the turbulent diffusivity coefficients.

For particles at the surface and dispersed, Equation 46 takes the form

\[ dx_k(t) = \begin{bmatrix} U(x_k, y_k, z_k, t) \\ V(x_k, y_k, z_k, t) \\ 0 \end{bmatrix} dt + \begin{bmatrix} \frac{dx_k'(t)}{dt} \\ \frac{dy_k'(t)}{dt} \\ \frac{dz_k'(t)}{dt} \end{bmatrix} \]  

The particle position does not change for particles at the surface. It can only change when the particle becomes dispersed and the horizontal velocity at the vertical position of the particle is used to displace the dispersed particles.

The Equation 47 can be expanded in different components according to the particle status index:

\[ \sigma = 0 \quad x_k(t) = \begin{bmatrix} U_c(x_k, y_k, 0, t) + U_w(x_k, y_k, 0, t) + U_s(x_k, y_k, 0, t) \\ U_c(x_k, y_k, 0, t) \end{bmatrix} dt + dx_k' \]

\[ \sigma = 1 \quad dx_k(t) = U_c(x_k, y_k, 0, t)dt + dx_k' \]  

where \( U_c \) is the Eulerian current velocity due to non-local wind, \( U_w \) is due to local wind and \( U_s \) is the velocity due to wave-induced currents or Stokes drift.

### 3.6.1. Current and local wind velocity terms

Ocean currents near the surface are generated by the atmospheric forcing. Although it can be subdivided into buoyancy fluxes and wind stresses, the last one is the most important in terms of kinetic energy of the induced motion (Wunsch, 1998). The wind-induced currents are attributable to non-local winds, and are dominated by geostrophic or quasi-geostrophic
dynamic balances (Pedlosky, 1986), where the pressure gradient force is balanced by the Coriolis effect and creates a flow parallel to the isobars. By definition, geostrophic and quasi-geostrophic motions have a timescale of several days and characterize oceanic mesoscale motion, a very important component of the large-scale flow field included in $U$. These currents dominate below the mixed layer, which its dynamics are typically considered ageostrophic. The dominant time-dependent, wind-induced currents in the surface layer are the Ekman currents due to local winds (Price, et al., 1987). All these components have to be considered in the $U_C$ field in Equation 48.

Ekman transport is described in Figure 5; surface currents flow at a 45º angle to the wind due to a balance between the Coriolis force and the drags generated by the wind and the water (Mann, 2006). If the ocean is divided vertically into thin layers, the main value of the velocity decreases until it dissipated. The direction also shifts slightly across each subsequent layer (right in the northern hemisphere, left in the southern hemisphere). This is called Ekman spiral (Knauss, 1978). If all flow over the length of the spiral is integrated, the net transportation is at 90º to the right (left) of the surface wind in the northern (southern) hemisphere (Colling, 2001).

![Figure 5: Ekman transport graphic](image)

Therefore, Ekman currents at the surface can be parametrized as a function of wind intensity and angle between winds and currents, i.e.

\[
U_W = \alpha \left[ W_x \cos \beta + W_y \sin \beta \right] \\
V_W = \alpha \left[ -W_x \sin \beta + W_y \cos \beta \right]
\]

(49) (50)

where $W_x$ and $W_y$ are the wind zonal and meridional components and $\alpha$ and $\beta$ are two parameters referred as drift factor and drift angle.

Current velocity fields can be obtained from analyses and forecasts produced by high-resolution ocean general circulation models (OGCMs).
3.6.2. Wave current term

Waves push pollutants by wave-induced velocities that are known as Stokes drift velocity, \( U_s \). It has to be added to the currents from OGCMs since most of the ocean models are not coupled with wave models (Röhrs, et al., 2012). Stokes drift is the net displacement of a particle in a fluid due to wave motion, resulting essentially from the fact that the particle moves faster forward when the particle is at the top of the wave circular orbit than it does backward when it is at the bottom of its orbit. Wave-driven transport is often ignored, although it may be the dominant mechanism transporting oil to adjacent beaches and coastal wetlands, whose environment has a big susceptibility to oil spills. It provides a natural mechanism for beaching of surface oil, one of the most damaging outcomes of a coastal oil spill (Sobey & Barker, 1997).

Considering the surface, the Stokes drift velocity intensity in the direction of the wave propagation is

\[
D_S(z = 0) = 2 \int_0^\infty \partial K(\partial)S(\partial)d\partial 
\]

where \( w \) is angular frequency and \( S(w) \) is wave spectrum.

To implement Equation 51, it is considered that the direction of the wave propagation is the same as the wind direction. Therefore,

\[
U_s = D_s \cos \theta \quad (52)
\]

\[
V_s = D_s \sin \theta \quad (53)
\]

Where \( \theta \) is the wind direction.

3.6.3. Turbulent diffusivity terms

It is assumed that the particle moving through the fluid gets a random impulse at each time step, due to the action of incoherent turbulent motions and that it has no memory of its previous turbulent displacement:

\[
dx'_k(t) = (2r - 1)\delta \quad (54)
\]

Where \( d \) is the particle mean path and \( r \) is a random real number.

The mean square displacement of Equation 54 is

\[
|dx'_k(t)^2| = \int_0^1 [(2r - 1)\delta]^2 dr = \frac{1}{3} \delta^2 \quad (55)
\]

while the mean square displacement of the turbulent terms in Equation 47 is \( dx'_k(t)^2 = 2Kdt \). Equalizing both terms the following is obtained:

\[
\delta^2 = 6K_x dt
\]
\[ \delta^2 = 6K_y dt \]
\[ \delta^2 = 6K_z dt \]  
(56)

Replacing these terms in Equation 54, it is possible to write the stochastic transport terms:

\[ dx'_{k}(t) = Z_1 \sqrt{2K_x dt} = (2r - 1) \sqrt{6K_h dt} \]
\[ dy'_{k}(t) = Z_2 \sqrt{2K_y dt} = (2r - 1) \sqrt{6K_h dt} \]
\[ dz'_{k}(t) = Z_3 \sqrt{2K_z dt} = (2r - 1) \sqrt{6K_v dt} \]  
(57)

where \( K_h \) and \( K_v \) are prescribed turbulent horizontal and vertical diffusivities.

### 3.7. Numerical considerations

In this section, the interpolation method between input fields and the oil tracer grid, to the numerical scheme used to solve Equations 33, 34 and 47, to the model time step and to the oil tracer grid selection is explained.

#### 3.7.1. Interpolation method

The environmental variables which are used in this model (atmospheric wind, ocean currents and sea surface temperature) are normally supplied on a different numerical grid than the oil slick centre or particle locations. Therefore, to calculate the advection process, it is necessary to compute the currents and winds at the particle locations. However, for the transformation processes calculation, the sea surface temperature and winds are interpolated at the slick centre.

To show these interpolation processes, \((x_E, y_E, z_E)\) is designed as the numerical grid on which the environmental variables are provided by the Eulerian models.

A pre-processing procedure is needed to reconstruct the currents in the area between the last water grid node of the oceanographic model and the real coastline. MEDSLIK\_II extrapolates the currents in a way that gives a velocity field to the land points:

\[ q_{x_E(i),y_E(i)} = \frac{q_{x_E(i+1),y_E(i)} + q_{x_E(i-1),y_E(i)} + q_{x_E(i),y_E(i-1)} + q_{x_E(i),y_E(i+1)}}{N_P} \]  
(58)

Then, the winds and currents are computed at the particle position \((x_k, y_k)\) for a fixed depth \(z_E\) with the following interpolation algorithm:

\[ q1 = q_{x_E(i),y_E(i)}[x_E(i + 1) - x_k] \]
\[ q2 = q_{x_E(i+1),y_E(i)}[x_k - x_E(i)] \]
\[ q3 = q_{x_E(i),y_E(i+1)}[x_E(i + 1) - x_k] \]
Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna

\[ q^4 = q_{x_E(i+1),y_E(i+1)}[x_k - x_E(i)] \]
\[ q_{x_k,y_k} = \frac{(q_1+q_2)[y_E(i+1)-y_k]+(q_3+q_4)[y_E(i)-y_k]}{\Delta x_E \Delta y_E} \] (59)

where \((x_k,y_k)\) is the particle position referenced to the oil tracer grid, \((x_E(i),y_E(i))\) and subsequent are the external four grid points.

The same algorithm is used to interpolate to the oil slick centre, \((x_C(t),y_C(t))\) the wind and sea surface temperature.

