Alquimia v1.0: A generic interface to biogeochemical codes– A tool for interoperable development, prototyping and benchmarking for multiphysics simulators

Sergi Molins[1], Benjamin J. Andre[1], Jeffrey N. Johnson[1,8], Glenn E. Hammond[2], Benjamin N. Sulman[3], Konstantin Lipnikov[4], Marcus S. Day[5], James J. Beisman[3], Daniil Svyatsky[4], Hang Deng[6], Peter C. Lichtner[7], Carl I. Steefel[1], and J. David Moulton[4]

[1]Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley CA 94720
[2]Pacific Northwest National Laboratory
[3]Oak Ridge National Laboratory
[4]Los Alamos National Laboratory
[5]National Renewable Energy Laboratory
[6]Peking University
[7]University of New Mexico
[8]Cohere Consulting, LLC, Seattle, WA

Correspondence: Sergi Molins (smolins@lbl.gov)

Abstract. Alquimia v1.0 is a generic interface to geochemical solvers that facilitates development of multiphysics simulators by enabling code coupling, prototyping and benchmarking. The interface enforces the function arguments and their types for setting up, solving, serving up output data, and other common auxiliary tasks, while providing a set of structures for data transfer between the multiphysics code driving the simulation and the geochemical solver. Alquimia relies on a single-cell approach that permits operator splitting coupling and parallel computation. We describe the implementation in Alquimia of two widely-used open-source codes that perform geochemical calculations, PFLOTRAN and CrunchFlow. We then exemplify its use for the implementation and simulation of reactive transport in porous media by two open-source flow and transport simulators, Amanzi and Parflow. We also demonstrate its use for the simulation of coupled processes in novel multiphysics applications including the effect of multiphase flow on reaction rates at the pore scale with openFOAM, the role of complex biogeochemical processes in land surface models such as the E3SM Land Model (ELM), and the impact of surface-subsurface hydrological interactions on hydrogeochemical export from watersheds with the Advanced Terrestrial Simulator (ATS). These applications make it apparent that the availability of a well-defined yet flexible interface has the potential to improve the software development workflow, freeing up resources to focus on advances in process models and mechanistic understanding of coupled problems.

1 Introduction

Numerical modeling has become an integral part of the investigation of some of the world's most pressing environmental challenges, such as climate change, pollution prevention, contaminant remediation or nuclear waste management (Steefel et al., 2015; Li et al., 2017). For an accurate system representation, models must consider the gamut of processes affecting mass balances and underlying biogeochemical transformations. Different models exist for representing biogeochemical processes including aqueous complexation, mineral dissolution-precipitation, surface sorption, and microbiologically-mediated reactions. The mathematical expressions for biogeochemical reaction models are diverse and generally express nonlinear relationships with complex parameterizations between primary variables such as concentrations (Steefel et al., 2015).

Multicomponent reactive transport codes couple biogeochemical models with solvers for flow and transport and other relevant processes such as heat transfer or geomechanics (Steefel and MacQuarrie, 1996; Steefel et al., 2005, 2015). Many of these codes are the legacy of years of development and research, a period over which model complexity has increased incrementally. Further development is driven by continued advances in process model descriptions across spatial scales and hydrological domains, from single pores to the subsurface reservoirs or watersheds. There is also a growing need to expand the role of reactive transport to explore nonlinear interactions among the atmosphere, hydrosphere, biosphere, and geosphere in combination with other community models such as land surface models (Li et al., 2017; Sulman et al., 2024). In parallel, hardware architectures evolve continuously and new numerical approaches become available for application. Jointly, these developments enable the simulation of tighter process coupling with increasing resolution and mechanistic detail, but they also demand continuous code development and, at times, refactoring. This increasing degree of sophistication also presents some challenges. How does one develop, test and ultimately incorporate biogeochemical capabilities in codes for new applications? How does one ensure the validity of a coupled simulator formulation prior to application? How does one continuously develop code to incorporate new numerical approaches or to take advantage of new computational resources?

Often the complexity of implementing a comprehensive and flexible treatment of biogeochemistry is a significant obstacle to the development of new biogeochemical capabilities. As a result, this step is often circumvented by coupling flow and transport codes to existing biogeochemical codes, an approach widely used, e.g. HYTEC+CHESS (van der Lee et al., 2003), Chombo-Crunch (Molins et al., 2012), Comsol+PHREEQC (Nardi et al., 2014; Jara et al., 2017), or ParCrunchFlow (Beisman et al., 2015). The use of PHREEQC as geochemical solver in this role for multiple codes, e.g. PHT3D (Prommer and Post, 2010), Hydrus/HPx (Simunek et al., 2013) and PHAST (Parkhurst et al., 2010), has led to the development of dedicated coupling tools such as IPhreeqc (Charlton and Parkhurst, 2011) or PhreeqcRM (Parkhurst and Wissmeier, 2015). However, these tools are specific to PHREEQC and thus tied to its capabilities.

Model development entails (among other tasks) prototyping, implementing and verifying the new coupled capabilities. Prototyping makes it possible to evaluate approach feasibility and often benefits from using high-level weakly-typed languages such as Python. Implementation requires a clear, well-documented description of the data structures and function calls that must allow enough flexibility such that a broad range of applications is possible. Ensuring the validity of complex coupled models involves intercomparison studies where multiple codes solve the same problem (Steefel et al., 2015; Maxwell et al., 2014; Molins et al., 2020).

In this manuscript we introduce Alquimia, a new open-source software library that provides a generic interface to existing biogeochemical capabilities. This software is intended to facilitate interoperable code development by exposing tried-and-true biogeochemical capabilities in existing software. Sec. 2.1 describes a general formulation of the problems the interface is designed for, and Sec. 3 the functions and data structures of the interface along with software design approaches. The implementation and use of Alquimia are given by the way of examples (Sec. 4). First, the geochemical capabilities in the open-source reactive transport codes PFLOTRAN and CrunchFlow are implemented in Alquimia. Then, these geochemical capabilities are made available for the simulation of reactive transport in porous media in two codes, Amanzi and ParFlow. Because Alquimia allows for different geochemical codes to share a common flow and transport solver, and therefore the same spatial discretization, time stepping control, and coupling schemes, it may be a useful tool for multi-way model intercomparison. Sec. 5 includes examples of how Alquimia has enabled incorporation of geochemical capabilities to codes for a range of applications, including prototyping land-surface processes in Python as well as in high-performance computing simulations using an OpenFOAM-based code and the Advanced Terrestrial Simulator (ATS).

2 Model description

2.1 Mass balance equations

The mass balance of each aqueous species may be written as

$$\frac{\partial \theta c_i}{\partial t} = L(c_i) + r_i, \qquad i = 1, N_s \tag{1}$$

where c_i is the concentration of species *i* (mass per unit water), θ is the volumetric water content, $L(c_i)$ represents the transport operator and r_i is the contribution of reactions to the mass balance of species *i*. While $L(c_i)$ represents the solute transport operator here, it may be also read as a more general operator that includes other processes.

Equilibrium aqueous complexation reactions make it possible to rewrite Eq. 1 as

$$\frac{\partial \theta \Psi_i}{\partial t} = L(\Psi_i) + R_i, \qquad i = 1, N_c \tag{2}$$

where the aqueous concentration of each component (Ψ_i) is defined as the sum of the concentration of a primary species and a set of secondary species (Steefel and MacQuarrie, 1996)

$$\Psi_i = c_i + \sum_{j=1}^{N_x} \nu_{ij} c_j, \qquad i = 1, N_c \tag{3}$$

where ν_{ij} is the stoichiometric coefficient of component i in reaction j and R_i is the contribution of kinetic reactions to the mass balance of component *i*.

This approach reduces substantially the number of governing equations and unknowns from $N_s = N_c + N_x$ species in Eq. 1 to N_c components in Eq. 2 by using the mass of law action equations of the N_x equilibrium reactions

$$c_j = \frac{\prod_{i=1}^{N_c} (\xi_i c_i)^{\nu_{ij}}}{\xi_j K_j}, \qquad j = 1, N_x \tag{4}$$

where ξ_i and ξ_j are the activity coefficient of primary and secondary species, and K_j is the equilibrium constant of reaction j.

Mass action law equations for heterogeneous equilibrium reactions follow a similar form (Steefel et al., 2015, e.g.). When these are considered, the mass of each component ($\Psi_{t,i}$) also includes the mass present as a mineral ($\Psi_{m,i}$) or sorbed/exchanged on a surface (Ψ_s, i)

$$\Psi_{t,i} = \Psi_i + \Psi_{m,i} + \Psi_{s,i}, \qquad i = 1, N_c \tag{5}$$

The kinetic reactions rates are calculated as functions of primary variables such as concentrations (c_i) and sets of intrinsic parameters (p_k) that are specific to each reaction k

$$R_i = \sum f_k(c_i, p_k), \qquad i = 1, N_c \tag{6}$$

The f_k functions take different mathematical forms for different reaction types. Different codes may implement somewhat different formulations or allow for generalized formulations (Mayer et al., 2002, e.g.) and custom rate expressions (Hammond, 2022). Examples of established formulations include the transition-state-theory-type (TST) rate law for mineral-dissolution-precipitation or the Monod-type rate expression for microbially-mediated reduction-oxidation reactions. The interface presented in Sec. 3 does not stipulate any specific mathematical form. Hence, we do not provide a specific form for f_k here. In Sec. 4.3, we compare codes when well-established reaction models are used in different geochemical engines. In Sec. 5, we give some specific reaction rate expressions connected to example applications.

2.2 Coupling approaches

The set of equations for all components along with the geochemical equilibrium and kinetic equations (Eqs. 2-6) (e.g., Steefel et al., 2015), is in general a coupled system of nonlinear equations. Two broad approaches are used to solve this system. The global implicit approach entails the simultaneous solution of the coupled system including reactions and transport, often directly substituting Eqs. 4, 6 in Eq. 2. In contrast, operator splitting consists of, first, simulating transport for each component (and phase) separately, and second, updating the concentrations by solving the geochemical equations. Mathematically, the method can be represented as a two-step sequential process consisting of a transport step (tr)

$$\frac{(\theta^{tr}\Psi_i^{tr} - \theta^n \Psi_i^n)}{\Delta t} = L(\Psi_i^n) \tag{7}$$

followed by a reaction step

$$\frac{\theta^{tr}(\Psi_i^{n+1} - \Psi_i^{tr})}{\Delta t} = R_i^{n+1} \tag{8}$$

where the reaction step includes the time-discretized form of the component balance over the different phases (Eq. 5), equilibrium aqueous speciation (Eq. 3), and kinetic reactions (Eq. 6)

$$\begin{aligned}
\Psi_{t,i}^{n+1} &= \Psi_i^{n+1} + \Psi_{m,i}^{n+1} + \Psi_{s,i}^{n+1} \\
\Psi_i^{n+1} &= c_i^{n+1} + \sum_{j=1}^{N_x} \nu_{ij}^{n+1} c_j^{n+1} \\
R_i^{n+1} &= f_k(c_i^{n+1}, p_k)
\end{aligned} \tag{9}$$

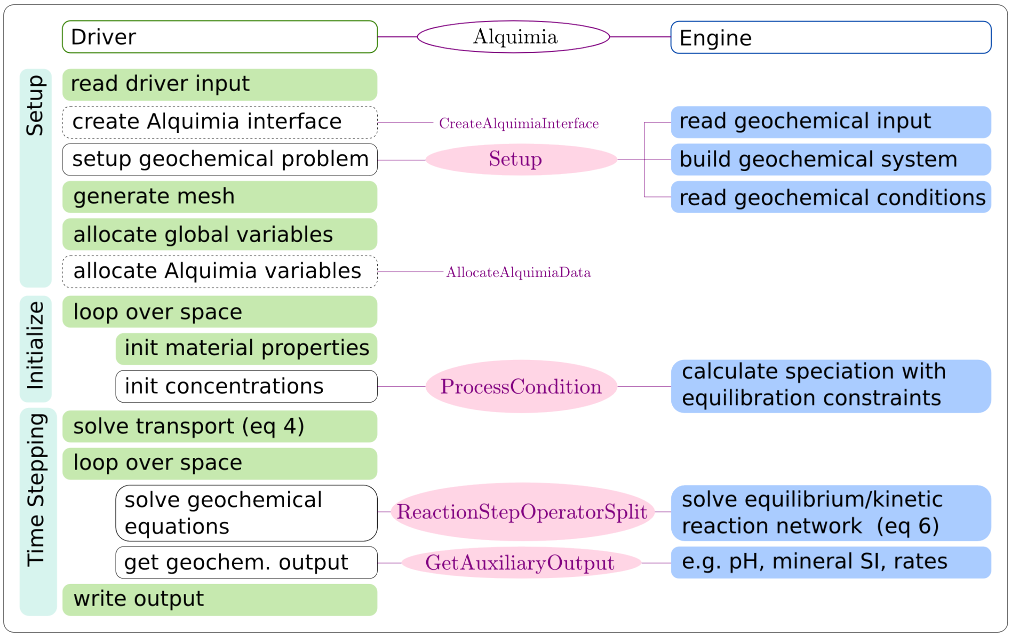

Figure 1. Example of a workflow in a code coupling a transport driver (left) and a geochemical engine (right) via Alquimia indicating the responsibilities for each the driver (green), Alquimia (pink) and engine (blue) codes. Calls to Alquimia functions in the driver code (unfilled boxes) replace development of native capabilities. Calls to Alquimia's utility library are indicated with dashed lines in the bounding box. The prefix Alquimia is omitted from Alquimia functions for clarity (i.e. *AlquimiaSetup, AlquimiaProcessCondition, AlquimiaReactionStepOperatorSplit, AlquimiaGetAuxiliaryOutput*).

