User's Manual

The Trash-Tracker

The Trash-Tracker: A Macroplastic Transport and Fate Model at River Basin Scale

Y.A.M. Mellink[1,2], T.H.M. van Emmerik[1], M. Kooi[3], C. Laufkötter[4,5], and H. Niemann[6,2,7]

[1] Hydrology and Quantitative Water Management Group, Wageningen University, Wageningen, the Netherlands

[2] Department of Earth Sciences, Faculty of Geosciences, Utrecht University, Utrecht, the Netherlands

[3] Aquatic Ecology and Water Quality Group, Wageningen University, Wageningen, the Netherlands

[4] Climate and Environmental Physics, Physics Institute, University of Bern, Bern, Switzerland

[5] Oeschger Centre for Climate Change Research, University of Bern, Bern, Switzerland

[6] Department of Marine Microbiology and Biogeochemistry, NIOZ Royal Netherlands Institute for Sea Research, 't Horntje, the Netherlands

[7] CAGE - Centre for Arctic Gas Hydrate, Environment and Climate, Department of Geosciences, UiT The Arctic University of Norway, Tromsø, Norway

---

The Trash-Tracker is a numerical code developed by Mellink et al. (2021) [7] to simulate the transport and accumulation of plastic trash through terrestrial and freshwater systems. The two plastic transport agents are wind and surface runoff. The transport of plastic trash over land occurs when the wind and/or surface runoff conditions are sufficient to overcome the wind and/or surface runoff thresholds, respectively. These thresholds reflect the resistance to plastic transport and are a function of land use and terrain slope.

Table of Contents

Python version

The *Trash-Tracker* is a Python 3.8.3 code written in the open-source web application Jupyter Notebook (Version 6.0.3) available by Anaconda Software Distribution [1].

NumPy arrays and plotted geographical maps

The Trash-Tracker codes makes use of NumPy arrays. The elements stored in an array can be called for by referring to its indices. In 2D arrays the index of the first and second dimensions are denoted by `i` and `j`, respectively. The data that are stored in arrays are often plotted on geographical maps (e.g. topography or land use maps).

If you wish to manually fill arrays and subsequently plot that data on geographical maps, then it must be noted that the indices of a 2D array are not equivalent to geodetic coordinates and that a 2D array is not simply an overlay over a geographical map. A data value (e.g. topography) that belongs to location 'x', which lies in the southwest, i.e. lower left, corner of a geographical map (grey shaded box in Fig. 1a) is stored in the upper left corner of the 2D array (grey shaded box in Fig. 1b). In addition, in a 2D NumPy array the row number (i) increases when going down (see Fig. 1b), while on a map the latitude decreases when going down (see Fig. 1a). Therefore, in order to project data stored 2D NumPy arrays to a geographical map, the array elements must be flipped over the first dimensional axis (the 0 axis), i.e. the rows.

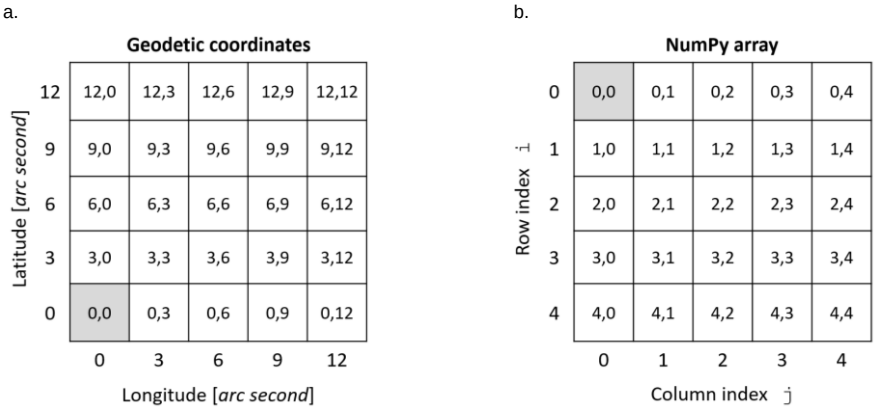

Fig. 1. (a) Map showing the geodetic coordinates of 25 grid cells with a grid resolution of 3 arc seconds. In grey location 'x' with 0 latitude and 0 longitude. (b) 5 by 5 2D NumPy array showing the indices `i` and `j` for the 25 elements that can be stored in this array. In grey the location at which the data value (element) that belongs to location 'x' is stored.

