TrackMPD v.1

A 3D numerical model to Track Marine Plastic Debris

QUICK-START GUIDE

Isabel Jalón-Rojas

TrackMPD 2019

Erick Fredj

1. About this guide

1.1. Purpose

The main purpose of this Quick-Start Guide is to enable you to use the TrackMPD modelling framework v.1, for applications involving tracking of marine plastic debris. The performance of the framework has been tested in a variety of applications. Future applications, however, might reveal errors that were not detected in the test simulations. Users are kindly requested to send notification of any errors found in the code.

TrackMPD v.2 will be released soon, together with a full user manual and description of the code (Fredj et al., in prep.).

1.2. Assumed user Background

TrackMPD is developed for MATLAB. We recommend installing the M_MAP toolbox or the function *m_fdist* of this toolbox before using TrackMPD.

Running TrackMPD requires very little knowledge of MATLAB or programming in general. However, proper use of the model requires knowledge of physical oceanography. This guide is not intended as a substitute for basic knowledge of this area.

Users might wish to use TrackMPD with a specific Ocean General Circulation Model (OGCM) other than POM and FVCOM, described below; this requires basic knowledge of MATLAB. The authors are happy to help users make TrackMPD compatible with other OGCMs.

1.3. Terms of use

When using TrackMPD in any scientific publication, technical report or otherwise formal writing, please cite our paper:

Jalón-Rojas, I., Wang, X.H., Fredj, E., 2019. *"A 3D numerical model to Track Marine Plastic Debris (TrackMPD): Sensitivity of microplastic trajectories and fates to particle dynamical properties and physical processes"*. Marine Pollution Bulletin, 141, 256-272.

Additionally, you may refer to this manual as Jalón-Rojas and Fredj (2019). *TrackMPD v.1. Quick-Start Guide.*

The TrackMPD code is licensed under LGPL (GNU Lesser General Public License). In summary, this means that the code is open source and may be used freely for non-commercial and commercial purposes. Any alterations to the TrackMPD source code or new modules must be licensed under LGPL as well.

2. TrackMPD

TrackMPD is a three-dimensional particle-tracking model for the transport of marine plastic debris in oceans and coastal systems. It can compute forward and backward trajectories in two or three dimensions. The power of TrackMPD lies in: (1) its compatibility with diverse formats of current-velocity inputs; and (2) its ability to extend the Lagrangian modelling of advection-diffusion by adding more-complex and realistic particle behaviours and physical processes, which can either be included or excluded depending on the application. TrackMPD v.1 can include beaching, washing-off, degradation, biofouling, sinking and deposition. In particular, sinking and deposition depend on particle behaviour, which in turn depends on the particle density, size, shape, fouling state and degradation state. The model can incorporate new processes and behaviours, and change the implementation of already existing ones, with new experimental findings or particular applications.

The model consists of a set of coupled and mutually interacting modules. The modules are independent functions or classes that define behaviours, read the inputs from a given source, implement a given physical process or perform auxiliary tasks such as creating outputs. This allows independent development of modules that can be easily added to the model without the need to change the other modules.

Readers should refer to **Jalón-Rojas et al. (2019)** for a detailed description of the modular structure, equations and solution methods of TrackMPD v.1.

3. Getting started

3.1. Primary Framework Components

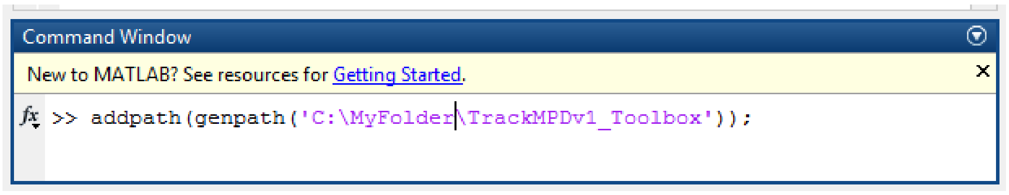
TrackMPD modelling framework v.1 has three main components:

- ***run.m***: Users can run the model using this script or directly from the MATLAB Command Window. See details in Section 3.2.
- ***input_conf.m***: TrackMPD v.1 works with input provided in a function with the format of *input_conf.m*. See details in Section 3.3.
- **TrackMPDv1_Toolbox**: This toolbox contains all the modules of the model. They are classified into the following packages/subfolders:
    - Advection
    - Behaviour
    - Dispersion
    - TRAJstruct (outputs)
    - TransformInputs
    - Windage
    - runTrackMPD

  The modules forming TrackMPD are not described in this short guide. Users should refer to **Jalón-Rojas et al. (2019**) for further description of the modular structure**.** A full user manual with a description of the modules will be released with Version 2.

