The Journal of Open Source Software

MF6RTM: a Python package for predictive reactive transport modeling via the MODFLOW 6 and PHREEQC APIs

Pablo Ortega-Tong [1], **Anthony Aufdenkampe** [2], **Andres Prieto-Estrada** [3], **Allan Foster** [4], **Paul Tomasula** [2], **and Lauren Mancewicz** [5]

1 INTERA Geosciences, Perth, WA, Australia **2** LimnoTech, Oakdale, MN, USA **3** INTERA INC., Houston, TX, USA **4** INTERA INC., Denver, CO, USA **5** Coastal and Hydraulics Lab, Engineer Research and Development Center, Vicksburg, MS, USA

DOI: 10.21105/joss.11173

Software

- Review
- Repository
- Archive

Editor: Mengqi Zhao

Reviewers:

- @martinvonk
- @vcantarella

Submitted: 24 December 2025
Published: 17 November 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Reactive transport modeling (RTM) plays a central role in predicting the coupled behavior of groundwater flow, solute transport, and geochemical reactions in subsurface systems (Prommer et al., 2019). This paper presents `MF6RTM` (MODFLOW 6 Reactive Transport Module), a Python package that couples `MODFLOW 6` (Langevin et al., 2024) with the geochemical code `PHREEQC-3` (Parkhurst & Appelo, 2013). The coupling is achieved through the `MODFLOW API` (Hughes et al., 2022) and `PhreeqcRM` (Parkhurst & Wissmeier, 2015), which implement the Basic Model Interface (BMI) version 2.0 (Hutton et al., 2020), extended in `MODFLOW 6` by the eXtended Model Interface (XMI), without modifying the source code of either program.

`MF6RTM` supports the core features of both codes, from contaminant migration to mineral dissolution, precipitation, and redox reactions, and can add geochemical reactions to existing `MODFLOW 6` models built with `FloPy`. Chemistry-related inputs are written as external array files, so that reactive transport models can be built, run, and history-matched within a single Python workflow, with the tools already used for flow models such as `PEST++` (White, 2018) and `pyEMU` (White et al., 2016).

State of the field

Actively developed open-source reactive transport codes include implicitly coupled simulators such as `CrunchFlow` (Steefel, 2014), `PFLOTRAN` (Hammond, 2022), and `OpenGeoSys` (Kolditz et al., 2012), and codes that couple separate transport and reaction models, such as `PHAST` (Parkhurst et al., 2010), `PHT3D` (Prommer et al., 2003), and `eSTOMP` (Nieplocha et al., 2006) (see Steefel et al. (2015) for a review).

Within the `MODFLOW` ecosystem, `PHREEQC` has been coupled to `MODFLOW-2005` in `PHT3D` (Prommer et al., 2003) and to `MODFLOW-USG Transport` in `PHT-USG` (Panday et al., 2019). `PHT3D` has been used extensively in academia and practice (Appelo & Rolle, 2010), and `PHT-USG` is increasingly used by practitioners. Both couplings, however, require modification of the source code, which imposes a heavy maintenance burden and has tied them to older software versions: they still rely on `PHREEQC-2`, and updating to `PHREEQC-3` (Parkhurst & Appelo, 2013) through `PhreeqcRM` would require substantial refactoring.

Statement of need

Despite the vast ecosystem of reactive transport simulators, to our knowledge, no open-source software comprehensively couples the current major versions of MODFLOW and PHREEQC. This gap is significant because MODFLOW remains the dominant platform for groundwater flow and transport modeling in regulatory, consulting, and applied research contexts. Existing integrated RTM codes generally require users to rebuild models in alternative frameworks, limiting their adoption for MODFLOW-based workflows. Moreover, as MODFLOW 6 and PHREEQC continue to expand in capability and adoption, keeping a coupled code current requires an approach that does not depend on changes to either source code, and that preserves transparency, extensibility, and computational efficiency. MF6RTM addresses this need by providing a fully open, API-based integration between MODFLOW 6 and PHREEQC.

Groundwater models are also increasingly expected to represent uncertainty explicitly and to support automated history matching and optimization (Langevin & Panday, 2012; White, 2017). Historically, most reactive transport workflows have relied on manual modification of input files to perform these tasks, creating a substantial burden for modelers and limiting reproducibility. Because MF6RTM exposes the geochemical inputs as array files that can be modified in the same way as MODFLOW 6 inputs, reactive parameters such as mineral amounts in the solid phase can be included directly in uncertainty analysis and multi-objective optimization. MF6RTM therefore brings reactive processes into the history-matching and uncertainty quantification workflows already applied to groundwater flow models.