Storage of model parameters

All model parameters regarding plastic mobilisation and transport thresholds, runoff coefficients and mismanaged plastic waste generation are stored in the 1D `ParVal` array at the top of the program. If you wish to use different values for model parameters, simply change the value in the `ParVal` array and the model will use it in its calculations.

Model grid and domain

The code uses a rectangular domain of which the user sets the boundaries by assigning a minimum and maximum latitude (Lat_min_arc & Lat_max_arc) and longitude (Lon_min_arc & Lon_max_arc). Additionally, the user sets the zonal (Lon_res_arc) and meridional (Lat_res_arc) resolutions, which are used to fit rectangular grid cells within the domain boundaries. At the centre of each grid cell lies a grid point. The model calculates the coordinates (latitude and longitude) of the grid points and stores them in the 1D lat_arc[i] and lon_arc[i] arrays.

Directions of motion

The model allows for eight directions of motion. This is due to the rectangular geometry of the model grid whereby a(n) (interior) grid cell is surrounded by maximum eight other grid cells (Fig. 2). This approach is based on the encoding of water flow directions described by Jenson and Domingue [3]. The eight directions are each encoded with a label, i.e. a number. These numbers are different for each vector variable (see sections below).

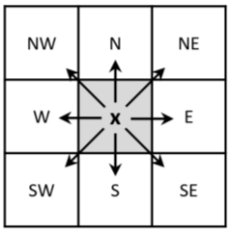

Fig. 2. The eight directions of motion.

Time frame

The user can set the time frame by assigning a time span (Time_span), i.e. the amount of time covered in a single model simulation, and a time resolution (Time_step), i.e. the amount of time covered within one time step. Dividing the time span by the time resolution yields the number of time steps (Time_range). The points in time at which input data is provided and for which output data is computed, are stored in the 1D Time[t] array.

Model input

The model input consists of terrain properties assigned to each grid cell in the domain and the weather conditions that occur within the grid cells. The terrain properties only have a spatial variability. Whereas the weather conditions (might) vary through space and time. For simple toy model applications, the input data arrays can be manually filled with values. For real life applications, the input data must be extracted from actual data bases (for examples of data bases that can be used see Mellink et al. (2021) [7]).

Land Use

The land use input data array, LU[i,j], contains for each grid cell the type of land use, whereby the model distinguishes between river and five types of land use. The land use types are labelled according to Tab. 1. The LU[i,j] array can be used to plot a land use map.

Tab. 1. Labels used in the LU array to indicate the type of land use.

Land use	Label
River	0
Bare land	100
Forest	200
Agricultural land	300
Urban land	400
Grass/shrub land	500

Population Density & mismanaged plastic waste generation

The population density input array, Y[i,j], contains for each grid cell the number of inhabitants. This array can be used to plot a population density map. The user can chose whether to assign artificial numbers of inhabitants to actually uninhabited grid cells in order to account for littering associated with recreational activities (e.g. in forest or on beaches).

The mismanaged plastic waste (MPW) generation for each grid cell and time step is stored in the 3D p_conc_INPUT[i,j,t] array. The MPW generation can either be achieved from an external data base or it can be calculated from the population density by:

$$MPW = MPW_{capita} \times MWF \times PP \times Y$$

where MSW_capita equals the amount of municipal solid waste generated per capita per time unit, MWF represents the fraction of waste that is mismanaged, PP is the fraction of waste that consists of plastics and Y is the number of inhabitants of the grid cell in question. These values can be achieved from Lebreton & Andrady (2019) [5].

Topography

The topography input array, Z[i,j], contains for each grid cell the elevation above (positive) or below (negative) mean sea level in meters. The Z[i,j] array can be used to plot a topography map.

From the topography data the model computes the terrain slopes for each grid cell. The slope in a certain direction is referred to as the distance weight drop (dwd). The model computes for each grid cell in the domain the distance weight drop to each of its neighbouring grid cells, e.g. for interior grid cells surrounded by eight other grid cells, the model calculates eight distance weight drops (Fig. 3a).

The distance weight drop of a grid cell towards a neighbouring grid cell is computed by dividing the change in elevation between the grid cell points, by the horizontal distance between these grid points. The elevations of the grid cells are stored in the Z[i,j] array and the horizontal distances are either the zonal resolution (Lon_res_m), the meridional resolution (Lat_res_m) or the diagonal distance (d_m), which equals $\sqrt{(Lon\_res\_m^2 + Lat\_res\_m^2)}$ (Fig. 3b).