Besides, a vertical interpolation of the currents at the particle positions is also needed:
\[ q_{x_k,y_k,z_k} = \frac{1}{z_E(i)-z_E(i+1)}\{q_{x_k,y_k,z_E(i+1)}[z_E(i) - z_k] + q_{x_k,y_k,z_E(i)}[z_k - z_E(i+1)]\} \] (60)

### 3.7.2. Numerical time integration scheme

The Lagrangian horizontal particle motion, Equation 41, is solved using a Euler forward scheme as follows
\[ x_k(t + \Delta t) = x_k(t) + U(x_k,t) \Delta t + \Delta x'_k(t) \] (61)
where \(x_k(t)\) represents the particle position at the current time step.

Equations 33 and 34 are also solved the same way as before but with a different time step, so-called weathering time step,
\[ V^{TK}(t + \delta t) = V^{TK}(t) + \frac{dV^{TK}}{dt} \delta t \] (62)
\[ V^{TN}(t + \delta t) = V^{TN}(t) + \frac{dV^{TN}}{dt} \delta t \] (63)

The two processes above described are not solved with the same speed: transformation processes are faster than advection-diffusion processes. Since transformation equations are stiff, to integrate them it is necessary that the time step is shorter, a fraction, of the Lagrangian time step (Butenschöhn, et al., 2012).

### 3.7.3. Particle status updates

After the particles have moved for a Lagrangian time step, their oil volume and status are updated. A particle becomes dispersed if the following probability function
\[ P^{(D)}(t) = \frac{f^{(D)}(x_C,t)-f^{(D)}(x_C,t-\Delta t)}{1-f^{(D)}(x_C,t-\Delta t)} \] (64)
Is greater than a random number defined between 0 and 1:
\[ r < P^{(D)}(t) = \nabla \sigma(n_{k},t) = 1 \] (65)
The factor \( f(D)(x_C,t) \) is defined as

\[
f(D)(x_C,t) = \frac{V^{TK}(x_C,t) + V^{TN}(x_C,t)}{V^{TK}(x_C,t_0) + V^{TN}(x_C,t_0)} \tag{66}
\]

To evaluate the particles that adhere to the coast, it is checked whether the particle crosses the coastline defined by the line segments \( L_i \). If it does, the particle is moved to the intersecting position and changes its status from surface to beached. This situation may not last for ever and the particle may go back to the water (Shen, et al., 1987). The probability that runs that phenomenon is

\[
P^{(C)}(L_i,t) = 1 - 0.5 \left( \frac{\Delta t}{T_W(L_i)} \right) \tag{67}
\]

Therefore,

\[
r < P^{(C)}(t) = \nabla \sigma(n_k,t) = 0 \tag{68}
\]

The model does not follow oil fraction that is permanently deposited on the coast; the whole particles are not lost as permanently beached, but only a fraction of them. The actual number of particles remains constant.

The model just loses particle only when they get to the bottom. There is not a good parameterization that calculates the particles that actually reach the bottom: those that are located less than 20 cm from the bottom are automatically considered not dispersed anymore but at the bottom:

\[
H_B(x_k,y_k) - z_k < 20cm \sigma(n_k,t) = 2 \tag{69}
\]

### 3.7.4. Oil tracer grid and number of particles

It is important to take into account some considerations regarding the size of the oil tracer grid and the number of particles in order to avoid problems during the whole calculation process. Also, to ensure the correct reproduction of oil distribution in space and time.

Analysing the Equation 44, two spatial limitations can be extracted:

\[
L_A = U \Delta t \approx 180m \tag{70}
\]

\[
L_T = \sqrt{K \Delta t} \approx 60m \tag{71}
\]

where \( L_A \) is the adyective scale and \( L_T \) the diffusivity scale.

Therefore, the spatial resolution and the model time step have to be chosen so

\[
L_T < \delta x_T < L_A \tag{72}
\]

Besides, a method is needed to estimate the minimum and maximum number of particles. Equation 19 gives the concentration at the surface, which at the initial moment can be
written, in the limit of one particle in the tracer grid cell and assuming no evaporation and beaching, using Equation 27, as

$$C_S(x_T, y_T, t) = \frac{N_S V_S(x_C, t_0)}{N} \frac{\rho}{\delta x \delta y_T}$$  \hspace{1cm} (73)

Equation 69 can be used to determine the maximum and minimum number of particles, deciding first the maximum and minimum concentrations (which change depending on the scenario) and for a given tracer grid size:

$$N_{max} = \frac{N_S V_S(x_C, t_0)}{C_S^{min} \delta x \delta y_T} \rho$$  \hspace{1cm} (74)

$$N_{min} = \frac{N_S V_S(x_C, t_0)}{C_S^{max} \delta x \delta y_T} \rho$$  \hspace{1cm} (75)

All these procedures and equations are included in MEDSLIK-II code, developed by De Dominicis, et al. (2013). Further and inside calculations can be seen in the whole article referenced by the end of this paper.

The following sections present the structure of the code, together with all the modifications added so it is upgraded to a next level so it can read an extended type of currents and temperature data: ROMS files.
4. MEDSLIK II. Code Architecture

This chapter describes the structure of the code behind MEDSLIK II, including the inputs needed (both files and variables). In the Appendix of this paper, a list of all the variables with their descriptions can be found. In this section, the main architecture is described.

Medslik II system is composed of six main parts:

1) Source code
2) Meteo-oceanographic files
3) Input data files
4) Script files to execute the model in a Linux operative system
5) Visualization software
6) Out-put data files

Figure 6 shows MEDSLIK II reference code architecture where all the codes are listed.

![Figure 6: MEDSLIK-II Reference Code Architecture (De Dominicis, 2012)](image)

4.1. Source Code

The folder source contains all the Fortran 77 based source codes. The main ones and, therefore, the ones that need most of the changes to upgrade the program, are:

- medslik_II.for: simulates the oil spill.
- Extract_II.for: extracts and processes the required currents and temperature data from generic netCDF files.

### 4.2. Meteo-oceanographic files

MEDSLIK II has its own archive of bathymetry which is stored in EXE/data folder. New bathymetry files will need to be generated if one wants to run the program in specific areas not listed there.

Moreover, the program has a directory (DATA/fcst_data) where the user has to place the currents and wind files according, again, to the area where the simulation wants to be performed and also to the type of data (hourly or daily). In here it is necessary to create a new folder where to put the ROMS file. Table 2 shows a list of the initial mete-oceanographic model outputs which can be used by MEDSLIK II.

<table>
<thead>
<tr>
<th>MODEL OUTPUT</th>
<th>DIRECTORY</th>
<th>INPUT FILE STRING</th>
<th>FLAG USED IN THE CODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mediterranean Forecasting System (MFS) – hourly output</td>
<td>01h</td>
<td>MFS</td>
<td>70</td>
</tr>
<tr>
<td>Adriatic Forecasting System (AFS) – hourly output</td>
<td>A1h</td>
<td>AFS</td>
<td>72</td>
</tr>
<tr>
<td>Mediterranean Forecasting System (MFS) – daily output</td>
<td>OPA</td>
<td>MFS24</td>
<td>10</td>
</tr>
<tr>
<td>Adriatic Forecasting System (AFS) – daily output</td>
<td>A24</td>
<td>AFS24</td>
<td>11</td>
</tr>
<tr>
<td>Sicily Channel Regional Model (SCRM) – hourly output</td>
<td>S1h</td>
<td>SCRM</td>
<td>71</td>
</tr>
<tr>
<td>Sicily Channel Regional Model (SCRM) – daily output</td>
<td>S24</td>
<td>SCRM24</td>
<td>12</td>
</tr>
<tr>
<td>Western Mediterranean Model (WME) – hourly output</td>
<td>WME</td>
<td>WME</td>
<td>75</td>
</tr>
<tr>
<td>Relocatable model</td>
<td>H3k</td>
<td></td>
<td>74</td>
</tr>
<tr>
<td>Tyrrenian Forecasting System (TFS) – hourly output</td>
<td>T1h</td>
<td>TYRR</td>
<td>73</td>
</tr>
<tr>
<td>Tyrrenian Forecasting System (TFS) – daily output</td>
<td>T24</td>
<td>TYRR24</td>
<td>13</td>
</tr>
<tr>
<td>ECMWF 0.5° winds</td>
<td>ECM</td>
<td>ECMWF02S</td>
<td>25</td>
</tr>
<tr>
<td>ECMWF 0.25°</td>
<td>E25</td>
<td>ECMWF05</td>
<td>6</td>
</tr>
</tbody>
</table>

*Table 2: Meteo-oceanographic model outputs initially run by MEDSLIK-II*

### 4.3. Input Data files

Together with the currents and bathymetry, information regarding the spill needs to be introduced in the model. The different variables are manually inserted in two different files:

- medslk_inputfile.txt: contains spill data such as location, duration, type of oil and others. In the Appendix, a list and description of all variables can be found.
- medslk5.inp: here the user establishes the simulation parameters such as the drift angle or the diffusivity. Again, in the Appendix there is a complete list and description of every parameter.

