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

Version 1.0.0

Rodrigo Pérez-Illanes

Daniel Fernàndez-Garcia

Barcelona
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 to the source code of MODPATH, released by the U.S. Geological Survey (USGS) (Pollock, 2016; Pollock & Provost, 2017). Besides the base functionalities, the program implements: (i) new particles displacements based on the RWPT, (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. The RWPT method requires more computational resources than the advective displacements, so intial developments considered the implementation of parallel processing with the OpenMP library (Pérez-Illanes & Fernàndez-Garcia, 2022) in order to distribute the computation load. Additional functionalities and program characteristics supporting the previous features are addressed in more detail throughout this document.

Specific topics regarding the routines for reading values from MODFLOW simulation files, or the different grid files supported by MODPATH and their characteristics, are not addressed in much detail in this report. This is because said utilities remain essentially the same than in the base code and users are refered to the MODPATH documentation for more details (Pollock, 2016).

The main motivation to extend MODPATH source code lies in the inherent interoperability with a variety of MODFLOW models, allowing the immediate integration of the solute transport functionalities with this suite of groundwater flow codes. The program is delivered with complementary online repositories and tools to assist the process of writing input files. The release considers the following repositories

- `modpath-rw`: Main source code for the program.
- `flopyrw`: `python` utilities for writing program input files extended from the `flopy` package (Bakker *et al.*, 2016).

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

Input files for MODPATH-RW follow in general a similar logic than those from the base code, so users are encouraged to follow closely the documentation of MODPATH-v7 (Pollock, 2016). Specifically, the main simulation file considers only minor changes, so only these notable differences are remarked. Most of the solute transport functionalities are grouped into specific packages, and the program will look for package-specific files while interpreting the `mpnam` file (Section 2.2). Package files configure specific aspects of a RWPT simulation like for example the protocols for time step selection, the specification of dispersion parameters, the solute sources, just to name a few. This 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 files for MODPATH-RW. These are an extension from the classes currently available at the project `flopy`[2] (Bakker *et al.*, 2016) for particle tracking simulations based on 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 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 interpreta-

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

```
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 1 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. EndpointOutputOption     | 10 20                           |10:BudgetCellNumbers[BudgetCellCount]
   ix.   ParticlesMassOption      | 1                               |11:ReferenceTimeOption
   x.    SpeciesDispersionOption  | 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:DelayingFactorOption           | 2                               |
24:DelayingFactor[NCPL] or        | CONSTANT 2.0                    |
   DelayingFactor[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.

tions or modifications are summarized in Table (2.1). Besides the MODPATH `SimulationTypes`, MODPATH-RW introduces additional interpretations for this parameter that modify the procedures for tracking particles to a version considering Random Walk (RW). By default, all of the simulation types write an endpoint file compiling the last position of particles.

In the last public version of the base code (Pollock & Provost, 2017), an additional parameter was added to the simulation file, controlling the writing of timeseries records (`TimeseriesOutputOption`). This variable defines whether records are written only for active particles or for all particles. An additional interpretation introduced here is to skip the writing of records, which is useful for simulations pursuing aggregated metrics like concentrations at observation cells or spatial density reconstruction. These are generally based on timeseries simulations, and skipping the writing of individual records removes the overhead of this operation, also the sometimes very large output files for simulations with high particle resolution. On a similar note, a new parameter (`EndpointOutputOption`) allows to control the writing of the endpoint file, for which a new alternative format has been introduced as discussed in Section (3.4).

Two additional options allow the user to indicate further information for the particle groups provided in the main simulation file. Specifically, the parameter `ParticlesMassOption` indicates whether a particle mass should be read for the particle groups given at the simulation file. Further, it also

controls whether a solute identifier should be read. In this case, the reading of the solute identifiers applies to all of the particles created in the program, even for particles created from other packages, as remarked in later sections. The solute/species identifier is in general useful for organizing multiple particle groups into a single substance for the purposes of concentration monitoring. The last new parameter in this file is the `SpeciesDispersionOption` which indicates whether mass particles are displaced with the same dispersion parameters or with species specific dispersion. Dispersion parameters are provided through the `DSP` package which is detailed in Section (2.5).

Table 2.1: New and modified parameters of the simulation file introduced by MODPATH-RW, identified with the indexation shown in Figure (2.1). By default, 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	EndpointOutputOption	int	0: Write the classical endpoint file. 1: Write the simplified endpoint file for mass particles. 2: Skip the writing of the enpoint file.
3.ix	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.x	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 coordinate the packages included on a simulation. In classical MODPATH, this file link the MODFLOW model files and the basic package (`MPBAS`). For MODPATH-RW, this file 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 specifications. The new packages are summarized in Table (2.2), and an example name file for a MODPATH-RW simulation considering a MODFLOW-6 flow 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. Some packages are allowed to contain multiple instances of the parameters specification and begin with a number indicating how many instances are included in the file. As a remark, this does not mean that the package is given multiple times in the name file.

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`	GPKDE	Smoothed density reconstruction	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 store cell porosities and default `IFACEs` for stress packages not able to store this parameter as auxiliary variable. The structure and interpretation of this file remains exactly the same as in MODPATH-v7 so no additional comments are provided here (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).

The last parameter in this file allows to select the function for random number generation. The latter is optional and by default selected by compiler. While using the `ifort` compiler some parallel scalability issues where detected while obtaining random numbers based on the intrinsic `random_number`. An external module generating random numbers with the Ziggurat method was included into the program, which is explicitly parallelized and solves the scalability issue.

