Universitat Politècnica de Catalunya
Department of Civil and Environmental Engineering
Hydrogeology Group

MODPATH-RW: A Random Walk Particle Tracking Code for Solute Transport in Heterogeneous Aquifers

Documentation of Input-Output

Rodrigo Pérez-Illanes
Daniel Fernàndez-Garcia

Barcelona
February 17, 2023

Contents

Chapter 1

Introduction

This report provides the documentation of input and output for the Random Walk Particle Tracking (RWPT) program MODPATH-RW, implemented as an extension of the source code of the particle tracking model MODPATH, released by the U.S. Geological Survey (USGS) (see Pollock, 2016; Pollock & Provost, 2017). Besides the base functionalities, the program implements: (i) new particles displacements based on the RWPT method, (ii) Grid Projected Kernel Density Estimation (GPKDE) for smoothed concentration reconstruction, (iii) cell observations for resident and flux concentrations, (iv) and the new interpretation of MODPATH particles as individual solute mass elements with specific dispersion properties. In this sense, the RWPT method requires more computational effort in comparison to the base code, so in order to provide efficient runtimes, development also considered the implementation of parallel processing of particles with the OpenMP library (Pérez-Illanes & Fernàndez-Garcia, 2022). Additional functionalities and program characteristics supporting the previous features are addressed in more detail throughout this document.

Some specific topics regarding the routines for reading variables from simulation files, or the different grids supported by the program and their characteristics are not addressed in much detail in this report, because essentially these remain the same as in the base code, so users are refered to the base MODPATH documentation for more details (Pollock, 2016). The main motivation to extend MODPATH lies in the inherent interoperability with a variety of MODFLOW models, allowing in this sense to integrate the solute transport functionalities with this suite of groundwater flow codes. Following the practices of the USGS, the program is delivered with complementary online repositories and tools to assist the process of writing input files. The program release considers the following repositories

- `MODPATH-RW`: Main Fortran source code for the program.
- `gpkde`: Fortran module for GPKDE reconstruction.
- `flopyrw`: `python` utilities for writing program input files extended from the `flopy` package.

In the following, specific changes to the main simulation file and the instructions for writing the input files for the new model packages are addressed.

Chapter 2

Input files

Input files for MODPATH-RW follow in general the same structure and logic than those from the base code. In this sense, users are encouraged to follow the documentation of MODPATH-v7 (Pollock, 2016) and in this chapter only the main notable differences are remarked. Documentation is written considering that the program executable is `mpathrw`, and in general, considering a simulation file `mpathsim.mprw`.

A complementary project of this development are the `python` classes grouped in the repository `flopyrw`[1], which contains automated utilities to write the necessary input simulation files for MODPATH-RW. These are an extension from the classes currently available at the project `flopy`[2], extended from the classes for MODPATH-v7. At the time of this documentation, `python` utilities are consistent with the structure of files discussed in the following sections.

2.1 Simulation file

This is the main entry point for a MODPATH-RW simulation. It configures the simulation kind and general model parameters. Is the input file required to execute the program

```
mpathrw mpathsim.mprw
```

The file parameters are illustrated in Figure (2.1). It should be considered that not all of the parameters are included simultaneously and the presence of some of them is dependent on the values assigned to other parameters. In this regard, for MODPATH-RW, this file has only minor differences and users are encouraged to follow the guidelines explained in Pollock (2016). The differences introduced into this file for the purposes of MODPATH-RW are outlined in the following.

The structure of the main simulation file is very dynamic, so an objective of the implementation was to intervene this file only on the strictly necessary parts. The variables that have been given new interpretations or modifications are summarized in Table (2.1). Besides the MODPATH `SimulationTypes`, MODPATH-RW introduces additional interpretations for this parameter, that modify the function that track particles to a version considering Random Walk (RW) which performs the necessary operations for obtaining local quantities determining particles displacement. As

[1]https://github.com/upc-ghs/flopyrw
[2]https://github.com/modflowpy/flopy

it the case in MODPATH, all of the simulations write an endpoint file compiling the last position of particles

```
0 :Optional comment line          | # MODPATH-RW configuration file |
1 :NameFileName                   | mpathsim.mpnam                  |
2 :ListingFileName                | mpathsim.mplst                  |
3 :i.    SimulationType           | 5 1 2 1 0 0 0 1                 |
   ii.   TrackingDirection        | mpathsim.mpend                  | 4:EndpointFileName
   iii.  WeakSinkOption           | mpathsim.pathline               | 5:PathlineFileName
   iv.   WeakSourceOption         | mpathsim.timeseries             | 6:TimeseriesFileName
   v.    BudgetOutputOption       | mpathsim.trace                  | 7:TraceFileName
   vi.   TraceMode                | 1 100                           | 8: i. TraceParticleGroup ii. TraceParticleID
   vii.  TimeseriesOutputOption   | 2                               | 9:BudgetCellCount
   viii. ParticlesMassOption      | 10 20                           |10:BudgetCellNumbers[BudgetCellCount]
   ix.   SpeciesDispersionOption  | 1                               |11:ReferenceTimeOption
                                  | 0.0                             |12:ReferenceTime
                                  | 1 1 0                           |13:i.   StressPeriod
14:StopTimeOption                 | 3                               |   ii.  TimeStep
15:StopTime                       | 100.0                           |   iii. TimeStepFraction
16:TimePointOption                | 1                               |
17:i.  TimePointCount             | 50 2.0                          |
   ii. TimePointInterval          | 3                               |18:TimePointCount
                                  | 1.0  2.0  3.0                   |19:TimePoints[TimePointCount]
20:ZoneDataOption                 | 1                               |
21:StopZone                       | 0                               |
22:Zones[NCPL] or Zones[NROW,NCOL]| CONSTANT 1                      |
23:RetardationFactorOption        | 2                               |
24:Retardation[NCPL] or           | CONSTANT 2.0                    |
   Retardation[NROW,NCOL]         | 1                               |25:ParticleGroupCount
                                  | PG1                             |26:ParticleGroupName
                                  | 1                               |27:ReleaseOption
                                  | 0                               |28:ReleaseTime
                                  | 10 0.0 1.0                      |29:i.   ReleaseTimeCount
30:ReleaseTimeCount               | 2                               |   ii.  InitialReleaseTime
31:ReleaseTimes[ReleaseTimeCount] | 0.0  1.0                        |   iii. ReleaseInterval
32:StartingLocationsFileOption    | EXTERNAL PG1.sloc               |
   (StartingLocationsFileName)    | 1                               |
33:ParticlesMass                  | 0.1                             |
34:SpeciesID                      | 1                               |
```