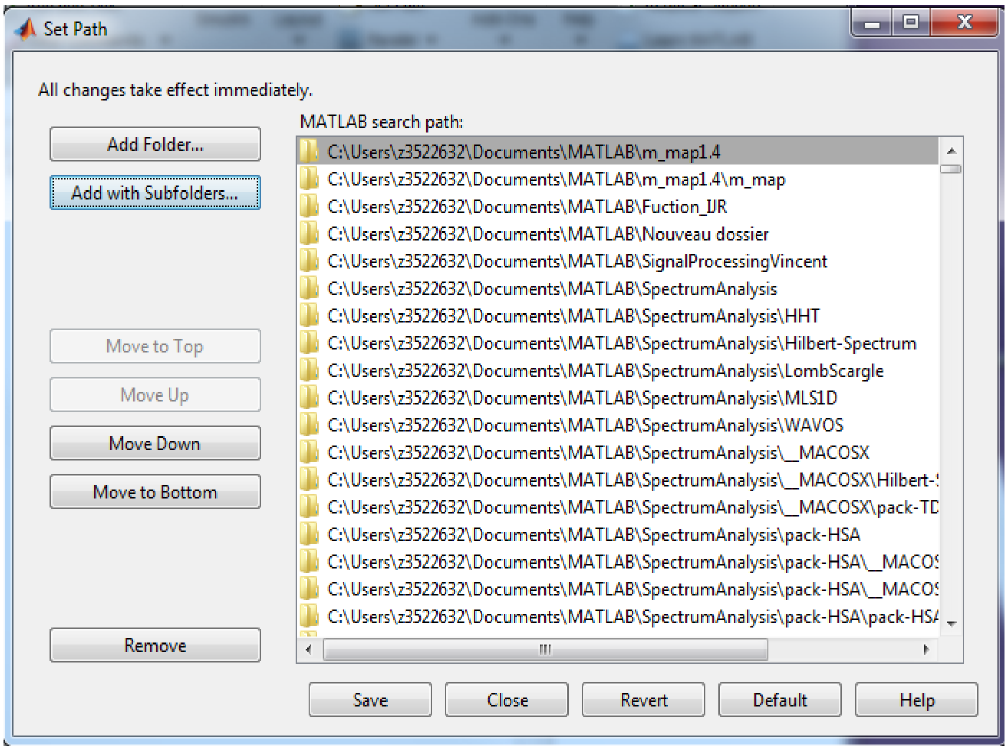
  To use this toolbox, users must add the toolbox directory to the MATLAB search plan. There are several options to do this:

    - Typing the following command in the MATLAB command window:



```
>> addpath(genpath('C:\MyFolder\TrackMPDv1_Toolbox'));
```

- Using the Set Path dialog box. To open it, select Set Path from the Home menu in the MATLAB desktop.



```
C:\Users\z3522632\Documents\MATLAB\m_map1.4
C:\Users\z3522632\Documents\MATLAB\m_map1.4\m_map
C:\Users\z3522632\Documents\MATLAB\Fuction_JJR
C:\Users\z3522632\Documents\MATLAB\Nouveau dossier
C:\Users\z3522632\Documents\MATLAB\SignalProcessingVincent
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\HHT
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\Hilbert-Spectrum
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\LombScargle
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\MLS1D
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\WAVOS
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\__MACOSX
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\__MACOSX\Hilbert-
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\__MACOSX\pack-TD
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\pack-HSA
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\pack-HSA\__MACOS
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\pack-HSA\__MACOS
C:\Users\z3522632\Documents\MATLAB\SpectrumAnalysis\pack-HSA\pack-HSA
```

Click *Add with Subfolders* ⟶ select the *Toolbox* folder ⟶ click *Save* ⟶ click *Close*.

3.2. Running a simulation

TrackMPD should be run in two steps:

1. Transform the hydrodynamic input files (OGCM outputs) to an “standard” TrackMPD input format. This step is implemented using the function:

   ```
   >> transformInputs(conf_name)
   ```

2. Run the tracking model using the function:

   ```
   >> runTrackMPD(conf_name)
   ```

   where *conf_name* is the name of the input file (e.g. `conf_name='inputs_conf';` Section 3.3).

The script **run.m** includes these two step. Users can run the whole file or select one of the two steps. Step 1 must be performed when the output of a given OGCM simulation is used for the first time as inputs to TrackMPD. Once Step 1 has been performed once, Step 2 can be performed multiple times using different model settings. The model can be also run by using the function *transformInputs* and *runTrackMPD* in the MATLAB command window.

The present version provides compatibility with POM and FVCOM outputs. However, the model can be easily adapted to new OGCMs. It supports velocities defined in rectangular Arakawa A and C grids, unstructured grids, and in depth (z) and sigma (σ) vertical coordinates. See Section 4 for details about the model extendibility to other OGCMs.

