MEDSLIK-II version 2.0 – List of changes and complementary manual

Antonio Augusto Sepp Neves, University of Bologna, Italy

July 2018

1 Introduction

MEDSLIK-II was originally conceived to carry out deterministic, short term simulations supporting oil spill emergency situations. The oil spill field has evolved and ensemble forecasting is now a reality and a must. The new MEDSLIK-II applications have generated new demands, mostly related to improving the model performance, delivering rawer outputs and running simulations outside the Mediterranean sea. This annex is aimed at describing the recent changes applied to MEDSLIK-II v1.01 (available at the model website) attending some of the new demands in the oil spill field. The modified code, hereinafter MEDSLIK-II v2.0, is now able to:

- ingest global meteo-oceanographic inputs
- ingest global bathymetry and coastlines
- generate netCDF4 outputs

More information about each of these items above are described in the following sections. On the other hand, some of the functions present in version 1.01 have been removed:

- preparation for specific meteo-oceanographic inputs (e.g. MFS, CYCOFOS, ECMWF winds,etc.)
- graphical outputs
- ASCII format outputs

2 MEDSLIK-II installation

MEDSLIK-II installation should be relatively simple. Firstly, certify yourself that all the system requirements have been fulfilled. Secondly, follow the few installation steps in order to get your model compiled and working on you computer. Finally, test your installation with the case study made available at the MEDSLIK-II website. For some extra information on the case study, see Section 7.

2.1 System requirements

The model currently runs on Linux with a couple of software requirements:

- gfortran
  - sudo apt-get install gfortran
- netCDF4 library
  - sudo apt-get install netcdf-bin
  - sudo apt-get install libnetcdf-dev

2.2 Installation

- download the model and put the tarball in your home directory.
- extract the contents of the tarball: tar –xvzf MEDSLIK_II_v2.0.tar.gz MEDSLIK_II_v2.0
- open /home/user/MEDSLIK_II_v2.0/EXE/source/compile.sh and give the correct address of your netCDF installation
- now go to /home/user/MEDSLIK_II_v2.0/EXE and compile the code: sh source/compile.sh
- the model should be compiled now. Two last steps:
  - go to /home/user/MEDSLIK_II_v2.0/EXE/medslik_II.sh and put the correct MEDSLIK installation folder to the HOME_MEDSLIK variable
  - repeat the same procedure for /home/user/MEDSLIK_II_v2.0/EXE/RUN.sh

3 MEDSLIK-II meteo-oceanographic inputs

Unlike observed in the previous versions, MEDSLIK-II v2.0 supports one single type of ocean fields and one type of atmospheric fields. The end user will be responsible for translating his/her input data into MEDSLIK format and he/she is strongly encouraged to perform it outside the Fortran code (**Extract.for**). Two Python scripts were prepared to translate MERCATOR currents fields (**preproc_currents.py**) and ERA-Interim atmospheric fields (**preproc_winds.py**), which can be found at the model webpage.

In case the user opts for using the Python scripts available, we suggest installing Python Anaconda which includes all libraries necessary to pre-process the meteo-oceanographic fields. The following Python libraries are needed for the pre-processing:

- netCDF4 (http://unidata.github.io/netcdf4-python/)
- Scipy (https://www.scipy.org)

Subversion: In case the user wishes to employ different data sources, he/she is advised to check the format of the processed meteo-oceanographic netCDF files. Things should work accordingly if the structure, variable names and variable characteristics (e.g. units, order) are respected.

Ocean fields are divided into three different files (meridional current, zonal current and sea surface temperature). The user is expected to deliver currents in four different depths: 0m, 10m, 30m and 120m.