```
0: TimeStepSelection | MIN_ADV_DISP
1: (CFL)             | 0.1
2: (CTDisp)          | 0.1
3: (TimeStep)        | 0.01
4: AdvectionKind     | EULERIAN
5: DimensionMask     | 1 1 0
6: (RandomGenerator) | 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.
6	`RandomGenerator`	`int`	`0`: selected by compiler. `1`: based on intrinsic `random_number`. `2`: uses the Ziggurat method.

2.5 Dispersion (DSP)

This package organizes 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. Two different alternatives for specifying dispersion parameters are implemented on this release. The first is the classical dispersion model with linear isotropic dispersivities (Fig. 2.4), interpreting pore-diffusion plus the longitudinal and transverse values. The second alternative employs the axisymmetric formulation from Lichtner *et al.* (2002), considering the vertical axis for these purposes (Fig. 2.5). In this case, besides pore-diffusion, two longitudinal dispersivities plus two transverse dispersivities are specified. Following the notation of the axisymmetric formulation, the index H indicates that the value is perpendicular to the symmetry axis (the vertical axis), and the index V indicates that the value is parallel to the symmetry axis. Regardless of the given dispersion model, parameters can be specified as domain constants or read with the free-format input of MODPATH for each model layer. More details are presented in Tables (2.4) and (2.5).

If multiple dispersion specifications are given, but `SpeciesDispersionOption=0`, then only the first 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. Each specification begins with an identificatory string name for the dispersion data, followed by the dispersion model and input format, which configures the reading of dispersion parameters.

```
0 :DspName                    |DSP1
1 :i.DspModel ii.InputFormat  |0      1
2 :DMEff                      |CONSTANT    0.000000E+00       #dmeff  layer 1
3 :AlphaL                     |CONSTANT    10.00000E+00       #alphal layer 1
4 :AlphaT                     |CONSTANT     1.00000E+00       #alphat layer 1
```

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

```
0 :DspName                    |DSP1
1 :i.DspModel ii.InputFormat  |1      1
2 :DMEff                      |CONSTANT    0.000000E+00       #dmeff   layer 1
3 :AlphaLH                    |CONSTANT    10.00000E+00       #alphalh layer 1
4 :AlphaLV                    |CONSTANT    5.000000E+00       #alphalv layer 1
5 :AlphaTH                    |CONSTANT    1.000000E+00       #alphath layer 1
6 :AlphaTV                    |CONSTANT    1.000000E-01       #alphatv layer 1
```

Figure 2.5: Illustrative structure for specification axisymmetric dispersion model, for a single layer domain.

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

Id	Parameter	Type	Values
0	DspName	string	Less than 20 characters.
1.i	DspModel	int	1: linear, isotropic. The only implemented.
1.ii	InputFormat	int	0: Domain uniform parameters. 1: Distributed parameters, free-format input.
2	DMEff	float	Effective molecular diffusion (corrected by tortuosity).
3	AlphaL	float	Longitudinal dispersivity.
4	AlphaT	float	Transverse dispersivity.