### 4.4. Script files and executables

The script RUN.sh first launches the model run (medslk_II.sh) and afterwards the visualization execution (medslk_plots/medslk_plots.sh). The operations performed by the Linux shell script medslk_II.sh are listed below. The modification of the original program maintains this structure.

1. Read input data: medslik_inputfile.txt is processed.
2. Read the oil characteristics (using the routine read_oil_data.py) from the oil-type database (oilbases.txt and oil_list.txt).
3. Read slick contour. It can be read from satellite data file or from medslik_inputfile.txt (if the coordinates have been entered manually).
4. Save input data in the medslk5.inp file (which is read by medslk_II.exe)
5. Area selection: the routine lat_lon calculates the geographical limits of the area affected by the spill (assuming that the slick travels at less than 1.5 nauticalmiles/h from the spill site).
6. Check the currents and wind files needed for the simulation.
7. Extract currents, wind and SST data: this section calls the routine Extract_II.exe, one of the source codes that needs major changes due to the different input file (ROMS).
8. Run: medslk_II.exe is called so the simulation of the transport and weathering of the oil is performed.
9. Archive of the MEDSLIK_II output files, currents and winds used in the simulation in one unique folder ($HOME/MEDSLIK_II_1.01/EXE/output).

### 4.5. Visualization software

The MEDSLIK_II visualization software reads the oil-on-surface output (.srf) and plots the oil-slick concentration in space and time, as well as the winds and currents. It generates maps for each time and stores them in the output directory.

The user can modify some parameters of the visualization such as the lat/lon limits or the time step between two images. The file medslk_plots.ncl contains all the modifiable parameters.

### 4.6. Output Data files

After each simulation the input files are saved in a subfolder inside the $HOME/MEDSLIK_II_1.01/EXE/output directory. It contains the following:

- The input files in text format (medslik_inputfile.txt, medslk5.inp and medslk.tmp).
- The satellite data file (if used).
- The parameter file in text format (medslik5.par).
- The medslik_plots.ncl visualization parameter file.
- The medslik.fte file, which contains the trend over time of: the oil volume spilled, the percentage of oil evaporated, the percentage of oil on the surface, the percentage of oil dispersed, the percentage of oil on the coast and other parameters.
- outhhhh.srf files: contain the values of the oil concentration on the surface.
- outhhhh.dsp files: contain the values of the dispersed oil concentration.
- outhhhh.cst files: contain the values of the oil concentration on the coast.

In addition, the folder output contains one subfolder (plot) containing two subfolders where the data of the wind and currents in the area affected by the spill is stored.
Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna

5. MEDSLIK-II. Input files analysis

5.1. Introduction

All the information related to the currents, winds, temperature and bathymetry is stored in netCDF format files.

The Network Common Data Form (netCDF) software was developed by Unidata, a National Science Foundation-sponsored program empowering U.S. universities which has extended its influence around the world. It works as an I/O library, which can be called by C, FORTRAN, C++ and other languages. The library stores and retrieves data in self-describing, machine-independent datasets in the form of arrays. An array is an n-dimensional rectangular structure containing terms which all have the same data type. The type of data encompasses from single-point observations and time series to satellite or radar images. Array values are accessed directly, without knowing how data is stored. Extra information such as units is stored with the data (Rew, et al., 2016).

The netCDF libraries allow simultaneous access to multiple netCDF datasets which are identified by ID numbers in addition to ordinary file names.

Each dataset contains dimensions, variables and attributes. The dimensions have a name and a length, which is a positive integer. The variables have a name and an ID number. They represent an array of values of the same type. Each variable has a name, a data type, and associated attributes. It is important to identify these aspects so they can be properly called from the main program.

MEDSLIK II is prepared to work with netCDF files in a certain way. Here relays the aim of this dissertation, to upgrade MEDSLIK II so it can read an extensively used netCDF type of file: ROMS. The following section describe ROMS structure, although first, the original input files are reviewed so it is easier to identify conflict points.

5.2. MEDFF files

The Medff system provides with one file for each variable: temperature, u-currents and v-currents. One file contains hourly the information of one day. However, this system does not provide the data from 00:00 to 23:59; instead, it is given from 13:00 to 12:59 of the following day.

Each variable is calculated in the middle of the cell, so the string where they are stored has the same size: imx x jmx x kmx x ktmx (longitude length, latitude length, depth levels, measurement hours). Important is to notice that the depth levels are measured from the surface down, levels that are constant at any (lon , lat) position. There is a variable which stores the depth values in meters. Figure 7 shows a diagram of how the data is stored.
The file also contains information relatively the longitude and the latitude of each cell centre, although the program does not use it because it works with already set up areas.

5.3. ROMS files

ROMS is a free-surface, terrain-following, primitive equations ocean model widely used by the scientific community for a diverse range of applications. The data resolution is quite convenient, since the user can adjust both vertical and horizontal resolution.

A unique ROMS file contains all three variables: temperature, u-currents and v-currents. Also, when extracting the file, the user can ask for the duration of the forecast. In other words, instead of having one file per day, there is only one file containing all the information. Since the user selects when downloading the data which period of time they want the information about, once they have the file, the only way of figuring out the date is through an internal calculation based on the file’s variable scrum_time. This variable refers to the time since initialization since 1st January at 00:00.

Special attention needs to be given to where ROMS calculates each variable. Scalar ones are at the cell’s centre (temperature) while vector ones are calculated at the contour (u-currents and v-currents). Figure 9 shows the grid and the location of each variable.

Even though currents (both components) are defined in all the cells contour, when extracting data, one can usually just find the interior range of parameters, as it is shown in Figure 10. That results in not having any current data in the external contour. Furthermore, the dimensions of the three variables (temperature, u-currents and v-currents) are not the same, as it is shown in Table 3.
Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna

Figure 9: ROMS horizontal grid (Anon., 2016)

Figure 10: ROMS horizontal extracted grid (Anon., 2016)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>imx x jmx x kmx x ktmx</td>
</tr>
<tr>
<td>u-currents</td>
<td>imx-1 x jmx x kmx x ktmx</td>
</tr>
<tr>
<td>v-currents</td>
<td>imx x jmx-1 x kmx x ktmx</td>
</tr>
</tbody>
</table>

*Table 3: ROMS variables’ size*
Regarding depths, ROMS has its own reference system. It does not have, as Medff, horizontal levels at determined depths, which means that at shallow waters it stores null data at a certain point. Instead, it only takes into consideration the actual water levels by stretching curves over the ocean bottom. Figure 11 shows an example of these curves.

There is a simple equation which transforms from ROMS reference system to $z$-system in case it is needed to know to which $z$-depth a certain data belongs. In Section 6.2 this transformation is reviewed.

ROMS data starts at the bottom and goes up to the ocean surface. Figure 12 shows a diagram of how the data is stored (the same type of diagram as in Medff files has been used, although the depth levels do not correspond since they are not horizontal).
ROMS files contain many other variables regarding the currents and temperature. For this dissertation, special relevance is given to bathymetry at RHO-points variable and to the coefficients needed to transform S-coordinates to Z-coordinates.

5.4. **Comparison between inputs**

Once the structure of both input types is clear, the main differences are identified. The target is to create a subroutine that extracts the data from the ROMS file and transforms it so it is equal to the one MEDSLIK II is programmed to work with. Overall, the main differences are:

- Medff files store data of one day counting from 13:00 to 12:59 (including then data from two different calendar days). Therefore, there will be several files, one for each day. ROMS, however, contains all the information of every day in a unique file, starting at 00:00 to 23:59 of the last day of forecast.
- MEDSLIK_II identifies the files by its file name which has the following structure: YYMMDDHH. ROMS files have a standard name, “ocean_his”. Therefore, when extracting the data, the files need to be renamed so medslik_II.for can read them.
- The original MEDSLIK_II is programmed in order to get information from three different files, one for each current component and one for the temperature. However, ROMS includes these three variables in a unique file.
- In the original program, one can find the bathymetry information already stored in a library in .bath files. However, the ROMS file includes the bathymetry of the area to model. Therefore, it is necessary to create the .bath file from the information in there. This will allow the user to perform the model in any area, not only in the ones predefined.
- The three variables of interest, both components of the currents and temperature, are taken from the centre of the grid cell. However, ROMS presents the currents on the border of the cell and the temperature in the centre (see Figure 10).
6. MEDSLIK_II. Modifications

6.1. Introduction

MEDSLIK_II was programmed to run at specific areas of the Mediterranean Sea. This dissertation brings a modification which makes MEDSLIK_II universal; it can be used anywhere if the user has the required input files.