While the transport step requires the solution over the entire domain, the geochemical equations can be solved independently within each cell. By separating the problem into two steps, operator splitting allows for solving the transport problem with linear solvers, confining the nonlinearity to the geochemical problem within a single cell. The time step size, however, is limited by the Courant criterion to avoid operator splitting error (Steefel and MacQuarrie, 1996).

3 Interface description

3.1 Functions

The Alquimia interface is designed to act as a generic, intermediary layer between a code that solves Eq. (7) and a code that solves Eq. (8). We will refer to the former code as the *driver* and the latter as the *engine* (Fig. 1).

The *driver* is the code that drives the simulation, handles the spatial description of the problem, including the meshing and spatial discretization, and solves Eq. (7). It is responsible for managing global variable storage, including reading spatially and/or temporally varying material properties, looping through space, managing time stepping, unpacking and moving data from the mesh dependent storage into data transfer containers, and input/output (I/O) operations.

The geochemical *engine* defines the geochemical problem and solves Eq. 8 at each point in space independently. It is responsible for reading the geochemical reaction data, managing the geochemical system (e.g. reading the thermodynamic database), reading and processing the initial and boundary solutions (e.g. equilibrating the solution with specific minerals, or a pH value), and providing access to geochemical data for output (pH, mineral saturation indices, reaction rates).

The Alquimia interface itself handles the engine-dependent implementation for setting up, processing the speciation constraints, performing the reaction step solution, and shutting down. Alquimia does not do any geochemical calculations, but does perform any unit conversions required by each engine. At the start of each operation, it unpacks data from the Alquimia data transfer containers provided by the engine, and packages them into the correct format for that engine. Then, calls to engine subroutines are made to perform the appropriate calculations. At the end of the operation, it packages the results back into the Alquimia containers for use by the driver.

3.2 Software

Alquimia has two parts: (1) an engine-independent application programming interface (API) consisting of all relevant functions, data structures, constants, and their respective types (Table 1) and (2) an optional utility library.

The API works by enforcing the arguments and types of the geochemical subroutines using a single-cell model. That is, the calls to the geochemical solver are carried out for a single element of a given spatial discretization of a driver, and thus are made within a loop over space.

The main calls in the workflow include *AlquimiaSetup*, *AlquimiaProcessCondition*, and *AlquimiaReactionStepOperatorSplit*. *AlquimiaSetup* initializes the engine by reading the geochemical engine's input file and database. This builds the geochemical system, sets the values of the reaction parameters, and generates a list of aqueous solutions with appropriate equilibration constraints. *AlquimiaProcessCondition* performs the speciation calculations on this list of aqueous solutions to obtain initial and boundary concentrations as needed by the engine, solving a steady-state form of Eq. (8), with additional constraints such as fixed species concentration or pH values, and charge balance or mineral equilibrium. *AlquimiaReactionStepOperatorSplit* performs the solution of the geochemical problem (Eq. 8).

Variables that may change in the engine with each call to reaction time-stepping are part of Alquimia's *AlquimiaState* data structure, including porosity, fluid density and pressure and the total concentrations (Ψ_i , $\Psi_{m,i}$, and $\Psi_{s,i}$), the reactive surface areas of the minerals, the surface site density for sorption and the cation exchange capacity. Variables that do not change in the engine are part of the *AlquimiaProperties* data structure, including water saturation and cell volume and miscellaneous reaction parameters (sorption constants, Freundlich sorption exponents, Langmuir sorption coefficients, mineral rate constants, and kinetic reaction rate constants). The units used for each set of variables are required by the interface as outlined in the documentation. This implies that the driver must supply in the values in these units but it is the responsibility of the Alquimia interface implementation of each engine to perform the necessary conversions to the engine's internal units.

Other structures contain information about the available functionality in the engine (*AlquimiaFunctionality*) (e.g. is porosity updated by the engine?), the status of the geochemistry engine after the last operation (*AlquimiaEngineStatus*) (e.g. did the solution converge for the time step?), data such as names of all species (*AlquimiaProblemMetaData*), and data for output

Table 1. Summary of Alquimia's API (data transfer containers, functions, constants) and C utilities library

Data Transfer Containers	
struct: AlquimiaVectors	struct: AlquimiaEngineFunctionality
void pointer: engine_state	struct: AlquimiaProblemMetaData
struct: AlquimiaSizes	struct: AlquimiaAuxiliaryOutputData
struct: AlquimiaState	struct: AlquimiaGeochemicalCondition
struct: AlquimiaProperties	struct: AlquimiaAqueousConstraint
struct: AlquimiaAuxiliaryData	struct: AlquimiaMineralConstraint
struct: AlquimiaEngineStatus	
Functions	
void AlquimiaSetup	void AlquimiaProcessCondition
void AlquimiaShutdown	void AlquimiaReactionStepOperatorSplit
void AlquimiaGetEngineMetaData	void AlquimiaGetAuxiliaryOutput
Constants	
error codes	
string lengths	
strings	
C Utilities Library	
struct: AlquimiaInterface	void AllocateAlquimiaXXX
void CreateAlquimiaInterface	void FreeAlquimiaXXX
struct: AlquimiaData	void PrintAlquimiaXXX
void AllocateAlquimiaData	

purposes (*AlquimiaAuxiliaryOutputData*). Another structure (*AlquimiaAuxiliaryData*) is used to store data necessary for the engine, the type of which is known but on which the driver should not do any operation. These data include for example the initial guesses for the next nonlinear solution, and thus the driver must return them on the next call and write them to checkpoint files. Last, a standalone pointer variable contains all the persistent internal state data for the chemistry engine that is not mesh dependent and can be reinitialized from the input file upon restart (*AlquimiaEngineState*).

Generally, the parameter values of the geochemical model are set by the engine. This simplifies code development on the driver side as the engines already have facilities to read in these parameters. However, there are cases where finer-grained control of certain parameter values by the driver. For example, in parameter estimation simulations or inverse problems that involve varying these values, it may be easier to control model parameters from the driver rather than having to write scripts that change the values in engine input files. For this purpose, Alquimia provides the ability to control a limited set of reaction parameters. These are part of the *AlquimiaProperties* data structure and include the linear sorption constant, the Freundlich sorption exponent, the Langmuir sorption coefficient, the mineral dissolution/precipitation kinetic rate constant, and the aqueous kinetic reaction rate constant. A single flag controls this behavior. When turned on ("hands-on" mode), Alquimia will use the parameter values coming from the driver for each cell at every time step to populate the appropriate data holders in the engine. Otherwise ("hands-off" mode), the default behavior is to let the engine control these parameters. While enabling this option gives the driver more control over some engine parameters, this approach is involved in terms of coding and must be used with caution, especially because it may expand the capabilities of the engines. For example, one can specify a different value of the rate constants in each grid cell, which in general is not an option available in most geochemical models.

The second part of the Alquimia library is a C utility library that contains reusable code for common tasks such as allocating memory, printing data and other miscellaneous auxiliary tasks. These are optionally used in driver codes to facilitate implementation of these common tasks.

For wide compatibility with mixed language programming, Alquimia is implemented in the C language as it offers the most flexibility to mix with other languages, including C++, Python, or Fortran. Examples of this flexibility are given in Secs. 4 and 5. The two engines currently available in Alquimia are implemented in Fortran (Sec. 4.1).

3.3 Development practices

Alquimia adheres to best practices set forth by the Extreme-scale Scientific Software Development Kit (xSDK) (Bartlett et al., 2017). Among other things, this means that Alquimia uses a CMake-based build system, provides a comprehensive test suite, provides a documented, reliable way to contact the development team, and has an accessible repository (https://github.com/LBL-EESA/alquimia-dev, 2024).

The required parts of the Alquimia are compiled using CMake-based build system into libalquimia_c.a and libalquimia_fortran.a. Semantic versioning is used for its public API. The source code and tagged releases are hosted in a GitHub repository, publicly available via a three-clause BSD licence. A test suite is provided that is based on two simple driver codes: one for batch geochemistry calculations and another for coupled reactive transport calculations. Both the build system and the test suite are included in the automated continuous integration framework available from GitHub Actions (GitHub, 2024), which is triggered by pull requests and used as condition for their approval. Alquimia is documented using restructured text files included in the source distribution and may be built and exported to different formats using Python's Sphinx package (Sphinx, 2024).

4 Implementation and use

The implementation and use of Alquimia are illustrated here by describing selected examples of the two tasks that Alquimia separates for engine and driver codes. One is the implementation of the geochemical calls in driver codes (e.g. left column in Fig. 1). This implementation is independent of the engines available; it only depends on Alquimia's data structures and call signatures. The second task is the implementation of the engine-specific function calls in Alquimia for a given engine (e.g. right column in Fig. 1). This also includes the necessary transfer of data between Alquimia data transfer containers and the engine's internal data structures. This implementation is independent of the use any driver makes of Alquimia and does not need to be repeated every time the interface is implemented in a new driver code. That is, if a new engine is added, no changes are needed in any driver that uses Alquimia to make use of these new engine's capabilities. This allows for performing simulations using the same driver replacing the geochemical engine. We use this feature to compare results from simulations performed using different driver-engine combinations of the codes presented in what follows.

4.1 Geochemical engines

The widely-used open-source codes PFLOTRAN and CrunchFlow have been implemented as engines in Alquimia. PFLOTRAN is an open-source, massively parallel multiscale and multiphysics code for subsurface multiphase flow, reactive transport, geomechanics and geophysics applications (Hammond et al., 2014; Lichtner et al., 2015; Jaysaval et al., 2023). CrunchFlow is an open-source software package for simulating reactive transport (Steefel et al., 2015). Although both codes also solve for flow and transport processes and are known for implementing the global implicit approach, they also give the user the possibility of running reactive transport simulations in operator splitting mode, e.g. Steefel and MacQuarrie (1996). This facilitated the isolation of the geochemical capabilities of these codes for implementation in Alquimia. Although this applies to *AlquimiaSetup*, *AlquimiaProcessCondition*, and *AlquimiaReactionStepOperatorSplit*, we focus here on the latter function for brevity to exemplify the steps to implement engine capabilities in Alquimia.

The implementation of *AlquimiaReactionStepOperatorSplit* for both engines follows essentially the same steps. These include copying the data from the transfer containers, passing the initial guesses to the appropriate variables, performing the iterative nonlinear solve of the geochemical problem, and upon checking for convergence updating mineral concentrations (Code blocks 1-2). However, the details of the implementation differ somewhat owing to the differences between the two codes.

Code Block 1. Selected sections of Alquima's implementation of the PFLOTRAN operator splitting step in *AlquimiaReactionStepOperatorSplit*. Lines starting with "!..." indicate portions of the code omitted for brevity. Some sections have been edited for legibility. The reader is directed to the actual code for full details.

```fortran
subroutine ReactionStepOperatorSplit(pft_engine_state, &
     delta_t, properties, state, aux_data, status)
  ! ... include and "use module" statements, and variable declarations
  ! ... assigning of the target of a C pointer to the Fortran engine_state pointer, check for
    integrity
  reaction => engine_state%reaction
  call CopyAlquimiaToAuxVars(copy_auxdata, engine_state%hands_off, &
       state, aux_data, properties, &
       reaction, engine_state%global_auxvar, &
       engine_state%material_auxvar, engine_state%rt_auxvar)
  ! copy free ion primaries into initial guess array
  allocate(guess(reaction%ncomp))
  do i = 1, reaction%naqcomp
     guess(i) = engine_state%rt_auxvar%pri_molal(i)
  enddo
  do i = 1, reaction%immobile%nimmobile
     guess(i+reaction%offset_immobile) = engine_state%rt_auxvar%immobile(i)
  enddo
  ! perform batch calculation
  vol_frac_prim = 1.0
  engine_state%option%tran_dt = delta_t
  call RReact(guess, engine_state%rt_auxvar, engine_state%global_auxvar, &
       engine_state%material_auxvar, num_newton_iterations, &
       reaction, natural_id, engine_state%option, &
       PETSC_FALSE, PETSC_FALSE, ierror)
  deallocate(guess)
  ! check for errors, copy solution back to alquimia data struct
  if (ierror /= 1) then
     call RUpdateKineticState(engine_state%rt_auxvar, engine_state%global_auxvar, &
          engine_state%material_auxvar, engine_state%reaction, engine_state%option)
     call CopyAuxVarsToAlquimia( &
          engine_state%reaction, &
          engine_state%global_auxvar, &
          engine_state%rt_auxvar, &
          porosity, &
          state, aux_data)
  else
    ! ...
  endif
end subroutine ReactionStepOperatorSplit
```

PFLOTRAN uses an object-oriented programming model introduced in a code refactoring (Hammond, 2022), while CrunchFlow uses a legacy procedural modular programming. These two different approaches require different ways of storing and manipulating engine data within Alquimia.