Table 2.5: Dispersivities for specification shown in Figure (2.5).

Id	Parameter	Type	Values
3	AlphaLH	float	Longitudinal dispersivity perpendicular to the vertical axis.
4	AlphaLV	float	Longitudinal dispersivity parallel to the vertical axis.
5	AlphaTH	float	Transverse dispersivity perpendicular to the vertical axis.
6	AlphaTV	float	Transverse dispersivity parallel to the vertical axis.

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. If `ParticlesMassOption!=2` and the package is not given, 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 (Fig. 2.6). In case `ParticlesMassOption=2`, the program will read the specifications and relate the `SpeciesID` given by users while defining particle groups. In this case, it is internally understood that the identifying numbers given to the 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.

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

Figure 2.6: Illustrative structure for specification of species. Parentheses remark conditional parameters.

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.

Table 2.6: Parameter details for dispersion specification shown in Figure (2.4).

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 scale of particles mass. The expected values are explicitly interpreted as resident concentrations, in all cases interpreted with the free-format input of MODPATH. This information combined with porosities, cell volumes and delaying factors, allow to estimate the total mass inside model cells. An example specification is shown in Figure (2.7) illustrating only one concentration distribution, however, the program is able to interpret multiple specifications. Details of parameters are show in in Table (2.7).

The program will enforce consistency of total mass by recalculating the value of particles mass with two different methodologies. The first integrates the solute plume at the domain level in order to obtain the total mass. Combined with the estimated total number of particles, recompute particles mass and assign this value to all particles. A second alternative allows to perform a similar procedure, but in this case at a cell level. This means that particles from different cells might have different characteristic mass. The user can specify a parameter for determining the placement of particles inside each cell. The latter can be equispaced, similar to a uniform distribution. A second alternative allows to perturbate the previous distribution by a random number proportional to half the particle spacing, known as quasi-random. A third alternative allows to place particles randomly inside each cell.

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 :i.  ParticlesMass      |500.0     0                                |
    ii. ParticlesDistrib   |1                                          |  4 :(SpeciesID)
 5 :ResidentConcentration  |INTERNAL     1  (3E15.6) -1 #ic1 layer 1   |
...                        |1.593200E+00   3.470600E+00   4.769200E+00 |
/1]                        |8.877000E+00   1.032400E+01   1.181560E+01 |
                           |1.661340E+01   1.834760E+01   2.016920E+01 |
                           |2.634600E+01   2.872420E+01   3.131140E+01 |
                           |INTERNAL     1  (3E15.6) -1 #ic1 layer 2   |
                           |0.000000E+00   0.000000E+00   0.000000E+00 |
                           |4.184800E+00   6.062600E+00   7.428200E+00 |
                           |1.189620E+01   1.346920E+01   1.506860E+01 |
                           |1.996520E+01   2.161440E+01   2.326660E+01 |
```

Figure 2.7: Example input for initial condition package, considering a structured model with $(N_{lay}, N_{row}, N_{col}) = (2, 4, 3)$. Square brackets in parameters column indicate a list of sub-specifications and parenthesis remarks conditional parameters.

Table 2.7: Parameter details for initial condition shown in Figure (2.7).

Id	Parameter	Type	Values
0	NInitialConditions	int	At least 1, determine how many are interpreted.
1	InitialConditionName	string	Less than 20 characters.
3	InitialConditionFormat	int	Read concentrations with free-format input. 0: Particles with unique mass, global mass consistency. 1: Particles with cell-specific mass, local consistency.
3.i	ParticlesMass	float	To tranform the total mass concentration into a number of particles. Determines particle resolution.
3.ii	ParticlesDistrib	int	Determines the placement of particles inside cells. 0: Particles are equispaced. 1: Particles are in quasi-random array. 2: Random locations.
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 (Langevin *et al.*, 2017) and MT3DMS (Zheng & Wang, 1999). 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 were 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 given auxiliary variable, under the assumption that these represent solute concentrations.

Figure (2.8) presents the variables structure and example values for a source specification with the `AUXILIARY` format, and the interpretation of parameters is shown in Table (2.8). 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

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

and parameters that allow to transform the concentration data into particles. The program detects automatically which flow-model cells are associated with the budget name, running over all the simulation stress periods. 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, which determine the resolution with which the source is represented. 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 program will look for an integer solute identifier 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. For these models, the value of `BudgetName` can be given as the classical package text-label or as the specific identificatory name of an instance (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 (2.9).

Table 2.8: Parameter details for specification with AUXILIARY format shown in Figure (2.8).

Id	Parameter	Type	Values
0	SourceName	string	An identificatory name for the source. Less than 20 characters.
1	SourceFormat	string	The kind of source specification. AUX/AUXILIARY (or SPEC/SPECIFIED).
2	NBudgets	int	The number of budget headers from where to extract data. At least 1.
3.i	BudgetName	string	The budget header label or the specific TXT2ID2 for MODFLOW-6.
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 2.9: 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 2.9: 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 (2.9), 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 (2.10).

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 2.10: Parameter details for specification with SPECIFIED format shown in Figure (2.9). Only parameters with different meaning than those from Table (2.9) 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. Spatial reconstruction is performed by default at the end of each timeseries step configured in the main simulation file. However, users can indicate in the package configuration file specific times where reconstruction should be performed, similar to the time point data indicated for the timeseries.

Reconstruction is performed on a regular grid, which is independent of the flow-model grid, considering reference axes aligned with the MODPATH interpretation of coordinates. The module receives the three-dimensional global particle coordinates as input data, with their respective particle mass.

Based on the grid specifications, the GPKDE 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 while calculating the histogram, then the bin size should be specified to be zero in said dimension. Additional set of parameters allow the user to perform reconstruction on a sliced manner. The latter is useful, for example, for three-dimensional problems where the horizontal extent is significantly larger than the vertical. In said case, a two-dimensional reconstruction can be performed for each model layer and in general this is the recommended approach for quasi two-dimensional problems, avoiding issues related to the reflection correction for kernels intersecting domain boundaries.

Kernels can be computed in real-time or precalculated for a set of non-dimensional bandwidths (the ratio between smoothing and cell size). A kernel database is expected to provide faster reconstruction particularly for large domains.