The minimum distance weight drop indicates the steepest *downhill* slope for that grid cell. The magnitude of the steepest downhill slope for each grid cell is stored in the 2D Q[i,j] array. The direction of the steepest downhill slope for each grid cell is stored in the 2D q[i,j] array using the direction labels that are shown in Fig. 4. The Q[i,j] and q[i,j] arrays can be used to plot steepest downhill slope magnitudes and directions maps.

In case a grid cell is only surrounding by grid cells that have a higher elevation, the minimum distance weight drop value and its corresponding direction indicate the gentlest *uphill* slope.

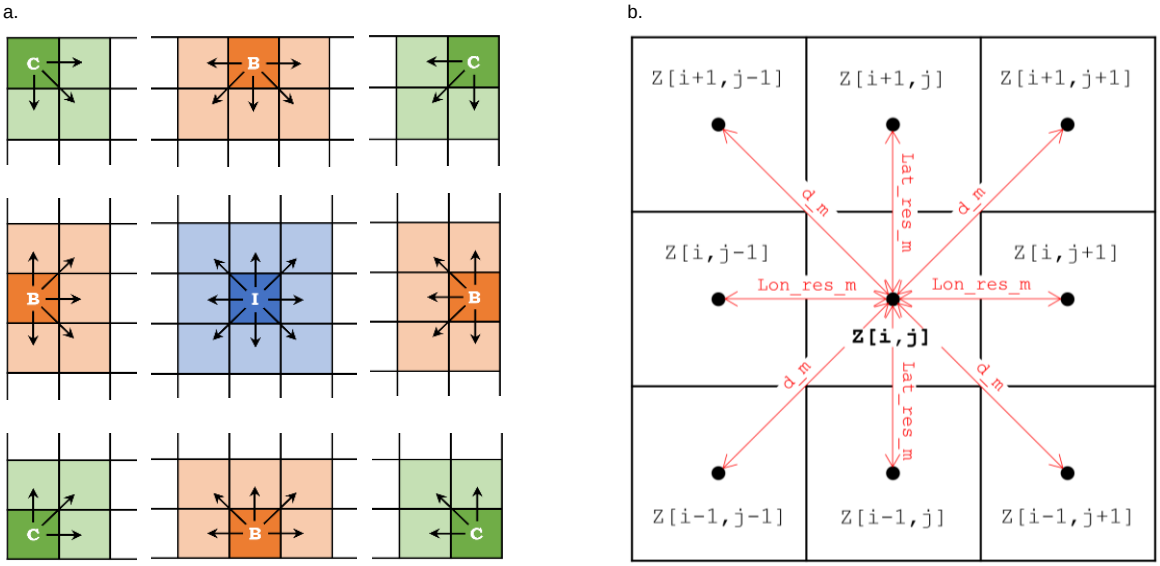

Fig. 3. (a) The directions in which the distance weight drop (dwd) is calculated for grid cells that are located in the interior (I), corner (C) or at the border (B) of the model domain. (b) Geographical map representation of an interior grid cell whose elevation is stored at Z[i,j]. Red arrows indicate the horizontal distance between grid points. Lat_res_m is the meridional grid resolution (m), Lon_res_m is the zonal grid resolution (m), and d_m is the diagonal distance (m) between grid points (= $\sqrt{(Lon\_res\_m^2 + Lat\_res\_m^2)}$).

1000	1001	1002
1007	x	1003
1006	1005	1004

Fig. 4. Map view of an interior grid cell 'x' and its eight neighbouring grid cells. Numbers correspond to the labels used to indicate towards which neighbouring grid cell the steepest downhill slope of grid cell 'x' occurs. For example, if the steepest downhill terrain slope is towards the north, then grid cell 'x' will have label 1001 in the q[i,j] array.

River flow

The river flow input data array, ri[i,j], contains for each river grid cell the direction in which the river water flows and the number 0 for land grid cells. The direction labels that are used for the river flow are shown in Fig. 5. The river water can only flow towards adjacent river grid cells.

3000	3001	3002
3007	x	3003
3006	3005	3004

Fig. 5. Map view of an interior grid cell 'x' and its eight neighbouring grid cells. Numbers correspond to the labels used to indicate towards which neighbouring grid cell the river water in grid cell 'x' flows. For example, if the river flows towards the north, then river grid cell 'x' will have label 3001 in the ri[i,j] array.