Software design

Four design requirements guided the architecture: (i) reproducing established reactive transport benchmarks and agreeing with PHT3D and PHREEQC; (ii) supporting programmatic model construction, as MODFLOW workflows increasingly rely on FloPy (Bakker et al., 2016; Hughes et al., 2024); (iii) integrating with uncertainty analysis frameworks; and (iv) separating model construction from execution, so that simulation partners like PEST++ can run a model without the Python objects used to build it.

MF6RTM is therefore organized into two main modules. mup3d (Model Utility Preprocessor 3D) plays for MF6RTM a role similar to that of FloPy for MODFLOW 6: it constructs the geochemical inputs and the arrays needed to build the MODFLOW 6 files, and is used once, when the model is built. simulation reads the files written by mup3d and MODFLOW 6 and coordinates initialization, time stepping, and data exchange through the APIs, from Python or from the mf6rtm command line. Supporting modules handle array input/output (io), run configuration (config), and shared utilities (utils):

```
mf6rtm
├── mup3d
│   └── base.py
├── simulation
│   ├── solver.py
│   ├── mf6api.py
│   ├── phreeqcbmi.py
│   └── discretization.py
├── io
│   ├── externalio.py
│   └── yaml_reader.py
├── config.py
└── utils.py
```

Geochemical inputs

In `mup3d`, a geochemical system is defined with one class per PHREEQC input block: `Solutions`, `EquilibriumPhases`, `ExchangePhases`, `KineticPhases` (including rate parameters), and `SurfacePhases`. Each class holds a dictionary of numbered PHREEQC definitions and an integer array, matching the model grid, that assigns a definition to each cell; `ChemStress` holds the solutions assigned to MODFLOW 6 boundary conditions. The `Mup3d` class assembles these blocks and runs the initial equilibration in `PhreeqcRM`, which determines the transported components and their initial concentrations in mol m^{-3}. These arrays are passed to `FloPy` to build one MODFLOW 6 groundwater transport (GWT) model per component; alternatively, `Mup3d.from_mf6` replicates the transport model of an existing `FloPy` simulation for each component. `Mup3d` finally writes the PHREEQC input, the `PhreeqcRM` settings, and the run configuration read by `simulation`.

Coupling and data exchange

The `simulation` module wraps the MODFLOW 6 shared library through `modflowapi` and `PhreeqcRM` through its BMI implementation, and couples them by sequential non-iterative operator splitting. For each transport time step: (i) the transport models of all components are solved to convergence; (ii) the cell saturation is read from the flow model; (iii) the concentrations are read from MODFLOW 6, converted from mol m^{-3} to mol L^{-1}, and passed to `PhreeqcRM` as a single component-by-cell array; (iv) `PhreeqcRM` integrates the reactions over the same time step; and (v) the reacted concentrations are converted back and written to MODFLOW 6 before the time step is finalized. Data are exchanged by copy through the BMI `get_value` and `set_value` functions rather than shared memory, as each transfer requires a unit conversion, and porosity is applied by MODFLOW 6, with the `PhreeqcRM` porosity set to one. Reactions can be restricted to user-defined time steps and, in models without kinetic reactions, to cells whose relative concentration change exceeds a threshold, as kinetic reactions proceed even where transport leaves concentrations unchanged.

External input files

When external input is enabled, the initial amounts of the exchange, equilibrium, and kinetic phases are written as one array file per layer, from which the PHREEQC initialization file is regenerated at the start of each run. PEST++ can therefore modify these files like any MODFLOW 6 array, and the forward run reduces to a call to the `mf6rtm` command.

Benchmarks

Nine benchmark cases are included in the codebase, covering from mineral dissolution and precipitation fronts and cation exchange to pyrite oxidation and the kinetic biodegradation of hydrocarbons with multiple electron acceptors, and are compared against PHT3D and, in a few cases, PHREEQC. A 1D case with first-order kinetic decay is also compared against the analytical solution of Genuchten & Alves (1982), isolating the integration of kinetic reactions over each transport time step. Four documentation tutorials cover both model-building workflows.