- Temperature
  - `File name:` MDK_ocean_yymmdd_T.nc, where *yy* correspond to the last two digits of the year, *mm* month and *dd* day. Files are daily containing hourly fields.
  - `Dimensions:` *time_counter* for time, *y* for latitude and *x* for longitude, and *depth_t* for depth. Latitude starting from the southernmost grid point. Longitude starting from the westernmost point.
  - `Variables:` *nav_lat* and *nav_lon* for latitude and longitude fields dependent on the *(y)* and *(x)* dimensions, respectively. The temperature fields are included in *votemper* which depends on *(time_counter, depth_t, y, x)*.
  - `Temporal resolution:` hourly starting from 13:00.
  - `Missing value:` 9999
  - `Units:` degrees Celsius
- Zonal currents
  - `File name:` MDK_ocean_yymmdd_U.nc, where *yy* correspond to the last two digits of the year, *mm* month and *dd* day. Files are daily containing hourly fields.
  - `Dimensions:` *time_counter* time, *y* for latitude and *x* for longitude, and *depth_u* for depth. Latitude starting from the southernmost grid point. Longitude starting from the westernmost point.
  - `Variables:` *nav_lat* and *nav_lon* for latitude and longitude fields dependent on *(y,x)* dimensions. The current fields are included in *vozocrtx* which depends on *(time_counter, depth_u, y, x)*.
  - `Temporal resolution:` hourly starting from 13:00.
  - `Missing value:` 9999
  - `Units:` m/s

- Meridional currents
  - `File name:` MDK_ocean_yymmdd_V.nc, where *yy* correspond to the last two digits of the year, *mm* month and *dd* day. Files are daily containing hourly fields.
  - `Dimensions:` *time_counter* time, *y* for latitude and *x* for longitude, and *depth_t* for depth. Latitude starting from the southernmost grid point. Longitude starting from the westernmost point.
  - `Variables:` *nav_lat* and *nav_lon* for latitude and longitude fields dependent on *(y,x)* dimensions. The current fields are included in *vomecrty* which depends on *(time_counter, depth_v, y, x)*.
  - `Temporal resolution:` hourly starting from 13:00.
  - `Missing value:` 9999
  - `Units:` m/s

Wind fields are given using a single file containing meridional and zonal 10m winds in m/s. Currently, the model ingests hourly resolution wind fields. The wind files are organized as follows:

- `File name:` YYYYmmdd.nc, where *YYYY* correspond to all the four digits of the year, *mm* month and *dd* day. Files are daily.
- `Dimensions:` *time* for time, *lat* for latitude and *lon* for longitude. Latitude starting from northernmost grid point. Longitude starting from westernmost point.
- `Variables:` *lat* and *lon* for latitude and longitude fields dependent on *(lat)* and *(lon)* dimensions, respectively. The wind fields are included in *U10M* and *V10M* and depend on *(time,lat,lon)*.
- `Temporal resolution:` hourly starting from 00:00.

4 Bathymetry and coastline generation

MEDSLIK-II allows users to generate their own bathymetry and coastline files for any part of the globe and, of course, two Python scripts are available at the model webpage doing this trick in a few seconds. From now on, bathymetries in MEDSLIK-II are based on the GEBCO database with original 30" resolution and coastline segments are based on the NOAA GSHHS full resolution data set.

4.1 Preparing your GEBCO-based bathymetry

The script **bathymetry_mdk2.py** generates the bathymetry file for your experiment based on the already prepared ocean fields. As long as you have the global GEBCO 30" bathymetry netCDF file in your computer (GEBCO_2014 Grid dataset, global grid, 1D netCDF found at:

https://www.gebco.net/data_and_products/gridded_bathymetry_data/), you should be able to generate a bathymetry subset for your experiment coincident with the current fields grid. The script will be able to load it and generate a **.bath** file that should be placed in your *MEDSLIK_II/EXE/data/* folder. Do not change the file name (**medf_.bath**), this is the best approach and we strongly encourage the user to stick to it.