Wind speed and direction

The wind speed input array, W_speed[i,j,t], indicates for each time step for each grid cell the speed of the air flow (m/s). The wind speed values can be directly extracted from a wind speed data set or be generated based on wind speed frequency tables. In the current version of the Trash-Tracker it is assumed that each grid cell has the same wind speed during one time step. The wind speed values were generated on the basis of a wind speed frequency table from the Royal Netherlands Meteorological Institute (KNMI) [9]. The KNMI wind speed frequency table was split and assigned to 12 frequency arrays, one for each month (e.g. W_speed_freq_Jan[i], W_speed_freq_Feb[i], etc.). The values in these arrays indicate the frequencies with which the wind speeds measured at 'De Bilt' weather station (52.1015441 latitude and 5.1779992 longitude, the Netherlands) in the period 1981-2000, fell within 20 wind speed classes. The W_speed_classes[i] array contains the numbers 1 to 20, because there are 20 wind speed classes. The model uses the NumPy function numpy.random.choice([A], p = [B]), with A = is W_speed_classes[i] and B = W_speed_freq_*month*[i] in order to pick a wind speed class for each time step. The selected wind speed classes are stored in the w_speed_class[t] array. The wind speed value (m/s) for a time step is generated by randomly selecting a wind speed value that falls within the limits of the selected wind speed class. The wind speed values (m/s) are stored in the w_speed_value[t] array. The W_speed[i,j,t] array is created from the w_speed_value[t] array, by simply addressing the wind speed value for each time step to all the grid points.

The wind direction input array, W_dir[i,j,t], indicates for each time step for each grid cell the direction towards which the air flows. The wind directions can be directly extracted from a wind direction data set or be generated based on wind direction frequency tables. In the current version of the Trash-Tracker it is assumed that each grid cell has the same wind direction during one time step. The wind directions were generated on the basis of a wind direction frequency table from the Royal Netherlands Meteorological Institute (KNMI) [10]. The KNMI frequency table was split and assigned to 12 separate arrays, one for each month (e.g. W_dir_freq_Jan[i], W_dir_freq_Feb[i], etc.). The values in these arrays indicate the frequencies with which the wind directions measured at 'De Bilt' weather station (52.1015441 latitude and 5.1779992 longitude, the Netherlands) in the period 1981-2000, fell within 37 wind speed classes. The W_dir_classes[i] array contains the numbers 1 to 37, because there are 37 wind direction classes. The model uses the NumPy function numpy.random.choice([A], p = [B]), in which A = W_dir_classes[i] and B = W_dir_freq_*month*[i] in order to pick a wind direction class for each time step. The selected wind direction classes are stored in the w_dir_class[t] array. The wind direction for a time step is generated by randomly selecting a wind direction that falls within the limits of the selected wind direction class. The wind direction values (°) are stored in the w_dir_value[t] array. It must be noted that the wind direction is defined as the direction from which it originates. However, (plastic) particles carried by the wind are transported in the opposite direction. Therefore, 180° must be added to the 'official' wind direction in order to obtain the wind transport direction. Finally, the wind transport directions, in degrees (0° - 360°), are converted to the eight directions of motion using Tab. 2. The Wind-displacement_dir[t] array contains for each time step the wind transport directions, i.e. the directions in which the wind transports (plastic) particles, using the direction labels shown in Fig. 6. The W_dir[i,j,t] array is created from the Wind-displacement_dir[t] array, by simply addressing the wind direction labels for each time step to all the grid points.

Tab. 2. Wind direction conversion table. From degrees to the wind direction labels used in the Trash-Tracker code (Fig. 6).

Wind direction* [*degrees*]	**Label**
292.5 - 337.5	2000
337.5 - 22.5	2001
22.5 - 67.5	2002
67.5 - 112.5	2003
112.5 - 157.5	2004
157.5 - 202.5	2005
202.5 - 247.5	2006
247.5 - 292.5	2007

***** *Direction towards which the wind blows*

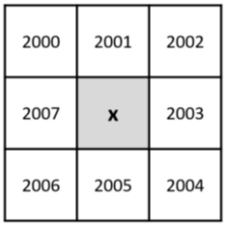

Fig. 6. Map view of an interior grid cell 'x' and its eight neighbouring grid cells. Numbers correspond to the labels used to indicate towards which neighbouring grid cell the wind in grid cell 'x' blows. For example, if the wind blows towards the north, then grid cell 'x' will have label 2001 in the W_dir array.