Figure 2.1: Illustrative MODPATH-RW simulation file. Example parameters are enclosed between vertical dividers and their names are given in left/right columns.

Table 2.1: New and modified parameters of the main simulation file introduced by MODPATH-RW, identified following the indexation shown in Figure (2.1). All `SimulationTypes` write an endpoint file.

Id	Parameter	Type	Values
`3.i`	`SimulationType`	`int`	`1`: MODPATH endpoint `2`: MODPATH pathline `3`: MODPATH timeseries `4`: MODPATH pathline-timeseries `5`: MODPATH-RW timeseries `6`: MODPATH-RW pathline-timeseries `7`: MODPATH-RW endpoint
`3.vii`	`TimeseriesOutputOption`	`int`	`0`: Write timeseries only for active particles `1`: Write timeseries for all particles `2`: Skip timeseries writer
`3.viii`	`ParticlesMassOption`	`int`	`0`: Particles groups follow classical specification `1`: Request `33:ParticlesMass` for particle groups `2`: Request both mass and `34:SpeciesID`. In this case, `SpeciesID` is also requested for other packages leading to the creation of particle groups (`IC`,`SRC`)
`3.ix`	`SpeciesDispersionOption`	`int`	`0`: All particles with the same dispersion parameters. `1`: Specific dispersion related to `SpeciesID`.
`33`	`ParticlesMass`	`float`	Particles mass used for concentration postprocesses.
`34`	`SpeciesID`	`int`	Identifies the particle group as a specific species. In case `ParticlesMassOption=2` and single dispersion, the `SpeciesID` is identificatory and organizes particle groups as a species for postprocess (`OBS,GPKDE`).

2.2 Name file

The name file acts as coordination of the packages included on a simulation. For classical MODPATH, this file groups the information related to the MODFLOW model and the basic package (`MPBAS`), which stores the cell porosities and default `IFACE`s. For MODPATH-RW, this is the entry point for the new packages implemented in the program following a similar structure than the classical approach for MODFLOW-based software, where an identificatory package code is provided followed by the name of the file storing the package specifications.

MODPATH-RW introduces new packages to a MODPATH-v7 simulation which are summarized in Table (2.2). An example name file for a MODFLOW-6 model is shown in Figure (2.2). For a RW simulation, the `DSP` and `RWOPTS` packages are mandatory as they control the necessary parameters for computing particles displacements. Similarly, some packages accept multiple specifications, which could be understood as multiple instances of the package. For example, multiple dispersion models can be specified and then related to different substances.

Table 2.2: Packages implemented for MODPATH-RW. The column "Multi" indicates whether the package accepts multiple specifications on a single file. The "Optional" column is in terms of a RW simulation. For classical MODPATH runs, none of these packages is required.

Code	Name	Explanation	Multi	Optional
`RWOPTS`	Random Walk options	Controls RW displacements, time step	No	No
`DSP`	Dispersion	Dispersion parameters	Yes	No
`GPKDE`	Reconstruction	Grid, kernels	No	Yes
`IC`	Initial conditions	Initial distribution of concentrations	Yes	Yes
`SPC`	Species	Individualize species parameters	Yes	Yes
`OBS`	Observations	Monitor resident and flux concentrations	Yes	Yes
`SRC`	Sources	Mass injection cells	Yes	Yes
`IMP`	Impermeable cells	Specular reflection cells	No	Yes

```
# Name file for MODPATH-RW
MPBAS      mprwsim.mpbas
GRBDISV    mf6sim.disv.grb
TDIS       mf6sim.tdis
HEAD       mf6sim.hds
BUDGET     mf6sim.bud
DSP        mprwsim.dsp
SPC        mprwsim.spc
RWOPTS     mprwsim.rwopts
GPKDE      mprwsim.gpkde
IC         mprwsim.ic
IMP        mprwsim.imp
SRC        mprwsim.src
OBS        mprwsim.obs
```

Figure 2.2: Example MODPATH-RW name file for MODFLOW-6 simulation with `DISV` grid.

2.3 Basic file

This file remains exactly the same as in MODPATH-v7 so no additional comments are provided (see Pollock, 2016).

2.4 Random walk options (RWOPTS)

This package is mandatory for a RW simulation. It specifies basic parameters for computing the displacement of particles like the time step selection criteria, the advection model and the dimension mask. An example specification file is shown in Figure (2.3) and the interpretation of parameter names is shown in Table (2.3).

```
0: TimeStepSelection |MIN_ADV_DISP
1: (CFL)             |0.05
2: (CTDisp)          |0.05
2: (TimeStep)        |0.01
3: AdvectionKind     |EULERIAN
4: DimensionMask     |1 1 0
```

Figure 2.3: Illustrative specification for Random Walk options package. Parentheses remark conditional parameters (see Table 2.3).

