

TrackMPD

A 3D numerical model to Track Marine Plastic Debris



TrackMPD

TUTORIAL

TrackMPD v2.3

Isabel Jalón-Rojas

Vincent Marieu

1. About this guide

1.1. Purpose

The main purpose of this tutorial is to enable you to use the TrackMPD modelling framework v.2.3 for applications involving the tracking of particles (and more in particular plastic debris) in marine and coastal systems. It is an updated and enlarged version of the Quick-Start Guide of TrackMPD v1 by Jalón-Rojas and Fredj.

The performance of TrackMPD 2.3 has been tested in a variety of applications. Future applications, however, might reveal some errors that were not detected in the test simulations. Users are kindly requested to send notifications of any potential error found in the code. TrackMPD is available on GitHub, with ongoing updates made available through its version control and change tracking features.

1.2. Assumed user Background

TrackMPD is developed for MATLAB. We recommend the installation of the *Matlab Parallel toolbox* as TrackMPD v2.3 uses it for the parallel computation of particle trajectories.

TrackMPD also uses the function *m_fdist* of the external [M_MAP toolbox](#) (Pawlowicz, 2020). This function is already provided in TrackMPD v2.3 (folder “External”), but users may still install the M_MAP toolbox.

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 cannot be used as a substitute for basic knowledge of this area.

TrackMPD can use velocity data from various sources (offline coupling), such as different ocean general circulation and hydrodynamic models (OGCM). Available applications have already been implemented with POM, FVCOM, TELEMAC, MOHID, MARS and NEMO, among others. Users might wish to use TrackMPD with another hydrodynamic model. Some indications to read and transform velocity data from other models are available in Section 4, but note that this requires more advanced knowledge of MATLAB. The authors are happy to help users make TrackMPD compatible with other circulation models.

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.

Another publication about the resuspension, deposition and bed load processes added in version 2.3 will be published soon.

Additionally, you may refer to this manual as Jalón-Rojas and Marieu (2023). *Tutorial. TrackMPD v2.3*.

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 also be licensed under LGPL.

2. TrackMPD

2.1. Model description

TrackMPD is a three-dimensional particle-tracking model for the transport of marine plastic debris in oceans, coastal, and estuarine 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.2 can include beaching, washing-off, degradation, biofouling, sinking, deposition, resuspension and bedload. In particular, sinking, deposition and resuspension 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 the independent development of modules that can be easily added to the model without the need to change the other modules.

TrackMPD can work on 2DH, 2DV, and 3D modes. Only the 2DH and 3D modes are available in the public toolbox. The 3D mode can be used with all the available transport processes. The 2D(H) mode is typically used for floating particles transported in surface waters so it includes advection and horizontal dispersion. Nevertheless, the code may be easily modified to consider the vertical movement of particles by dispersion and settling, even if the horizontal velocities are considered constant over the water column.

TrackMPD v2.3 supports velocities defined in (spherical or cartesian) rectangular (e.g. Arakawa A and C grids) and unstructured grids, and in depth (z), sigma (σ) and hybrid (z - σ) vertical coordinates, depending on the coordinate system used in the hydrodynamic model. See Section 4 for details about the model extendibility to different OGCMs.

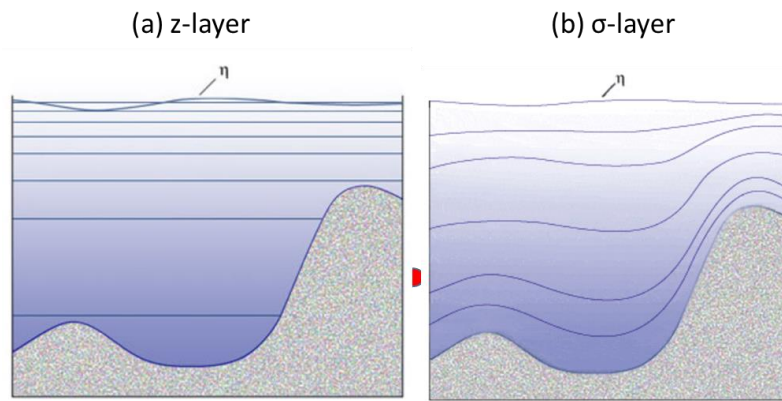


Figure 1. Sketch of z -layer and σ -layer coordinates. Source: <https://www.oc.nps.edu>

2.2 Transport processes

Here we provide a description of the transport processes considered in TrackMPD v2.3. The text is an updated version of the description provided in [Jalón-Rojas et al. \(2019\)](#) for v1.