In this section, the main additions and modifications to the code are listed and reviewed. One has to consider that minor changes are not mentioned, although in the appendix of this paper the entire code is attached with the modifications and additions marked.

Most of the changes affect the source code, specially Extract_II.for. This code processes the input files (in this case only one) based on the data of the spill and extracts the information needed for the simulation.

One of the targets of this dissertation is to create a clean and clear code so it can be read and modified easily by other users. Therefore, the new code follows the structure of the old one so it does not affect its harmony.

6.2. Extract_II.for

One of the ROMS highlights is its data resolution. Since the area covered by the data is rather small (compared with original MEDSLIK_II data, which covers the entire Mediterranean), it is also the area where MEDSLIK_II performs the modelling. Therefore, it is not necessary, as it is with the other type of files, to select a subarea from the original area. This results in having a much bigger grid (in terms of cells number), but with better resolution.

First step is to interpolate the u and v-currents strings so the values are located in the cells centre. Since only two values (from each border of the cell) are available, the interpolation is lineal.

Since temperature values are already stored at the centre, after the interpolation is done, the three variables available to the next step, which is the vertical interpolation.

As it has been described before, ROMS data has a terrain-following vertical grid, so at shallow waters it does not store null numbers (as it would with a standard vertical grid). MEDSLIK_II works with the data at certain depths (10, 30 and 120 m) in all cells, so it is necessary to perform a transformation of the vertical grid to know the depth location of each cell:

\[ z(x, y, \sigma, t) = \zeta(x, y, t) + [\zeta(x, y, t) + h(x, y)]S(x, y, \sigma) \] (75)

\[ S(x, y, \sigma) = \frac{h_c + h(x, y)C(\sigma)}{h_c + h(x, y)} \] (76)
Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna

Where $S(x,y,\sigma)$ is a nonlinear vertical transformation functional, $\zeta(x,y,t)$ is the time-varying free-surface, $h(x,y)$ is the unperturbed water column thickness (a.k.a. bathymetry), $\sigma$ is a fractional vertical stretching coordinate ranging from -1 to 0, $C(\sigma)$ is a nondimensional, monotonic, vertical stretching function ranging from -1 to 0 and $h_c$ is a positive thickness controlling the stretching. It is important to notice that,

$$S(x, y, \sigma) = \begin{cases} 0 & \text{if } \sigma = 0, \quad C(\sigma) = 0 \quad \text{at free surface} \\ 1 & \text{if } \sigma = -1, \quad C(\sigma) = -1 \quad \text{at the ocean bottom} \end{cases} \quad [78]$$

As it is shown by the transformation above, ROMS files have a changing vertical grid throw time. This, again, gives even more precision to the modelling, since the data ($u$ and $v$ currents) is exactly measured at the exact depth.

Once the depth in meters string is available ($z$), several loops are performed to define the value of both $u$ and $v$ string that is located at the desired depth.

Finally, all the information which MEDSLIK_II requires to run is written in a file. The is a file for each time step.

![Figure 13: Structure of a file extracted from the main ROMS file](image)

6.3. Bathymetry and coast line data

A part from the $u$ and $v$-currents and temperature, MEDSLIK_II also needs the bathymetry of the modelling area, its coast line and the type of coast for each coast line segment. MEDSLIK_II has a library where the user can find bathymetry, coast line and type of coast files for each area for which it was programmed to run for. Therefore, it is necessary that the user manually introduce these files for their case since ROMS files do not include this information (only the bathymetry which is extracted in Extract_II.for and stored in a file ready to be read by MEDSLIK_II).

These files need to have the following structure:

- Bathymetry (*.bath):
  Heading
  Min Longitude  Max Longitude  Min Latitude  Max Latitude
  No. ‘x-cells’  No. ‘y-cells’
  Cells bathymetry starting from the higher left one and moving to the left.

- Coast line (*.map)
  Total no. of contours (coast lines)
No of-points at the coast line 0 (island) or 1 (open coast)
lon  lat

- Type of coast (*.cst): For each coordinate of a coast point, the user must define a type of coast according to this code:
  1  sand beach
  2  sand and gravel beach
  3  cobble beach
  4  rocky shore
  5  seawall; concrete, wharf, etc
  6  exposed headland
  7  sheltered sand or gravel beach
  8  sheltered rocky shore
  9  sheltered marsh or mud flats

These three files have to be placed at the MEDSLIK_II/EXE/data folder. For each simulation, these files need to be created accordingly.

6.4. medslick_II.for

medslick_II.for contains the simulation code based on the theory reviewed in section X. Minor modifications need to be applied since one of the improvement targets is to upgrade the code following its original structure: the ROMS file is processed so the information obtained has the same structure as the one medslick_II.for is programmed for.
7. MEDSLIK_II. Case study

To check the new code, a ROMS file containing data from Vigo has been used. The main effort has been generating through the routine Extract_II.for forecast data files which the same structure are the previously admitted by the program. A test case of the original program (which modelled a region of Algeria) has also been used to compare different aspects.

In this case, the ROMS file contains information of a 176x153x10x169 grid. This means a 7-days forecast of the region showed in Figure 14:

![Region for the case study simulation](image)

In this case, the grid will contain information regarding land areas. This increases unnecessarily the size of the files, resulting in a higher computational time if these points weren’t considered. However, the geography and size of the area make impossible to reduce the grid because the leakage can disperse from its original point (e.g. Vigo’s harbour located inside the ria) to any part of the current grid.

The files are correctly extracted and stored in the forecast folder for the subsequent simulation (a total of 192 files).

![Partial view of the files generated by Extract_II.for](image)
The bathymetry and coast line where already available. The coast line includes both Cíes islands although only part of them are considered in the simulation (see Figure 14).

Finally, medslik_II.for processes all input data so files containing oil concentration are created for each hour after the spill.

![Structure of the hourly output file](image1)

*Figure 16: Structure of the hourly output file*

In this case, the graphic output is not valid due to the resolution. Further modifications need to be added to adapt ROMS resolution to the plot’s. However, in Figure 17 one of the plots obtained in the case study is shown. This plot is useful to know the direction of the spill’s movement. However, it is not really useful regarding numerical data.

![Plot of Algeria test](image2)

*Figure 17: Plot of Algeria test*

Overall, except the plot section, the model is considered as validated.
8. Conclusions and future modifications

The original MEDSLIK_II brings to the hydrocarbons modelling field a new system of both Lagrange and Stoke particle movement. This innovation needs to be universal, therefore, the software has to be able to process as many types of data as possible. This dissertation brings a modification of the original code so it is now able to process ROMS data, a widely used forecast system.

Further work needs to also be done to the plot routine. It is currently programmed for big areas’ resolution so when simulating much smaller areas (as Vigo), its output results useless. It would be interesting to relate this function to the extract one so, the ROMS grid size is correlated to the plot resolution.

The grid which MEDSLIK_II uses is quadratic, which means that, sometimes, land data is considered, increasing the simulation time. Further modifications should allow to relocate the original grid so it is parallel to the coast (e.g.: when modelling Barcelona’s harbour, which has an approximately 45º inclination, being able to locate the grid parallel to it). This would include interpolating again ROMS data, so it has to be carefully studied.

Understanding the original code has been one of the most complicated parts: as a free software, MEDSLIK_II should be readable and understandable and it is currently not the case. Therefore, one of the future added improvements should be the revision of the entire code so it is more universal. The subroutine added to Extract_II.for has followed this vision and every modification of any part of the generic code has been precisely specified.
References


Available at: http://www.unidata.ucar.edu/software/netcdf/docs/ [Accessed 21 July 2016].


Modelling of hydrocarbons’ dispersion in the sea

Paula Martín Laguna


Appendix

Appendix .........................................................................................................................43
Input parameters in medslk_inputfile.txt ......................................................................44
Input parameters in medslk5.par ..................................................................................47
Subroutine added to Extract_II.for ..............................................................................49
Input parameters in medslik_inputfile.txt

The following are the input data entered manually.

Simulation name (SIM_NAME). Name of the simulation (any name without blank spaces and symbols), e.g. SIM_NAME=TEST

Model data (MODEL). A string representing each of the different current model outputs for which MEDSLIK-II can read the data. For example, MFS represents the MFS hourly output, MFS24 the MFS daily output, AFS the AFS hourly output and so on (for the complete list see Table 1), e.g. MODEL=MFS.

Wind data (WIND). A string representing each of the different wind fields for which MEDSLIK-II can read the data. ECMF025 represents ECMWF data with a resolution of 0.25, while ECMF05 represents ECMWF data with a resolution of 0.5 (for the complete list see Table 1), e.g. WIND=ECMF025.