Table 2.3: Parameter details for `RWOPTS` file shown in Figure (2.3).

Id	Parameter	Type	Values
0	`TimeStepSelection`	`string`	`ADV`: selection of time step with advection criteria using `CFL`. `DISP`: selection of time step with dispersion criteria using `CTDisp`. `MIN_ADV_DISP`: selects the minimum, requires `CFL` and `CTDisp`. `FIXED`: the same for all particles, given in `TimeStep`.
1	`CFL`	`float`	Include if `TimeStepSelection` is `ADV` or `MIN_ADV_DISP`.
2	`CTDisp`	`float`	Include if `TimeStepSelection` is `DISP` or `MIN_ADV_DISP`.
3	`TimeStep`	`float`	Include if `TimeStepSelection` is `FIXED`.
4	`AdvectionKind`	`string`	`EULERIAN` (default) or `EXPONENTIAL`.
5	`DimensionMask`	`int`	RW displacements in active (`1`) dimensions. `0` means inactive.

2.5 Dispersion (DSP)

Package groups dispersion parameters. Accepts multiple specifications following the simplified structure

```
 0 :NDsp
[1]:DspSpecification
```

where NDsp is the number of dispersion specifications (DspSpecification) given in the file. This release considers a single dispersion model based on classical linear dispersivities for isotropic porous medium, but more implementations are considered for future releases. An illustrative input structure for this package is shown in Figure (2.5). Dispersion parameters are all read with the free-input format of MODPATH for each model layer. Parameter details are presented in Table (2.4).

If in the main simulation file, the SpeciesDispersionOption=0, then only the first dispersion specification will be read. In case the simulation is configured with species specific dispersion, then all specifications will be loaded. However, the assignment of the different dispersion models will be conditioned to the presence of the SPC package, as discussed in the following section. The specification begins with an identificatory string name for the dispersion data, followed by a reading format. Until this release, there is a single format that reads the effective molecular diffusion (corrected by tortuosity), longitudinal and transverse dispersivities.

```
 0 :DspName   |DSP1
 1 :DspFormat |1
 2 :DMEff     |CONSTANT    0.000000E+00       #dmeff layer 1
 3 :AlphaL    |CONSTANT    10.00000E+00       #alphal layer 1
 4 :AlphaT    |CONSTANT    1.000000E+00       #alphat layer 1
```

Figure 2.4: Illustrative structure for specification of dispersion parameters for a single layer model.

Table 2.4: Parameter details for dispersion specification shown in Figure (2.5).

Id	Parameter	Type	Values
0	DspName	string	Less than 20 characters.
1	DspFormat	int	1: the only implemented.
2	DMEff	float	Corrected by tortuosity. Layered, free-input format.
3	AlphaL	float	Longitudinal dispersivity. Layered, free-input format.
4	AlphaT	float	Transverse dispersivity. Layered, free-input format.

2.6 Species (SPC)

This package organizes the different species instances. These serve as a link between different particle groups, also coordinating the dispersion parameters for simulations with species specific dispersion. The package accepts the specification of multiple species, following the simplied structure

```
 0 :NSpecies
[1]:SpeciesSpecification
```

where NSpecies indicates how many specifications are given. For MODPATH-RW, this package is optional and different scenarios are possible depending on the values given to the simulation parameters SpeciesDispersionOption and ParticlesMassOption, and if the SPC package was specified or not.

In case the package is not specified and ParticlesMassOption=2, then different species will be internally created, using the SpeciesID given by the user. Otherwise, a single placeholder species is created, to which all of the particle groups in the simulation are linked.

If the package is given the program will interpret the specifications. If ParticlesMassOption!=2, then the program will require that the user indicates the particle groups associated with the species. In case ParticlesMassOption=2, the program will read the specifications and relate the SpeciesID given by users at particle groups. In this case, it is internally understood that identifying numbers provided to particle groups have a minimum value of one and a maximum equal to NSpecies given in the SPC package.

For simulations with a single dispersion model for all particles, then the grouping provided by species will be only identificatory. Concentration related postprocesses will consider that species are different, but the transport model will employ the same dispersion parameters for all particles. If the simulation considers species specific dispersion, then this package becomes mandatory because is the only way for the program to know which dispersion model should be related to which species.

In case that the particle groups are specified by the user to the species in the SPC package, the group numbers should be considered carefully. Besides the classical particle groups, some packages will increase the number of groups (IC,SRC) and the user needs to keep track of their ids. In this sense, for simulations with complex specification of sources and initial conditions, it might be easier to provide the SpeciesID with ParticlesMassOption=2 and let the program interpret which particle group belongs to which species.

```
0 : SpcName                     |SRC1
1 : (NPGroups)                  |2
2 : (ParticleGroups[NPGroups]) |1 2
3 : (DspName)                   |DSP1
```

Figure 2.5: Illustrative structure for specification of species. Parentheses remark conditional parameters.

Table 2.5: Parameter details for dispersion specification shown in Figure (2.5).

Id	Parameter	Type	Values
0	`SpcName`	`string`	Less than 20 characters.
1	`NPGroups`	`int`	Read if `ParticlesMassOption!=2`. At least 1.
2	`ParticleGroups`	`int`	Read if `ParticlesMassOption!=2`.
3	`DspName`	`string`	Identifier of dispersion to which the species is related. Read if `SpeciesDispersionOption=1`.