2.2.1 Advection

Each particle position is directly advected with no drag effect using the velocity field given by the hydrodynamic model. In TrackMPD v2.3, the advection can be computed using the Runge-Kutta method in first order (Euler scheme), 2nd order, or 4th order. For instance, at 1st order,

the new particle position vector \mathbf{x}_{n+1} is obtained from the previous one \mathbf{x}_n , using the velocity field \mathbf{U} at the particle location:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t_i \mathbf{U}, \quad (1)$$

where Δt_i (TimeStepCalc) is TrackMPD internal time step. The velocity field is interpolated in time and space at the particle location. This interpolation is time-consuming, which implies that the advection time step must be chosen carefully, considering the considered physical problem. Higher-order schemes are more accurate but also more time-consuming.

2.2.2. Dispersion

A random-walk model is used to simulate turbulent particle motion in the horizontal (x or y) directions:

$$x_{n+1} = x_n + R[2K_h\Delta t_i]^{1/2} \quad (2)$$

where K_h in m^2/s is the horizontal diffusivity, x_n is the original particle position, R is a random number with mean zero and standard deviation $r=1$, and Δt_i is the internal time step (TimeStepCalc). TrackMPD reproduces sub-grid-scale turbulent particle motion in the vertical (z) direction to mimic the debris turbulent particle motion in that direction:

$$z_{n+1} = z_n + R[2K_v\Delta t_i]^{1/2} \quad (3)$$

where z_n is the initial particle location and K_v is the vertical diffusivity.

K_h and K_v can be set constant or can be read from the OGCM outputs at each grid point and each time step if the OGCM has this option.

2.2.3. Sinking/Raising (biofouling and degradation)

The settling (rising) velocity w_s (m/s) occurs when the net gravitational force (gravity minus buoyancy) is higher (lower) than the drag force. It is a characteristic feature of any negatively (positively) buoyant particle, and determines its sinking (rising) displacement:

$$z_{n+1} = z_n - w_s(t_i)\Delta t_i \quad (4)$$

where z_n is the initial particle location, Kw_s is the vertical (settling or rising) velocity, and Δt_i is the internal time step (TimeStepCalc).

w_s can be directly defined by users ('value' option) or calculated by TrackMPD ('formulated' or 'rate' options) according to the particle behaviour (physical properties, biofouling, degradation). When

defining a constant value, pay attention to the sign convention for this parameter: 0=neutral buoyancy; + value: settling; - value: rising. a

The available formulations to calculate the settling velocity are:

- Waldschlager's formulation: recommended for spheres, fragments and fibers ([Waldschlager and Schuttrumpf, 2019a](#); [Waldschlager et al., 2020](#); [Jalón-Rojas et al., 2022](#))
- Dellino's formulation: recommended for sheets ([Jalon-Rojas et al., 2022](#))
- Zhiyao's formulation ([Zhiyao et al., 2008](#)), already available in TrackMPD v1.
- Khatmullina's formulation ([Khatmullina and Isachenko, 2017](#)).

Note that these formulations depend on different particles' physical parameters. Users only need to provide the variable required for the selected formulation. New formulations can be easily added to the code.

The calculated vertical velocity can include the effect of biofilms by defining a biofilm thickness (BT) and biofilm density (ρ_f) and calculating the total density of the particle ρ_p (only available for spheres and cylinders for now):

Spheres:

$$\rho_p = \rho_0 \frac{R_0^3}{(R_0+BT)^3} + \rho_D \left[1 - \frac{R_0^3}{(R_0+BT)^3} \right], \quad (5)$$

Cylinders:

$$\rho_p = \rho_0 \frac{R_0^2}{(R_0+BT)^2} + \rho_D \left[1 - \frac{R_0^2}{(R_0+BT)^2} \right], \quad (6)$$

where R is the radius of the original particle and ρ_0 is the density of the polymer.

The vertical velocity can remain constant (above options) or vary over time under the influence of biofouling and degradation. The different options to get a variable settling velocity are:

- Define and increasing/decreasing rate of ws ($WsRate$, m/s/day) so that:

$$ws = ws_0 + WsRate \Delta t \quad (7)$$

where Δt is the internal time step ($TimeStepCalc$).

- Define and increasing (or decreasing) rate of density (DR , g/cm³/day) due to biofouling

$$\rho_p = \rho_0 + DR \Delta t \quad (8)$$

- Define and decreasing (or increasing) rate of particle size (SR) due to degradation/fragmentation:

$$d_{equi} = d_{equi,0} + SR \Delta t \quad (9)$$

This option is only available using Waldschlager's formulation for the calculation of the settling velocity.

Note that all these behaviours can be easily updated or improved.