Example 5 is presented here to demonstrate usage and verify the implementation. It reproduces the column experiment of Appelo et al. (1998), in which marine sediment containing pyrite was equilibrated with a 280 mmol L^{-1} $MgCl_2$ solution, flushed with a more dilute solution, and oxidized for four pore volumes with H_2O_2. Pyrite oxidation drives the hydrochemical evolution, accompanied by competing organic matter oxidation, kinetic calcite dissolution, cation exchange, and CO_2 sorption on goethite. The geochemical model is set up with `mup3d` as follows (abbreviated from `benchmark/ex5.ipynb`):

```
from mf6rtm import mup3d
```

```
solution = mup3d.Solutions(solutions)              # PHREEQC SOLUTION definitions
solution.set_ic(1)                                 # solution number per cell
exchanger = mup3d.ExchangePhases(exchanger_dict)
exchanger_ic = np.repeat([1, 2, 3, 4], 4)          # four zones of four cells
exchanger.set_ic(exchanger_ic.reshape(nlay, nrow, ncol))

# rate laws come from the database RATES block
kin_phases = {1: {"Pyrite":   {"m0": 4e-2, "parms": [3.42, 0.0, 0.5, 0.0]},
                  "Orgc_sed": {"m0": 10.0, "parms": [9.5e-10],
                               "formula": "Orgc_sed -1.0 C 1.0"}}}
kinetics = mup3d.KineticPhases(kin_phases)
kinetics.set_ic(1)                                 # all cells use KINETICS block 1

model = mup3d.Mup3d("ex5", solution, nlay, nrow, ncol)
model.set_database("ex5.dat")
model.set_exchange_phases(exchanger)
model.set_phases(kinetics)
model.initialize()                                 # initial equilibration in PhreeqcRM

wellchem = mup3d.ChemStress("wel")
wellchem.set_spd([2, 3])                           # injected solution per stress period
model.set_chem_stress(wellchem)

# then build one mf6 GWT model per component with FloPy
model.run()
```

MF6RTM closely reproduces the PHT3D simulation for all major ions, pH, alkalinity, and the calcite saturation index, and both models capture the main features of the measured effluent composition (Figure 1). Both models overestimate the measured Ca concentrations after about 250 mL of outflow. As this discrepancy is shared by both codes, it is attributed to the reaction network rather than to the coupling.