2.7 Initial conditions (IC)

This package is intended to specify an initial distribution of concentrations which is internally transformed into a distribution of particles, for a given particles mass. The expected values are explicitly interpreted as resident concentrations, which combined with porosities, cell volumes and the delaying factor due to sorption, allows to estimate the total mass inside the model cells. An example specification is shown in Figure (2.6) illustrating only one distribution, however, the package is able to interpret multiple specifications. Until this release of the code, there is only one initial condition format, which reads the distribution of concentrations using the free-format input of MODPATH.

For each initial condition specification, MODPATH-RW will create a particle group whose name is given by `InitialConditionName`, which will be appended to the currently existing particle groups. An aspect to consider for the particles created with this method, is that their release time is zero by definition, for consistency with the idea of an initial condition.

```
 0 :NInitialConditions     |1
[1 :InitialConditionName   |IC1
 2 :InitialConditionFormat |1
 3 :ParticlesMass          |500.0
 4 :(SpeciesID)            |1
 5 :ResidentConcentration  |INTERNAL             1  (4E15.6) -1 #ic1 layer 1
...                        |1.593200E+00   3.470600E+00   4.769200E+00   6.102200E+00
/1]                        |8.877000E+00   1.032400E+01   1.181560E+01   1.335640E+01
                           |1.661340E+01   1.834760E+01   2.016920E+01   2.209500E+01
                           |2.634600E+01   2.872420E+01   3.131140E+01   3.413840E+01
                           |INTERNAL             1  (4E15.6) -1 #ic1 layer 2
                           |0.000000E+00   0.000000E+00   0.000000E+00   0.000000E+00
                           |4.184800E+00   6.062600E+00   7.428200E+00   8.864800E+00
                           |1.189620E+01   1.346920E+01   1.506860E+01   1.668820E+01
                           |1.996520E+01   2.161440E+01   2.326660E+01   2.491700E+01
```

Figure 2.6: Example input for initial condition package, considering a structured model with $(N_{lay}, N_{row}, N_{col}) = (2, 4, 4)$. Square brackets in parameters column indicate a list of sub-specifications and parenthesis remarks conditional parameters.

Table 2.6: Parameter details for initial condition shown in Figure (2.6).

Id	Parameter	Type	Values
0	`NInitialConditions`	`int`	At least 1, determine how many are interpreted.
1	`InitialConditionName`	`string`	Less than 20 characters.
2	`InitialConditionFormat`	`int`	1: the only implemented.
3	`ParticlesMass`	`float`	Determines the number of particles per cell.
4	`SpeciesID`	`int`	Read if `ParticlesMassOption=2`.
5	`ResidentConcentration`	`float`	The initial condition specified with free-format input.

2.8 Sources (SRC)

This package configures the source mass influx from a flow-model boundary condition, in cases where the flow-rate through the boundary is positive, entering the groundwater flow-model. The combination of flow-rates, concentrations and particles mass is transformed into a release of particles characterizing the boundary. Specifications for this package share some similarities with the Source and Sink Mixing (SSM) packages from MODFLOW and MT3DMS. However, in case of negative flow-rates, the boundary will not release particles and the number of extracted particles is an output result that could be monitored, for example, with observation cells.

The package accepts multiple specifications on a single file, following the simplified structure:

```
 0 :NSources
[1]:SourceSpecification
```

where the first parameter `NSources` is a positive integer indicating the length of the list of specifications (`SourceSpecification`) to be interpreted. Two specification formats are implemented:

- `AUXILIARY` (`AUX`): concentrations are extracted from auxiliary variables stored in the MODFLOW budget file. MODPATH-RW generates a timeseries of flow-rates from a given budget header.
- `SPECIFIED` (`SPEC`): concentrations, injection times and cells are given by the user for a determined budget header.

The package file can contain combined (multiple) specification formats. In the following, both specification formats are addressed with more details.

2.8.1 AUXILIARY

Auxiliary variables have been progressively integrated into the different versions of MODFLOW. In MODFLOW-2005 only some packages are compatible with this approach accepting up to 20 auxiliary variables (see Harbaugh, 2005), whereas in MODFLOW-6 most of the stress packages write output files capable of storing an unlimited number of auxiliary variables (Table 36, p. 288 in Langevin *et al.*, 2022). In MODPATH-v7, auxiliary variables are employed for interpretation of the `IFACE` parameter (Pollock, 2016). For practical purposes, auxiliary variables offer high flexibility to users allowing them to directly relate concentrations and flow-rates for a set of cells, at specific flow-model stress periods. This format will create a MODPATH particle group for each auxiliary variable, under the assumption that these represent solute concentrations.

Figure (2.7) presents the variables structure and example values for a source specification with the `AUXILIARY` format, and the interpretation of parameters is shown in Table (2.7). In plain words, the specification begins with an identificatory name, after which the `AUX` format is indicated. The format allows for multiple budget names to be read, indicating the set of auxiliary variables to be extracted and parameters that allow to transform the concentration data into particles. The program detects

```
 0 :SourceName                    | SRC1
 1 :SourceFormat                  | AUX
 2 :NBudgets                      | 1
[3 :i.BudgetName ii.IFaceOption | WEL-1   1
 4 :NAuxVariables                 | 3
[5 :i.  AuxVarName                | CONCSPC1    1.0  2 2 2   1
    ii. ParticlesMass             | CONCSPC2    0.5  4 2 2   2
    iii.Nx iv.Ny v.Nz             | CONCSPC3    0.1  4 4 1   3
/5] vi. (SpeciesID)               |
...                               |
/3]                               |
```

Figure 2.7: Example specification structure for source format `AUXILIARY`. Square brackets indicate a list of sub-specifications and parenthesis remarks optional parameters.

automatically which flow-model cells are associated with the budget name, running over all the simulation stress periods, as it is possible that some budget names are not enabled during the whole simulation. For each auxiliary variable, the user should specify the magnitude of particles mass and a template distribution (`Nx,Ny,Nz`) to be applied for each cell detected in the budget name. This distribution of particles is modified according to the dimensions mask provided in the `RWOPTS` package, and for inactive dimensions a default value of 1 is assigned to the template. For each auxiliary variable, the user should an integer solute identifier to which the variable is related in case that `ParticlesMassOption=2`. For each budget name, a flag indicates whether the `IFACE` auxiliary variable is read, and in such a case, the template of particles is applied on the face of cells. For consistent results, the value of `IFACE` should not change in time.