2.2.4 Beaching and washing-off

When a particle reaches the land (defined in the domain file, see inputs in Section 3.3) it is considered as beached if “Beaching” is activated. When “Beaching” is deactivated, it comes back to its last position in the water domain.

Beached particles can be washed-off if “Refloating” is activated. For this purpose, TrackMPD includes a Monte Carlo approach with probability P of being washed off, based on Lagrangian oil-spill models (*Al-Rabeh et al., 2000*):

$$P = 0.5^{-t/T}, \quad (10)$$

where t is the time from the last beaching and T is the half-life for debris to remain on the beach before being washed off again. This approach assumes that the probability of washing-off decreases exponentially with time due to interaction with the coastline (*Liubarzeva et al., 2018*, Figure 2).

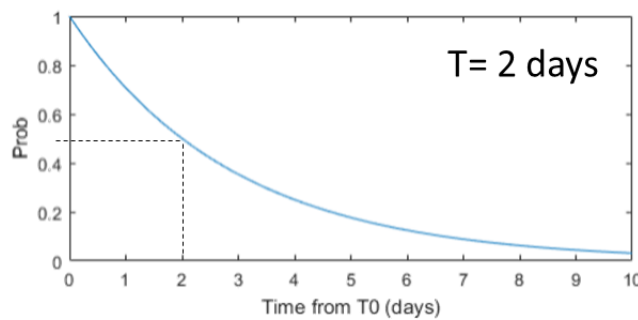


Figure 2. Example of temporal evolution of the washing-off probability for T (half-life for debris to remain on the beach before being washed off again) equal to 2 days.

Beached debris can be washed off from the coast to the sea to its last position before beaching if a randomly generated number (between 0 and 1) is less than P .

TrackMPD also has the option of activating washing-off only at high tide. In that case, the Monte Carlo method explained above is only implemented at high tide in a point of the domain (coordinates of the point to be defined as input parameter).

TrackMPD can include more-complex formulations in future versions to better reproduce the washing-off by tides, and also by waves.

2.2.5 Deposition, Resuspension and Bedload

When a particle reaches the bottom is deposited. It can be resuspended or transported as bedload (sliding) if ‘Resuspension’ and ‘Bedload’ are activated by users. Bedload cannot be activated without resuspension. Resuspension can be activated alone.

When these modules are activated, bed shear stress τ_0 is calculated for deposited particles as:

$$\tau_0 = \rho_w K_v \frac{U_{bot}}{\Delta z} \quad (11)$$

where ρ_w is the water density, U_{bot} is the current velocity from the layer closest to the bottom at the particle position, and Δz is the layer separation distance.

TrackMPD assumes the existence of two critical shear stress thresholds $\tau_{cr,1}$ and $\tau_{cr,2}$, so that:

- if $|\tau_0| < \tau_{cr,1}$ no motion, the particle remains deposited at the floor
- if $\tau_{cr,1} \leq |\tau_0| < \tau_{cr,2}$ bedload motion (if “Sliding” is activated), no motion (if “Sliding” is not activated).
- $|\tau_0| \geq \tau_{cr,2}$ resuspension

These thresholds are calculated following the Soulsby’s formulations ([Soulsby, 1997](#)) or applying the Waldschlager’s formulation to take into account the hiding/exposure effect by sediments ([Waldschlager and Schuttrumpf, 2019b](#)).

$$\theta_{cr} = 0.5588 \theta_{cr,Soulsby} \left(\frac{D_i}{D_{50}} \right)^{-0.503} \quad (12)$$

Other kind of parameterizations may be easily added to the code.

When a particle is resuspended it starts back from its position where it was deposited and it can be advected again. When a particle is transported by bedload, it experiences a horizontal transport proportional to the near-bed velocity:

$$U_{bedload} = C_m U_{bot} , \quad (13)$$

where C_m is an empirical coefficient for bedload transport defined through the keyword “conf.Beh.Cm” in TrackMPD input file.

Please note that the bedload module is still a trial version. It has been tested in 2DV (Jalón-Rojas et al., in prep.) with a different approach but its use in the 3D approach should be still considered carefully.