Surface runoff flux and direction

The surface runoff input data array, R_speed[i,j,t], indicates for each time step for each land grid cell the amount of surface runoff (mm/d). The surface runoff for river grid cells is set to 1000 (mm/d). The surface runoff values can be directly extracted from a surface runoff (or rainfall) data set or be generated based on rainfall frequency tables. In the current version of the Trash-Tracker the surface runoff is computed from rainfall values and it is assumed that each grid cell receives the same amount of rainfall during one time step. The rainfall values were generated on the basis of a rainfall frequency table, which was calculated from a rainfall data set of the Royal Netherlands Meteorological Institute (KNMI) [8].The KNMI rainfall data set contains for each day (from 1 January 1981 up and until 31 December 2000) the total amount of rainfall (mm) recorded by the Royal Netherlands Meteorological Institute (KNMI) at 'De Bilt' weather station (52.1015441 latitude and 5.1779992 longitude, the Netherlands). Each day was assigned to one of the 23 rainfall classes based on the total amount of rainfall that fell during that day. Subsequently, the frequencies for each rainfall class were computed and stored in the Rainfall_freq[i] array. Note that the model does not take monthly variations in the rainfall frequencies into account and thus the frequencies in the Rainfall_freq array hold for all months. The Rainfall_classes[i] array contains the numbers 1 to 23, because there are 23 rainfall classes. The model uses the NumPy function numpy.random.choice([A], p = [B]), with A = Rainfall_classes[i] and B = Rainfall_freq[i] in order to pick a rainfall class for each time step. The selected rainfall classes are stored in the rainfall_class[t] array. The rainfall value (mm/d) for a time step is generated by randomly selecting a rainfall value that falls within the limits of the selected rainfall class. The rainfall values (mm/d) are stored in the rainfall_value[t] array. Next, the rainfall values are converted to surface runoff values by using runoff coefficients. The runoff coefficient is the fraction of the rainwater that does not infiltrate in the soil and consequently becomes surface runoff. The surface runoff coefficients for river, bare, forest, agricultural, urban and grass\shrub lands are referred to as RC_river_value, RC_bare_value, RC_forest_value, RC_agricul_value, RC_uban_value and RC_grass_value, respectively. The values for these runoff coefficients are based on typical reported runoff coefficients for these types of areas [2, 4]. The rainfall value (mm/d) multiplied with a runoff coefficient yields the surface runoff (mm/d) for each grid cell and is stored for each time step in the R_speed[i,j,t] array.

The surface runoff direction input array, R_dir[i,j], indicates for each time step for each grid cell the direction towards which the surface runoff flows. For land grid cells, the surface runoff directions are equal to the directions of the steepest terrain slope, i.e. equal to the q[i,j] array. For river grid cells, the surface runoff directions are equal to the river flow directions, i.e. equal to the ri[i,j] array). The R_dir[i,j] array is in fact a combination of the q[i,j] and ri[i,j] arrays. The direction labels used for the surface runoff directions are shown in Fig. 7.

3000	3001	3002
3007	x	3003
3006	3005	3004

Fig. 7. Map view of an interior grid cell 'x' and its eight neighbouring grid cells. Numbers correspond to the labels used to indicate towards which neighbouring grid cell the surface runoff in grid cell 'x' flows. For example, if the surface runoff flows towards the north, then grid cell 'x' will have label 3001 in the R_dir array.

Plastic mobilisation and transport thresholds

Wind speed thresholds

The wind speed threshold can be calculated as a function of only the type of land use – Option 1 – or as a function of the type of land use and the (combination of) terrain slope and wind direction – Option 2. In the ParVal array at the top of the program, the user choses Option 1 or 2 by filling in '1' or '2', respectively, for the W_T variable, stored at ParVal[6].

Option 1

The wind speed threshold array, W_T_lu[i,j] indicates for each grid cell the critical wind speed (m/s) required to mobilise and transport plastic over at least a distance of d_m meters (i.e. the horizontal distance between two diagonal grid points – see Fig. 3b). The wind speed threshold values only depend on the type of land use and are referred to as W_T_R, W_T_B, W_T_F, W_T_A, W_T_U and W_T_G, for river, bare, forest, agricultural, urban and grass\shrub lands, respectively. The values for these parameters are filled in by the user at ParVal[0] to ParVal[5] at the top of the program. An explanation for the wind speed threshold values that are currently used in the Trash-Tracker can be found in the Supplementary Information S3 from Mellink et al. (2021) [7].