Subversion: considering the wide range of areas and scales where oil spills can take place, it is very likely that GEBCO data will not fit all the scenarios or it may not be the most applicable solution. In those cases, the user is encouraged to use different data sources and it should not be that hard to build his/her own bathymetry file as long as the following guidelines are respected:

- be sure to name it **medf_.bath**
- follow the file structure:
  - line 1: a good-enough description of your bathymetry file. For instance, "GEBCO 30sec - derived bathymetry for the Patagonian shelf"
  - line 2: [minlong] [maxlong] [minlat] [maxlat] (space between variables with seven digits - 4 decimals - (in degrees))
  - line 3: [nx] [ny] (number of columns (E-W) and rows (S-N) in your source 2D bathymetry field)
  - following lines: [depth value] (positive for values below the water line) with no decimals (4 digits). In case of land, put 9999. The bathymetry vector is based on the 2D bathymetry fields looping like:

```
for i in n_rows:
      for j in n_columns:
            bat1d.append(bat2d[i,j])
```

For further understanding, check the procedure done in the Python script available in our website.

4.2 Preparing your GSHHS-based coastline

The Python script **coastline_mdk2.py** will do the trick. Be sure you have downloaded the correct binary, **gshhs_f.b**, from the NOAA GSHHS website (https://www.ngdc.noaa.gov/mgg/shorelines/data/gshhg/latest/) and that the path given to the python script is correct. The geographical limits of your coastline file will be based on your ocean fields. Therefore, certify yourself to have correctly informed the path to the python script. A **.map** file will be generated and it should be placed in *MEDSLIK_II/EXE/data/* folder. Once again, do not change the file name.

Depending on the spatial scale of your study and on the number of simulations you are expected to perform, you may consider whether to use the full resolution file (more points and improved representation of the coastline and longer simulations) or high/intermediate (less points, poorer representation of the coastline but faster simulations) resolution.

Subversion: Very high resolution applications are becoming more and more common in oceanography and, just like in the bathymetry case, the user may be interest in using his/her own coastline map in MEDSLIK-II. Again, it should not be that hard to build his/her own coastline file as long as the following guidelines are respected:

- be sure to name it **medf.map**
- be sure your polygons never remain open.
- be sure it follows the file structure:
  - line 1: [number of coastline polygons, p, in your study area - 4 digits]
  - line 2: [number of points, n, forming the polygon p_1] [0]
  - line 3: [p_{1_x}] [p_{1_x}] - 8 digits with 5 decimals
  - line 4: [p_{2_x}] [p_{2_x}] - 8 digits with 5 decimals
  - line 3+n: [p_{n_x}] [p_{n_x}] - 8 digits with 5 decimals
  - line 3+n+1:[number of points, n, forming the polygon p_2] [0]
  - repeat procedure writing polygon points
  - this sequence should be repeated for all the polygons in your study area

For further understanding, check the procedure done in the Python script available in our website.

5 MEDSLIK-II netCDF4 outputs and processing scripts

MEDSLIK-II outputs have changed. The old ASCII files have been replaced by a single netCDF4 file allowing faster writing, taking less space and, mainly, containing much more information. The file, so called spill_properties.nc, contains information relative to the evolution of each oil parcel at each time step. The output file is built as follows:

- `File name:` spill_properties.nc
- `Dimensions:` *parcel_id* refers to the parcel number and *time* refers to simulation time steps.
- `Variables:`
  - *latitude:* latitude of each oil parcel (degrees)