```
 0: i.   OutputFile             | mprwsim.gpkdeout      0     0  |
    ii.  (ColumnFormat)         |   0.0    0.0   0.0             | 1: DomainOrigin [x y z]
    iii. (FileFormat)           | 100.0   50.0   5.0   1   0.05 | 2: i.   DomainSize [x y z]
 3: BinSize [x y z]             |   2.0    2.0   2.5             |    ii.  (GridAllocFormat)
 4: i.  SlicedReconstruction    | 1      3                       |    iii. (BorderFraction)
    ii. (SlicedDimension)       | 10                             | 5: NOptLoops
 6: i.  SkipErrorConvergence    | 0     0.02                     |
    ii. (RelativeConvergence)   | 1     0       0      0         | 7: i. KDB ii.(BoundKernelSize)
 8: KDB (MinHD DeltaHD MaxHD)   | 0.5   0.1   10.0               |    iii. (IsotropicKernels)
 9: i.  InitialSmoothingFormat  | 0     2.0                      |    iv.  (AnisotropicKernelsIC)
    ii. (BinSizeFactor)         | 1                              |10: AsResidentConcentration
11: EffectiveWeightFormat       | 0                              |
12: i. (TimePointOption)        | 1     0                        |
    ii.(SkipInitialCondition)   | 3     2.0                      |13: i.  TimePointCount
14: TimePoints[TimePointCount]  | 1.0   2.0   3.0                |    ii. TimePointInterval
```

Figure 2.10: Illustrative specification for spatial reconstruction module. Parentheses remark optional parameters

Table 2.11: Parameter details for specification `GPKDE` package shown in Figure (2.10).

Id	Parameter	Type	Values
`0.i`	`OutputFile`	`string`	Less than 200 characters.
`0.ii`	`ColumnFormat`	`int`	`0`: Output bin ids and densities. `1`: Output bin ids, cell coordinates and densities. `2`: Output cell coordinates and densities.
`0.iii`	`FileFormat`	`int`	`0`: Output file as text-plain. `1`: Output file as binary.
`1`	`DomainOrigin`	`float`	Coordinates of the origin for the GPKDE grid.
`2.i`	`DomainSize`	`float`	Extent of the domain for the GPKDE grid.
`2.ii`	`GridAllocFormat`	`int`	`0`: Grid allocated according to domain. `1`: Grid adapts to the particle coordinates.
`2.iii`	`BorderFraction`	`int`	Read if `GridAllocFormat=1`. Defines buffer distance for grid allocation, added to the extext of the particle distribution.
`3`	`BinSize`	`float`	Size of the reconstruction cells, regular.
`4.i`	`SlicedReconstruction`	`int`	For models with $d > 1$ dimensions, performs reconstruction with $d-1$ dimensional kernels on each slice of `SlicedDimension`. `0`: Disabled. `1`: Enabled.
`4.ii`	`SlicedDimension`	`int`	The dimension to be considered compressed for kernels functions.
`5`	`NOptLoops`	`int`	Maximum number of bandwidth optimization loops.
`6.i`	`SkipErrorConvergence`	`int`	`0`: Break bandwidth optimization by error checks. `1`: Skip error checks and do `NOptLoops`.
`6.ii`	`RelativeConvergence`	`int`	Threshold to determine reconstruction convergence. Not read if `SkipErrorConvergence=1`.
`7.i`	`KDB`	`int`	`0`: Kernels calculated in real time for a given bandwidth. `1`: Precompute a kernel database (`KDB`) for a set of bandwidths.
`7.ii`	`BoundKernelSize`	`int`	`0`: Bound kernel size based on domain restrictions. `1`: Bound kernel size based on `MinHD`, `MaxHD` `2`: Unbounded kernel sizes.
`7.iii`	`IsotropicKernels`	`int`	`0`: Kernels are anisotropic in multidimensional problems. `1`: Kernels are isotropic.
`7.iv`	`AnisotropicKernelsIC`	`int`	`0`: Kernels are isotropic for initial condition. `1`: Kernels are anisotropic for initial condition. `2`: Kernels for initial condition follow `7.iii IsotropicKernels`.
`8`	`KDB (MinHD DeltaHD MaxHD)`	`float`	If `7.i KDB=1`, define a range of non-dimensional bandwidths for precomputing the kernel database. If `BoundKernelSize=1` will read `MinHD` and `MaxHD`.

Table 2.12: Parameter details for specification GPKDE package shown in Figure (2.10), continued.

Id	Parameter	Type	Values
9.i	InitialSmoothingFormat	int	0: Automatic initial kernel selection (Silverman, 1986). 1: Initial kernel as a factor amplifying cell size.
9.ii	BinSizeFactor	float	Amplifies cell size for initial bandwidth selection. Only read if InitialSmoothingFormat=1.
10	AsResidentConcentration	int	0: Report reconstruction as total mass density. 1: Report as resident concentration (aqueous), if possible.
11	EffectiveWeightFormat	int	Defines the protocol for processing mass particles. 0: Global effective mass to obtain histogram (Kish, 1965, 1992) 1: Average mass to obtain equivalent histogram. 2: Bandwidth selection based on real histogram. 3: Local effective histogram for bandwidth selection.
12.i	TimePointOption	int	0: Reconstruction performed at timeseries time points. 1: Read TimePointCount and TimePointInterval. 2: Read TimePointCount and TimePoints,
12.ii	SkipInitialCondition	int	0: Reconstruction performed for initial particle positions. 1: Skip reconstruction of tracking time zero.
13.i	TimePointCount	int	The number of reconstruction time points. Read if TimePointOption=1 or TimePointOption=2.
13.ii	TimeInterval	float	The interval for equispaced reconstruction stages if TimePointOption=1.
14	TimePoints	float	The array of times if TimePointOption=2.

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-format input, 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 2.11: 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 2.13: Parameter details for impermeable cells specification shown in Figure (2.11).

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 are implemented in 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 cumulative 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, users should 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`. Users may provide some additional parameters for the reconstruction process. In case of smoothed reconstruction, these are a simplified set of the options given for the spatial reconstruction package.

An example specification is shown in Figure (2.12) with parameter details indicated in Table (2.14).