Length of the Simulation (sim_length). The number of hours from the start of the spill or trajectory for which the prediction of position and state of the oil are required. This written using 4 characters, e.g. sim_length=0072.

Day of Spill/Start (day). The day on which the spill or the trajectory starts. This item must be written using 2 characters, e.g. day=03.

Month of Spill/Start (month). The month on which the spill or the trajectory starts. This item must be written using 2 characters, e.g. month=02.

Year of Spill/Start (year). The year on which the spill or the trajectory starts. This item must be written using 2 characters, e.g. year=09.

Hour of Spill/Start (hour). The hour of day between 0 and 23. This item must be written using 2 characters, e.g. hour=08.

Minutes of Spill/Start (minutes). The minutes after the hour of start, between 0 and 59, at which the spill or trajectory is started. This item must be written using 2 characters, e.g. minutes=05.

Location of the Spill/Start (lat_degree). The latitude at which the spill occurred is entered in degrees and decimal minutes. This item has 2 characters, e.g. lat_degree=42.

Location of the Spill/Start (lat_minutes). The latitude at which the spill occurred or the trajectory started is entered in degrees and decimal minutes. This item has 4 characters, e.g. lat_minutes=22.20.

Location of the Spill/Start (lon_degree). The longitude at which the spill occurred or the trajectory started is entered in degrees and decimal minutes. This item has 2 characters, e.g. lon_degree=10.
Location of the Spill/Start (lon_minutes). The longitude at which the spill occurred or the trajectory started is entered in degrees and decimal minutes. This item has 4 characters, e.g., lon_minutes=55.50.

Duration of the Spill/Releases (duration). The number of hours during which oil or pollutant was spilling. For an instantaneous spill enter 0. It is written using 4 characters, e.g., duration=0072 or duration=0000.

Rate of Spillage (spillrate). The rate (in tons per hour) at which oil was spilt. For an instantaneous spill, the total volume has to be written, e.g., spillrate=10.50.

Slick age (age). The slick age, which can be 0, 24 or 48 hrs, e.g., age=0.

Oil tracer grid size (grid_size): Each time the results are printed in one of the output files, the particles are aggregated inside ‘pixels’, the size of which is specified by the grid_size parameter, e.g., grid_size=150.

Oil (OIL). If the precise name of the oil is known, write OIL=NAME, if it is unknown write OIL=API.

Type of Oil. The name of the oil type that has been spilt can be chosen from a list of over 200 oils (oil_list.txt) e.g. OIL_TYPE=Arabian Heavy. If the precise name of the oil is unknown, you must enter the API number of the oil, e.g., OIL_TYPE=17.

Point or areal source of spill (ContourSlick and SAT_DATA): MEDSLIK-II allows the simulation of (1) a point source of spill or (2) an areal source of spill with manual insertion of contour coordinates or (3) an areal source of spill using satellite data.

1) Use SAT_DATA=NO and ContourSlick=NO to perform a simulation of a point source of spill (no need to fill the Number of Slicks - Nslick and list of latitude and longitude of slick points contour -- S1lon[1], S1lat[1], S1lon[2], S1lat[2]…)

2) Use SAT_DATA=NO and ContourSlick=YES to perform a simulation of an areal source of spill. If ContourSlick=YES is chosen, then the following parameters have to be specified (otherwise leave them blank):

   Number of Slicks (Nslick): number of oil slick to be simulated, written using 1 character, e.g., NSlick=2.

   List of latitude and longitude of slick points contour (S1lon[1], S1lat[1], S1lon[2], S1lat[2]…): latitude and longitude (in decimal degrees) of the oil slick polygon vertices.

3) Use SAT_DATA=YES and ContourSlick=NO to perform a simulation of an areal source of spill using satellite data (no need to fill the Number of Slicks - Nslick and list of latitude and longitude of slick points contour -- S1lon[1], S1lat[1], S1lon[2], S1lat[2]… ). When MEDSLIK-II is used for forecast of a slick observed by satellite or by other means, the data
for the observation is read from a file named initial.txt in the main model directory ($HOME/MEDSLIK_II_1.01/EXE). This file has been converted by the ReadSatData.py from a ‘.gml’ or ‘.xml’ file derived from the satellite image. Satellite data is received from EMSA in a file of type ‘*.gml’ which can be visualized on Google Earth. Such files may contain data for several oil slicks, with the boundary of each digitized to a fine resolution, especially in the case of a gml file. If SAT_DATA=YES is chosen, the following parameters have to be filled (otherwise leave them blank):

- **namefileGML** has to be filled with the name of the GML file (e.g. namefileGML=ASA_WSM_1PNACS20080806_095116_00000612071_00022_33643_0001.N1.0000_Oil.gml). The file has to be saved in the directory $HOME/MEDSLIK_1.01/EXE.
- **N_OS** has to be filled with the number of the oil slick in the file to be simulated (e.g. $N_OS=1$)
Input parameters in medslik5.par

The following will provide a brief description of the significance of the parameters in the medslik5.par. These are the parameters that a user is most likely to want to change from their default values.

Stokes drift correction. Choosing 01 allows the model to use the wave-induced velocity (Stokes drift) calculated using an empirical formulation. Choosing 00 Stokes drift velocity will not be applied. The default value is 01.

Wind correction (Drift Factor). The drift speed of the slick is equal to this factor multiplied by the wind speed. The default value is 0.

Wind correction (Drift Angle). The wind-driven drift of the slick occurs at this angle to the right of the wind direction. The default value is 0.0 degrees, which causes the slick to move directly downwind.

Variable Drift Angle. Choosing 01 allows the model to use a drift angle that decreases as the wind speed increases; a wind speed at which the drift angle is reduced by 50% must then be entered in the line below. The default is that such a reduction is not made.

Reduction of Forecast Wind Speed. When using forecast water circulation in a simulation, the forecast water velocities already include the effect of the wind forces on the water surface. It may thus be considered appropriate in some cases to reduce the wind speed used in the drift formula by a fraction of the winds used in the forecast. This can be done writing 01 in effective wind speed and entering in the line below the reduction fraction (between 0 and 1). The default is that such a reduction is not made.

Smagorinsky Scheme. Choosing 01 allows the horizontal diffusivity to be computed from the water currents using the Smagorinsky scheme. The default value is 0.

Horizontal Diffusivity. Enter the diffusivity that determines the horizontal diffusive spreading of the slick. The default value is 2.0 m²/s. A larger value will cause the slick to spread faster.

Vertical Diffusivities. The model allows the use of two values of vertical diffusivity: a larger value in the top well-mixed layer and a smaller value below the mixed layer. The defaults are 0.01 and 0.0001 m²/s respectively.

Depth of Mixed Layer. The default for this depth is 30 m.

Number of parcels. Diffusion and dispersion are modelled using a Monte Carlo algorithm, representing the oil by a large number of particles which are then given appropriate random displacements, and the resulting cloud of parcels used to estimate the concentration of the oil in the slick. MEDSLIK-II uses 10,000 parcels as the default, but the user can increase this up to 300,000.
Depths of Forecast Currents. The current fields are extracted from the Nectdf and are given in the text files in \$HOME/MEDSLIK_II_1.01/EXE/fcst_data at fixed depths. Whatever depths are used in the text files (they can only be changed by modifying the Extract_II.for code), the same depths must be entered in the parameters form. In the MEDSLIK-II_1.01 version the current fields are given at 3 depths (10 m, 30 m and 120 m) plus the surface.

Selection of currents for convection of slick. In the MEDSLIK-II_1.01 version it is possible to choose between 4 options: 00 for surface currents, 01 for 10 m deep currents, 02 for 30 m deep currents and 03 for 120 m deep currents. The default strategy is to use the surface velocities for advection of the slick and to ignore the wind drift altogether (by setting the drift factor equal to zero).

Number of Time Steps per Hour. The default value is 2. The default time step for computation of the convection and diffusion is 30 minutes. (A shorter step is used for the fate processes.) In general, this is adequate, but for a continuous spill with strong winds and/or currents, the 30-minute step may cause the computed slick to appear as a number of discrete slicks that do not merge for several hours. This lack of reality can be reduced by using a shorter time step.