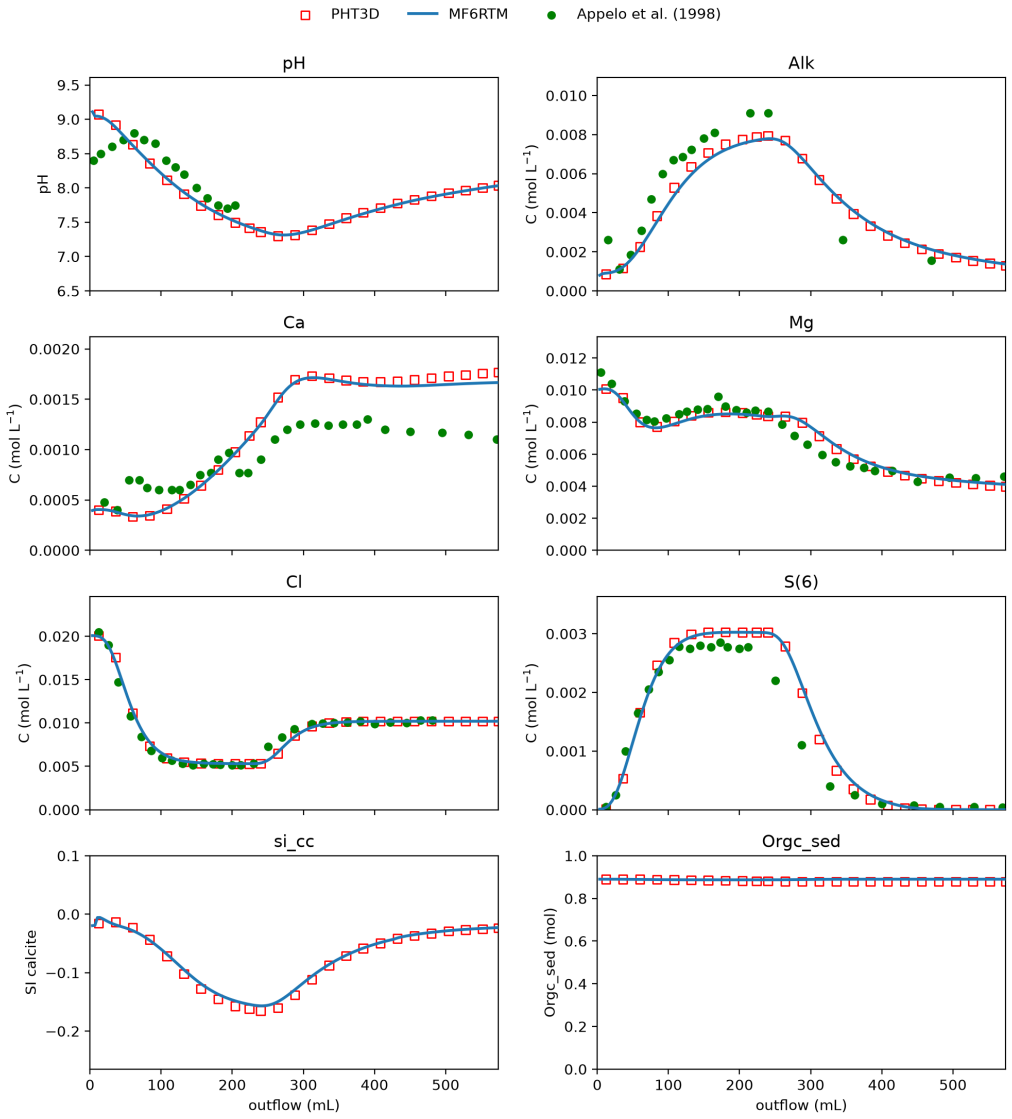


Figure 1: Comparison of effluent concentrations simulated with `MF6RTM` and `PHT3D` for Example 5, with the measured data of Appelo et al. (1998).

Research impact statement

The main contribution of `MF6RTM` is to make reactive transport modeling with `MODFLOW 6` and `PHREEQC` accessible from Python through their APIs. `MF6RTM` has been applied to a field-scale 3D model of a deep-well injection trial, including a `PEST++` setup for history matching and uncertainty analysis (Ortega-Tong & Hugman, 2026), and to a synthetic aquifer storage and recovery (ASR) case on a 3D unstructured grid (MF6RTM ASR Example).

AI usage disclosure

Generative AI tools were used in a limited capacity during the development of `MF6RTM` to draft docstrings, explore causes of software bugs, suggest code optimizations, and improve the grammar of the manuscript. During the revision stage, Claude (Anthropic; Opus 5.5), accessed through Claude Code, was also used to (i) draft docstrings, documentation tutorials,

and the contribution guidelines, (ii) restructure the package modules and write unit tests, (iii) investigate software bugs reported by the reviewers, (iv) rerun and post-process benchmark simulations, and (v) help edit the revised manuscript. The architecture of the code and the coupling scheme were designed by the authors. All AI-generated content was reviewed, edited, and tested by the authors, who take full responsibility for the software and the manuscript.

Acknowledgements

MF6RTM development was supported by INTERA INC. and its Research and Development initiative. We also thank Henning Prommer for his insights and discussions during the benchmarking of MF6RTM.

References

Appelo, C. A. J., & Rolle, M. (2010). PHT3D: A reactive multicomponent transport model for saturated porous media. *Ground Water*, *48*, 627–632. https://doi.org/10.1111/j.1745-6584.2010.00732.x

Appelo, C. A. J., Verweij, E., & Schäfer, H. (1998). A hydrogeochemical transport model for an oxidation experiment with pyrite/calcite/exchangers/organic matter containing sand. *Applied Geochemistry*, *13*, 257–268. https://doi.org/10.1016/S0883-2927(97)00070-X

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*, 733–739. https://doi.org/10.1111/gwat.12413

Genuchten, M. Th. van, & Alves, W. J. (1982). *Analytical solutions of the one-dimensional convective-dispersive solute transport equation* (Technical Bulletin No. 1661). U.S. Department of Agriculture, Agricultural Research Service. https://doi.org/10.22004/ag.econ.157268

Hammond, G. E. (2022). The PFLOTRAN Reaction Sandbox. *Geoscientific Model Development*, *15*. https://doi.org/10.5194/gmd-15-1659-2022

Hughes, J. D., Langevin, C. D., Paulinski, S. R., Larsen, J. D., & Brakenhoff, D. (2024). FloPy workflows for creating structured and unstructured MODFLOW models. *Groundwater*, *62*, 124–139. https://doi.org/10.1111/gwat.13327

Hughes, J. D., Russcher, M. J., Langevin, C. D., Morway, E. D., & McDonald, R. R. (2022). The MODFLOW Application Programming Interface for simulation control and software interoperability. *Environmental Modelling and Software*, *148*. https://doi.org/10.1016/j.envsoft.2021.105257

Hutton, E. W. H., Piper, M. D., & Tucker, G. E. (2020). The Basic Model Interface 2.0: A standard interface for coupling numerical models in the geosciences. *Journal of Open Source Software*, *5*, 2317. https://doi.org/10.21105/joss.02317