```
0: ObsName                  |OBS1               |
1: ObsFormat                |RESIDENT           |
2: ObsOutputFile            |obscells1.obs      |
3: i.  OutputOption         |1    1    1        |
   ii. PostprocessOption    |10  0.01           | 4: i. NOptLoops ii. RelativeConvergence
   iii.ReconstructionOptions|1   3.0            | 5: i.  InitialSmoothingFormat
6: EffectiveWeightFormat    |3                  |    ii. BinSizeFactor
7: i.  CellInputOption      |1   2   1          |
   ii. NCells               |145                |[8 : Cells
   iii.CellFormat           |146                |\8]:
```

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

Table 2.14: Parameter details for specification of RESIDENT observation shown in Figure (2.12).

Id	Parameter	Type	Values
0	ObsName	string	Less than 20 characters.
1	ObsFormat	string	RESIDENT or RES.
2	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.
3.iii	ReconstructionOptions	int	0: Do not interpret reconstruction options [4-6]. 1: Interpret reconstruction options.
4.i	NOptLoops	int	The limit number of bandwidth optimization loops for reconstruction.
4.ii	RelativeConvergence	float	Tolerance for determining reconstruction convergence.
5.i	InitialSmoothingFormat	int	Determines the procotol for initial bandwidth selection. 0: Automatic based on Gaussian distribution (Silverman, 1986). 1: Initial bandwidth as a factor amplifying timeseries timestep.
5.ii	BinSizeFactor	float	Read if InitialSmoothingFormat=1.
6	EffectiveWeightFormat	int	Processing protocol for the particles with non-uniform mass. 0: Global effective mass to obtain histogram (Kish, 1965, 1992) 1: Average mass to obtain equivalent histogram. 2: Bandwidth selection based on real histogram. 3: Local effective histogram for bandwidth selection.
7.i	CellInputOption	int	0: Cells are given as a list of cell ids. 1: Cells read with free-format input.
7.ii	NCells	int	Number of cells if CellInputOption=0. At least 1.
7.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.

The structure of the specification for this observation has some differences with respect to the RESIDENT kind. Some new parameters are included and the rest preserve the meaning indicated in Table (2.14), although their indexaction might be modified. In the first place, the values for ObsFormat in this case adopt the values FLUX or SINK. Observation records are written each time a particle is removed due to the strong-sink approach. That is, no records will be generated if the extraction flow-rate is zero. For these observations is not necessary to predefine beforehand a

list of times where the observation records are written. In this case, the concentration timeseries is obtained from postprocessing the individual particle records. Users are provided with a set of parameters to control the histogram employed for postprocessing the particle data (Fig. 2.13; Table 2.15). Bin sizes can be selected automatically based on the statistics of arrival times or be given by the user. Notice that the histogram is employed as the base for the smoothed reconstruction, so it is relevant to provide an adequate bin size.

The reconstructed timeseries can be interpolated to a user provided time step and this leads to a visible difference in the output file of the observation. Notice that for a multispecies simulation, the automatic selection of the histogram bin size will probably lead to different values for each substance. So if no interpolation is requested, the timeseries length and step is likely different for each substance. For these cases, the observation output file is written differently than for cases where the times are the same for all substances, as indicated in Section (3.2.2).