For simulations based on MODFLOW-6, multiple packages of the same kind can be specified, so the value of `BudgetName` should be the identificatory name given to each instance of a package (the value stored in `TXT2ID2`, see Langevin *et al.*, 2022). For simulations based on MODFLOW-2005, only some packages allow to specify auxiliary variables. The user can specify `BudgetName` as the three character identifier of the budget or its full budget header (`WEL` or `WELLS` for the well package). For budget header names with white spaces, the value should be specified in between quotes. Packages currently supported are shown in Table (2.8).

Table 2.7: Parameter details for specification with `AUXILIARY` format shown in Figure (2.7).

Id	Parameter	Type	Values
0	`SourceName`	`string`	Less than 20 characters.
1	`SourceFormat`	`string`	Less than 20 characters.
2	`NBudgets`	`int`	At least 1.
3.i	`BudgetName`	`string`	Less than 20 characters.
3.ii	`IFaceOption`	`int`	0: `IFACE` is not read from budget. 1: `IFACE` is read from budget.
4	`NAuxVariables`	`int`	At least 1.
5.i	`AuxVarName`	`string`	Less than 20 characters.
5.ii	`ParticlesMass`	`float`	Greater than zero.
5.iii	`Nx`	`int`	At least 1.
5.iv	`Ny`	`int`	At least 1.
5.v	`Nz`	`int`	At least 1.
5.vi	`SpeciesID`	`int`	Read if `ParticlesMassOption=2`.

Table 2.8: MODFLOW-2005 packages supported to be specified as sources with the `AUXILIARY` format.

Package id	Budget header
DRN	DRAINS
DRT	"DRAINS (DRT)"
GHB	"HEAD DEP BOUNDS"
RIV	"RIVER LEAKAGE"
WEL	WELLS

```
 0  :SourceName                                     |SRC2
 1  :SourceFormat                                   |SPEC
 2  :NBudgets                                       |1
[3  :i.BudgetName ii.IFaceOption iii.(DefaultIFace) |"CONSTANT HEAD"  1  2
 4  :i.CellInput ii.NCells iii.CellFormat iv.ConcPerCell |1 2 0 1
[5  :i.Cells ii.(IFace)                             |1 1 5
\5] :                                               |1 1 10 4
 6  :i.NSpecies ii.TemplateOption                   |2 1
[7  :i.Nx ii.Ny iii.Nz                              |2 2 2
/7]                                                 |4 4 2
[8] :ParticlesMass                                  |1.0 5.0
[9] :(SpeciesID)                                    | 1   2
 10 :NTimeIntervals                                 |4
[11 :i.TStart ii.TEnd iii.ConcSpeciesCell           |0.0   10.0  55.5  45.5  75.5  15.5
                                                    |15.0  17.5  66.6  56.6  26.6  26.6
                                                    |17.5  20    87.7  67.8  27.7  27.7
\11]                                                |22.5  30    57.7  37.1  17.7  27.7
...
\3]
```

Figure 2.8: Example specification structure for source format `SPECIFIED`. Square brackets indicate a list of sub-specifications and parenthesis remarks conditional parameters.

2.8.2 SPECIFIED

This second input format offers an additional alternative for users to indicate mass sources. It may be useful for users not familiarized with auxiliary variables, or to configure mass sources for those packages from MODFLOW-2005 that do not support said variables. In this case, the format is called `SPECIFIED` because users need to explicitly specify the cells and concentration timeseries for a given budget. A prototype file structure is presented in Figure (2.8), and most of the variables share the same meaning than for the case with auxiliary variables. The variables with different interpration are shown in Table (2.9).

The format provides different alternatives for reading source cells. These can be interpreted from the budget file, specified as a list of cell ids or as a three dimensional array. Only in the case the cells are given as list the program offers the users to read different concentration values for each cell. Meaning that in the case the source cells are read from the budget file or as three dimensional array, then the same concentration is assumed for all cells. Format requires that the user indicates time intervals, where concentrations are interpreted as step-wise quantities. Start and end times are relative to the MODFLOW simulation time, and are made consistent with the MODPATH reference and stop times indicated in the simulation file. The only restriction for the intervals specified by the user is that the intervals are not overlapping, but the program allows discontinuous time. In this sense, MODPATH-RW will create a timeseries of flow-rates extracted from the budget file that is consistent with the MODFLOW simulation steps and the user given intervals. As before, this format will create a MODPATH particle group for each of the species indicated in the input file.

Table 2.9: Parameter details for specification with `SPECIFIED` format shown in Figure (2.8). Only parameters with different meaning than those from Table (2.8) are presented.