3.3. Input

TrackMPD v.1 needs three input files:

- **OGCM inputs** (see Section 4 for details)
- The **initial particle location** is defined from a .cvs file which contains three columns: longitude; latitude; depth (in meters).
- The **study domain** is defined from a .dat or .txt file which contains two columns: longitude and latitude of the coastline.
- All the other inputs are defined in the function ***input_conf.m***. Users can change the name of this function for different applications, but the structure and the name of the variables included in this file (described below) should be preserved.

It is recommended to save the OGCM inputs, the particle location file and the study domain file in a folder (e.g. InputData). It is also recommended to create a folder to save the model outputs (e.g. Outputs).

3.3.1. Structure and variables of *inputs_conf.m*

The structure and variables of the main input file are described below. Users can also refer to the examples provided with TrackMPD v.1 for a better understanding of the input variables.

a) Input data

conf.Data.BaseDir	path of the folder that contains the particle and domain input files
conf.Data.Domain	name of the “domain file” (full path)
conf.Data.ParticlesFile	name of the particle location file (full path)

b) Ocean General Circulation Model (OGCM)

Conf.OGCM.Model_name	name of the OGCM
conf.OGCM.DomainName	name of the domain
conf.OGCM.BaseDir	path of the folder that contains the OGCM inputs
conf.OGCM.TimeStep	time step of the OGCM inputs in days
conf.OGCM.VerticalLayer	type of vertical layer: — *Rectangular* — *Sigma2depthVar* : sigma layer; depth not constant over time, dependent on water elevation — *Sigma2depthCte* : sigma layer; depth constant over time

Then, specify the variables of the selected OGCM.

For POM (SARCCM version):

conf.OGCM.POM_Prefix	prefix of POM outputs
conf.OGCM.POM_Suffix	suffix of POM outputs
conf.OGCM.t0	POM reference time variable (date of the first time step)
conf.OGCM.Hmin	depth for land grid points

For FVCOM:

conf.OGCM.FVCOMFile	FVCOM output file
conf.OGCM.FVCOMGrid	gdr file used as input to FVCOM
conf.OGCM.NumLonGrid	number of points in the new rectangular grid (longitude dimension)
conf.OGCM.NumLatGrid	number of points in the new rectangular grid (latitude dimension)
conf.OGCM.NameTime	Name of the variable time in conf.OGCM.FVCOMFile (i.e. ´time´or ´t´)
conf.OGCM.NameW	Name of the variable “vertical velocity” in conf.OGCM.FVCOMFile (i.e. ´w´or ´ww´)

c) Trajectory

Mode

conf.Traj.Mode	‘2D’ or ‘3D’ approach for tracking MPD

Time parameters

conf.Traj.ReleaseTime	date of particle release: datenum(y,m,d,H,M,S)
conf.Traj.TrajectoryDuration	trajectory duration in days
conf.Traj.TimeStep	time step for the trajectory output
conf.Traj.Direction	*forward* or *backward*

Output

conf.Traj.ScenarioName	name of the output file (free choice)
conf.Traj.BaseDir	path of the output folder

Chunk method

To avoid memory problems, the time domain is divided into different partitions (chunks) for calculation purposes. Necessary for long simulations and/or high-resolution grids.

conf.Traj.chunklen	duration of each partition in days (e.g. 5 days)

Dispersion

conf.Traj.Kh	Horizontal diffusion coefficient (m^2/s)
conf.Traj.Kv	Vertical diffusion coefficient (m^2/s). Only for 3D model.

Beaching

conf.Traj.Beaching	*yes* : particle beaching is considered *no* : particle beaching is not considered

Washing off

conf.Traj.Refloating	*yes* : particle washing off considered

	no : particle washing off not considered

If conf.Traj.Refloating = *yes*, specify the following three variables:

conf.Traj.Tw	half-life of particles remaining on the beach before washing off again (in days)
conf.Traj.xcoord_elev	grid cell (x-coord) to calculate the time series of water elevation
conf.Traj.ycoord_elev	grid cell (y-coord) to calculate the time series of water elevation

d) Behaviour (3D mode)

BehaviourType	Select the type of behaviour: 1: macro or microplastic, plastic density <1 g/cm^3, no biofouling, no degradation 2: macroplastic or microplastic with a known ws, plastic density >1 g/cm^3, no biofouling, no degradation 3: microplastic (defined by shape, size and density), plastic density >1 g/cm^3, no biofouling, no degradation 4: microplastic (defined by shape, size and density), biofouling (constant parameters over time), no degradation 5: microplastic (defined by shape, size and density), biofouling (no constant parameters over time), no degradation