Dimension of the Array used for representing the slick. Each time the results are printed in one of the output files, the particles are aggregated inside ‘pixels’, the size of which was entered in the medslik_inputfile.txt (grid_size). These pixels form an array the dimension of which is set by this parameter. The default array dimension is 2000×2000 and the maximum is 4000×4000. Choosing this maximum causes the run program to slow very significantly. On the other hand, too small a dimension causes the displayed slick to have straight, barrier-like, edges. If this occurs, you must either choose a larger array dimension or a larger pixel size.
Modelling of hydrocarbons' dispersion in the sea

Paula Martín Laguna

Subroutine added to Extract_II.for
This routine reads winds and currents from meteo-oceanographic model output (NetCDF files). The development of the MEDSLIK-II model is supported by a formal agreement Memorandum of Agreement for the Operation and Continued Development of MEDSLIK-II signed by the following institutions:

INGV - Istituto Nazionale di Geofisica e Vulcanologia
OC-UCY - Oceanography Center at the University of Cyprus
CNR-IAMC - Consiglio Nazionale delle Ricerche – Istituto per lo Studio dell’Ambiente Marino Costiero
CMCC - Centro Euro-Mediterraneo sui Cambiamenti Climatici

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

character regn*, indate(30)*8, indate_wind(30)*8, fc_dir*120, filename*9
common regn, alon1, alon2, alat1, alat2, numfiles, indate, numfiles_wind, indate_wind, iviod, icurrents, fc_dir, filename
integer len_dir

call getarg(1, fc_dir)
len_dir=120
do while(fc_dir(len_dir:len_dir).eq. ' ')
len_dir=len_dir-1
enddo

open(1, file='medslik.tmp')
read(1,*) regn, icurrents, iwind
print*, icurrents
read(1,*) alon1, alon2
read(1,*) alat1, alat2
read(1,*) numfiles
if(icurrents.ne.100) then
  do n=1, numfiles+1
    read(1, '(a8)') indate(n)
  enddo
  read(1, '(a8)') filename
  print*, filename
else
  read(1, '(a8)') filename
  print*, filename
endif
endif
read(1,*) iviod
close(1)

open(99,file='Extract.log')
if(icurrents.eq.10) call ExtractOPA(fc_dir,len_dir)
if(icurrents.eq.11) call ExtractADRI24(fc_dir,len_dir)
if(icurrents.eq.12) call ExtractSICI24(fc_dir,len_dir)
if(icurrents.eq.13) call ExtractTYRR24(fc_dir,len_dir)
if(icurrents.eq.70) call ExtractOPA_1hr(fc_dir,len_dir)
if(icurrents.eq.71) call ExtractSICI(fc_dir,len_dir)
if(icurrents.eq.72) call ExtractADRI(fc_dir,len_dir)
if(icurrents.eq.73) call ExtractTYRR(fc_dir,len_dir)
if(icurrents.eq.74) call ExtractRELO(fc_dir,len_dir)
if(icurrents.eq.75) call ExtractWESTMED(fc_dir,len_dir)
if(icurrents.eq.100) call ExtractROMS_1hr(fc_dir,len_dir,filename)
if(iwind.eq.6) call ExtractECMWF(fc_dir,len_dir)
if(iwind.eq.25) call ExtractECMWF25(fc_dir,len_dir)
stop
eend

c******************************************************************************
c     Extract medslik files from ROMS data 1hr

c******************************************************************************
subroutine ExtractROMS_1hr(fc_dir,len_dir,filename)

parameter(ktmx=169, imx=176, jmx=153, kmx=10, imx_U=175, jmx_V=152, theta_s=6, thetha_b=0.9, hc=1)

real sc_r(10), css_r(10)
real fmis !$netcdf
parameter(fmis=0.) !$netcdf
integer start(4), start_lonlat (2), start_mask(2), start_h(2), &
    start_st(2), count(4), count_lonlat(2), count_mask(2), &
    count_h(2), count_st(2) !$netcdf
integer id, idlon, idlat, idU, idV, idT, idmask_rho, idh, idst !$netcdf
integer Status !$netcdf
integer msk(imx,jmx), mask(imx,jmx), st(ktmx,1)
real oplon(imx), oplat(jmx), &
    ts(imx,jmx,kmx), u(imx,jmx,kmx), v(imx,jmx,kmx), &
    ts_tmp(imx,jmx,kmx), u_tmp(imx,jmx,kmx), v_tmp(imx,jmx,kmx), &
    vs_tmp(imx,jmx,kmx), lon_tmp(imx), lat_tmp(jmx), &
    vs 169(imx,jmx,kmx,ktmx), u 169(imx,jmx,kmx,ktmx), &
    v 169(imx,jmx,kmx,ktxm), lon(imx,jmx), lat(imx,jmx), &
    u_in(imx_U,jmx,kmx,ktmx), v_in(imx,jmx_V,kmx,ktmx), &
    z(imx,jmx,kmx), h(imx,jmx)
character Startd*6, filename*9, prdate*16, outfile*40, infile*120, &
    heads*150, empty*80, regn*4, ora*2, ore*2, fc_dir*120, &
    dia*2, hour*2, dayc*2, monthc*2
 logical ex
 integer t, o, kount, nore, len_dir, Startdi, Startdd, d, s, dur, &
     month, day, dayS
 common regn, alon1, alon2, alat1, alat2, numfiles, indate, &
     iviod, icurrents
 data udef /9999., rhoa /1.19/

c----------------------------------------------------------------------------
c     Read ROMS file

c----------------------------------------------------------------------------
c Get to the directory

Status = 0
infol=fc_dir(1:len_dir):'/fcst_data/ROMS/
&
    //filename(1:9)''.'nc'
len_file=120
do while(infol=len_file:len_file).eq.' ')
len_file=len_file-1
enddo

c Open netCDF file

Status = nf_open(infile(1:len_file), nf_nowrite, id)
print *, id
call handle_err(Status)

c------------------------------------------------------------------
c     ROMS horizontal grid
c------------------------------------------------------------------
c Check of the variables from ROMS file

Status = nf_inq_varid(id, 'lon_rho', idlon)
call handle_err(Status)
Status = nf_inq_varid(id, 'lat_rho', idlat)
call handle_err(Status)
Status = nf_inq_varid(id, 'u', idU)
call handle_err(Status)
Status = nf_inq_varid(id, 'v', idV)
call handle_err(Status)
Status = nf_inq_varid(id, 'temp', idT)
call handle_err(Status)
Status = nf_inq_varid(id, 'scrum_time', idst)
call handle_err(Status)
Status = nf_inq_varid(id, 'mask_rho', idmask_rho)
call handle_err(Status)
Status = nf_inq_varid(id, 'h', idh)
call handle_err(Status)

print *, idh

c Since longitudes and latitudes at RHO points are available, the grid
c is already built. It's just necessary to extract the strings holding
c the data.

c Longitudes string (at RHO points)

start_lonlat(1) = 1
start_lonlat(2) = 1
count_lonlat(1) = imx
count_lonlat(2) = 1

Status = nf_get_vara_real(id, idlon, start_lonlat, count_lonlat,
&                       lon_tmp)
call handle_err(Status)
oplon = lon_tmp

c Latitudes string (at RHO points)

start_lonlat(1) = 1
start_lonlat(2) = 1
count_lonlat(1) = 1
count_lonlat(2) = jmx

Status = nf_get_vara_real(id, idlat, start_lonlat, count_lonlat,
&                       lat_tmp)
call handle_err(Status)
oplat = lat_tmp

c The spill will affect the entire grid due to the size of the area.
c Therefore, it is easy to update the coordinates of the RHO points
c and number of total cells affected by the spill.

alon1 = oplon(1)
alon2 = oplon(imx)
alat1 = oplat(1)
alat2 = oplat(jmx)

i_first = 1
i_last = imx
j_first = 1
j_last  = jmx
imax = imx
jmax = jmx

Write information in the file Extract.log

write(99,*) 'i-limits = ',i_first,i_last,imax
write(99,*) 'j-limits = ',j_first,j_last,jmax
write(99,*) 'lon-limits = ',alon1,alon2,imx
write(99,*) 'lat-limits = ',alat1,alat2,jmx

c------------------------------------------------------------------
c     read ROMS data file
c------------------------------------------------------------------

All the information regarding currents and temperature is stored in the same netcdf file, which has been already open to extract the longitudes and latitudes. In this section, a part from extracting the relevant strings, an interpolation is performed, since the data corresponding to the currents has been measured at the borders of the grid and MEDSLIK II works with all the data at the centre of it. A linear interpolation is performed, taking into account that the borders of the grid do not have information, therefore no interpolation is possible there. More details can be found at the report.

c------------------------------------------------------------------
c Extract U

c Linear interpolation. Because of the data available, first and last columns can't be interpolated