Option 2

This approach assumes that if the wind blows uphill/downhill, the ability of the wind to mobilise and transport plastics (in the direction of the wind) proportionally decreases/increases with the steepness of the slope, respectively. As the wind directions (can) vary through time, the wind speed thresholds calculated via Option 2 depend are time dependent as well. The wind speed threshold array, W_T_lus[i,j,t,], indicates for each time step for each grid cell the critical wind speed (m/s) required to mobilise and transport plastic over at least a distance of d_m meters (i.e. the horizontal distance between two diagonal grid points – see Fig. 3b). The thresholds are computed by calculating for each time step and grid cell the terrain slope angle (rad) in the direction of the wind at that time step. If the slope angle in the direction of the wind equals 0 (°), then the wind speed thresholds from option 1 (i.e. W_T_R, W_T_B, W_T_F, W_T_A, W_T_U or W_T_G) hold. For non-zero terrain slope angles, a value of 4.2 (m/s), referred to as deltaTW (stored at ParVal[8]), is added (for uphill winds) or subtracted (downhill winds) from the W_T_R, W_T_B, W_T_F, W_T_A, W_T_U or W_T_G value for each radian of terrain slope angle. An explanation for the deltaTW value can be found in Mellink et al. (2021) [7].

Surface runoff thresholds

The surface runoff threshold array, R_T_lus[I,j], indicates for each grid cell the critical amount of surface runoff (mm/d) required to mobilise and transport plastic over at least a distance of d_m meters (i.e. the horizontal distance between two diagonal grid points – see Fig. 3b). For river grid cells the threshold is set to 0 (mm/d), because once in the river the plastics will automatically follow the river flow and no surface runoff is required. The surface runoff thresholds depend on the type of land use and terrain slope, both are constant through time. Consequently, the surface runoff thresholds are time independent. The surface runoff threshold is determined by the terrain slope angle (°) of the steepest downhill slope, which is calculated from the steepest slope values (m/m) ($Q[i,j]$). The slope angle category in which the terrain slope angel falls, combined with the type of land use, determines the surface runoff threshold. There are 10 slope angle categories of which 'Category 0' corresponds to uphill slopes (in case the grid cell is only surrounded by higher topographies). The remaining 9 categories cover the downhill slope angles between 0^0 and 90^0 with intervals of 10^0. The surface runoff threshold values (mm/d) for each combination of land use and slope angle category are determined by the user and filled in at $ParVal[9]$ to $ParVal[58]$ at the top of the program. An explanation for the surface runoff threshold values that are currently used by the Trash-Tracker can be found in Mellink et al. (2021) [7].

Mobilisation maps

At each time step the wind speed and surface runoff values are compared to the wind speed and surface runoff thresholds, respectively. The outcome of this comparison is labelled with the numbers 1 to 4 (see Tab. 3). The $Check[i,j,t]$ array contains for each time step for each grid cell the outcome label. The $Check[i,j]$ array can be used to plot a plastic mobilisation map.

Tab. 3. Labels used for the outcome of the comparison between wind speed and surface runoff values are compared to the wind speed and surface runoff thresholds.

Outcome of the comparison between weather conditions and thresholds	Label
Only the wind speed threshold is surpassed	1
Only the surface runoff threshold is surpassed	2
Both the wind speed and the surface runoff threshold is surpassed	3
Neither the wind speed or surface runoff threshold is surpassed	4

Tracking trash

Each time step begins with a "*start MPW distribution*", stored in the $p\_conc\_start[i,j,t]$ array. During each time step the MPW trash clusters may or may not, depending on the wind and surface runoff conditions of that time step, be transported to a different (neighbouring) grid cell. After all the MPW trash cluster have been moved (or not) the model has stored the "end MPW distribution" of this time step in the $p\_conc\_end[i,j,t]$ array. For the first time step ($t = 0$), the start MPW distribution is equal to the MPW generation distribution (stored in the $p\_conc\_INPUT[i,j,t]$ array). For all other time steps, the start MPW distribution equals the sum of the MPW generation distribution for that time step and the end MPW distribution of the previous time step (Fig. 8).