3. Getting started

3.1. Primary Framework Components

TrackMPD modelling framework v.2 has four 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*** and ***input_ModelName***: TrackMPD v.2 works with input provided in two functions:
 - *input_conf.m* compiles most of the input parameters (which are common to any application of TrackMPD, independently of the hydrodynamic model used to force it). See more details in Section 3.3.
 - *input_ModelName* includes the specific parameters needed to read the input velocity and grid data from a hydrodynamic model. Some examples are provided in the folder “ModelInputExamples”. See more details in Section 4.
- **TrackMPD_Toolbox**: This toolbox contains all the modules of the model. They are classified into the following packages/subfolders:
 - Advection
 - Behaviour
 - Dispersion
 - TRAJstruct (outputs)
 - TransformInputs
 - runTrackMPD
 - External

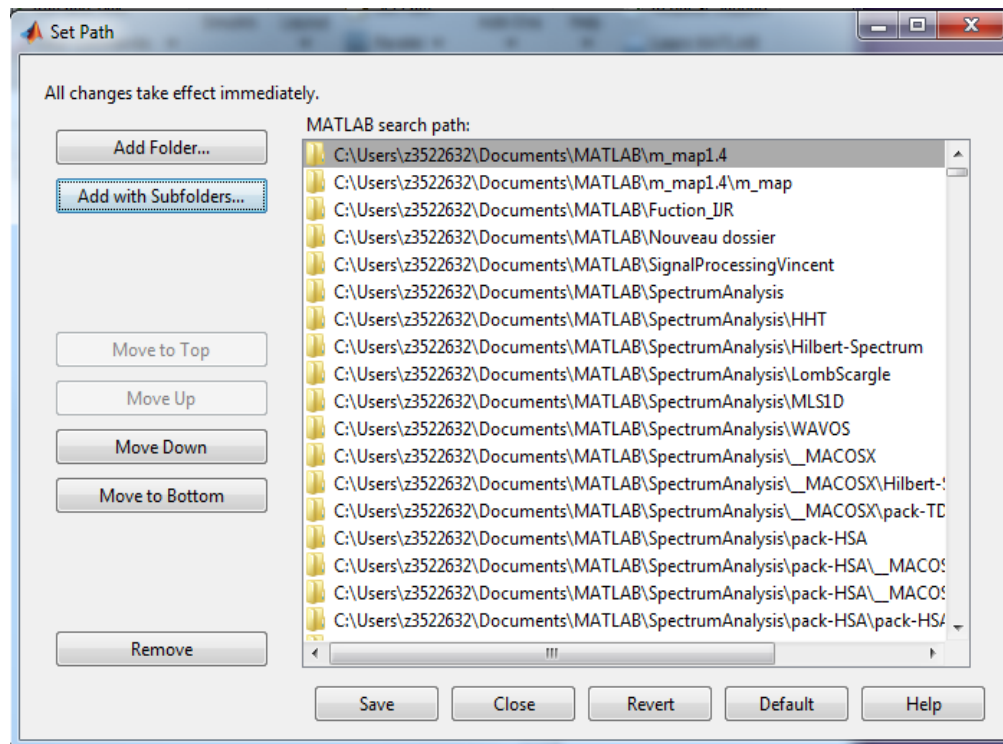
The modules forming TrackMPD are not described in this tutorial but each process is described above in Section 2.2.

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:

```
Command Window
New to MATLAB? See resources for Getting Started.
fx >> addpath(genpath('C:\Myfolder\TrackMPD_Toolbox'));
```

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



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 a “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 steps. 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.

3.3. Input

TrackMPD v.2 needs three input files:

- **OGCM inputs** (see Section 4 for details)
- The **initial particle location** is defined from a .csv file which contains three columns: longitude; latitude; depth (in meters).
- The **domain and coastline of the case study** are defined from a .dat or .txt file which contains two columns: the longitude and latitude coordinates of the coastline. These coordinates describe closed polygons of land. If a case study has several “pieces of land”, not connected between them (e.g. continent and islands, two continents), the coordinates of each piece should be separated by a line of 2 NaNs (see the coastline in the example provided for the Gironde estuary).

Note that many users make errors when preparing the coastline file for the first time.

To minimize this issue, we have prepared the script “check_domain.m”, available in the folder “Other scripts”. You can run this script with your coastline file and check if your land is plotted by closed grey polygons (see example below). If it’s the case, your file is correct. Otherwise, you should correct it (e.g. closing a polygon by adding new lon lat coordinates; separating islands from land with NaNs). Note that the type of

coordinates (spherical or cartesian) should agree with the coordinates type of the OGCM grid.