Id	Parameter	Type	Values
`3.ii`	`IFaceOption`	`int`	0: `IFACE` is not read 1: `IFACE` is read after cell id (if `CellInput=1`)
`3.iii`	`DefaultIFace`	`int`	Default if `IFaceOption=1` and no value is given after cell id. If `CellInput=0 or 2` applies `DefaultIFace` to all cells.
`4.i`	`CellInput`	`int`	0: cells are read from budget 1: cells are given as a list of cell id's 2: cells are read with U3D reader
`4.ii`	`NCells`	`int`	Number of cells if `CellOption=1`
`4.iii`	`CellFormat`	`int`	0: `Cells` read as `Lay Row Col` 1: `Cells` read as nodes (`CellNumber`)
`4.iv`	`ConcPerCell`	`int`	0: Same species concentration(s) for all cells. 1: Different concentration(s) values for each cell.
`5.i`	`Cells`	`int`	Do not include if `CellInput=0` Cell identifiers written according to `CellFormat` if `CellInput=1` U3D array with `0` and `1` for active cells if `CellInput=2`
`5.ii`	`IFace`	`int`	`IFace` value for cell if `IFaceOption=1` and `CellInput=1`
`6.i`	`NSpecies`	`int`	Number of concentrations to be specified. At least 1
`6.ii`	`TemplateOption`	`int`	0: Only one particle template read, same for all species. 1: Read `NSpecies` templates.
`8`	`ParticlesMass`	`float`	One column for each specie
`9`	`SpeciesID`	`int`	Read if `ParticlesMassOption=2`. One column per specie.
`10`	`NTimeIntervals`	`int`	At least 1. Referent to MODFLOW times.
`11.i`	`TStart`	`float`	Beginning of the time interval.
`11.ii`	`TEnd`	`float`	End of the time interval. Intervals should not overlap.
`11.iii`	`ConcSpeciesCell`	`float`	Concentration values during the time interval. Read `NSpecies` columns if `CellInput=0` or `CellInput=2`. Read `NSpecies` columns if `CellInput=1` and `ConcPerCell=0`. Read `NSpecies` for each `Cell` if `CellInput=1` and `ConcPerCell=1`.

2.9 Grid projected kernel density estimation (GPKDE)

This package configures the parameters for the spatial concentration reconstruction. Is an optional specification and performs kernel density estimation over the set of particles considered to be of the same species. Until this document, spatial reconstruction is performed at the end of each timeseries step configured in the main simulation file. Reconstruction is performed on a regular grid, which is independent of the flow-model grid. In this regard, the module considers the reference axes aligned with the MODPATH interpretation of coordinates and receives the global particle coordinates as input data, with their respective particle mass.

Based on the grid specifications, the GPKDE Fortran module will determine whether reconstruction should be performed with 1, 2 or 3-dimensional kernels. The latter is decided based on the relation between the domain and bin sizes. In case the bin size on a given direction equals to the domain size, then reconstruction will consider that said dimension is compressed. The same is not true while computing the histogram. In this case, if the user wants to ignore a coordinate of the particles, then the bin size should be specified to be zero in said dimension.

```
0: OutputFile                 |mprwsim.gpkdeout
1: DomainOrigin               |  0.0    0.0    0.0
2: DomainSize                 |100.0  50.0  5.0
3: BinSize                    |5.0   5.0  5.0
4: NLoops                     |2
5: KernelDatabase             |0
6: minHLam (deltaHLam) maxHLam |1.0  5.0
7: AsResidentConcentration    |1
```

Figure 2.9: Illustrative specification for spatial reconstruction module. Parentheses remark conditional parameters

Table 2.10: Parameter details for specification GPKDE package shown in Figure (2.9).

Id	Parameter	Type	Values
0	OutputFile	string	Less than 200 characters.
1	DomainOrigin	float	Coordinates of the origin for the GPKDE grid.
2	DomainSize	float	Extent of the domain for the GPKDE grid.
3	BinSize	float	Size of the cells, are regular.
4	NLoops	int	Maximum number of optimization loops.
5	KernelDatabase	int	0: Compute kernels for exact smoothings. 1: Allocate a kernel database for discrete smoothings.
7	*HLam	float	minHLam minimum kernel smoothing. deltaHLam step to discretize kernel database. maxHLam maximum kernel smoothing.
7	AsResidentConcentration	int	0: Report reconstruction as total mass. 1: Report reconstruction as resident concentration, if possible.

2.10 Impermeable cells (`IMP`)

This is an optional package for MODPATH-RW, which determine which cells are impermeable for mass particles. Similarly, the first parameter of this package (`DefaultBoundary`) indicates the default behavior when reaching a flow-model boundary face. By default, if the package is not given, flow-model boundaries are considered to be impermeable (`DefaultBoundary=1` meaning that particles will rebound specularly. A value of zero (`0`) indicates that boundaries are open.

The second parameter (`ImpFormat`) indicates the method in which the impermeable cells are specified. Impermeable cells can be linked to the flow-model `IBOUND`, which is static in time. Similarly, these could be linked to the `IBOUNDTS` variable, which is introduced by MODPATH to monitor those flow-model cells that get dry during the simulation. The latter changes in time and is offered to users as an alternative to specify impermeable cells. The final format will read a layered array of integers from the variable (`ImpCells`) using the free-input format, where the value of zero (`0`) indicates that the cell is permeable and a value of `1` that the cell is impermeable. The main restriction for the latter is that not all cells can be marked as impermeable. In case the package is not specified all the interior cells are marked as permeable.

```
0 : DefaultBoundary | 0
1 : ImpFormat       | 2
2 : ImpCells        | INTERNAL      1    (6I4) -1 #impcells layer 1
                    | 0    0    0    0    0    0
                    | 0    0    0    0    0    0
                    | 0    0    1    1    0    0
                    | 0    0    1    1    0    0
                    | 0    0    0    0    0    0
                    | 0    0    0    0    0    0
```

Figure 2.10: Illustrative specification of a flow-model with $(N_{lay}, N_{row}, N_{col}) = (1, 6, 6)$ cells with an interior region of mass impermeable cells.