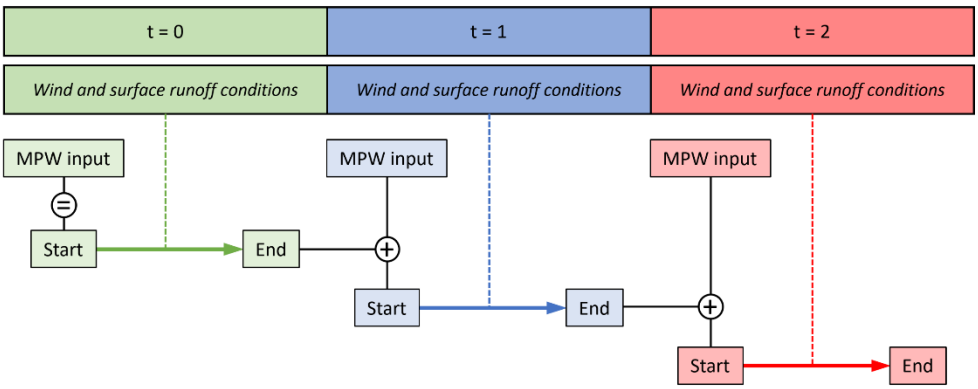

Fig. 8. Schematic representation of the model framework in which the start and end MPW distributions for each time step are computed. The start MPW distribution of a time step is the sum of the MPW input for that time step and the end MPW distribution of the previous time step. Except for t = 0, where the start MPW distribution equals the MWP input for t = 0.

For each time step, the model loops through all the grid cells in the model domain, if no MPW is present in a grid cell, it goes to the next. If a grid cell contains MPW, the value of the move_to variable will be determined, which is an indication of the displacement direction. Fig. 9 shows to which directions the values of move_to correspond (e.g. move_to = 3 means that the MPW is displaced to the neighbouring grid cell in the east). For river grid cells the move_to variable is directly derived from the river flow direction (stored in R_dir[i,j]). For land grid cells, the value of move_to depends on whether and which thresholds are surpassed (stored in Check[i,j]). If the Check[i,j] array indicates that only the wind threshold is surpassed, then the value of move_to is derived from the wind direction (stored in W_dir[i,j,t]). If the Check[i,j] array indicates that only the surface runoff threshold is surpassed, then the move_to value is derived from the surface runoff direction (stored in R_dir[i,j,t]). If the Check[i,j] array indicates that both thresholds are surpassed, then the code randomly picks either the wind or the surface runoff direction to determine the value of move_to. If the Check[i,j] array indicates that none of the thresholds are surpassed, then the MPW does not move and stays in its current grid cell and the value of move_to equals 8.

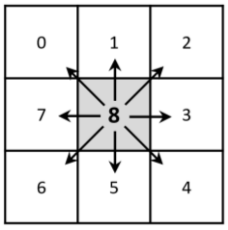

Fig. 9. The value of the move_to variable indicates towards which neighbouring grid cell the MPW is displaced. If no displacements occurs: move_to = 8.

Based on the value of move_to the model computes the indices of the grid cell location towards which the MPW has moved. These new array indices (i_new and j_new) are computed using the indices (i and j) of the current grid cell (see Tab. 4). If the new location of the MPW lies outside the model domain, it means that these plastics have left the river basin. The Trash-Tracker records for each time step the amount of MPW that has left the river basin via the river mouth, i.e. river emissions (stored in the OUT_river[t] array), via the coast, i.e. coastal emissions (stored in the OUT_coast[t] array) and via an inland border, i.e. land-to-land emissions (stored in the OUT_land[t] array). In case the new location of the MPW lies inside the model domain, this new location marks its “end” location for this time step. Note that in case the MPW has not moved the start location is equal to the end location.

The amount of MPW that by the end of each time step is located in the river and on land is documented in the Riverine_plastic[t] and Terrestrial_plastic[t] arrays, respectively. The Present_in_river_basin[t] array contains the total amount of MPW present in the whole model domain (i.e. river basin) by the end of each time step. This amount is computed by subtracting the cumulative amount of MPW that has left the river basin since that time step, from the cumulative amount of MPW that has been generated since that time step.