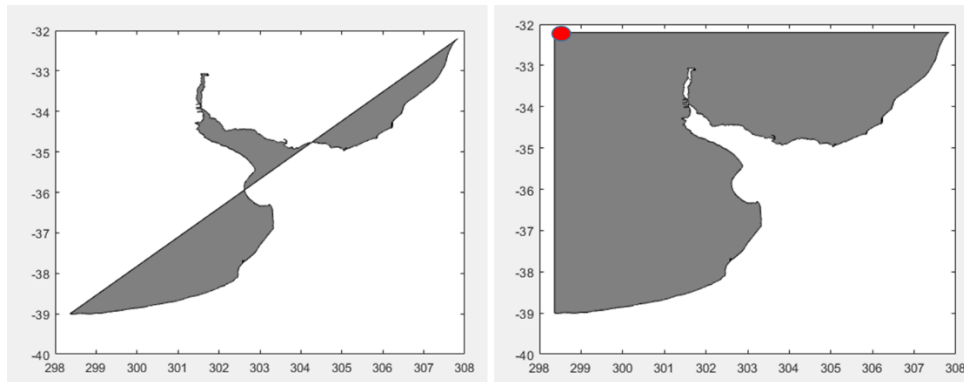


Figure 3. Example of plotting a coastline with `check_domain.m`. The left panel shows that there is a problem with the coastline as grey polygons do not correspond to land. The right panel shows the correct representation of the domain after including a new coordinate (red dot) to “close” the land polygon.

- All the other inputs are defined in the function *input_conf.m* and *input_OGCMname.m*. Users can change the name of these functions for different applications, but the structure and the name of the variables included in *input_conf.m* (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 and TrackMPD v2 to better understand the input variables.

a) Input data

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

b) Ocean General Circulation or Hydrodynamic Model (OGCM)

Conf.OGCM.Model_name	name of the OGCM
conf.OGCM.DomainName	name of the domain or case study
conf.OGCM.BaseDir	path of the folder that contains the OGCM files containing currents velocities and grid information
conf.OGCM.TimeStep	time step of the OGCM outputs in days (all the type of time steps involved in TrackMPD are detailed in Table 1).
Conf.OGCM.Coordinates	OGCM grid coordinates: spherical (degrees) or cartesian (km). Pay attention to select the correct type!
conf.OGCM.VerticalLayer	type of vertical layer: — <i>Rectangular</i> — <i>Sigma2depthVar</i> : sigma layer; depth not constant over time, dependent on water elevation — <i>Hybrid</i> : mix of sigma and rectangular layer
Conf.OGCM.BottomType	Bottom type options: — <i>Cte</i> : constant bed depth over time — <i>Var</i> : variable bed depth over time
Conf.OGCM.cut	<p>This option allows the users to select a portion of the original OGCM domain for TrackMPD simulations. If activated, trajectories will be only calculated in this portion. Options: 'yes' or 'no'</p> <p>When 'yes' is activated the portion of the OGCM domain to take into account is defined through the coordinates: <i>conf.OGCM.minLon</i>, <i>conf.OGCM.maxLon</i>, <i>conf.OGCM.minLat</i>, <i>conf.OGCM.maxLat</i>.</p>

c) Trajectory

Mode

conf.Traj.Verbose	<p>The verbose mode allows users to select the degree of detail in the progress messages during running.</p> <p>Options: 0: no information; 1: basic information; 2: full information</p>
-------------------	---

conf.Traj.Mode	‘2D’ or ‘3D’ approach (see Section 2.1 for more details)
conf.Traj.NbCores	Number of cores for particle parallel processing.

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.TimeStepCalc	Internal time step for the calculation of advection, dispersion and settling/raising (see Table 1). CFL condition is checked and warnings appear on the screen with the $\text{Verbose} \geq 1$.
conf.Traj.Direction	<i>forward</i> or <i>backward</i>
conf.Traj.Scheme	Numerical scheme for the solution of the advection equation. Options: <i>Euler</i> : Euler 1st order, <i>RK2</i> and <i>RK4</i> : Runge Kutta 2nd and 4th order. RK4 recommended.

Output

conf.Traj.TimeStepOut	Time step for beaching, washing-off, deposition and resuspension conditions verification. Time step of trajectory outputs. It should be higher or equal to conf.Traj.TimeStepCalc. Values lower or equal to conf.OGCM.TimeStep are recommended. (See Table 1)
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 computation purposes. Necessary for long simulations and/or high-resolution grids. This parameter does not impact the results. It is only important if Matlab warns you of memory problems. In that case, you need to decrease its value. A chunk length of 1-5 days should work well, but lower values may be necessary for large high-resolution grids.