In PFLOTRAN, most data structures needed to describe the geochemical problem are passed as arguments and thus must be included as part of the *AlquimiaEngineState* in Alquimia. This include those that contain geochemical reaction data such as stoichiometric coefficients, *engine_state%reaction*, global variables such as aqueous saturation, *engine_state%global_auxvar*, reactive transport variables such as total concentrations, *engine_state%rt_auxvar*, or material properties such as porosity, *engine_state%material_auxvar*. As shown in Code Block 1, handling this within Alquimia is straigthforward using *AlquimiaEngineState* and also makes it for easy-to-maintain code. When new variables are added in PFLOTRAN the Alquimia interface does not need to change; only if the capabilities of Alquimia are expanded and there are new variables that need to be passed explicitly, the interface require modification. These changes would mostly be limited to *CopyAlquimiaToAuxVars* and *CopyAuxVarsToAlquimia*, which are helper subroutines that copy data from Alquimia transfer containers to the engine's variables and back.

CrunchFlow relies on global variables declared in modules, which are dynamically allocated upon initialization from the inputs. This requires that these modules are included in the Alquimia interface. Examples include aqueous saturation (*satliq*) from the transport module, porosity (*por*) from the medium properties module, or total concentrations (*sn*) from the concentration module (see *CopyAlquimiaToAuxVars* subroutine in *crunch_alquimia_interface.F90*). By contrast, CrunchFlow passes the dimensions of the geochemical problem from high-level subroutines to low-level subroutines, thus the *AlquimiaEngineState* data structure is used to store them (e.g. *engine_state%ncomp*, *engine_state%nspec*, *engine_state%nkin*, among other geochemical sizes).

While Alquimia's approach to engine data sometimes introduces more detail in the code, its flexibility allows Alquimia to accommodate the needs of very different engines. Ultimately, this makes it possible for example for the subroutine ReactionStepOperatorSplit to share the same arguments for different engines (compare lines 1 and 2, Code Blocks 1 and 2).

Code Block 2. Selected sections of Alquimia implementation of the CrunchFlow operator splitting step in *AlquimiaReactionStepOperatorSplit*. Lines starting with "!..." indicate portions of the code omitted for brevity. Some sections have been edited for brevity. The reader is directed to the actual code for full details.

```
1:  subroutine ReactionStepOperatorSplit(cf_engine_state, &
2:    delta_t, properties, state, aux_data, status)
3:    ! ... "use module" statements and variable declarations
4:    ! ... assigning of the target of a C pointer to the Fortran engine_state pointer, check for
        integrity
5:    call CopyAlquimiaToAuxVars(copy_auxdata,engine_state%hands_off,state,aux_data,properties,&
6:     engine_state%ncomp,nspec,engine_state%nkin,engine_state%nrct,engine_state%ngas,&
7:     engine_state%nexchange,engine_state%nsurf,engine_state%ndecay,engine_state%npot,&
8:     engine_state%nretard)
9:    ! ...
10:   delt = delta_t / secyr
11:   newtmax = 0
12:   jx=1; jy=1; jy=1
13:   CALL keqcalc2(ncomp,nrct,nspec,ngas,nsurf_sec,jx,jy,jz)
14:   IF (igamma == 3) THEN
15:     IF (Duan) THEN
16:       CALL gamma_co2(ncomp,nspec,ngas,jx,jy,jz)
17:     ELSE
18:       CALL gamma(ncomp,nspec,jx,jy,jz)
19:     END IF
20:   END IF
21:   CALL AqueousToBulkConvert(jx,jy,jz,AqueousToBulk)
22:   CALL os3d_newton(engine_state%ncomp,engine_state%nspec,engine_state%nkin,nrct,engine_state%ngas,&
23:     engine_state%ikin,engine_state%nexchange,engine_state%exch_sec,engine_state%nsurf,&
24:     engine_state%nsurf_sec,engine_state%npot,engine_state%ndecay,engine_state%neqn,&
25:     engine_state%igamma,delt,engine_state%corrmax,engine_state%jx,engine_state%engine_state%jy,&
26:     engine_state%jz,iterat,icvg,engine_state%nx,engine_state%ny,engine_state%nz,time,AqueousToBulk)
27:   CALL SpeciesLocal(ncomp,nspec,jx,jy,jz)
```

```
  CALL totconc(ncomp,nspec,jx,jy,jz)
  ! ...
  if (icvg == 1) then
  ! not converged, do nothing
    ! ... recover appropriate initial guesses
  else
    CALL mineral_update(nx,ny,nz,nrct,delt,dtnewest,ineg,jpor,deltmin)
    ! ...
  end if
  ! ... update variables holding previous time step solutions with new
  ! send state variables back to Alquimia state with solution of this solve
  call CopyAuxVarsToAlquimia(engine_state%ncomp,engine_state%nspec,engine_state%nkin,&
    engine_state%nrct,engine_state%ngas,engine_state%nexchange,engine_state%nsurf,&
    engine_state%ndecay,engine_state%npot,engine_state%nretard,state,aux_data)
  return
end subroutine ReactionStepOperatorSplit
```

4.2 Drivers

The widely-used open-source codes Amanzi and ParFlow use Alquimia to implement geochemical capabilities. Amanzi is a multi-process high-performance-computing simulator that provides a flexible and extensible simulation capability. ParFlow is a parallel, integrated hydrology model that simulates spatially distributed surface and subsurface flow, as well as land surface processes including evapotranspiration and snow. The implementation of Alquimia in these codes responded to the particular needs and capabilities in each case.

Code Block 3. Amanzi implementation of Alquima's *AlquimiaReactionStepOperatorSplit*. Lines starting with "// ..." indicate portions of the code omitted for brevity.

```
bool
ChemistryEngine::Advance(const double delta_time,
                         const AlquimiaProperties& mat_props,
                         AlquimiaState& chem_state,
                         AlquimiaAuxiliaryData& aux_data,
                         AlquimiaAuxiliaryOutputData& aux_output,
                         int& num_iterations)
{
  // ...

  // Advance the chemical reaction all operator-split-like.
  chem_.ReactionStepOperatorSplit(&engine_state_,
                                  delta_time,
                                  &(const_cast<AlquimiaProperties&>(mat_props)),
                                  &chem_state,
                                  &aux_data,
                                    &chem_status_);
  // ...

  // Retrieve auxiliary output.
  chem_.GetAuxiliaryOutput(&engine_state_,
                           &(const_cast<AlquimiaProperties&>(mat_props)),
                           &chem_state,
                           &aux_data,
                           &aux_output,
                           &chem_status_);

  // Did we succeed?
  if (chem_status_.error != kAlquimiaNoError) return false;

  // Did we converge?
  if (!chem_status_.converged) return false;

  // Write down the (maximum) number of Newton iterations.
  num_iterations = chem_status_.num_newton_iterations;
  return true;
}
```

In Amanzi, unstructured-mesh and structured-mesh discretizations are available in a single C++ code base and Alquimia was the right solution for a unified geochemical interface that worked for the different data structures holding the state variables for both meshes. (In this work we use the labels Amanzi-U and Amanzi-S for convenience to distinguish between unstructured and structured capabilities, respectively.) *ChemistryEngine::Advance* is the C++ function that provides a unified access within Amanzi to Alquimia's *AlquimiaReactionStepOperatorSplit* C function (Code Block 3). After calling *AlquimiaReactionStepOperatorSplit*, it obtains the auxiliary output variables. This is by choice here; there is no requirement to do so every time step but in Amanzi, it is done so that when requested by the user, certain output variables can be written to the output file. This function is called for each cell of the discretization, thus the driver is responsible for handling how their data structures are accessed for each cell.

The structured-mesh capabilities rely on block-structured adaptive mesh refinement (AMR) from the BoxLib library (Dubey et al., 2014), build upon Fortran Array Boxes (*FArrayBox*). The function that solves the geochemical problem is written for this *FArrayBox*. Using the available box iterators, the codes iterates for each grid cell in the box, performing three operations (Code Block 4). First, the Alquimia transfer containers are updated using the structured variables, then the *ChemistryEngine::Advance*) function (described above) is called with these updated data structures, solving the geochemical problem, and last the new solution is passed back to the (*FArrayBox*) structures for the next transport step.

Code Block 4. Amanzi-S call to advance chemistry. Lines starting with "// ..." indicate portions of the code omitted for brevity.

```
void
AlquimiaHelper_Structured::Advance(const FArrayBox& aqueous_saturation,       int sSat,
                                  const FArrayBox& aqueous_pressure,         int sPress,
                                  const FArrayBox& porosity,                 int sPhi,
                                  const FArrayBox& volume,                   int sVol,
                                  FArrayBox&       primary_species_mobile,   int sPrimMob,
                                  FArrayBox&       fcnCnt,                   int sFunc,
                                  FArrayBox&       aux_data, Real water_density, Real temperature,
                                  const Box& box, Real dt, int chem_verbose)
{
    // ...
    Box thread_box(box);
    thread_box.setSmall(BL_SPACEDIM-1,tli);
    thread_box.setBig(BL_SPACEDIM-1,tli);

    for (IntVect iv=thread_box.smallEnd(), End=thread_box.bigEnd(); iv<=End; thread_box.next(iv)) {

      BL_to_Alquimia(aqueous_saturation,       sSat,
                     aqueous_pressure,         sPress,
                     porosity,                 sPhi,
                     volume,                   sVol,
                     primary_species_mobile,   sPrimMob,
                     aux_data, iv, water_density, temperature,
                     alquimia_properties[threadid],
                     alquimia_state[threadid],
                     alquimia_aux_in[threadid],
                     alquimia_aux_out[threadid]);

      // ...

      int newton_iters;
      engine->Advance(dt,alquimia_properties[threadid],alquimia_state[threadid],
                      alquimia_aux_in[threadid],alquimia_aux_out[threadid],newton_iters);

      // ...

      Alquimia_to_BL(primary_species_mobile,   sPrimMob,
                     aux_data, iv,
                     alquimia_properties[threadid],
                     alquimia_state[threadid],
                     alquimia_aux_in[threadid],
                     alquimia_aux_out[threadid]);

      fcnCnt(iv,sFunc) = newton_iters;
    }
  }
```

```
}
```

The three operations are also present in the unstructured counterpart of the *Advance* function (Code Block 5). The copy operations from and to Alquimia data structures are different than those in the structured function and thus cannot be re-used. However, the call to the chemistry engine *Advance* is the same as in the structured function. This exemplifies how the single-cell model adopted in Alquimia offers significant flexibility in enabling a broad range of discretizations.

Code Block 5. Amanzi-U call to advance chemistry

```
int
Alquimia_PK::AdvanceSingleCell(double dt,
                               Teuchos::RCP<Epetra_MultiVector>& aqueous_components,
                               int cell)
{
  // Copy the state and property information from Amanzi's state within
  // this cell to Alquimia.
  CopyToAlquimia(
    cell, aqueous_components, alq_mat_props_, alq_state_, alq_aux_data_, Tags::DEFAULT);

  int num_iterations = 0;
  if (alq_mat_props_.saturation > saturation_tolerance_) {
    bool success = chem_engine_->Advance(
      dt, alq_mat_props_, alq_state_, alq_aux_data_, alq_aux_output_, num_iterations);
    if (not success) {
      if (vo_->os_OK(Teuchos::VERB_MEDIUM)) {
        Teuchos::OSTab tab = vo_->getOSTab();
        *vo_->os() << "no convergence in cell: " << mesh_->cell_map(false).GID(cell) << std::endl;
      }
      return -1;
    }
  }

  // Move the information back into Amanzi's state, updating the given total concentration vector.
  CopyAlquimiaStateToAmanzi(
    cell, alq_mat_props_, alq_state_, alq_aux_data_, alq_aux_output_, aqueous_components);

  return num_iterations;
}
```

In ParFlow, earlier work entailed coupling solute transport in the subsurface to a subset of the geochemical capabilities in CrunchFlow via a custom interface (Beisman et al., 2015). In that case, the same C-to-Fortran macros used for coupling Parflow with the CLM land model (Maxwell and Miller, 2005) were applied. This earlier work served as the basis for the implementation of Alquimia in the code, although the availability of Alquimia's C interface here enabled a more seamless coupling, without the need of C to Fortran macros. For example, the *AlquimiaReactionStepOperatorSplit* C function is called directly within a loop inside block structured discretization (Code Block 6). The function in Code Block 6 shows how the coupling utilizes extensively the C utility library. ParFlow calls *CopyAlquimiaState*, *CopyAlquimiaProperties*, and *CopyAlquimiaAuxiliaryData* are used to store and retrieve data before and after the call to *AlquimiaReactionStepOperatorSplit* in case of non convergence of the solution (Code Block 6). This is in contrast to Amanzi, where similar functions are used but they are written as new C++ code in Amanzi itself and encapsulated in *BL_to_Alquimia* and *Alquimia_to_BL* for the structured capabilities and *CopyToAlquimia* and *CopyAlquimiaStateToAmanzi* for the unstructured.