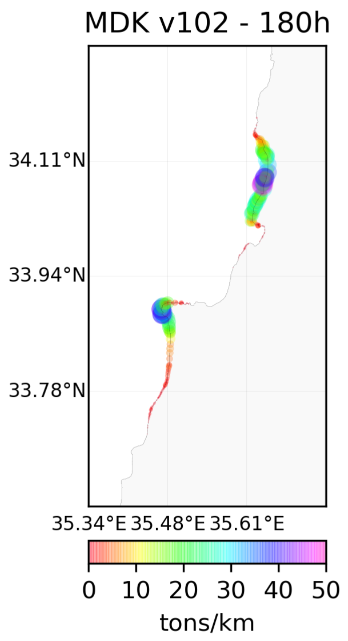
- *longitude:* longitude of each oil parcel (degrees)
- *evaporative_volume:* parcel volume relative to the evaporative component of the oil (cubic meters)
- *non_evaporative_volume:* parcel volume relative to the non-evaporative component of the oil (cubic meters)
- *water_fraction:* percentage of water in each parcel (%)
- *particle_status:*
  - 0: not released
  - 1: released and spreading
  - 2: released and not spreading
  - 3: released and dispersed
  - 4: bottom
  - -n = parcel is beached. "n" corresponds to the total volume of oil seeped at the coastline.
  - 9: beyond boundary limits
- *time:* time index
- *total_fixed_oil:* total amount of oil beached and fixed at the coast (tonnes)
- *viscosity_emulsion_1:* viscosity emulsion water-oil for mini-spill 1($Pa.s$)
- *viscosity_oil_1:* viscosity oil for mini-spill 1($Pa.s$)
- *density_emulsion_1:* density water-oil for mini-spill 1 (kg/m^3)
- *water_fraction_1:* percentage of water in the emulsion for mini-spill 1 (%)
- *viscosity_emulsion_2:* viscosity emulsion water-oil for mini-spill 2 ($Pa.s$)
- *viscosity_oil_2:* viscosity oil for mini-spill 2 ($Pa.s$)
- *density_emulsion_2:* density water-oil for mini-spill 2 (kg/m^3)
- *water_fraction_2:* percentage of water in the emulsion for mini-spill 2 (%)
- *volume_ratio:* volume ratio water/oil

- `Temporal resolution:` depends on the time step defined *a priori*.

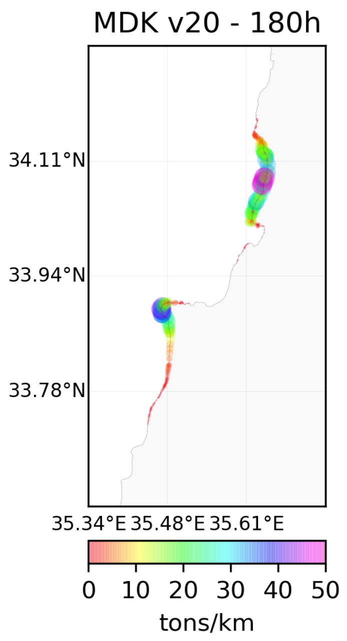
The automatic generation of graphic outputs has been removed from the latest version of MEDSLIK-II. Instead, the end user will now find two Python scripts able to plot the surface/dispersed (**oil_track.py**) and beached (**oil_beached.py**) oil at each time step. The Python scripts and the netCDF file are self-explanatory and follow the terminology used in DeDomenicis et al., (2013).

6 Consistency check

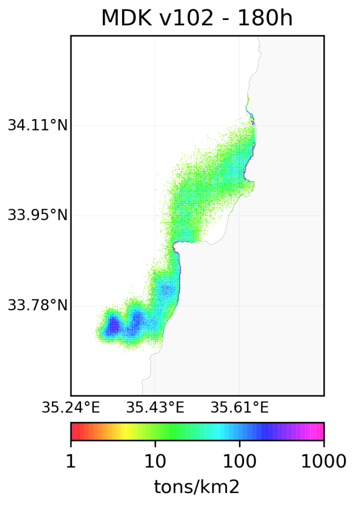
The new MEDSLIK-II source code does not include changes in the weathering and advection-difusion components of MEDSLIK-II v2.0. Consequently, we should expect results obtained by versions 1.01, 1.02 and 2.0 to be equivalent. Taking the Lebanon spill case study as our reference, we compared the trajectory of the spill and beached oil concentrations obtained by the two version of the model (Figure 1). As expected, the new model version predicted results equivalent to MEDSLIK-II v1.02 (the latest version available online) demonstrating its consistency.



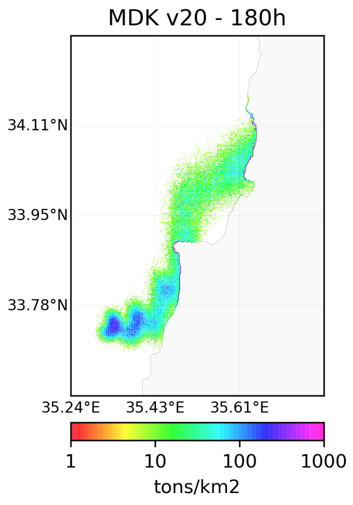
(a)



(b)



(c)



(d)

Figure 1: Beached oil concentrations after 180h of simulation for the Lebanese coast (in tons/km) by versions (a) 1.02 and (b) 2.0.