6: microplastic (shape,size,density), no biofouling, degradation

Specify the variables of the selected behaviour. These variables may include:

conf.beh.PolymerDensity	polymer density (g/cm^3)
conf.beh.Ws	settling velocity (m/s)
conf.beh.PolymerType	polymer type, e.g. *polystyrene*, *high-density polyethylene*, *polypropylene*
conf.beh.PolymerAcronym	polymer acronym, e.g. *PS*, *HDPE*, *PP*
conf.beh.Category	*Sphere* or *LongCylinder* or *ShortCylinder*
conf.beh.Shape	*Sphere(R)* or *Cylinder(R,L)* where *R* is the radius (m) and *L* is the length (m)
conf.beh.WaterDensity	water density (g/cm^3)
conf.beh.WaterViscosity	water viscosity (m^2/s)
conf.beh.BiofoulingThickness	biofilm thickness (m)
conf.beh.FilmDensity	biofilm density (g/cm^3)
conf.beh.BiofoulingThickness0	initial biofilm thickness (m)
conf.beh.BiofoulingRate	biofuling rate (m/day)
conf.beh.Shape0	initial shape: *Sphere(R)* or *Cylinder(R,L)* where *R* is the radius (m) and *L* is the length (m)

conf.beh.DegradationRate	percentage size decrease per day

An example of an input file (info_conf.m) is provided with the code.

4. Include a new OGCM

TrackMPD v.1 reads the OGCM inputs in a specific format. To use the outputs of a given OGCM as TrackMPD inputs, their format should be changed to this specific format in Step 1 of running TrackMPD (Section 3.1). TrackMPD v.1 is compatible with POM (Arakawa C rectangular grid, sigma layer) and FVCOM (unstructured grid, sigma layer) outputs, but it can be adapted to a new OGCM easily. The steps to include a new OGCM option in TrackMPD v.1 are:

1. Create a function *transformOGCMNAMEinputs_3D* and save it in the folder *TransformInputs* of the toolbox. Users should use the functions *transformPOMinputs_3D* or *transformFVCOMinputs_3D* as a guide. Instruction for this step are given in Section 4.1.
2. Include the option of using the new OGCM in the function *TransformInputs.m* :

```
elseif strcmpi(OGCMmodel_name,'OGCMNAME') && strcmpi(Mode,'3D')
 transformOGCMNAMEinputs_3D(conf_name)
```

3. Include the variables that you need to read the OGCM files in *info_conf.m*.

Users are invited to contact the authors for support on adding new OGCMs to TrackMPD.

4.1. Creating *transformOGCMNAMEinputs_3D*

The function *transformOGCMNAMEinputs_3D.m* should read the original OGCM files, then create three types of MAT files (*grid.mat*, *timestamp.mat* and various *TrackMPDInputX.mat files*). To read the OGCM files, the function can use some variables defined in input_conf.m such as the name of the file/s and the name of some variable/s. The three types of MAT files are the TrackMPD input files that will be read in Step 2 of running TrackMPD (Section 3.1). These input files should have the following format:

1. *grid.mat* contains the grid information in four variables:
   a. *Lat* : a vector with the grid latitude;
   b. *Lon* : a vector with the grid longitude;
   c. *BottomDepth* : a matrix (lat x lon) with the bottom depth (m) at each grid point;
   d. *water_mask* : a matrix (lat x lon) with values equal to 1 for water, 0 for land.
2. *Timestamps.mat* contains the time information in one variable:
   a. *timestamps* : a vector with the date of the different TrackMPDInputX.mat files. For example, the first value corresponds to the date of TrackMPDInput1.mat, the second value to the date of TrackMPDInput2.mat, and so on.
3. *TrackMPDInputX.mat* is a set of files containing the hydrodynamic information. Each file contains the information at a given time stamp of the OGCM simulation. The number of files (or time stamps) is equal to the duration of the OGCM simulation divided by the time step. For example, for a

OGCM simulation of 1 day with a time step of 1 hour (1/24 days), *transformOGCMNAMEinputs_3D* would create 24 files.

Each *TrackMPDInputX.mat* file contains 7 variables:

a. *u* : a matrix (lat x long x layers) with the horizontal velocity (longitude direction; m/s) at each grid point;
b. *v* : a matrix (lat x long x layers) with the horizontal velocity (latitude direction; m/s) at each grid point;
c. *w* : a matrix (lat x long x layers) with the vertical velocity (m/s) at each grid point;
d. *depth* : a matrix (lat x long x layers) with the depth of each grid point. It can vary over time as a function of elevation (vertical layer type: Sigma2depthVar, Section 3.3.1.b) or be constant over time (vertical layer type: Sigma2depthCte, Section 3.3.1.b);
e. *E* : a matrix (lat x long) with the water elevation (m) at each grid point;
f. *time* : single numerical value containing the time stamp;
g. *time_str* : single string value containing the time stamp.

Users should use the functions *transformPOMinputs_3D* or *transformFVCOMinputs_3D* as a guide to create this new function.