Interpolation for the rest

u_169(l,m,n,o)=u_in(l,m,n,o)+
                   (u_in(l,m,n,o)-u_in(l-1,m,n,o))/2

enddo
enddo
enddo
enddo

Extract V

start(1) = 1
start(2) = 1
start(3) = 1
start(4) = 1

count(1) = imx
count(2) = jmx - 1
count(3) = kmx
count(4) = 1

do t = 1,ktmx
start(4) = t
  Status = nf_get_vara_real (id, idV, start, count, v_tmp)
call handle_err(Status)
endo
c Linear interpolation. Because of the data available,
c first and last rows can't be interpolated
v_169(l:imx,1:kmx,1:ktmx) = v_in(l:imx,1:kmx,1:ktmx)
v_169(1:imx,l:jmx,1:kmx,1:ktmx) = v_in(1:imx,l:jmx,1:kmx)

do l=1,imx
do m=2,jmx-1
do n=1,kmx
do o=1,ktmx
  v_169(l,m,n,o)=v_in(l,m-1,n,o)+
  (v_in(l,m,n,o)-v_in(l,m-1,n,o))/2
endo
endo
endo
endo
c Extract T
start(1) = 1
start(2) = 1
start(3) = 1
start(4) = 1
count(1) = imx
count(2) = jmx
count(3) = kmx
count(4) = 1

do t = 1,ktmx
start(4) = t
  Status = nf_get_vara_real (id, idT, start, count, ts_tmp)
call handle_err(Status)
endo
c There is no need to interpolate temperature since this variable
c is available at them RHO points.
--------------
Obtain starting and ending date of the data
--------------
MEDSLIK_II input files are hour files. These files' names have the
following structure: nameYYMMDDHH, where name is a 4 digits
description of the region where the data comes from.
The main target of this work is to adapt the ROMS files so MEDSLIK_II
can process them. Therefore, it is necessary to create these hour
c files and give them a name that follows the structure showed above.
ROMS data, unlike MFS data, start at 00:00. The netcdf file contains
c a string (scrum_time) which stores the time since initialization.
This variable will be used to obtain: initial date, final date and
c number of days.
--------------
Extract scrum_time (st)
start_st(1) = 1
start_st(2) = 1
count_st(1) = ktmx
count_st(2) = 1

Status = nf_get_vara_int (id, idst, start_st, count_st, st)
call handle_err(Status)

c Starting month and day of the data
im = st(1, 1)/86400

im1 = im - 31
im2 = im - 31 - 28
im3 = im - 31 - 28 - 31
im4 = im - 31 - 28 - 31 - 30
im5 = im - 31 - 28 - 31 - 30 - 31
im6 = im - 31 - 28 - 31 - 30 - 31 - 30
im7 = im - 31 - 28 - 31 - 30 - 31 - 30 - 31
im8 = im - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 30
im9 = im - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 30 - 31
im10 = im - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 30 - 31 - 30
im11 = im - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 30 - 31 - 30 - 31
im12 = im - 31 - 28 - 31 - 30 - 31 - 30 - 31 - 30 - 31 - 30 - 31 - 30

if (im1 .le. 0) then
  month = 1
  day = im1 + 31
else if (im2 .le. 0) then
  month = 2
  day = im2 + 28
else if (im3 .le. 0) then
  month = 3
  day = im3 + 31
else if (im4 .le. 0) then
  month = 4
  day = im4 + 30
else if (im5 .le. 0) then
  month = 5
  day = im5 + 31
else if (im6 .le. 0) then
  month = 6
  day = im6 + 30
else if (im7 .le. 0) then
  month = 7
  day = im7 + 31
else if (im8 .le. 0) then
  month = 8
  day = im8 + 31
else if (im9 .le. 0) then
  month = 9
  day = im9 + 30
else if (im10 .le. 0) then
  month = 10
  day = im10 + 31
else if (im11 .le. 0) then
  month = 11
  day = im11 + 30
else if (im12 .le. 0) then
  month = 12
  day = im12 + 31
else
  print *, 'There is an error with the scrum time string'
endif

c Variable with the starting day of the data (YYMMDD)
write (monthc, '(i2)') month
write (dayc, '(i2)') day

if (month .lt. 10) then
  if (day .lt. 10) then
    Startd = '150'//monthc(2:2)//'0'//dayc(2:2)
  else
    Startd = '150'//monthc(2:2)//dayc(2:2)
  endif
else
  if (day .lt. 10) then
    Startd = '150'//monthc(2:2)//'0'//dayc(2:2)
  else
    Startd = '150'//monthc(2:2)//dayc(2:2)
  endif
endif
Startd='150'/monthc(2:2)//dayc(1:2)
endif
else
  if (day 10) then
    Startd='15'/monthc(1:2)//'0'/dayc(1:2)
  else
    Startd='15'/monthc(1:2)//dayc(1:2)
  endif
endif

c Finishing month and day of the data
fm=st(169,1)/86400
fm1=fm-31
fm2=fm-31-28
fm3=fm-31-28-31
fm4=fm-31-28-31-30
fm5=fm-31-28-31-30-31
fm6=fm-31-28-31-30-31-30
fm7=fm-31-28-31-30-31-30-31
fm8=fm-31-28-31-30-31-30-31-31
fm9=fm-31-28-31-30-31-30-31-31-30
fm10=fm-31-28-31-30-31-30-31-30-31
fm11=fm-31-28-31-30-31-30-31-30-31-30
fm12=fm-31-28-31-30-31-30-31-30-31-30-31

if (fm1 0) then
  fmonth=1
  fday=fm1+31
else if (fm2 0) then
  fmonth=2
  fday=fm2+28
else if (fm3 0) then
  fmonth=3
  fday=fm3+31
else if (fm4 0) then
  fmonth=4
  fday=fm4+30
else if (fm5 0) then
  fmonth=5
  fday=fm5+31
else if (fm6 0) then
  fmonth=6
  fday=fm6+30
else if (fm7 0) then
  fmonth=7
  fday=fm7+31
else if (fm8 0) then
  fmonth=8
  fday=fm8+31
else if (fm9 0) then
  fmonth=9
  fday=fm9+30
else if (fm10 0) then
  fmonth=10
  fday=fm10+31
else if (fm11 0) then
  fmonth=11
  fday=fm11+30
else if (fm12 0) then
  fmonth=12
  fday=fm12+31
else
  print *, 'There is an error with the scrum time string'
endif