Code Block 6. Parflow call to Alquima. Lines starting with "// ..." indicate portions of the code omitted for brevity.

```c
void AdvanceChemistry(ProblemData *problem_data, AlquimiaDataPF *alquimia_data, Vector **
    concentrations, Vector *saturation, double dt, double t, int *any_file_dumped, int dump_files,
    int file_number, char *file_prefix)
{
  // ...
  // copy the transported primary mobile concentrations to alquimia
  AdvectedPrimaryToChem(alquimia_data->chem_state, &alquimia_data->chem_sizes, concentrations,
    problem_data);

  // solve chemistry cell-by-cell
  ForSubgridI(is, subgrids)
  {
    subgrid = SubgridArraySubgrid(subgrids, is);

    // ...
    por_sub = VectorSubvector(ProblemDataPorosity(problem_data), is);
    por = SubvectorData(por_sub);
    sat_sub = VectorSubvector(saturation, is);
    sat = SubvectorData(sat_sub);

    GrGeomInLoop(i, j, k, gr_domain, r, ix, iy, iz, nx, ny, nz,
    {
      por_index = SubvectorEltIndex(por_sub, i, j, k);
      sat_index = SubvectorEltIndex(sat_sub, i, j, k);

      chem_index = (i-ix) + (j-iy) * nx + (k-iz) * nx * ny;

      alquimia_data->chem_properties[chem_index].volume = vol;
      alquimia_data->chem_properties[chem_index].saturation = sat[sat_index];

      // Set the thermodynamic state.
      alquimia_data->chem_state[chem_index].water_density = water_density;
      alquimia_data->chem_state[chem_index].temperature = 25.0;
      alquimia_data->chem_state[chem_index].porosity = por[por_index];
      alquimia_data->chem_state[chem_index].aqueous_pressure = aqueous_pressure;

      //copy pre-solution state and aux data to temp containers in case non-convergence is an issue
      CopyAlquimiaState(&alquimia_data->chem_state[chem_index], &alquimia_data->chem_state_temp);
      CopyAlquimiaAuxiliaryData(&alquimia_data->chem_aux_data[chem_index], &alquimia_data->chem_aux_data_temp);
      CopyAlquimiaProperties(&alquimia_data->chem_properties[chem_index], &alquimia_data->chem_properties_temp);

      // Solve the geochemical system
      alquimia_data->chem.ReactionStepOperatorSplit(&alquimia_data->chem_engine,
                                             dt_seconds, &alquimia_data->chem_properties[chem_index],
                                             &alquimia_data->chem_state[chem_index],
                                             &alquimia_data->chem_aux_data[chem_index],
                                             &alquimia_data->chem_status);
      if (alquimia_data->chem_status.error != 0)
      {
        amps_Printf("ReactionStepOperatorSplit() error: %s\n",
                    alquimia_data->chem_status.message);
        PARFLOW_ERROR("Geochemical engine error, exiting simulation.\n");
      }

      if (!(alquimia_data->chem_status.converged))
      {
        // ...
      }

      alquimia_data->chem.GetAuxiliaryOutput(&alquimia_data->chem_engine,
                                      &alquimia_data->chem_properties[chem_index],
                                      &alquimia_data->chem_state[chem_index],
                                      &alquimia_data->chem_aux_data[chem_index],
                                      &alquimia_data->chem_aux_output[chem_index],
                                      &alquimia_data->chem_status);
      if (alquimia_data->chem_status.error != 0)
      {
        amps_Printf("GetAuxiliaryOutput() auxiliary output fetch failed: %s\n",
                    alquimia_data->chem_status.message);
        PARFLOW_ERROR("Geochemical engine error, exiting simulation.\n");
      }
    });
  }

  // copy solved primary concentrations back to PF
  ReactedPrimaryToPF(alquimia_data->chem_state, &alquimia_data->chem_sizes, concentrations, problem_data);

  // ...
}
```

4.3 Multi-way comparison

We build on the tests in the Alquimia test suite to develop a set of one-dimensional simulations of reactive transport with Amanzi and ParFlow. The test suite (Sec. 3.3) ensures the correct functioning in the simulation of specific reaction types separately: mineral dissolution-precipitation, aqueous kinetics, ion exchange, surface complexation, and isotherm-based sorption in batch and reactive transport scenarios. In addition, a non-reactive tracer simulation is used to identify how the different discretization schemes affect the differences in the results but also to rule out numerical issues by the data transfer steps involved in using Alquimia.

The simulations are simple with regard to the transport processes and the spatial distribution of properties in this domain. The domain is one-dimensional, 100 meters in length and discretized with 100 cells, with a porosity of 0.25. A uniform flow rate is applied along the domain in fully-saturated conditions such as the infiltrating front is half-way through the domain in the 50-year simulations. Diffusive-dispersive processes are not considered. In the tests with heterogeneous reactions, the solution initially in the domain is in equilibrium with the mineral or surface. A solution with a distinct composition infiltrates from the left boundary, displacing the initial solution and driving the geochemical reactions considered in each case (except for the non-reactive tracer.)

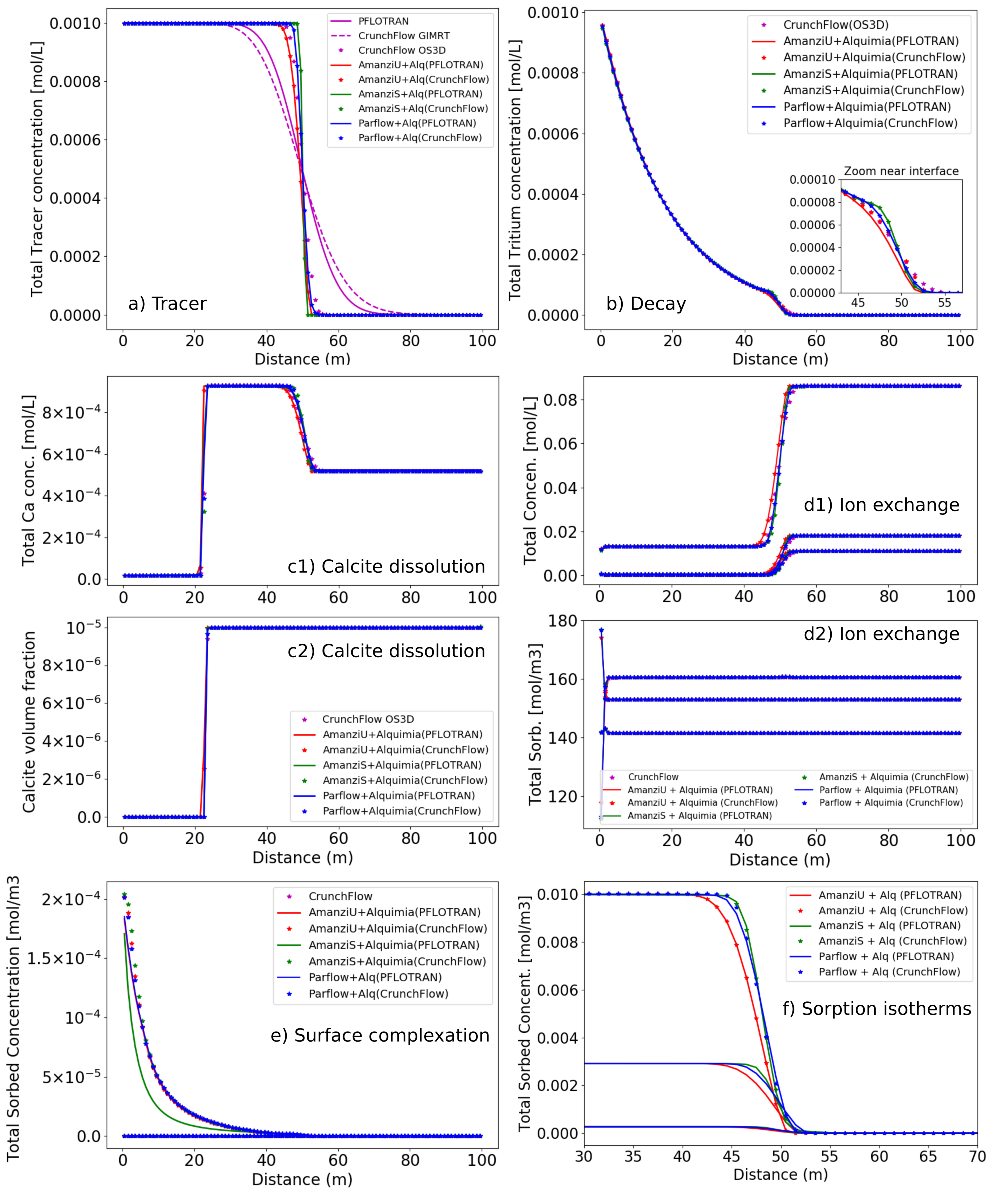

Figure 2. Selected results from six 1D reactive transport simulations that consider common geochemical reactions separately: (a) non-reactive tracer, (b) tritium decay, (c) calcite dissolution, (d) ion exchange, (e) surface complexation and (f) isotherm-based sorption. Each simulation was performed with different code combinations Amanzi (-S and -U) and Parflow were used as driver codes, with PFLOTRAN and CrunchFlow as engine codes. Additionally, PFLOTRAN and CrunchFlow were also used as reactive transport simulators, where GIMRT and OS3D refer to the global implicit and operator splitting capabilities of CrunchFlow. In global implicite mode, PFLOTRAN and CrunchFlow solve transport implicitly resulting in diffuse solutions, which are omitted in the figure. The Langmuir and Freundlich sorption isotherms are not presented for CrunchFlow. While they can be simulated via a surface complexation model and a single sorbing species, no specific keyword in the input deck is available and this was not pursued further here. Standalone CrunchFlow does not output sorbed concentrations for Kd linear sorption.

Each problem is simulated eight times. This includes six times with different driver-engine combinations (two each with Amanzi-S, Amanzi-U, and ParFlow, using PFLOTRAN and CrunchFlow as engines). Two additional simulations are performed with PFLOTRAN and CrunchFlow as standalone codes using their own transport solvers. In all cases, the Courant number was kept to 1, as needed by the operator splitting approach.

Results from the non-reactive tracer show that the advective front is at 50 m at 50 years (Figure 2a). Spreading of the front can be attributed exclusively to numerical dispersion added by the numerical scheme employed in each case. Results from Amanzi-S show the sharpest advective front, consistent with the high-order methods in BoxLib (Dubey et al., 2014), especially when Courant number equals 1. CrunchFlow's third-order time-diminishing variation (TVD) scheme (Gupta et al., 1991) in operator splitting mode results in a numerical dispersion similar to that of Amanzi-U's explicit second order scheme. PFLOTRAN and CrunchFlow in global implicit mode use both implicit solvers that result in larger numerical dispersion.

The differences in the discretization schemes affect the results from the reactive transport simulations differently. For kinetically-controlled aqueous reactions such as first-order tritium radioactive decay, $r = -\lambda c$, differences arise at the leading edge of the infiltration front but disappear with time behind the front (Figure 2b). For fast or equilibrium heterogeneous reactions, two fronts may be present, one associated with the advective infiltration front and one with the heterogeneous reaction.

In the calcite simulation, dissolution is treated as a kinetic reaction with a transition-state-theory-type rate law:

$$R_i = kA(1 - Q/K_s) \tag{10}$$

where k is the rate constant, Q is the ion activity product, K_s is the equilibrium constant of the reaction, and A is the reactive surface area. The intrinsic rate of reaction (kA) is much faster than the rate of advective transport, which effectively results in equilibrium conditions. The incoming solution drives dissolution, depleting over time the initial mass of calcite. Where calcite is still present, the solution is in equilibrium with respect to calcite (between 22-100 m at 50 years, 2c). Where calcite has been depleted, the concentrations reflect the incoming solution (between 0-22 m at 50 years). The reactive front is thus sharp. All codes capture accurately the location of this sharp front, with only minor differences in the values in the grid cells near the front compared to the absolute variation in values in the front. Thus, this front is not as affected by the numerical dispersion discussed earlier. In contrast, the infiltration front at 50 m shares the same issues discussed previously with regard to the front spreading. The solution resulting from calcite dissolution upstream mixes with the initial solution owing to numerical dispersion (Figure 2c). In this example, the reaction reduce differences that may arise from the transport calculations such as those from discretization schemes. In reactive transport systems at steady-state or quasi-steady state conditions (such as this slow-moving calcite dissolution front), numerical dispersion is often less of a concern to practitioners in contrast to the small time steps required with the operator splitting approach.

Surface reactions (sorption isotherms, ion exchange, and surface complexation) are all treated as equilibrium reactions by the two engines available in Alquimia. Results for the associated test examples also show a very reasonable agreement between codes capturing the resulting fronts (Fig. 2). Sorption isotherms (linear-Kd, Freundlich and Langmuir) express relationships between aqueous and sorbed concentrations. As such, there is only the advective front present in the results, with the sorbed concentrations tracking this front (Fig. 2f, compare with Fig. 2a), and the codes perform according to their performance in the conservative tracer simulations.

The ion exchange simulation shows two sets of fronts (Fig. 2d). The fronts are limited to very narrow bands near the inlet and at 50 m (not visible in Fig. 2d). The infiltrating solution is dilute in comparison to the initial solution but the proportions of the solutes are different. Sorbed concentrations change accordingly, with Mg^{2+} and Ca^{2+} increasing relatively to initial and Na^{+} decreasing. Surface complexation of Zn^{2+} also shows good agreement between codes, where the greater Zn^{2+} concentrations in the infiltrating solution result in increasing sorbed concentrations with time (Figure 2e).