```
0: ObsName                    |OBS2              |
1: ObsFormat                  |FLUX              |
2: ObsOutputFile              |obscells2.obs     |
3: i.  OutputOption           |1    1    1    1 |
   ii. PostprocessOption      |2   1.0  0.5      | 4: i. BinOption ii. BinParam iii. TimeStepOut
   iii.HistogramOptions       |10  0.01          | 5: i. NOptLoops ii. RelativeConvergence
   iv. ReconstructionOptions  |1   3.0           | 6: i.  InitialSmoothingFormat
6: EffectiveWeightFormat      |3                 |    ii. BinSizeFactor
7: i.  CellInputOption        |1   2   0         |
   ii. NCells                 |1   10  4         |[8 : Cells
   iii.CellFormat             |1   8   3         |\8]:
```

Figure 2.13: Example observation specification employing the format `FLUX`.

Table 2.15: Parameter details for specification of `FLUX` observation shown in Figure (2.13).

Id	Parameter	Type	Values
`3.iii`	`HistogramOptions`	`int`	`0`: Skip interpretation of histogram options `1`: Interpret histogram options
`4.i`	`BinOption`	`int`	Determines the protocol for selection of the histogram bin size `0`: Automatic selection with Scott (1979) rule. `1`: Automatic selection with Freedman & Diaconis (1981) rule. `2`: User given value in `4.ii BinParam` `3`: Infered from timeseries, assumed uniform time step.
`4.ii`	`BinParam`	`float`	Auxiliary parameter for bin size specification. If `BinOption=0 or 1` is a fraction multiplying the automatic size. If `BinOption=2` is the bin size. If `BinOption=3` is not read.
`4.iii`	`TimeStepOut`	`float`	Optional parameter determining a time step resolution for the timeseries. If given, concentrations are interpolated to the given step. All concentrations follow the same time vector. Interpolation is done for points within the limits of the original series.

Both observation formats can be combined on the same input file, as many times as the user require (see for example Fig. 2.14). 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 0
1 2 1
145
146
OBS2
FLUX
obscells2.obs
2 1 1 0
1  0.75  1.0
1 2 0
1 10 4
1 8  3
```