Conf.Traj.chunklen	Duration of each partition in days (e.g. 5 days)
--------------------	--

Dispersion

conf.Traj.KvOption	Options to define the vertical diffusion coefficient (3D approach): <ul style="list-style-type: none">- <i>Cte</i>: Constant value over space and time.- <i>FromOGCM</i>: $K_v(x,y,z,t)$ varies over space and time; values are provided in the OGCM outputs and incorporated into TrackMPDInput.mat files in the transform phase.
--------------------	--

conf.Traj.KhOption	Options to define the horizontal diffusion coefficient: <ul style="list-style-type: none">- <i>Cte</i>: Constant value over space and time.- <i>FromOGCM</i>: $K_h(x,y,z,t)$ varies over space and time; values are provided in the OGCM outputs and incorporated into TrackMPDInput.mat files in the transform phase.
--------------------	--

conf.Traj.Kh	Horizontal diffusion coefficient (m^2/s) when <code>conf.Traj.KhOption = 'cte'</code>
--------------	---

conf.Traj.Kv	Vertical diffusion coefficient (m^2/s) when <code>conf.Traj.KvOption = 'cte'</code>
--------------	---

Beaching

conf.Traj.Beaching	<i>yes</i> : particle beaching is considered <i>no</i> : 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 variables:

conf.Traj.Tw half-life of particles remaining on the beach before
 washing off again (in days)

conf.Traj.RefloatingAtHighTide ‘yes’: washing-off only allowed at high tide
 ‘no’: washing-off allowed at each time step

conf.Traj.xcoord_elev grid cell (x-coord) to calculate the time series of water
 elevation when conf.Traj.RefloatingAtHighTide = ‘yes’

conf.Traj.ycoord_elev grid cell (y-coord) to calculate the time series of water
 elevation when conf.Traj.RefloatingAtHighTide = ‘yes’

Deposition and resuspension

conf.Traj.Deposition *yes* : particle deposition at the bottom considered
 no : particle deposition at the bottom not considered

conf.Traj.Resuspension *yes* : particle resuspension from the bottom considered
 no : particle resuspension from the bottom not considered

conf.Traj.ResOption *Soulsby* or *waldschlager*

conf.Traj.Sliding *yes* : particle bedload (sliding) considered (resuspension
 should be also activated)
 no : particle bedload (sliding) not considered.
 Please note that the bedload module is still a trial version.
 It has been tested in 2DV applications but it’s use in 3D
 approaches should be still considered carefully.

Table 1. Comparison of the different types of time step in TrackMPD v1 and TrackMPD v2.

Time step	TrackMPD v1	TrackMPD v2
TrackMPD inputs (OGCM outputs)	conf.OGCM.TimeStep	conf.OGCM.TimeStep
Advection calculation	Defined through odeset (abs_tol, rel_tol, max_step)	conf.Traj.TimeStepCalc
Dispersion calculation	conf.Traj.TimeStep	conf.Traj.TimeStepCalc
Settling calculation	conf.Traj.TimeStep	conf.Traj.TimeStepCalc
Beaching, washing- off, deposition, and out of domain check	conf.Traj.TimeStep	conf.Traj.TimeStepOut
Resuspension	-	conf.Traj.TimeStepOut
Outputs	conf.Traj.TimeStep	conf.Traj.TimeStepOut

d) Behaviour (3D mode)

Behaviour parameters will be only read by TrackMPD if settling/rising, deposition and/or resuspension are activated. If you are not using these modes you can leave the parameters by default. **Please note that the definition of behaviours is different from TrackMPD v1.**

Water and bed sediment properties: only useful for resuspension, sliding and/or for the internal calculation of settling/rising velocity (ws) from the physical parameters of the particle

conf.Beh.WaterDensity	water density (g/cm^3); for resuspension, sliding and/or internal calculation of ws
conf.Beh.WaterViscosity	water viscosity (m^2/s); for internal calculation of ws and/or sliding
conf.Beh.SediD50	D50 of bed sediments (m); for resuspension and sliding
Conf.Beh.Cm	Empirical drag coefficient (adim) to take into account pressure drag, added mass effect (see Jalón-Rojas et al., in prep.); for sliding only. Value between 0 and 1.
Conf.Beh.Cd	Bed drag coefficient (adim); for sliding only.

Particle properties: only useful for resuspension, sliding and/or for the internal calculation of settling/rising velocity (ws) from the physical parameters of the particle

conf.Beh.PolymerAcronym	polymer acronym, e.g. <i>PS</i> , <i>HDPE</i> , <i>PP</i> ; Optional, just for records.
conf.Beh.ParticleDensity	Polymer/particle density (g/cm ³); for resuspension, sliding and internal calculation of ws
conf.Beh.ParticleSize	Particle length (m). It refers to the largest side: e.g diameter for sphere, length for fibres, longest side for sheets); for resuspension, bedload, and internal calculation of ws.
conf.Beh.ParticleDequi	Particle equivalent diameter (a*b*c) ^(1/3) (m); for resuspension, sliding, and internal calculation of ws.
conf.Beh.Shape	'sphere','fragment','fibre','sheet'; for internal calculation of ws.