5 Applications

This section presents examples of how Alquimia can be deployed within very different applications and purposes. Alquimia's interface is well-defined (i.e. variables and parameters given and returned are unequivocally specified with their units, and functions are clearly described), and is flexible enough to allow for the peculiarities associated with each application, making it broadly applicable to the simulation of geochemical systems. From a software perspective, the interface must couple with codes written in different programming languages, and the code performance must not be hindered by the coupling. This section also documents issues associated with the development and how they were overcome in each case.

5.1 Pore-scale multiphase flow and reactive transport

In recent years, there has been an increase in the use of reactive transport models to simulate pore-scale processes, e.g. Molins et al. (2020, 2024b). A distinctive aspect of pore-scale models is that they represent explicitly the fluid and solid phases that make up porous media. From a geochemical perspective, this allows for consideration of mineral surface areas directly from the pore space geometry and thus for the accessibility of the reactive fluids to the mineral surfaces (Molins, 2015).

A recent area of interest is the coupling of pore-scale models for multiphase flow to reactive transport. In this direction, Li et al. (2022) developed a multiphase flow and reactive transport simulator building on capabilities of openFOAM (noa, 2022). OpenFOAM (for “Open-source Field Operation And Manipulation") is a very popular open-source platform written in C++ to develop computational models in fluid dynamics applications and beyond. For the application presented in Li et al. (2022), openFOAM provides the solver tools for multiphase flow and transport of solutes in the aqueous phase. However, no geochemical packages are available to represent multicomponent aqueous speciation and mineral dissolution-precipitation reactions. CrunchFlow has been used previously in this role, including in pore-scale applications (Molins et al., 2012; Beisman et al., 2015; Zhang et al., 2022, 2024), and thus its use was a preferred choice.

Alquimia facilitated the development of the application. Because it is written in C, it was straightforward to use in OpenFOAM's C++-based code. It also provided enough flexibility to incorporate the pore-scale conceptualization of reactive processes. Importantly for pore-scale applications, the reactive surface area (A') in the mineral rate (R_i') is in units of area. e.g., m^2. Mineral dissolution in the pore-scale model is simulated with a transition-state-theory-type rate law:

$$R_i' = kA'(1 - Q/K_s) \tag{11}$$

where k is the rate constant, Q is the ion activity product, K_s is the equilibrium constant of the reaction, and A' is the reactive surface area. Alquimia was originally designed with porous-media applications and it also assumes mass balance is performed per unit volume. As such, it requires this surface area to be in units of specific area, namely $m^2/m^3 - bulk$ as the conservation equation 8 is written per unit volume. In other words, this implies that the surface area must be normalized by the volume of medium it occupies. In a discretized form, this volume is the volume of the grid cell n (V^n). Hence, in the single-cell Alquimia model, the surface area is the area of the interface as computed by openFOAM normalized by the grid cell volume ($A^n = A'^n/V^n$). At the same time, the volumetric water content (θ) on the left-hand side of Eq. 8 is set to be the water volume in the cell normalized to the grid cell volume ($\theta^n = V_{aq}^n/V^n$). Hence, the function f_k in Eq. 6, is in discretized form:

$$R_i^n = kA^n(1 - Q^n/K_s) \tag{12}$$

In a cell n where A'^n is not zero (i.e. in contact with the solid phase) and where V_{aq}^n is not zero (i.e. the aqueous phase is present), R_i^n can be non-zero (i.e. dissolution or precipitation may take place).

OpenFOAM uses header files for setting up and initializing the problem at hand. This implies that initial values for primary variables such as species concentrations in solute transport problems are given in these files. In multicomponent geochemical models, the value of the initial and boundary concentrations is generally not known before the simulation, rather, a set of constraints is given to obtain species concentrations. The specialized Alquimia *AlquimiaProcessCondition* function call is used for this purpose in Li et al. (2022). In a first step, the dimensions of the geochemical problem are given in an openFOAM header files, which then are used to allocate and initialize the concentration variables with dummy values. In a second step, the Alquimia *AlquimiaSetup* function is called which provides the dimensions of the geochemical system, which must be checked for consistency with those in the header files, followed by a call to *AlquimiaProcessCondition* (Fig. 3) to initialize and set the boundary conditions of the pore-scale problem in OpenFOAM.

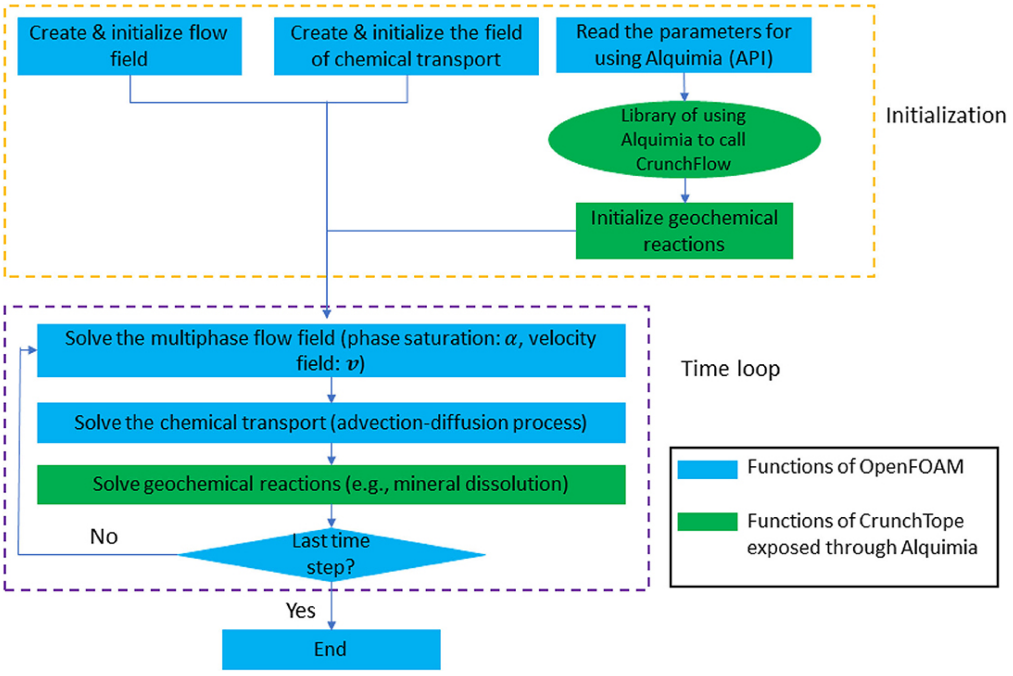
Create & initialize flow field
Create & initialize the field of chemical transport
Read the parameters for using Alquimia (API)
Library of using Alquimia to call CrunchFlow
Initialize geochemical reactions
Initialization
Solve the multiphase flow field (phase saturation: α, velocity field: v)
Time loop
Solve the chemical transport (advection-diffusion process)
Solve geochemical reactions (e.g., mineral dissolution)
No
Last time step?
Yes
End
Functions of OpenFOAM
Functions of CrunchTope exposed through Alquimia

Figure 3. Diagram summarizing the computational workflow of CrunchFOAM, including initialization of the CrunchFlow geochemical problem and calls to its solver via Alquimia. Reprinted from Li et al. (2022)

5.2 Land surface processes

In the Earth System Model (ESM) context, the land surface represents the lower boundary of the atmosphere. It also acts as host of biogeochemical cycles such as soil organic matter decomposition that influence the global carbon budget. Land surface models (LSMs) couple the relevant processes in terrestrial environments in order to quantify key interactions including greenhouse gas exchange fluxes between soil and atmosphere or the capacity of soils to sequester carbon. LSMs generally use simplified representations of biogeochemical cycling that consider the role of carbon, nitrogen, phosphorus and water but neglect other components, pH, and the aqueous reactions associated with it, including detailed terminal electron accepting processes (Sulman et al., 2024). There is an opportunity to significantly improve LSMs by incorporating the broad range of reactions that are available in general-purpose geochemical models. In particular, PFLOTRAN offers flexible options via its Reaction Sandbox feature to readily implement custom reaction rate models (Hammond, 2022). LSMs are already complex codes, in that they include many processes, and thus development often benefits from prototyping new process models, testing them considering a reduced number of coupled processes or exploring scenarios systematically. This process typically also benefits from using tools that automate processing and visualization of the results.

In this example application, Sulman and co-authors (Wang et al., 2024; Sulman et al., 2024) used Alquimia to incorporate geochemical processes in land surface modeling. Rather than tackling a full implementation to an existing land surface model, a two-step approach was used.

In a first step, a Python-based prototyping simulator was developed and applied to the simulations of methane processes in Arctic soils. A Python interface implemented Alquimia's API functions and data structures as Python functions using the CFFI package (CFFI, 2023). Alquimia's C interface was used to couple it to this Python implementation. As a prototyping tool, the model consisted of a single-cell representation of the system and the main use was to systematically prescribe fluxes in the Python code directly. This allowed for setting up the domain in Python data structures while retaining the PFLOTRAN reaction network capabilities. Time-stepping was done in the Python code. The *hands-on* mode in Alquimia allowed setting and updating reaction rate constants from the Python code in each time step. This is not necessarily possible with PFLOTRAN directly. This Python-Alquimia prototyping system was further applied to simulate one-dimensional soil column processes in coastal wetlands (Wang et al., 2024).

In a second step, Alquimia was used to couple PFLOTRAN to the existing Energy Exascale Earth System Model (E3SM) Land Model (ELM). This allowed for representing complex redox dynamics, aqueous and solid-phase chemistry, and pH dynamics in ELM (LaFond-Hudson and Sulman, 2023; Sulman et al., 2024). This is especially important in tidal wetlands, which are subject to both saltwater and freshwater inputs driven by tidal hydrological dynamics. Saltwater inputs inputs are associated with elevated sulfate concentrations that provide alternative terminal electron acceptors and reduce methane emissions in saltwater-affected wetlands. This work built on the prototype developed earlier but relied on the FORTRAN interface in Alquimia for convenience given that ELM is written in FORTRAN. Although PFLOTRAN is also in FORTRAN, from the engine's perspective (ELM in this case), the implementation is independent of the language the engine is written. Alquimia initialization (*AlquimiaSetup*) and initial condition equilibration (*AlquimiaProcessCondition*) subroutines were added to the ELM initialization code, and the Alquimia time stepping subroutine was added to ELM (Figure 4).

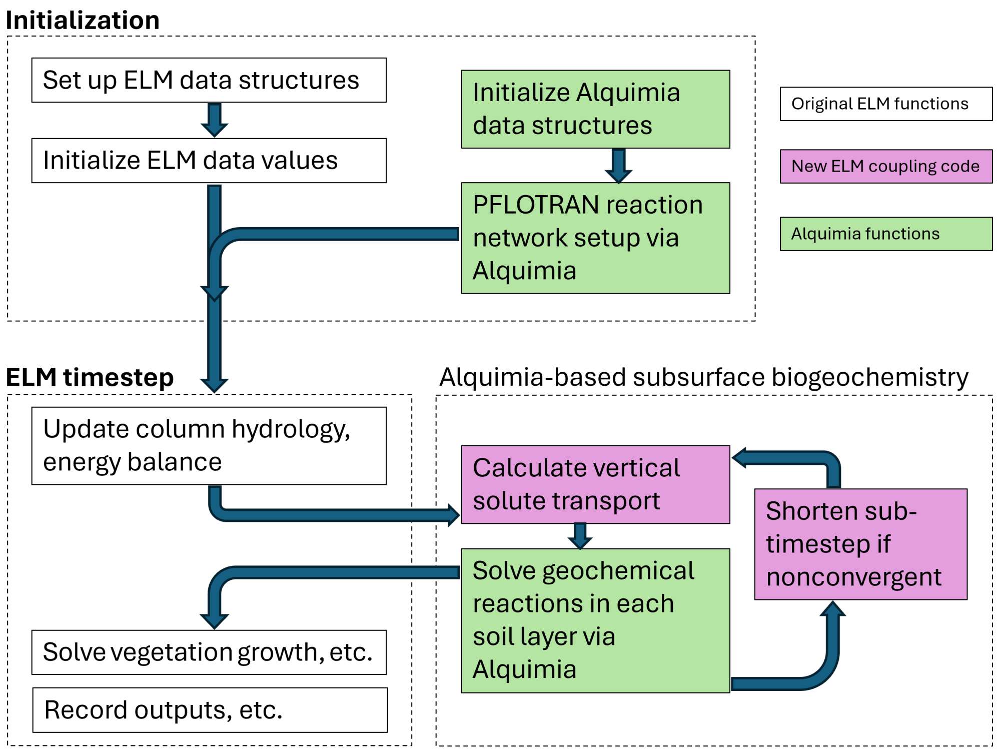
Initialization
Set up ELM data structures
Initialize ELM data values
Initialize Alquimia data structures
PFLOTRAN reaction network setup via Alquimia
Original ELM functions
New ELM coupling code
Alquimia functions
ELM timestep
Alquimia-based subsurface biogeochemistry
Update column hydrology, energy balance
Calculate vertical solute transport
Shorten sub-timestep if nonconvergent
Solve geochemical reactions in each soil layer via Alquimia
Solve vegetation growth, etc.
Record outputs, etc.

Figure 4. Diagram summarizing the computational workflow of ELM-PFLOTRAN, including initialization of the PLFOTRAN geochemical problem and calls to its solver via Alquimia.