Figure 2.14: Example observation file with two specifications of different kind.

Chapter 3

Output

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). However, a new alternative for endpoint files is implemented, including solute related data as discussed later in this chapter.

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`. Water volume in this file is the obtained for the individual observation cells.

```
1 : TimePointIndex       (int)       8 : CellNumber     (int)
2 : CumulativeTimeStep   (float)     9 : Layer          (int)
3 : TrackingTime         (float)     10: DelayingFactor (float)
4 : ParticleID           (int)       11: WaterVolume    (float)
5 : ParticleMass         (float)     12: GlobalX        (float)
6 : ParticleGroup        (int)       13: GlobalY        (float)
7 : SpeciesID            (int)       14: GlobalZ        (float)
```

`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`. In this case the water volume is aggregated over all the cells of the observation.

For `PostprocessOption=0`:

```
1 : TimePointIndex           (int)
2 : TimePoint                (float)
3 : WaterVolume              (float)
4 : ConcHistogram[NSpecies]  (float)
```

For `PostprocessOption=1`:

```
1 : TimePointIndex           (int)
2 : TimePoint                (float)
3 : WaterVolume              (float)
4 : ConcHistogram[NSpecies]  (float)
```

```
5 : ConcGPKDE[NSpecies]      (float)
```

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     (int)          6 : ParticleGroup  (int)
2 : CumulativeTimeStep (float)        7 : SpeciesID      (int)
3 : TrackingTime       (float)        8 : CellNumber     (int)
4 : ParticleID         (int)          9 : Layer          (int)
5 : ParticleMass       (float)        10: SinkFlowRate   (float)
```

`ObsPostprocess`

The difference of this file with the one presented for resident concentrations is the meaning of the third column, which in this case is the flow-rate of the observation, agregated over all the related cells. The following structure is written for observations where the timeseries for all substances follows the same discretization. The species identifier is implicit in the order of columns.

For `PostprocessOption=0`:

```
1 : TimePointIndex           (int)
2 : TimePoint                (float)
3 : SinkFlowRate             (float)
4 : ConcHistogram[NSpecies]  (float)
```

For `PostprocessOption=1`:

```
1 : TimePointIndex           (int)
2 : TimePoint                (float)
3 : SinkFlowRate             (float)
4 : ConcHistogram[NSpecies]  (float)
5 : ConcGPKDE[NSpecies]      (float)
```

In case that the timeseries is not the same for all species, the output file is written differently. The species identifier is included explicitly in the file and data for each substance is written concatenated vertically. Similarly, simulations where the histogram bin size is selected automatically and without a requested time discretization with the parameter `TimeStepOut` will be written with the following structures.

For `PostprocessOption=0`:

```
1 : SpeciesID             (int)
2 : TimePointIndex        (int)
3 : TimePoint             (float)
4 : SinkFlowRate          (float)
5 : ConcHistogram         (float)
```

For `PostprocessOption=1`:

```
1 : SpeciesID             (int)
2 : TimePointIndex        (int)
3 : TimePoint             (float)
4 : SinkFlowRate          (float)
5 : ConcHistogram         (float)
6 : ConcGPKDE             (float)
```

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 and bin indexes defined are with respect to a given origin. Similarly, the reported values by the module are interpreted as resident concentration (dissolved) in the case that both porosity and delaying factor were uniform in space. In case any of these is non-uniform, then values should be considered as total mass density (aqueous plus sorbed). In any case, both `Density` and `Histogram` columns are considered to have the same units/dimensions. Users need to consider that the time reported in these output files corresponds to the time points selected for reconstruction, that is, it could be coincident with the timeseries points array or following the independent time points defined for the reconstruction package.

Depending on the value of the parameter `ColumnFormat` (`0.ii` in Table 2.11), the output can have different structures.

`ColumnFormat=0`: Time and species id's, bin id's and densities.