Table 2.11: Parameter details for impermeable cells specification shown in Figure (2.10).

Id	Parameter	Type	Values
0	`DefaultBoundary`	`int`	`0`: If no cell connection, open boundary. `1`: If no cell connection, rebound boundary.
1	`ImpFormat`	`int`	`0`: Follow flow-model `IBound`. `1`: Follow flow-model `IBoundTS` which also includes dry cells. `2`: Read layered array of impermeable cells.
2	`ImpCells`	`int`	Layered array, read with free-format input.

2.11 Observations (OBS)

This package configures the monitoring of concentration with observation cells. The package allows the interpretation of multiple specifications on a single file following the simplified structure

```
 0 :NObs
[1]:ObsSpecification
```

where `NObs` is the number of specifications (`ObsSpecification`) included in the file. The program will interpret as many specifications as indicated by the parameter `NObs`. Two kind of observations have been implemented into MODPATH-RW: observation of resident concentrations, which is obtained by monitoring the dissolved mass inside a cell (or group of cells) and dividing by a cummulative water volume; and observation of flux-concentrations, which require a sink flow rate and the simulation configured as strong-sinks in order to remove the particles from the system. A more detailed description on the conceptual differences between resident and flux concentrations can be found in Parker & van Genuchten (1984) and Kreft & Zuber (1986). Both these observations can be configured for a group of cells and a concentration timeseries can be obtained as postprocess of time dependent records.

2.11.1 RESIDENT

Observations of `RESIDENT` kind are tightly connected to the timeseries specification of the simulation. This is because resident concentrations have to be computed for pre-established times, where dissolved mass within a cell is aggregated. In this sense, it is recommended that users configure the timeseries simulation with regular (uniform) timesteps. For the timeseries reconstruction stage, in scenarios where the observation considers postprocessing, the program will automatically compute the size of the histogram bin as the step given for `TimePoints`.

An example specification is shown in Figure (2.11) with parameter details indicated in Table (2.12).

```
 0 :ObsName                                  |OBS1
 1 :ObsFormat                                |RESIDENT
 2 :ObsOutputFile                            |obscells1.obs
 3 :i.OutputOption ii.PostprocessOption      |1 1
 4 :i.CellInput ii.NCells iii.CellFormat     |1 2 1
[5 :Cells                                    |145
\5]:                                         |146
```

Figure 2.11: Example observation specification employing the format `RESIDENT`.

Table 2.12: Parameter details for specification of `RESIDENT` observation shown in Figure (2.11).

Id	Parameter	Type	Values
0	`ObsName`	`string`	Less than 20 characters.
1	`ObsFormat`	`string`	`RESIDENT` or `RES`.
1	`ObsOutputFile`	`string`	Less than 200 characters.
`3.i`	`OutputOption`	`int`	0: `ObsOutputFile` with observation records. 1: Postprocesses in `ObsOutputFile`, second file with records. 2: `ObsOutputFile` with postprocessed records.
`3.ii`	`PostprocessOption`	`int`	0: Histogram timeseries reconstruction of records. 1: Histogram and smoothed timeseries reconstruction of records.
`4.i`	`CellInput`	`int`	1: `Cells` are given as a list of cell ids. 2: `Cells` read with free-format input.
`4.ii`	`NCells`	`int`	Number of cells if `CellInput=1`. At least 1.
`4.iii`	`CellFormat`	`int`	0: `Cells` read as `Lay Row Col`. 1: `Cells` read as nodes (`CellNumber`).

2.11.2 FLUX

The second kind of observations monitors flux concentrations. Based on the number of particles (and their total mass) exiting through a strong-sink, the program estimates the mass flux leaving the system. In combination with the extraction flow-rate, a timeseries of flux concentrations can be obtained. Specification for this observation is similar to the previous case, being the only difference the `ObsFormat` parameter which can adopt the values `FLUX` or `SINK`. An aspect to remark of this observation is that it requires the cells to have a sink flow-rate. In case a cell with no extraction flow-rate is given, no records will be generated and hence the observation will remain empty. An example specification is seen in Figure (2.12). Until this implementation of MODPATH-RW, the postprocessing of this observation is also linked to the timeseries points provided by the user, but not the writing of records, which is performed only once per particle when arriving to an extraction cell.

```
 0 :ObsName                                  |OBS2
 1 :ObsFormat                                |FLUX
 2 :ObsOutputFile                            |obscells2.obs
 3 :i.OutputOption ii.PostprocessOption      |1 1
 4 :i.CellInput ii.NCells iii.CellFormat     |1 2 0
[5 :Cells                                    |1 10 4
\5]:                                         |1 8  3
```

Figure 2.12: Example observation specification employing the format `FLUX`.

Both observation formats can be combined on the same input file, as many times as the user require (see for example Fig. 2.13). It is recommended that users verify that a cell is being specified only once in observations to avoid inconsistencies. In case a cell appears in more than one specification, the program will consider that the cell belongs to the last interpreted specification.

```
2
OBS1
RESIDENT
obscells1.obs
1 1
1 2 1
145
146
OBS2
FLUX
obscells2.obs
2 1
1 2 0
1 10 4
1 8  3
```

Figure 2.13: Example observation file with two specifications of different kind.

Chapter 3

Output files

The base output files from MODPATH remain as they are configured in the original program. The structure of endpoint, pathline and timeseries files can be considered to be the same as the one shown in Pollock (2016).