c Duration data (in days). Addition of one since the last data is the
  c one of the following day at 00:00
  dur=(ktmx/24)+1
c     if (fmonth.eq.month) then
506  c     datadays=fday-day+1
507  c   else if (month.eq.1).or.(month.eq.3).or.(month.eq.5).or.
508  c      (month.eq.7).or.(month.eq.8).or.(month.eq.10).or.
509  c      (month.eq.12) then
510  c     datadays=(31-day)+fday+1
511  c   else if (month.eq.2)
512  c     datadays=(28-day)+fday+1
513  c   else
514  c     datadays=(30-day)+fday+1
515  c endif
516
517 c--------------------------------------------------------------------
518 c     Create hourly files (romsYYMMDDHH)
519 c--------------------------------------------------------------------
520 c Now hourly files need to be created. Their names will follow the
521 c structure described below. Two loops are necessary: one for each day
522 c and one for each hour. One thing to take into c
523 One thing to take into c...
524 the hour 24 belongs to t...
525
do d=1,dur
526  60 do t=0,23
527
c the day of the data needs to be updated. Currently, this program only
528 c performs simulations during the same month.
529
dayS=day+d-1
530
write(6,*) 'Writing medslik file for date '//prdate
531 write(99,*) 'Writing medslik file for date '//prdate
532
533 if (t.lt.10) then
534 if (dayS.lt.10) then
535   prdate = '0'//dia(2:2)///Startd(3:4)///Startd(1:2)///
536 &     '0'/hour(2:2)/':00'
537 else
538   prdate =dia(1:2)///Startd(3:4)///Startd(1:2)///
539 &     '0'/hour(2:2)/':00'
540 endif
541 else
542 if (dayS.lt.10) then
543   prdate = '0'//dia(2:2)///Startd(3:4)///Startd(1:2)///
544 &     '0'/hour/:00'
545 else
546   prdate =dia(1:2)///Startd(3:4)///Startd(1:2)///
547 &     '0'/hour/:00'
548 endif
549
550 write(6,*) 'Writing medslik file for date '//prdate
551 write(99,*) 'Writing medslik file for date '//prdate
552
553 if (t.lt.10) then
554 if (dayS.lt.10) then
555   outfile='fcst_data/ROMS/'//roms'//Startd(1:2)//Startd(3:4)///
556 &   dia/'0'/hour(2:2)/'.opa'
557 else
558   outfile='fcst_data/ROMS/'//roms'//Startd(1:2)//Startd(3:4)///
559 &   dia//hour/'.opa'
560 endif
561 inquire(file = outfile, EXIST = ex)
562 if(ex) then
563 open(20,file = outfile)
564 read(20,*) empty
565 read(20,*) empty
566 read(20,'(4f9.5,2i5)') blon1,blon2,blat1,blat2,imax1,jmax1
567 if(blon1.eq.alon1.and.blon2.eq.alon2.and.blat1.eq.alat1.and.
568   &   blat2.eq.alat2.and.imax1.eq.imax.and.jmax1.eq.jmax) then
577    write(6,*),outfile//' already exists for this subregion'
578    go to 60
579  endif
580  close(20)
581  endif
582
583  c---------------------------------------------------------------
584  c     read OPA data files
585  c---------------------------------------------------------------
586  c Since hour unique file has ktmx hours, when defining the hour files
587  c it is necessary to define a criteria so when the loop overpasses the
588  c first day, it continues through the ktmx hours.
589  if(d.eq.1) then
590    u(1:imx,1:jmx,1:kmx) = u_169(1:imx,1:jmx,1:kmx,t+1)
591    v(1:imx,1:jmx,1:kmx) = v_169(1:imx,1:jmx,1:kmx,t+1)
592    ts(1:imx,1:jmx,1:kmx) = ts_169(1:imx,1:jmx,1:kmx,t+1)
593  else if(d.lt.dur) then
594    s=(d-1)*24+t+1
595    u(1:imx,1:jmx,1:kmx) = u_169(1:imx,1:jmx,1:kmx,s)
596    v(1:imx,1:jmx,1:kmx) = v_169(1:imx,1:jmx,1:kmx,s)
597    ts(1:imx,1:jmx,1:kmx) = ts_169(1:imx,1:jmx,1:kmx,s)
598  else
599    u(1:imx,1:jmx,1:kmx) = u_169(1:imx,1:jmx,1:kmx,ktmx)
600    v(1:imx,1:jmx,1:kmx) = v_169(1:imx,1:jmx,1:kmx,ktmx)
601    ts(1:imx,1:jmx,1:kmx) = ts_169(1:imx,1:jmx,1:kmx,ktmx)
602  endif
603
604  c---------------------------------------------------------------------
605  c     mask & nwp
606  c---------------------------------------------------------------------
607  c The ROMS file provides the mask on RHO-points. Therefore, it is not
608  c necessary to evaluate the values of u, v or ts since it is already
609  c know where are there water points.
610  c Extraction of the variable which stores the mask
611
612  start_mask(1) = 1
613  start_mask(2) = 1
614  count_mask(1) = imx
615  count_mask(2) = jmx
616  Status = nf_get_vara_int(id, idmask_rho, start_mask,
617                           & count_mask, mask)
618  call handle_err(Status)
619  print*,'nnn',maxval(mask)
620  nwp = 0
621  do i=1,imx
622    do j=1,jmx
623      if(mask(i,j).eq.1) then
624        nwp = nwp + 1
625      endif
626    enddo
627  enddo
628  nwp = nwp
629  write(99,*), 'nwp = ',nwp
630  write(99,*), 'mask: '
631  do j=j_last,j_first,-1
632    write(99,'(300i1)') (mask(i,j),i=i_first,i_last)
633  enddo
634  write(99,'(300i1)') (mask(i,j),i=i_first,i_last)
635  enddo
636
637  c     write(99,*), 'ts: '
638  c     do j=j_last,j_first,-1
639    write(99,*), (ts(i,j),i=i_first,i_last)
640  enddo
641
642  c     write(99,*), 'il = i_first-2
643  c     j1 = j_first-2
644  c     j2 = j_last+2
649 c       if(i1.lt.1) i1 = 1
650 c       if(i2.gt.imx) i2 = imx
651 c       if(j1.lt.1) j1 = 1
652 c       if(j2.gt.jmx) j2 = jmx
653
654 c       call extrap3d(ts, i1, i2, j1, j2, imx, jmx, kmx)
655 c       call extrap3d(u, i1, i2, j1, j2, imx, jmx, kmx)
656 c       call extrap3d(v, i1, i2, j1, j2, imx, jmx, kmx)
657
658 c       write(99,*)' ts after extrapolation: '
659 c       do j=j_last,j_first,-1
660 c         write(99,*) (ts(i,j),i=i_first,i_last)
661 c       enddo
662
663 c       do i=i_first,i_last
664 c       do j=j_first,j_last
665 c         if(msk(i,j).eq.1) then
666 c           do k=1,kmx
667 c             if(i.lt.imx) u(i,j,k) = (u(i,j,k) + u(i+1,j,k)) / 2.
668 c             if(j.lt.jmx) v(i,j,k) = (v(i,j,k) + v(i,j+1,k)) / 2.
669 c           enddo
670 c         endif
671 c       enddo
672 c       enddo
673 c     endif
674 c     enddo
675
676 c--------------------------------------------------------------------
677 c     write medslik files
678 c--------------------------------------------------------------------
679
680 open(20,file = outfile)
681 write(20,*)'ROMS forecast data for //'prdate
682 write(20,*)'Subregion of the Mediterranean with limits:'
683 write(20, '(4f9.5,2i5,''   Geog. limits'')')
684 & alon1,alon2,alat1,alat2,imax,jmax
685 write(20, '(i6,'' 0.0'')') nwp
686 heads = ' lat lon SST ' //
687 & ' u_srf v_srf u_10m v_10m'/
688 & ' u_30m v_30m u_120m v_120m'
689 write(20, '(a150)') heads
690
691 c MEDSLIK II uses as inputs the currents at the following depths: 10,
692 c 30 and 120 meters. Therefore, it is necessary to transform the S
693 c coordinates into z coordinates. The ocean_his file has all the
694 c parameters needed to obtain z(imx,jmx,kmx):
695
696 sc_r(1)=-0.95
697 sc_r(2)=-0.85
698 sc_r(3)=-0.75
699 sc_r(4)=-0.65
700 sc_r(5)=-0.55
701 sc_r(6)=-0.45
702 sc_r(7)=-0.35
703 sc_r(8)=-0.25
704 sc_r(9)=-0.15
705 sc_r(10)=-0.05
706
707 cs_r(1)=-0.972
708 cs_r(2)=-0.930
709 cs_r(3)=-0.882
710 cs_r(4)=-0.786
711 cs_r(5)=-0.588
712 cs_r(6)=-0.322
713 cs_r(7)=-0.128
714 cs_r(8)=-0.0417
715 cs_r(9)=-0.0116
716 cs_r(10)=-0.00198
717
718 start_h(1) = 1
719 start_h(2) = 1
count_h(1) = imx
count_h(2) = jmx

Status = nf_get_vara_real (id, idh, start_h,
  count_h, h)
call handle_err(Status)

do i=i_first,i_last
do j=j_first,j_last
  if(mask(i,j).eq.1) then
    blon = oplon(i)
    blat = oplat(j)
    sst = ts(i,j,kmx)
    us = u(i,j,kmx)
    vs = v(i,j,kmx)
    do l=1,imx
      do m=1,jmx
        do n=1,kmx
          z(l,m,n)=((hc*sc_r(n)+h(l,m)*cs_r(n))*h(l,m))/(hc+h(l,m))
        enddo
      enddo
    enddo
    if (z(i,j,1).ge. -10) then
      u10=u(i,j,1)
      v10=v(i,j,1)
    else
      do k=1,kmx
        if (z(i,j,k+1).gt. -10) then
          u10=(u(i,j,k)+u(i,j,k+1))/2
          v10=(v(i,j,k)+v(i,j,k+1))/2
          exit
        endif
      enddo
    endif
  endif
  if (z(i,j,1).ge. -30) then
    u10=u(i,j,1)
    v10=v(i,j,1)
  else
    do k=1,kmx
      if (z(i,j,k+1).gt. -30) then
        u30=(u(i,j,k)+u(i,j,k+1))/2
        v30=(v(i,j,k)+v(i,j,k+1))/2
        exit
      endif
    enddo
  endif
  if (z(i,j,1).ge. -120) then
    u10=u(i,j,1)
    v10=v(i,j,1)
  else
    do k=1,kmx
      if (z(i,j,k+1).gt. -120) then
        u120=(u(i,j,k)+u(i,j,k+1))/2
        v120=(v(i,j,k)+v(i,j,k+1))/2
        exit
      endif
    enddo
  endif
write(20,'(13f11.4)') blat,blon,sst,us,vs,
  & u10,v10,u30,v30,u120,v120
endif
enddo
enddo
enddo
endif
enddo
enddo
enddo
enddo
end
return
end

c******************************************************************************
c******************************************************************************
c******************************************************************************
SUBROUTINE HANDLE_ERR(Status)
include 'netcdf.inc'
c include '/usr/local/include/netcdf.inc'
INTEGER Status
IF (Status .NE. NF_NOERR) THEN
  PRINT *, NF_STRERROR(Status)
  STOP 'Stopped'
ENDIF
END