Settling velocity setup:

conf.Beh.InitialWsOption	Options to define the initial settling velocity: - <i>value</i> : known value, given by use (macro- or microplastic, other types of particles). To be set to 0 for passive behaviour. - <i>formulation</i> : select a specific formulation to calculate (internally) the settling velocity (microplastics) - <i>rate</i> : increasing or decreasing rate
--------------------------	---

If conf.Beh.InitialWsOption = 'value'

conf.Beh.Ws0	Initial settling velocity (0=neutral buoyancy; + value: settling; - value: rising)
--------------	--

If conf.Beh.InitialWsOption = 'formulation'

conf.Beh.WsForm	Available formulations:
-----------------	-------------------------

- *waldschlager*: recommended for spheres, fragments and fibers ([Waldschlager and Schuttrumpf, 2019a](#); [Waldschlager et al., 2020](#); [Jalón-Rojas et al., 2022](#))

- *dellino*: recommended for sheets ([Jalon-Rojas et al., 2022](#))

- *zhiyao*: ([Zhiyao et al., 2008](#)), already available in TrackMPD v1.

- *khatmullina*: ([Khatmullina and Isachenko, 2017](#)), already available in TrackMPD v1.

Note that new formulations can be easily added to the code.

For Waldschlager's formulation (see [Waldschlager and Schuttrumpf, 2019a](#)):

conf.Beh.CSF	Corey Shape Factor (adim): $c/(a*b)^{(1/2)}$
conf.Beh.P	Power roundness

For Khatmullina's formulation (see [Khatmullina and Isachenko, 2019](#)):

conf.Beh.Diameter	Fibers diameter (m)
-------------------	---------------------

For Dellino's formulation (see [Denillo et al. 2005](#) or [Jalón-Rojas et al., 2022](#)):

conf.Beh.Phi	Sphericity (adim)
--------------	-------------------

If conf.Beh.InitialWsOption = 'rate'

conf.Beh.Ws0	Initial settling velocity (m/s)
conf.Beh.WsRate	ws increasing or decreasing rate (m/s/day)

Biofouling

conf.Beh.Biofouling	Options for biofouling :
---------------------	--------------------------

- *no*: no biofouling
- *inWs*: We use this option just to indicate that we have already considered the effect of biofouling when defining the value of *ws* in `conf.Beh.InitialWsOptions = 'Value'`.
- *CteCalculated*: TrackMPD estimate an increase in *ws* as a function of the biofilm thickness and biofilm density as described in [Jalón-Rojas et al. \(2019\)](#).
- *rate*: *ws* increases or decreases over time as a function of a density rate (different from TrackMPD v1).

If `conf.Beh.Biofouling = 'CteCalculated'` (see formulations in [Jalón-Rojas et al., 2019](#))

<code>conf.Beh.BiofoulingThickness</code>	biofilm thickness (m)
<code>conf.Beh.FilmDensity</code>	biofilm density (g/cm ³)

If `conf.Beh.Biofouling = 'Rate'`

<code>conf.Beh.BiofoulingRate</code>	Rate of density increase (g/cm ³ /day). Note that this parameter is different from TrackMPD v1
--------------------------------------	---

Degradation

<code>conf.Beh.Degradation</code>	<ul style="list-style-type: none"> - <i>no</i>: no degradation - <i>rate</i>: the particle size (and <i>ws</i>) decrease or decrease as a function of a rate. Only compatible with 'waldschlager' formulation to calculate <i>ws</i>.
<code>conf.beh.DegradationRate</code>	Percentage of size decrease per day (mm/day) if <code>conf.Beh.Degradation = 'rate'</code> .

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

4. Include a new OGCM

TrackMPD 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.2 is already working with data from POM, FVCOM, MARS, TELEMAC, MOHID, NEMO, but it can be adapted to a new OGCM easily. The steps to include a new OGCM option in TrackMPD are:

1. Create a function *transformOGCMNAMEinputs_3D* and save it in the folder *TransformInputs* of the toolbox. Users can use the available functions in the *TransformInputs* folder as a model (e.g. POM or MARS for rectangular spherical grids, FVCOM for unstructured spherical grids, TELEMAC for unstructured cartesian grids). 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 *input_OGCMname.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 (matlab format) 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:

- u : a matrix (lat x long x layers) with the horizontal velocity (longitude direction; m/s) at each grid point;
- v : a matrix (lat x long x layers) with the horizontal velocity (latitude direction; m/s) at each grid point;
- w : a matrix (lat x long x layers) with the vertical velocity (m/s) at each grid point;
- $depth$: a matrix (lat x long x layers) with the depth of each grid point. It can be constant over time (rectangular), vary over time as a function of the water elevation (vertical layer type: Sigma2depthVar, Section 3.3.1.b, see examples for POM, FVCOM, TELEMAC, MARS; hybrid layer type: hybrid, see example for MOHID). For sigma or hybrid layers, this parameters should be transformed as $depth = depth - E$ so that depth of the first layer is equal to zero (figure 4).