ELM has its own description of biogeochemical processes, including consideration of carbon and nitrogen, which are present in multiple solid-state pools such as litter and soil organic matter, as well as aqueous soil nitrate and ammonium. In order to replace the ELM description of biogeochemical processes with that of PFLOTRAN, there has to be a one-to-one correspondence between ELM pools and the pools that are included in the PFLOTRAN reaction network used in the coupling (Tang et al., 2016). This is accomplished via the PFLOTRAN input file, along with soil organic matter decomposition kinetics defined using the PFLOTRAN Reaction Sandbox (Hammond, 2022). In this process, however, the Alquimia data structure interface was also augmented to treat solid-state SOM pools as immobile chemicals, allowing for transparent data transfer of the SOM pools from ELM to PFLOTRAN and back via the interface.

In the implementation, ELM decomposition processes were fully replaced by equivalent or modified calculations on the PFLOTRAN side and included the pools required by ELM. However, additional reactive processes are not affected by any requirements, enabling the consideration of reaction networks of arbitrary complexity using PFLOTRAN's flexible input file. For example, additional elemental cycles such as Mn redox processes or inorganic C interactions with soil minerals could be added to ELM simulations with minimal changes to ELM code. From a software design perspective, ELM stores all state variables of the geochemical problem (e.g., C and N concentrations as well as aqueous concentrations of H^+, SO_4^{2-}, HS^-, etc. and soil minerals such as iron oxides and iron sulfides if defined in the reaction network) but only those that are directly relevant to ELM state are visible to other model components (primarily organic matter and nutrient pools) while others are only handled by the interface. This allows for minimizing changes in ELM. We can, however, envision that certain process models in ELM could be improved on or refined with the addition of variables from the geochemical model currently not considered. For example, vegetation responses to phytotoxic sulfide concentrations or soil oxygen concentration could be added to improve representation of wetland vegetation (LaFond-Hudson and Sulman, 2023).

5.3 Reactive transport for integrated surface-subsurface hydrology

There is an increasing interest in using integrated hydrology models to quantify not only the water exports but also the solute exports from watersheds that impact human activities downstream (Bao et al., 2017). While integrated hydrology models have been used extensively to capture the feedback between flow in the surface and subsurface, solute transport and reactive processes are not represented in most models.

Molins et al. (2022) developed an approach to simulate reactive transport processes in integrated surface-subsurface hydrology problems. This approach was implemented in the open-source Advanced Terrestrial Simulator (ATS) (Coon et al., 2019), an integrated hydrology code built upon Amanzi solvers for subsurface flow and transport. The approach included geochemical processes both in the surface and subsurface compartments. Alquimia facilitated the implementation because ATS is built upon Amanzi, which already implemented Alquimia. In turn, the flexible multiphysics framework in ATS, which specifies interfaces for coupled processes (process kernels) and automates coupling strategies, allowed for defining separate process kernels for geochemistry in the surface and the subsurface (Fig. 5).

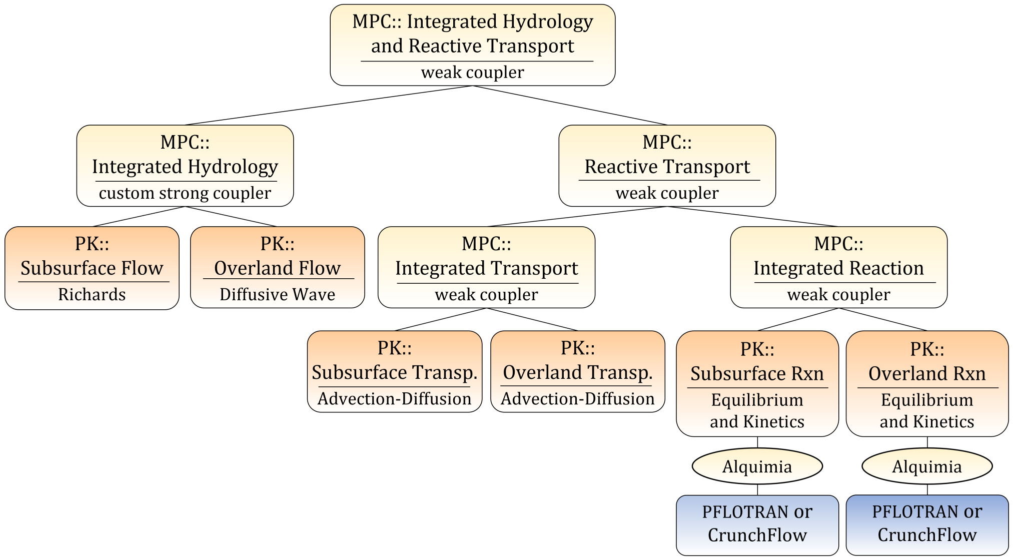

Figure 5. Process tree for a model of integrated hydrology and integrated reactive transport implemented in ATS (Coon et al., 2019; Molins et al., 2022) with calls to Alquimia, including both CrunchFlow and PFLOTRAN. In the example presented here, CrunchFlow is used in the Subsurface Reaction PK (left), and PFLOTRAN in the Overland Reaction PK (right). Reprinted from Molins et al. (2022).

The separation of multiphysics process kernels in ATS offers an excellent opportunity to showcase the flexibility that Alquimia provides in order to develop increasingly complex conceptual models and facilitate its implementation in software. This is exemplified here with simple simulations of reactive transport in a vertical column. To do so, we consider distinct geochemical models in the surface and subsurface, and then use both geochemical engines, CrunchFlow and PFLOTRAN, in the same simulation.

In the column, the subsurface soil is initially partially saturated and the surface is dry. Over the initial 105 days, a constant precipitation rate is applied to the surface domain, which exceeds the rate of drainage at the bottom of the column. As a result, water infiltrates into the subsurface, which becomes progressively saturated. The surface domain remains dry initially, but eventually the subsurface becomes fully saturated and water accumulates in the surface as well. The saturated hydraulic conductivity of the soil is larger than any of the prescribed water fluxes, precipitation or drainage. At 105 days, precipitation ceases and drainage increases. As a result, the ponded depth of water accumulated on the surface decreases. For the geochemical problem, we consider the presence of calcite and a set of dissolved species that describe the calcium carbonate system, including Ca^{2+}, HCO_3^-, and H^+ as primaries. The water in the column is initially in equilibrium with calcite. Rainwater is equilibrated with atmospheric CO_2, resulting in a relatively low pH, which drives calcite dissolution.

We consider 2 separate scenarios that represent two end-member conceptual models. Because in ATS surface water is treated with the shallow water approximation, water on the surface is well mixed. If heterogeneous reactions such as calcite dissolution are considered, this implies that the entire water column volume is in contact with mineral surfaces. This may overestimate the actual rate of dissolution if well mixed conditions are not achieved (e.g. gradients in concentrations exist along the water column). In scenario 1, surface water is assumed not to be in contact with the mineral at all and calcite dissolution is not included. In scenario 2, calcite is also present in the surface for dissolution.

In these simulations, we further demonstrate Alquimia's flexibility by using it to couple ATS with PFLOTRAN in the surface and CrunchFlow in the subsurface. These simultaneous couplings are not strictly necessary here because the capabilities to describe mineral dissolution and aqueous complexation in PFLOTRAN and CrunchFlow are very similar (yielding to the results shown in Section 4.3). However, one can envision situations where different engines have different capabilities, or that different process representations could be used in different parts of the domain with different codes. For example, a land surface model could be used to describe geochemical processes in the shallow subsurface in connection with SOM, vegetation and microbial dynamics, and a specialized geochemical model could be used to describe deeper subsurface processes, including mineral weathering.

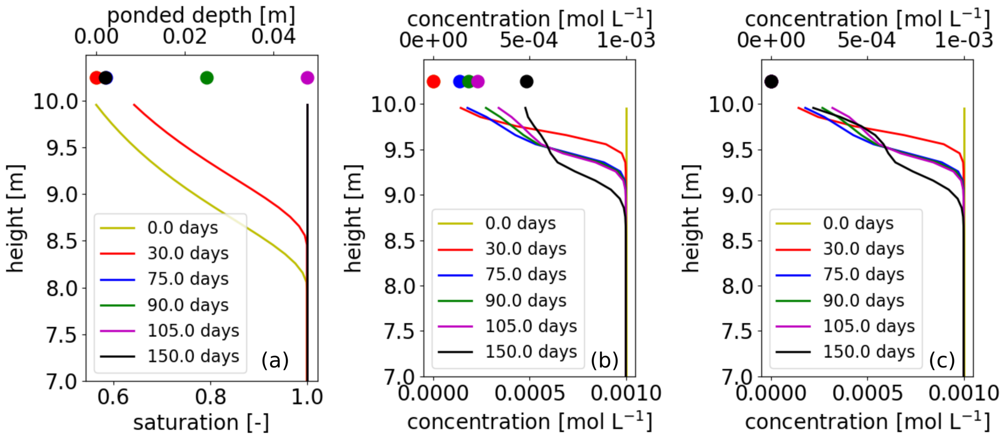

Figure 6. Results from surface-subsurface reactive transport simulation by ATS, with PFLOTRAN solving the geochemical problem in the surface and CrunchFlow in the subsurface, showing (a) ponded depth and water saturation, (b) Ca^{2+} concentrations in scenario 1 with dissolution in the surface, and (c) Ca^{2+} concentrations in scenario 2. Surface variables are shown as symbols at an arbitrary height and subsurface variables as solid lines as a function of height.

Results show differences in Ca^{2+} concentrations with depth in the two scenarios (Fig. 6). Before day 75, the surface is dry and there are no solutes in the surface. Rainwater infiltrates directly into the subsurface and in both scenarios results are identical for concentrations, with concentrations increasing with depth as calcite dissolves. From day 75, concentrations appear in the surface. In scenario 1, dissolution in the surface results in an increase of concentrations there with respect to rainwater values, while in scenario 2, the values reflect those in rainwater. As surface water infiltrates, it continues to drive calcite dissolution in both scenarios. Concentration profiles until day 105 are visually very similar in both scenarios, although numerical values differ by as much as 6%. This is due to how ATS calculates mass fluxes at the surface-subsurface interface (Molins et al., 2022). As indicated in the PK tree in Fig. 5, solute fluxes are obtained first for integrated transport, then the resulting concentrations from these fluxes are used to solve the geochemical problem. In this case, this means that fluxes into the subsurface not only reflect concentrations in the surface but also concentrations in the rainwater. Because concentrations in the rainwater are essentially zero, the solution infiltrating in the subsurface until day 105 is much more diluted than that in the surface. When precipitation ceases, the infiltrating concentrations reflect solely those in the ponded water and thus start to differ more clearly between scenario 1 and 2 (Fig. 6). This type of conceptual choices are made in the development of the driver code (ATS in this case) and enabled by the flexibility of the software interface.

6 Discussion and conclusions

6.1 Flexibility

The examples presented above highlight the flexibility of the generic interface Alquimia to expand the capabilities of multiphysics simulators to include multicomponent geochemical processes. Some of these applications go beyond the use initially envisioned for the interface, i.e. flow and reactive transport in subsurface porous media (Moulton et al., 2011). From this point of view, the operator L in Eq. 2 can be seen as a generic process operator that applies to concentration and affects the mass balance of component i.

For example, Alquimia can be used to perform simple batch chemistry calculations of the kind routinely carried out by geochemical models such as PHREEQc, without consideration of other processes that involve fluxes over a spatially discretized domain. This may be necessary for batch-scale laboratory experiments or, as in the land surface model example, to expand the range of reactive processes considered (Sulman et al., 2022, 2020). Further, when column-based land surface models such as ELM are used (Sulman et al., 2024), in addition to solute transport, the L operator can also include gas transport and lateral fluxes associated with tidal fluctuations along the depth of column with consideration of the corresponding salinity gradients.

Generally, however, the operator L includes fluxes over a spatial domain. In the pore-scale application, transport was considered in the aqueous phase only (Sec. 5.1). In the integrated hydrology application (Sec 5.3), the subsurface was simulated as a porous medium like in Amanzi or Parflow (Sec. 4.2) but the surface was represented as a 2D domain using the shallow water approximation. There, the solute mass balance is solved for the ponded depth of water. Because the surface may be wet or dry as a result of the dynamic conditions driven by rain events, the reactive transport processes are solved only for the wet portion of the surface.

The flexibility is also demonstrated by its application to different meshing and discretization schemes. The single-cell approach enabled the use of structured and unstructured meshes in Amanzi (Moulton et al., 2011). The unstructured capabilities in Amanzi allowed for traditional finite volume schemes, mimetic finite differences, and nonlinear finite volumes (Moulton et al., 2011), while the structured capabilities allowed for adaptive mesh refinement (AMR) Dubey et al. (2014).

6.2 Prototyping and benchmarking

Alquimia allows for rapid prototyping new capabilities and approaches. Examples include time stepping schemes or implementations of processes that are not readily available in engine codes. In the example of Sulman et al. (2022), this enabled prescribing the oxic/anoxic fluctuations of the system from the Python driver code by exchanging oxygen via a time-varying external boundary condition. Similarly, rate constants were updated by the driver at specified time points using the fine-grained access to the reaction parameters provided by Alquimia's hands-on mode.