```
1 : TimePointIndex    (int)
2 : TimePoint         (float)
3 : SpeciesID         (int)
4 : idBinX            (int)
5 : idBinY            (int)
6 : idBinZ            (int)
7 : Density           (float)
8 : Histogram         (float)
```

`ColumnFormat=1`: Time and species id's, bin id's, center cell coordinates and densities.

```
1  : TimePointIndex    (int)          7  : X                (float)
2  : TimePoint         (float)        8  : Y                (float)
3  : SpeciesID         (int)          9  : Z                (float)
4  : idBinX            (int)          10 : Density          (float)
5  : idBinY            (int)          11 : Histogram        (float)
6  : idBinZ            (int)
```

`ColumnFormat=2`: Time and species id's, center cell coordinates and densities.

```
1 : TimePointIndex    (int)
2 : TimePoint         (float)
3 : SpeciesID         (int)
4 : X                 (float)
5 : Y                 (float)
6 : Z                 (float)
7 : Density           (float)
8 : Histogram         (float)
```

3.4 Endpoint for mass particles

This file is a modified version from the original endpoint file in MODPATH, without the local cell coordinates and face information, and including parameters like the particle mass and the corresponding solute identification. The columns of this file follow the structure:

```
1 : SequenceNumber       (int)      9 : InitialCellNumber  (int)      17: FinalGlobalX      (float)
2 : ParticleGroup        (int)      10: InitialLayer       (int)      18: FinalGlobalY      (float)
3 : SpeciesID            (int)      11: InitialGlobalX     (float)    19: FinalGlobalZ      (float)
4 : ParticleID           (int)      12: InitialGlobalY     (float)    20: FinalZone         (int)
5 : Status               (int)      13: InitialGlobalZ     (float)
6 : ParticleMass         (float)    14: InitialZone        (int)
7 : InitialTrackingTime  (float)    15: FinalCellNumber    (int)
8 : TrackingTime         (float)    16: FinalLayer         (int)
```

As in MODPATH, records for unreleased particles are not written to the endpoint file. Notice that the information in this file can be used for mass balance purposes. For example, users may make use of the `Status` column to detect particles removed at sink cells, eventually grouping by cells and aggregating the particle mass.

Bibliography

Bakker, M., Post, V., Langevin, C. D., Hughes, J. D., White, J. T., Starn, J. J., & Fienen, M. N. 2016. Scripting MODFLOW Model Development Using Python and FloPy. *Groundwater*, **54**(5), 733–739.

Freedman, D., & Diaconis, P. 1981. On the histogram as a density estimator:L 2 theory. *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete*, **57**(4), 453–476.

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.

Kish, L. 1965. *Survey sampling.* John Wiley & Sons.

Kish, L. 1992. Weighting for unequal P_i. *Journal of Official Statistics*, **8**(2), 183–200.

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, C. D., Hughes, J. D., Banta, E. R., Niswonger, R. G., Panday, S., & Provost, A. M. 2017. *Documentation for the MODFLOW 6 Groundwater Flow Model.* U.S. Geological Survey Techniques and Methods 6-A55.

Langevin, C. D., Hughes, J. D., Banta, E., Provost, A., Niswonger, R., & Panday, S. 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`.

Lichtner, P. C., Kelkar, S., & Robinson, B. 2002. New form of dispersion tensor for axisymmetric porous media with implementation in particle tracking. *Water Resources Research*, **38**(8), 21–1–21–16.

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, R., & Fernàndez-Garcia, D. 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).

Scott, D. W. 1979. On optimal and data-based histograms. *Biometrika*, **66**(3), 605–610.

Silverman, B. W. 1986. *Density estimation for statistics and data analysis.* Vol. 26. CRC press.

Zheng, C., & Wang, P. P. 1999. MT3DMS: A Modular Three-Dimensional Multispecies Transport Model for Simulation of Advection, Dispersion, and Chemical Reactions of Contaminants in Groundwater Systems; Documentation and User's Guide.