Kolditz, O., Bauer, S., Bilke, L., Böttcher, N., Delfs, J. O., Fischer, T., Görke, U. J., Kalbacher, T., Kosakowski, G., McDermott, C. I., Park, C. H., Radu, F., Rink, K., Shao, H., Shao, H. B., Sun, F., Sun, Y. Y., Singh, A. K., Taron, J., … Zehner, B. (2012). OpenGeoSys: An open-source initiative for numerical simulation of thermo-hydro-mechanical/chemical (THM/C) processes in porous media. *Environmental Earth Sciences*, *67*. https://doi.org/10.1007/s12665-012-1546-x

Langevin, C. D., Hughes, J. D., Provost, A. M., Russcher, M. J., & Panday, S. (2024). MODFLOW as a configurable multi-model hydrologic simulator. *Groundwater*, *62*. https://doi.org/10.1111/gwat.13351

Langevin, C. D., & Panday, S. (2012). Future of groundwater modeling. *Groundwater*, *50*, 334–339. https://doi.org/10.1111/j.1745-6584.2012.00937.x

Nieplocha, J., Palmer, B., Tipparaju, V., Krishnan, M., Trease, H., & Aprà, E. (2006). Advances, applications and performance of the Global Arrays shared memory programming toolkit. In *International Journal of High Performance Computing Applications* (Vol. 20). https://doi.org/10.1177/1094342006064503

Ortega-Tong, P., & Hugman, R. (2026). *Dizon 36: A reactive transport benchmark of MF6RTM against PHT3D* (Version v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.23095990

Panday, S., Mori, H., Mok, C. M., Park, J., Prommer, H., & Post, V. (2019). *PHT-USG module of the BTN Package of MODFLOW-USG Transport*. GSI Environmental. https://www.gsienv.com/software/other-models-and-tools/pht-usg-module/

Parkhurst, D. L., & Appelo, C. A. J. (2013). *Description of input and examples for PHREEQC version 3: A computer program for speciation, batch-reaction, one-dimensional transport, and inverse geochemical calculations* (Techniques and Methods No. 6-A43; p. 519). USGS. https://doi.org/10.3133/tm6A43

Parkhurst, D. L., Kipp, K. L., & Charlton, S. R. (2010). PHAST version 2 - a program for simulating groundwater flow, solute transport, and multicomponent geochemical reactions. *Modeling Techniques, Book 6*. https://doi.org/10.3133/tm6a35

Parkhurst, D. L., & Wissmeier, L. (2015). PhreeqcRM: A reaction module for transport simulators based on the geochemical model PHREEQC. *Advances in Water Resources*, *83*, 176–189. https://doi.org/10.1016/j.advwatres.2015.06.001

Prommer, H., Barry, D. A., & Zheng, C. (2003). MODFLOW/MT3DMS-based reactive multicomponent transport modeling. *Ground Water*, *41*, 247–257. https://doi.org/10.1111/j.1745-6584.2003.tb02588.x

Prommer, H., Sun, J., & Kocar, B. D. (2019). Using reactive transport models to quantify and predict groundwater quality. *Elements*, *15*, 87–92. https://doi.org/10.2138/gselements.15.2.87

Steefel, C. I. (2014). CrunchFlow: User's manual. *Department of Energy Report*, *6*.

Steefel, C. I., Appelo, C. A. J., Arora, B., Jacques, D., Kalbacher, T., Kolditz, O., Lagneau, V., Lichtner, P. C., Mayer, K. U., Meeussen, J. C. L., Molins, S., Moulton, D., Shao, H., Šimůnek, J., Spycher, N., Yabusaki, S. B., & Yeh, G. T. (2015). Reactive transport codes for subsurface environmental simulation. *Computational Geosciences*, *19*, 445–478. https://doi.org/10.1007/s10596-014-9443-x

White, J. T. (2017). Forecast first: An argument for groundwater modeling in reverse. *Groundwater*, *55*, 660–664. https://doi.org/10.1111/gwat.12558

White, J. T. (2018). A model-independent iterative ensemble smoother for efficient history-matching and uncertainty quantification in very high dimensions. *Environmental Modelling & Software*, *109*, 191–201. https://doi.org/10.1016/j.envsoft.2018.06.009

White, J. T., Fienen, M. N., & Doherty, J. E. (2016). A Python framework for environmental model uncertainty analysis. *Environmental Modelling and Software*, *85*. https://doi.org/10.1016/j.envsoft.2016.08.017