7 Test case for MEDSLIK-II v2.0

A simple test case was prepared to find out if the model installation was successful. The 120h-long simulation reproduces an oil spill off the Venezuelan coast (see Table 1 for further information on the spill characteristics and MEDSLIK-II setup) and, in order to run the test case, proceed as follows:

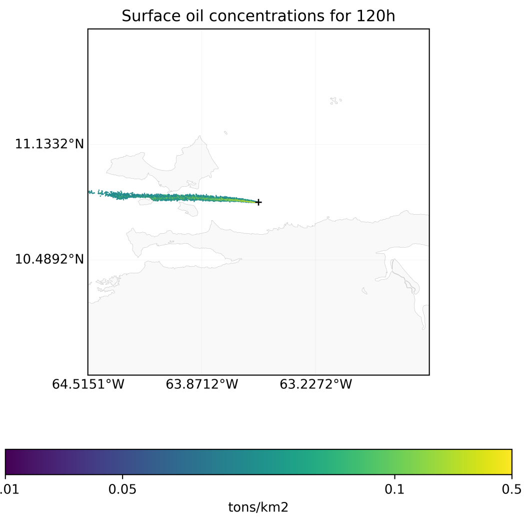
- unzip the test case file **paria_casestudy.tar.gz**
- copy all files found in *xp_files* inside the *RUN* folder of your MEDSLIK-II installation.
- copy the netCDF files found inside the *oce_files* directory into *MEDSLIK_II_v2/DATA/fcst_data/H3k*
- copy the netCDF files found inside the *met_files* directory into *MEDSLIK_II_v2/DATA/fcst_data/SK1*
- copy all files found in *bnc_files* into *MEDSLIK_II_v2/DATA/RUN/data*
- compile the code (in case you have not done it yet)
- go to *MEDSLIK_II_v2/DATA/RUN* and run the model with **./RUN.sh**

That is everything you should do. Feel free to try the post-processing Python scripts (**oil_track_mdk2.py** to plot the oil trajectory and **oil_beached_mdk2.py** to plot the beached oil evolution) with your new results and, if you feel confident, change them in order to obtain the best of your outputs!

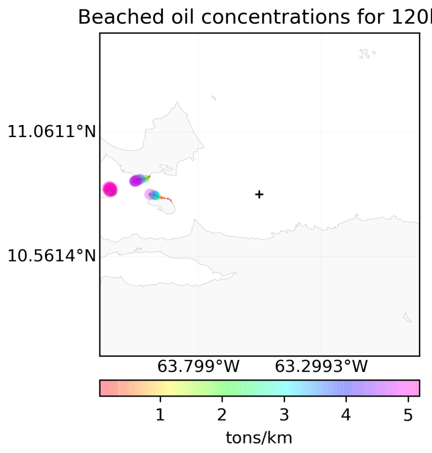
Spill time	24/04/2017 13:00
Spill position	10.811N -63.55W
Spill duration	120h
Spill volume	180 ton
Oil API	28
Simulation length	120h
Ocean fields	CMEMS GLOBAL_ANALYSIS_FORECAST_PHY_001_024
Wind fields	ERA-Interim
Stokes drift	MEDSLIK-II JONSWAP formulation
Number of parcels	25,000

Table 1: MEDSLIK-II setup applied in the test case

For those who opted for using the Python scripts available to generate the graphic outputs, your results should be very similar to Figure 2.



(a)



(b)

Figure 2: Surface and beached oil concentrations after 120h of simulation for Venezuelan test case. Surface concentrations in $tons/km^2$ and beached oil concentrations in $tons/km$. Black cross marks the origin of the spill.