As a generic interface, once Alquimia is implemented in a driver, one can swap engines for the same problem as long as these engines provide the same capabilities. This is of particular interest in benchmarking of reactive transport models. Because of the coupled nature of RTM, it is important to narrow down the source of discrepancies between codes. By sharing the same engine or by sharing the same driver, the use of Alquimia allows to isolate the potential sources of discrepancies. The example presented in Sec. 4.3 demonstrated how this approach can be used to investigate the impact of discretization schemes but this could be extended to other aspects.

6.3 Parallelization

Alquimia is a single-cell or 0-dimensional model that has no notion of the spatial problem. As discussed in Sec. 5.1 and Sec. 5.3, this assumes well-mixed geochemical conditions in this cell, which can be viewed as a batch reactor. The geochemical problem can be obtained independently from all other cells that make up the discretized spatial domain in a multiphysics problem. Hence, the driver determines the parallelization strategy for the solution of the spatially distributed, multiphysics problem. Because the geochemical equations are uncoupled across the domain, there is considerable freedom in staging their integration and load balancing.

All code examples presented here, except the python-based prototype in Sec. 5.2, are multithreaded CPU-based implementations and involve Message Passing Interface (MPI) parallelization by the driver. The work to integrate all the cells in the domain can be distributed across the available threads with no race or synchronization concerns. Load balance can be achieved by evenly distributing the work across processors, and within each processor across available threads. Code block 4 shows how Amanzi-S is explicitly exploiting multi-threading within a Fortran Array Box. However, the load balancing of chemical calculations may be at odds with load balancing strategies that incorporate stencil operations (such as advection or diffusion). While none of the examples presented tested this, it is possible to use different load balancing strategies for transport and chemistry when using Alquimia. The driver could choose to redistribute the state data between each stage of the time-split integration differently, including one that seeks better load balancing when sharp geochemical fronts are present in areas of the domain (and thus computations are more expensive) such as round-robin approaches, e.g. (De Lucia et al., 2021). The situation is more complex in GPU implementations (Balos et al., 2025). This contribution could not address the GPU use case with the capabilities available in the engines and drivers using Alquimia.

Geochemical engines could also implement a form of task parallelism for the geochemical solution within each cell. Although it is not the case for PFLOTRAN or CrunchFlow, it has been suggested that such a form of parallelism could speed up geochemical calculations in systems with a very large number of species and reactions. Anecdotal reports indicate that this has been effective for the codes Geochemist's Workbench (GWB) ChemPlugin (Bethke, 2024) and TOUGHREACT (Sonnenthal et al., 2021).

6.4 Limitations and Future work

As a generic interface, Alquimia is designed to allow for the implementation of different engines. However, the current number of implemented engines is still limited. As a result, only a small portion of all potential options have been demonstrated here.

Alquimia allows for the canonical approach (Eqs. 2-3), commonly used in reactive transport models such as PFLOTRAN and CrunchFlow (Lichtner, 1991), but this is not required. If this approach is not available in a given engine or equilibrium may not be assumed, there are no restrictions in setting the number of aqueous complexes (N_x) to zero and using the total mobile concentrations to hold all the species concentrations. Heterogeneous reactions present a similar case. For mineral dissolution-precipitation, mineral volume fractions and surface areas are passed between driver and engine but there is no implied assumption whether they are implemented as equilibrium or kinetic reactions in the engine. Likewise, only sorbed concentrations are required for surface complexation reactions, which have been modeled both as equilibrium (Steefel et al., 2015) and as kinetic processes (Greskowiak et al., 2015). The assumption on the driver side is only that total mobile concentrations are to be transported with the aqueous phase, while total immobile concentrations are not.

Alquimia currently only permits operator splitting coupling between engines and drivers. This poses a constraint on the time step size and thus hampers the solution of certain problems, e.g. rapid transport. However, the interface can be expanded to allow for the global implicit approach. The same single-cell model used for operator splitting would be applicable, but the Alquimia state variables would have to include the Jacobian matrix of the geochemical problem in each grid cell. The driver code would set up a solve the nonlinear problem resulting from coupling solute transport and reactions over the spatial domain. Parallelization responsibilities would remain with the driver with geochemical calculation still being performed for a cell. For the operator splitting coupling, there are no restrictions on how the nonlinear problem is solved and other schemes could be considered such as Strang symmetrization to speed up calculations if needed, e.g. in the case of rapid transport.

In the applications presented here, the choice of engine was often a function of the familiarity of the users with one of the engines available but also with the availability of specific capabilities from one engine. While most codes with geochemical capabilities share a set of basic capabilities (Steefel et al., 2015), one can anticipate that, as the number of engines connected to Alquimia grows, specialized capabilities will vary more widely between engines and this will open up the range of applications Alquimia can be used for. The example in Sec. 5.3 can serve an example of this, where different geochemical engines can be used in different parts of the domain as appropriate. Further, increasingly geochemical models make use of machine learning tools to accelerate calculations (Leal et al., 2020) or replace process-based calculations (Chang et al., 2023). These machine learning models could be seamlessly integrated in the interface as additional engines, which may be used in isolation or in combination with process-based engines as needed. While the two engines currently available are Fortran codes, the interface is prepared to couple with engines in other languages, e.g. PHREEQC, written in C++.

Against the backdrop of continuous evolution and development of multiphysics simulators driven and enabled by new approaches and capabilities, tools like Alquimia that simplify coupling by codifying a clear and flexible interface between codes and processes are increasingly valuable. Specifically, Alquimia enables access to existing true-and-tried geochemical models but also it facilitates future development and implementation of new models. Code interoperability gives access to a range of capabilities with simple, easily maintainable code bases that also facilitate prototyping and validation of new software. Ultimately, improved software productivity and sustainability have the potential to increase the pace of scientific discovery and promote more efficient and effective use of research resources.

Code availability. Alquimia (Andre et al., 2013) is developed and maintained using the repository at github.com/LBL-EESA/alquimia-dev, with releases generally timed with xSDK releases (Bartlett et al., 2017). Different versions of the code were used for results presented here. Snapshots of Alquimia, PFLOTRAN, CrunchFlow, Amanzi, Parflow and ATS for the versions involved in this work (as noted below) as well as the input files for the included simulations are available at dx.doi.org/10.5281/zenodo.11414442 (Molins et al., 2024a). The Amanzi source code used in this work is the version with hash 4867af7, which includes input files for Section 4.3 and reference solutions for

CrunchFlow and PFLOTRAN. The Parflow source code used in this work is the version with hash d4b20b9. The versions of Alquimia, PFLOTRAN, CrunchFlow and PETSC used with these versions of Amanzi and Parflow are 1.0.9, 3.0.2., 906e164, and 3.16.0, respectively. The crunchFOAM application is described in detail by Li et al. (2022) and is based on version 1.0.6 of Alquimia. The land surface model applications are described in detail by (Sulman et al., 2022; Wang et al., 2024; LaFond-Hudson and Sulman, 2023; Sulman et al., 2024), and are based on version 1.0.8 of Alquimia, which can be found in the model-data archives associated with these publications Sulman et al. (2020, 2023). The integrated hydrology application with ATS is described in detail by Molins et al. (2022), with results presented here obtained with the version with hash 37a7b6e of ATS, which include the input files of the simulations as part of the regression tests. The versions of Alquimia, PFLOTRAN, CrunchFlow and PETSC used in this version of ATS are 1.0.9, 5.0.0, cf938c8, and 3.20.0, respectively.

Author contributions. S.M. wrote the manuscript, developed Alquimia, Amanzi-ATS, implemented CrunchFlow in Alquimia, and performed the simulations in Sec. 4.3, 5.3. B.J.A. had the original idea for, designed and developed Alquimia, implemented PFLOTRAN in Alquimia, and developed PFLOTRAN. J.N.J. developed Alquimia, including compliance to xSDK policies, and implemented Alquimia in Amanzi. G.E.H. developed Alquimia, PFLOTRAN, and implemented PFLOTRAN in Alquimia. B.N.S. developed ELM-PFLOTRAN and Alquimia, including its Fortran and CFFI interfaces. K.L. developed Amanzi-U and Alquimia in Amanzi. M.S.D. developed Amanzi-S and Alquimia in Amanzi. J.J.B. implemented Alquimia in PARFLOW. D.S. developed Amanzi-ATS and Alquimia in ATS. H.D. led the development of CrunchFOAM. P.C.L. developed PFLOTRAN, and participated in the initial Alquimia design. C.I.S. developed CrunchFlow, and participated in the initial Alquimia design. J.D.M. developed Amanzi-ATS, and led the projects that funded this work.

Competing interests. The authors declare no competing interests.

Disclaimer. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy.

Acknowledgements. This material is based upon work supported as part of the IDEAS-Watersheds project funded by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research (Contract No DE-AC02-05CH11231). Initial development of Alquimia was funded by the U.S. Department of Energy, Office of Environmental Management (Contract No DE-AC02-05CH11231). B.N.S. is supported by the U.S. Department of Energy Office of Science Early Career Research program as part of research in Earth System Model Development within the Earth and Environmental Systems Modeling Program, and by the Next Generation Ecosystem Experiments (NGEE) Arctic project through the Office of Biological and Environmental Research in the U.S. Department of Energy Office of Science. Oak Ridge National Laboratory is managed by UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725. PNNL is operated for the DOE by Battelle Memorial Institute under contract DE-AC05-76RL01830.

References

OpenFOAM, https://www.openfoam.com/, 2022.

Andre, B., Molins, S., Johnson, J., and Steefel, C.: Alquimia, https://doi.org/10.11578/DC.20210416.49, language: en, 2013.

Balos, C. J., Day, M., Esclapez, L., Felden, A. M., Gardner, D. J., Hassanaly, M., Reynolds, D. R., Rood, J. S., Sexton, J. M., Wimer, N. T., and Woodward, C. S.: SUNDIALS time integrators for exascale applications with many independent systems of ordinary differential equations, The International Journal of High Performance Computing Applications, 39, 123–146, https://doi.org/10.1177/10943420241280060, publisher: SAGE Publications Ltd STM, 2025.

Bao, C., Li, L., Shi, Y., and Duffy, C.: Understanding watershed hydrogeochemistry: 1. Development of RT?Flux?PIHM, Water Resources Research, 53, 2328–2345, https://doi.org/10.1002/2016WR018934, 2017.

Bartlett, R., Demeshko, I., Gamblin, T., Hammond, G., Heroux, M. A., Johnson, J., Klinvex, A., Li, X., McInnes, L. C., Moulton, J. D., Osei-Kuffuor, D., Sarich, J., Smith, B., Willenbring, J., and Yang, U. M.: xSDK Foundations: Toward an Extreme-scale Scientific Software Development Kit, Supercomputing Frontiers and Innovations, 4, 69–82, https://doi.org/10.14529/jsfi170104, 2017.

Beisman, J., Maxwell, R., Navarre-Sitchler, A., Steefel, C., and Molins, S.: ParCrunchFlow: an efficient, parallel reactive transport simulation tool for physically and chemically heterogeneous saturated subsurface environments, Computational Geosciences, pp. 1–20, https://doi.org/10.1007/s10596-015-9475-x, 2015.

Bethke, C. M.: The Geochemist's Workbench® Release 17 ChemPlugin™ User's Guide, Aqueous Solutions, LLC, Champaign, Illinois, https://www.gwb.com/pdf/GWB/ChemPluginUsersGuide.pdf, 2024.

CFFI: CFFI documentation — CFFI 1.15.1 documentation, https://cffi.readthedocs.io/en/latest/, 2023.

Chang, E., Zavarin, M., Beverly, L., and Wainwright, H.: A chemistry-informed hybrid machine learning approach to predict metal adsorption onto mineral surfaces, Applied Geochemistry, 155, 105 731, https://doi.org/10.1016/j.apgeochem.2023.105731, 2023.

Charlton, S. R. and Parkhurst, D. L.: Modules based on the geochemical model PHREEQC for use in scripting and programming languages, Computers & Geosciences, 37, 1653–1663, https://doi.org/10.1016/j.cageo.2011.02.005, 2011.

Coon, E., Svyatsky, D., Jan, A., Kikinzon, E., Berndt, M., Atchley, A., Harp, D., Manzini, G., Shelef, E., Lipnikov, K., Garimella, R., Xu, C., Moulton, D., Karra, S., Painter, S., Jafarov, E., and Molins, S.: Advanced Terrestrial Simulator, https://doi.org/10.11578/DC.20190911.1, language: en, 2019.

De Lucia, M., Kühn, M., Lindemann, A., Lübke, M., and Schnor, B.: POET (v0.1): speedup of many-core parallel reactive transport simulations with fast DHT lookups, Geoscientific Model Development, 14, 7391–7409, https://doi.org/10.5194/gmd-14-7391-2021, publisher: Copernicus GmbH, 2021.

Dubey, A., Almgren, A., Bell, J., Berzins, M., Brandt, S., Bryan, G., Colella, P., Graves, D., Lijewski, M., L"offler, F., O'Shea, B., Schnetter, E., Van Straalen, B., and Weide, K.: A survey of high level frameworks in block-structured adaptive mesh refinement packages, Journal of Parallel and Distributed Computing, 74, 3217–3227, https://doi.org/10.1016/j.jpdc.2014.07.001, 2014.

GitHub: GitHub Actions documentation, https://docs.github.com/en/actions, 2024.