In the following, new output files and some modifications done to existing output files are detailed.

3.1 List file

It is typical in MODFLOW-based programs to write a file that registers some of the configuration parameters and decisions performed by the program based on the user input. MODPATH also generate said file and for the purposes of MODPATH-RW it has been extended to report information specific to each package. This file might be useful as a summary of the simulation for successful terminations, and a source of information in case the simulation terminated unexpectedly or by some of the errors handled by the program.

The interpretation of packages is preceded by a header indicating which package is being read. For example, for the `DSP` package

```
"MODPATH-RW DSP file data"
"------------------------"
```

Although the list file still contains some of the original messages implemented in MODPATH, most of the structure of this file has been modified to report aspects related to the MODPATH-RW simulations and users are encouraged to familiarize with it.

3.2 Observations

Observations generate different output files depending on their type, providing the necessary information for users to obtain a timeseries of concentrations. In this regard, both observation types may generate up to two output files, depending on the configuration of the observation. These output files are identified in the following as `ObsRecords`, which contains the source information for postprocessing and `ObsPostprocess`, containing the concentration timeseries.

3.2.1 RESIDENT

`ObsRecords`

File is written in text-plain format when users indicate `OutputOption=0 or 1`. In the first case, the records file adopts the name `OutputFileName`. In the second, the records file name is a modified version of `OutputFileName` with prepended substring `rec_`. For clarity, a observation with `OutputFileName=obscells1.obs` and `OutputOption=1`, will generate a file called `rec_obscells1.obs`. File columns are:

```
1 : TimePointIndex          8 : CellNumber
2 : CummTimeStep            9 : Layer
3 : TrackingTime            10: RetardationFactor
4 : ParticleID              11: WaterVolume
5 : ParticleMass            12: GlobalX
6 : ParticleGroup           13: GlobalY
7 : SpeciesID               14: GlobalZ
```

`ObsPostprocess`

This file is provided in cases where the user specify `OutputOption=1 or 2`. The number of file columns are dependent on the number of species and `PostprocessOption`.

For `PostprocessOption=0`:

```
1 : TimePointIndex
2 : TimePoint
3 : CummWaterVolume
4 : ConcHistogram[NSpecies]
```

For `PostprocessOption=1`:

```
1 : TimePointIndex
2 : TimePoint
3 : CummWaterVolume
4 : ConcHistogram[NSpecies]
5 : ConcGPKDE[NSpecies]
```

3.2.2 `FLUX`

`ObsRecords`

This kind of observation follow the same `OutputOption` behavior than the discussed for resident concentrations.

File columns are

```
1 : TimePointIndex          6 : ParticleGroup
2 : CummTimeStep            7 : SpeciesID
3 : TrackingTime            8 : CellNumber
4 : ParticleID              9 : Layer
5 : ParticleMass            10: SinkFlowRate
```

`ObsPostprocess`

The only difference of this file with the one presented for resident concentrations is the meaning of the third column, which in this case is the cummulative flow-rate of the observation.

For `PostprocessOption=0`:

```
1 : TimePointIndex
2 : TimePoint
3 : CummSinkFlowRate
4 : ConcHistogram[NSpecies]
```

For `PostprocessOption=1`:

```
1 : TimePointIndex
2 : TimePoint
3 : CummSinkFlowRate
4 : ConcHistogram[NSpecies]
5 : ConcGPKDE[NSpecies]
```

3.3 Spatial GPKDE

Writing of this file is controlled by the GPKDE library. As indicated previously, `GPKDE` considers an independent grid with respect to the flow-model grid and bin indexes are with respect to the given domain origin. Similarly, the reported values by the module are interpreted as resident concentration in the case that both porosity and retardation factor were uniform in space. In case any of these is not uniform, then the reported values should be considered as total mass density. In any case, both `Density` and `Histogram` columns are considered to have the same units/dimensions.

```
1 : TimePointIndex
2 : SpeciesID
3 : iBinX
4 : iBinY
5 : iBinZ
6 : Density
7 : Histogram
```

Bibliography

Harbaugh, A. W. 2005. *MODFLOW-2005, the U.S. Geological Survey modular ground-water model: the ground-water flow process.* U.S. Geological Survey Techniques and Methods 6-A16.

Kreft, A., & Zuber, A. 1986. Comments on "Flux-Averaged and Volume-Averaged Concentrations in Continuum Approaches to Solute Transport" by J. C. Parker and M. Th. van Genuchten. *Water Resources Research*, **22**(7), 1157–1158.

Langevin, Christian D., Hughes, Joseph D., Banta, Edward, Provost, Alden, Niswonger, Richard, & Panday, Sorab. 2022. *MODFLOW 6, the U.S. Geological Survey Modular Hydrologic Model.* Description of Input and Output, Version mf6.4.1. `https://water.usgs.gov/water-resources/software/MODFLOW-6/mf6io_6.4.1.pdf`.

Parker, J. C., & van Genuchten, M. Th. 1984. Flux-Averaged and Volume-Averaged Concentrations in Continuum Approaches to Solute Transport. *Water Resources Research*, **20**(7), 866–872.

Pérez-Illanes, Rodrigo, & Fernàndez-Garcia, Daniel. 2022. Multiprocessing for the Particle Tracking Model MODPATH. *Groundwater*, Dec.

Pollock, D. W. 2016. *User guide for MODPATH Version 7—A particle-tracking model for MODFLOW.* U.S. Geological Survey Open-File Report 2016-1086.

Pollock, D. W., & Provost, A. M. 2017. *MODPATH Version 7 public repository.* (Accessed Aug 28, 2020).