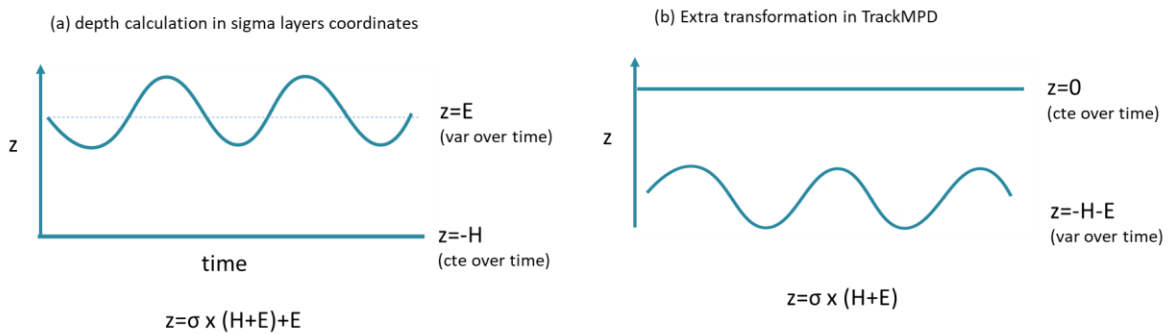


Figure 4. Sketch of depth evolution over time and transformation from sigma layer: (a) classical calculation, (b) extra transformation for TrackMPD to keep depth of water surface equal to zero. Cte refer to constant over time for a given grid point.

- E : a matrix (lat x long) with the water elevation (m) at each grid point;
- $time$: single numerical value containing the time stamp;
- $time_str$: single string value containing the time stamp.

Depending on the characteristics of your OGCM, you can use the functions *transformPOMinputs_3D*, *transformFVCOMinputs_3D*, *transformMARSinputs_3D*, *transformTELEMACinputs_3D*, or *transformMOHIDinputs_3D* (select the closest example to your application) as a guide to create this new function.

References

- Dellino, P.; Mele, D.; Bonasia, R.; Braia, G.; La Volpe, L.; Sulpizio, R. The Analysis of the Influence of Pumice Shape on Its Terminal Velocity. *Geophys. Res. Lett.* **2005**, *32*, L21306.
- Jalón-Rojas, I.; Wang, X. H.; Fredj, E. A 3D Numerical Model to Track Marine Plastic Debris (TrackMPD): Sensitivity of Microplastics Trajectories and Fate to Particle Dynamical Properties and Physical Processes. *Mar. Pollut. Bull.* **2019**, *141*, 256–272.
- Jalón-Rojas, I.; Romero-Ramirez, A.; Fauquembergue, K.; Rossignol, L.; Cachot, J.; Sous, D.; Morin, B. Effects of biofilm and particle physical properties on the rising and settling velocities of microplastic fibers and sheets. *Environ. Sci. Technol.*, **2022** *56*, 8114–8123.
- Khatmullina, L.; Isachenko, I. Settling Velocity of Microplastic Particles of Regular Shapes. *Mar. Pollut. Bull.* **2017**, *114*, 871–880.
- Pawlowicz, R. "M_Map: A mapping package for MATLAB", version 1.4m, [Computer software], **2020**, available online at www.eoas.ubc.ca/~rich/map.html.
- Soulsby, R.: Dynamics of marine sands, **1997**, Thomas Telford Publications.
- Waldschläger, K.; Schüttrumpf, H. Effects of Particle Properties on the Settling and Rise Velocities of Microplastics in Freshwater under Laboratory Conditions. *Environ. Sci. Technol.* **2019a**, *53*, 1958–1966.
- Waldschläger, K.; Schüttrumpf, H. Erosion behavior of different microplastic particles in comparison to natural sediments. *Environmental science & technology* **2019b**, *53*, 13219–13227.
- Waldschläger, K.; Born, M.; Cowger, W.; Gray, A.; Schüttrumpf, H. Settling and Rising Velocities of Environmentally Weathered Micro- and Macroplastic Particles. *Environ. Res.* **2020**, *191*, 110192.
- Zhiyao, S.; Tingting, W.; Fumin, X.; Ruijie, L. A Simple Formula for Predicting Settling Velocity of Sediment Particles. *Water Sci. Eng.* **2008**, *1*, 37–43.