Greskowiak, J., Gwo, J., Jacques, D., Yin, J., and Mayer, K. U.: A benchmark for multi-rate surface complexation and 1D dual-domain multi-component reactive transport of U(VI), Computational Geosciences, 19, 585–597, https://doi.org/10.1007/s10596-014-9457-4, 2015.

Gupta, A. D., Lake, L. W., Pope, G. A., Sepehrnoori, K. T. U., and King, M. J. B. R.: High-Resolution Monotonic Schemes for Reservoir Fluid Flow Simulation, In Situ; (United States), 15:3, https://www.osti.gov/scitech/biblio/5799660, 1991.

Hammond, G. E.: The PFLOTRAN Reaction Sandbox, Geoscientific Model Development, 15, 1659–1676, https://doi.org/10.5194/gmd-15-1659-2022, publisher: Copernicus GmbH, 2022.

Hammond, G. E., Lichtner, P. C., and Mills, R. T.: Evaluating the performance of parallel subsurface simulators: An illustrative example with PFLOTRAN, Water Resources Research, 50, 208–228, https://doi.org/10.1002/2012WR013483, 2014.

https://github.com/LBL-EESA/alquimia-dev: LBL-EESA/alquimia-dev, original-date: 2015-11-03T17:59:25Z, 2024.

Jara, D., de Dreuzy, J.-R., and Cochepin, B.: TReacLab: An object-oriented implementation of non-intrusive splitting methods to couple independent transport and geochemical software, Computers & Geosciences, 109, 281–294, https://doi.org/10.1016/j.cageo.2017.09.005, 2017.

Jaysaval, P., Hammond, G. E., and Johnson, T. C.: Massively parallel modeling and inversion of electrical resistivity tomography data using PFLOTRAN, Geoscientific Model Development, 16, 961–976, https://doi.org/10.5194/gmd-16-961-2023, 2023.

LaFond-Hudson, S. and Sulman, B.: Modeling strategies and data needs for representing coastal wetland vegetation in land surface models, New Phytologist, 238, 938–951, https://doi.org/10.1111/nph.18760, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/nph.18760, 2023.

Leal, A. M. M., Kyas, S., Kulik, D. A., and Saar, M. O.: Accelerating Reactive Transport Modeling: On-Demand Machine Learning Algorithm for Chemical Equilibrium Calculations, Transport in Porous Media, 133, 161–204, https://doi.org/10.1007/s11242-020-01412-1, 2020.

Li, L., Maher, K., Navarre-Sitchler, A., Druhan, J., Meile, C., Lawrence, C., Moore, J., Perdrial, J., Sullivan, P., Thompson, A., Jin, L., Bolton, E. W., Brantley, S. L., Dietrich, W. E., Mayer, K. U., Steefel, C. I., Valocchi, A., Zachara, J., Kocar, B., Mcintosh, J., Bao, C., Tutolo, B. M., Beisman, J., Kumar, M., and Sonnenthal, E.: Expanding the role of reactive transport models in critical zone processes, Earth-Science Reviews, https://doi.org/10.1016/j.earscirev.2016.09.001, 2017.

Li, P., Deng, H., and Molins, S.: The Effect of Pore-Scale Two-Phase Flow on Mineral Reaction Rates, Frontiers in Water, 3, 734 518, https://doi.org/10.3389/frwa.2021.734518, 2022.

Lichtner, P. C.: The Quasi-Stationary State Approximation to Fluid/Rock Reaction: Local Equilibrium Revisited, in: Diffusion, Atomic Ordering, and Mass Transport: Selected Topics in Geochemistry, edited by Ganguly, J., Advances in Physical Geochemistry, pp. 452–560, Springer US, New York, NY, https://doi.org/10.1007/978-1-4613-9019-0_13, 1991.

Lichtner, P. C., Hammond, G. E., Lu, C., Karra, S., Bisht, G., Andre, B. N. C. f. A. R., Lawrence Berkeley National Lab. (LBNL), B., Mills, R. I. C., Univ. of Tennessee, K., and Kumar, J.: Pflotran User Manual: A Massively Parallel Reactive Flow and Transport Model for Describing Surface and Subsurface Processes, Tech. Rep. LA-UR–15-20403, Los Alamos National Lab. (LANL), Los Alamos, NM (United States); Sandia National Lab. (SNL-NM), Albuquerque, NM (United States); Lawrence Berkeley National Lab. (LBNL), Berkeley, CA (United States); Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States); OFM Research, Redmond, WA (United States), https://www.osti.gov/scitech/biblio/1168703, dOI: 10.2172/1168703, 2015.

Maxwell, R. M. and Miller, N. L.: Development of a Coupled Land Surface and Groundwater Model, Journal of Hydrometeorology, 6, 233–247, https://doi.org/10.1175/JHM422.1, 2005.

Maxwell, R. M., Putti, M., Meyerhoff, S., Delfs, J.-O., Ferguson, I. M., Ivanov, V., Kim, J., Kolditz, O., Kollet, S. J., Kumar, M., Lopez, S., Niu, J., Paniconi, C., Park, Y.-J., Phanikumar, M. S., Shen, C., Sudicky, E. A., and Sulis, M.: Surface-subsurface model intercomparison: A first set of benchmark results to diagnose integrated hydrology and feedbacks, Water Resour. Res., 50, 1531–1549, https://doi.org/10.1002/2013WR013725, 2014.

Mayer, K. U., Frind, E. O., and Blowes, D. W.: Multicomponent reactive transport modeling in variably saturated porous media using a generalized formulation for kinetically controlled reactions, Water Resources Research, 38, 1174, https://doi.org/10.1029/2001WR000862, 2002.

Molins, S.: Reactive Interfaces in Direct Numerical Simulation of Pore-Scale Processes, Reviews in Mineralogy and Geochemistry, 80, 461–481, https://doi.org/10.2138/rmg.2015.80.14, 2015.

Molins, S., Trebotich, D., Steefel, C. I., and Shen, C.: An investigation of the effect of pore scale flow on average geochemical reaction rates using direct numerical simulation, Water Resources Research, 48, n/a–n/a, https://doi.org/10.1029/2011WR011404, 2012.

Molins, S., Soulaine, C., Prasianakis, N. I., Abbasi, A., Poncet, P., Ladd, A. J. C., Starchenko, V., Roman, S., Trebotich, D., Tchelepi, H. A., and Steefel, C. I.: Simulation of mineral dissolution at the pore scale with evolving fluid-solid interfaces: review of approaches and benchmark problem set, Computational Geosciences, https://doi.org/10.1007/s10596-019-09903-x, 2020.

Molins, S., Svyatsky, D., Xu, Z., Coon, E. T., and Moulton, J. D.: A Multicomponent Reactive Transport Model for Integrated Surface-Subsurface Hydrology Problems, Water Resources Research, 58, e2022WR032 074, https://doi.org/10.1029/2022WR032074, 2022.

Molins, S., Andre, B., Johnson, J., Hammond, G., Sulman, B., Lipnikov, K., Day, M., Beisman, J., Svyatskiy, D., Deng, H., Lichtner, P., Steefel, C., and Moulton, D.: Alquimia: A generic interface to biogeochemical codes– A tool for interoperable development, prototyping and benchmarking for multiphysics simulators., https://doi.org/10.5281/zenodo.11414442, language: en, 2024a.

Molins, S., Trebotich, D., and Steefel, C. I.: Approaches for the simulation of coupled processes in evolving fractured porous media enabled by exascale computing, Computing in Science & Engineering, pp. 1–10, https://doi.org/10.1109/MCSE.2024.3403983, 2024b.

Moulton, D., Berndt, M., Buskas, M., Garimella, R., Prichett-Sheats, L., Hammond, G., and Meza, J.: High-level design of Amanzi, the multi-process high performance computing simulator., Technical Report ASCEM-HPC-2011-03-1, US Department of Energy, Washington, DC, 2011.

Nardi, A., Idiart, A., Trinchero, P., de Vries, L. M., and Molinero, J.: Interface COMSOL-PHREEQC (iCP), an efficient numerical framework for the solution of coupled multiphysics and geochemistry, Computers & Geosciences, 69, 10–21, https://doi.org/10.1016/j.cageo.2014.04.011, 2014.

Parkhurst, D. L. and Wissmeier, L.: 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, 2015.

Parkhurst, D. L., Kipp, K. L., and Charlton, S. R.: PHAST Version 2 - A program for simulating groundwater flow, solute transport, and multicomponent geochemical reactions, in: U.S. Geological Survey Techniques and Methods, vol. 6-A35, p. 235, 2010.

Prommer, H. and Post, V.: PHT3D: A ReactiveMulticomponent Transport Model for Saturated Porous Media. User's Manual v2.10, http://www.pht3d.org, 2010.

Simunek, J., Jacques, D., Langergraber, G., Bradford, S. A., Šejna, M., and Genuchten, M. T. v.: Numerical Modeling of Contaminant Transport Using HYDRUS and its Specialized Modules, Journal of the Indian Institute of Science, 93, 265–284, http://journal.library.iisc.ernet.in/index.php/iisc/article/view/1224, 2013.

Sonnenthal, E., Spycher, N., Xu, T., and Zheng, L.: TOUGHREACT V4.12-OMP and TReactMech V1.0 Geochemical and Reactive-Transport User Guide, https://escholarship.org/uc/item/8945d2c1, 2021.

Sphinx: Welcome — Sphinx documentation, https://www.sphinx-doc.org/en/master/, 2024.

Steefel, C. I. and MacQuarrie, K. T. B.: Approaches to modeling of reactive transport in porous media, Reviews in Mineralogy and Geochemistry, 34, 85–129, http://rimg.geoscienceworld.org/content/34/1/85, 1996.

Steefel, C. I., DePaolo, D. J., and Lichtner, P. C.: Reactive transport modeling: An essential tool and a new research approach for the Earth sciences, Earth and Planetary Science Letters, 240, 539–558, https://doi.org/10.1016/j.epsl.2005.09.017, 2005.

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., and Yeh, G. T.: Reactive transport codes for subsurface environmental simulation, Computational Geosciences, 19, 445–478, https://doi.org/10.1007/s10596-014-9443-x, 2015.

Sulman, B., Yuan, F., O'Meara, T., Graham, D., Gu, B., Herndon, E., and Zheng, J.: Simulated hydrological dynamics and coupled iron redox cycling impact methane production in an Arctic soil: Modeling Archive, https://doi.org/10.5440/1814844, 2020.

Sulman, B., Wang, J., LaFond-Hudson, S., O'Meara, T., Yuan, F., Molins, S., Forbrich, I., Cardon, Z., and Giblin, A.: Model simulations of Plum Island Ecosystems LTER low marsh site using ELM-PFLOTRAN, https://doi.org/10.15485/1991625, publisher: urn:node:ESS_DIVE, 2023.

Sulman, B. N., Yuan, F., O'Meara, T., Gu, B., Herndon, E. M., Zheng, J., Thornton, P. E., and Graham, D. E.: Simulated Hydrological Dynamics and Coupled Iron Redox Cycling Impact Methane Production in an Arctic Soil, Journal of Geophysical Research: Biogeosciences, 127, e2021JG006 662, https://doi.org/10.1029/2021JG006662, 2022.

Sulman, B. N., Wang, J., LaFond-Hudson, S., O'Meara, T., Yuan, F., Molins, S., Hammond, G. E., Forbrich, I., Cardon, Z., and Giblin, A.: Integrating tide-driven wetland soil redox and biogeochemical interactions into a land surface model, JAMES, 2024.

Tang, G., Yuan, F., Bisht, G., Hammond, G. E., Lichtner, P. C., Kumar, J., Mills, R. T., Xu, X., Andre, B., Hoffman, F. M., Painter, S. L., and Thornton, P. E.: Addressing numerical challenges in introducing a reactive transport code into a land surface model: a biogeochemical modeling proof-of-concept with CLM–PFLOTRAN 1.0, Geoscientific Model Development, 9, 927–946, https://doi.org/10.5194/gmd-9-927-2016, publisher: Copernicus GmbH, 2016.

van der Lee, J., De Windt, L., Lagneau, V., and Goblet, P.: Module-oriented modeling of reactive transport with HYTEC, Computers & Geosciences, 29, 265–275, https://doi.org/10.1016/S0098-3004(03)00004-9, 2003.

Wang, J., O'Meara, T., LaFond-Hudson, S., He, S., Maiti, K., Ward, E. J., and Sulman, B. N.: Subsurface Redox Interactions Regulate Ebullitive Methane Flux in Heterogeneous Mississippi River Deltaic Wetland, Journal of Advances in Modeling Earth Systems, 16, e2023MS003 762, https://doi.org/10.1029/2023MS003762, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2023MS003762, 2024.

Zhang, Q., Deng, H., Dong, Y., Molins, S., Li, X., and Steefel, C.: Investigation of Coupled Processes in Fractures and the Bordering Matrix via a Micro-Continuum Reactive Transport Model, Water Resources Research, 58, e2021WR030 578, https://doi.org/10.1029/2021WR030578, 2022.

Zhang, Q., Dong, Y., Molins, S., and Deng, H.: The Impacts of Micro-Porosity and Mineralogical Texture on Fractured Rock Alteration, Water Resources Research, 60, e2023WR036 266, https://doi.org/10.1029/2023WR036266, 2024.