Tab. 4. Computation of the i_new and j_new indices from the i and j indices, respectively, for all the possible values of the move_to variable.

move_to	i_new	j_new
0	i + 1	j – 1
1	i + 1	j
2	i + 1	j + 1
3	i	j + 1
4	i – 1	j + 1
5	i – 1	j
6	i – 1	j – 1
7	i	j – 1
8	i	j

Potential plastic transport directions

The Trash-Tracker can be used to predict the (potential) transport directions of plastics without actually putting in plastics. In order to do so tally arrays, initially filled with zeros, are created for each of the eight directions of motion: tally_dir_0[i,j,t], tally_dir_1[i,j,t], tally_dir_2[i,j,t], tally_dir_3[i,j,t], tally_dir_4[i,j,t], tally_dir_5[i,j,t], tally_dir_6[i,j,t] and tally_dir_7[i,j,t]. For each time step the model determines for each grid cell in which direction MPW (if present) would have been transported under the provided set of wind and surface runoff conditions and thresholds. This transport direction is encoded with a move_to value (Fig. 9). Each transport direction, i.e. each move_to value, has its own tally array (e.g. $\text{move\_to} = 0$ belongs to tally_dir_0[i,j,t] and $\text{move\_to} = 1$ belongs to tally_dir_1[i,j,t] etc.). Each time wind and/or surface runoff conditions force MPW transport, +1 is added for that grid cell and that time step in the tally array that belongs to that transport direction.

In this way the Trash-Tracker is able to tally for each grid cell how often plastic transport occurs in all of the 8 possible transport directions. When all time steps have passed, the eight tally arrays contain the cumulative amount of times that plastics have been transported in each of the eight directions during the entire model run. The model extracts the final time steps from the eight tally arrays and refers to them as the “cumulative tally arrays”: tally_dir_0_cum[i,j], tally_dir_1_cum[i,j], tally_dir_2_cum[i,j], tally_dir_3_cum[i,j], tally_dir_4_cum[i,j], tally_dir_5_cum[i,j], tally_dir_6_cum[i,j] and tally_dir_7_cum[i,j]. Subsequently, the cumulative tally arrays are used to create the potential plastic routing map. To do so the transport directions are converted to 2D vectors. The thickness of the plotted transport vector is proportional to the frequency with which MPW was forced in that particular direction.

References

1. Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda. 2016. https://anaconda.com
2. Goel MK. Runoff coefficient. In: Encyclopedia of Snow, Ice and Glaciers. Dordrecht: Springer; 2011. p. 952–3. http://dx.doi.org/10.1007/978-90-481-2642-2_456
3. Jenson SK, Domingue JO. Extracting topographic structure from digital elevation data for geographic information system analysis. American Society for Photogrammetry and Remote Sensing. 1988;54(11):1593–600.
4. Karamage F, Zhang C, Fang X, Liu T, Ndayisaba F, Nahayo L, et al. Modeling rainfall-runoff response to land use and land cover change in Rwanda (1990–2016). Water (Basel). 2017;9(2):147 http://dx.doi.org/10.3390/w9020147
5. Lebreton LCM, Andrady A. Future scenarios of global plastic waste generation and disposal. Palgrave Commun. 2019;5(1):1–11 http://dx.doi.org/10.1057/s41599-018-0212-7
6. Meijer L, van Emmerik T, Lebreton L, Schmidt C, van der Ent R. Over 1000 rivers accountable for 80% of global riverine plastic emissions into the ocean. EarthArXiv. 2019; http://dx.doi.org/10.31223/osf.io/zjgty
7. Mellink YAM, van Emmerik T, Kooi M, Laufkötter C, Niemann, H. The Trash-Tracker: A Macroplastic Transport and Fate Model at River Basin Scale. Pre-print available at EarthArXiv. 2021;
8. Royal Netherlands Meteorological Institute (KNMI). Daggegevens van het weer in Nederland, RH, Etmaalsom van de neerslag (in 0.1 mm) (-1 voor <0.05 mm). http://projects.knmi.nl/klimatologie/daggegevens/selectie.cgi. Accessed 29 May 2020.
9. Royal Netherlands Meteorological Institute (KNMI). Frequentietabellen Windsnelheid in m/s, Distributief in procenten, De Bilt. http://projects.knmi.nl/klimatologie/frequentietabellen/maand.cgi. Accessed 29 May 2020.
10. Royal Netherlands Meteorological Institute (KNMI). Frequentietabellen Windrichting in graden, Distributief in procenten, De Bilt. http://projects.knmi.nl/klimatologie/frequentietabellen/maand.cgi. Accessed 29 May 2020.