Hindawi
Geofluids
Volume 2021, Article ID 4290248, 12 pages
https://doi.org/10.1155/2021/4290248

Research Article

Predicting Fluid Properties in the MUFITS Reservoir Simulator with User-Supplied Modules

Andrey Afanasyev and Ivan Utkin

Institute of Mechanics, Moscow State University, Moscow 119192, Russia

Correspondence should be addressed to Andrey Afanasyev; afanasyev@imec.msu.ru

Received 27 May 2021; Accepted 16 August 2021; Published 29 September 2021

Academic Editor: Jinze Xu

We present a recent development of the MUFITS reservoir simulator aiming at modelling the transport of fluids whose properties and phase equilibria are calculated in a user-supplied external shared library. Both the explicit correlations and tabulated data for the fluid parameters can be implemented in the library that we name the EoS-module (Equation of State-module). An iterative approach—which, for example, is based on the phase equilibria calculation through the Gibbs energy minimisation (GEM) method, can also be used in the EoS-module. A considerable effort has been undertaken to minimise the number of program procedures exported by the shared library. This should facilitate and ease the usage of the developed software extension by the scientific community. Furthermore, we supplement the article with the source code of two simple EoS-modules that can serve as templates in other modelling and software development efforts. The EoS-modules are also useful for coupling MUFITS with other elaborate software for fluid property prediction. To demonstrate such a possibility, we supplement the article with the source code of a more complicated EoS-module that couples MUFITS with the geochemical code GEMS3K. This module is used in a simple 1-D benchmark study showing the capabilities of MUFITS for modelling reactive transport in porous media.

1. Introduction

1.1. Reservoir Simulations. The numerical modelling of multicomponent multiphase flows in porous media is in demand in many areas of subsurface exploration and natural process prediction. These areas involve petroleum reservoir exploitation, subsurface CO_2 and natural gas storage, geothermal energy extraction, the prediction of interactions between hydrothermal and volcanic systems, and other applications [1, 2]. The related transport in porous media is often complicated by multiphase equilibria of reservoir fluids, nonisothermal processes, and reactions between fluid and rock [3]. Many other complications can be observed if the flows occur in fractured reservoirs, the transport in a porous medium is coupled with multiphase flows in wellbores and surface facilities, or there is a complicated, i.e., nonequilibrium, behaviour at the pore scale [4].

Given that the aforementioned complications often lead to nonlinear equations solved in a 3-D domain, the flows can be predicted only by means of numerical modelling, i.e., reservoir simulations [4, 5]. A computer program accounting for all relevant physical phenomena and exploitation processes in a convenient integrated manner is called a reservoir simulator. Certainly, reservoir simulators differ in several respects, i.e., target applications or available modelling options. There are academic, not-for-profit simulators often applied in the investigation of processes in nature, e.g., TOUGH2 [6], TOUGHREACT [7], and MODFLOW [8], among others. Usually, such simulators have a limited functionality, allowing users without sufficient backgrounds in reservoir simulations to do easy and quick assessments of some simple transport phenomena. Normally, the commercial software includes a much wider set of options to allow for modelling with engineering accuracy [4, 5].

Some simulators take an intermediate place between open-source and commercial software. Such reservoir simulation codes, e.g., MUFITS [9, 10] and AD-GPRS [11], are used to develop and test new methods for numerical modelling because their object-oriented architectures are closer to those of commercial software. If these methods prove their robustness, then they are applied in commercial software. Thus, such codes serve as platforms for the improvement of more elaborate simulators.

1.2. Governing Equations for Modelling Transport in Porous Media. If we neglect any specific complications, most reservoir simulators are aimed at solving the following governing equations [4–6]:

$$\frac{\partial}{\partial t}\left(\phi\sum_{i=1}^{np}\rho_i c_{i(j)}s_i\right)+\nabla\cdot\left(\sum_{i=1}^{np}\rho_i c_{i(j)}\boldsymbol{u}_i+\boldsymbol{\psi}_{(j)}\right)=0, \quad (1)$$

$$\frac{\partial}{\partial t}\left(\phi\sum_{i=1}^{np}\rho_i e_i s_i+(1-\phi)\rho_r e_r\right)+\nabla\cdot\left(\sum_{i=1}^{np}\rho_i h_i\boldsymbol{u}_i+\boldsymbol{\psi}_{(e)}\right)=0, \quad (2)$$

$$\boldsymbol{u}_i=-K\frac{K_{ri}}{\mu_i}(\nabla P_i-\rho_i g), \quad (3)$$

$$\Omega(P,T,c_t),\ \text{where}\ \Omega=\left\{np,\rho_i,e_i,h_i,\mu_i,s_i,c_{i(j)}\right\},c_t=\left\{c_{t(1)},c_{t(2)},\cdots,c_{t(nc)}\right\}, \quad (4)$$

$$K_{ri}=K_{ri}(s),P_i-P_k=P_{c,ik}(s),\ \text{where}\ s=\left\{s_1,\cdots,s_{np}\right\}, \quad (5)$$

$$\sum_{i=1}^{np}s_i=1,\ \sum_{j=1}^{nc}c_{i(j)}=1,\ \sum_{j=1}^{nc}c_{t(j)}=1. \quad (6)$$

Here, nc is the number of fluid components; np is the number of fluid phases; ϕ is the porosity; ρ is the density; $c_{i(j)}$ is the jth component mass fraction in the ith phase; s is the phase saturation (i.e., volume fraction of pore space occupied by the phase); $\boldsymbol{u}$ is the Darcy velocity; $\boldsymbol{\psi}_{(j)}$ and $\boldsymbol{\psi}_{(e)}$ are the fluxes of the jth component and energy caused by mechanical dispersion, molecular diffusion, or heat conduction; e is the specific energy; h is the specific enthalpy; K is the absolute permeability of the medium; K_r is the relative permeability; μ is the dynamic viscosity; P is the pressure; g is the gravity acceleration; $c_{t(j)}$ is the bulk mass fraction of the jth component; T is the temperature; P_c is the capillary pressure; the subscripts i, (j), and r denote the parameters of the ith phase, the jth component, and rock, respectively.

Equation (1) is the mass conservation equation for each fluid component, Equation (2) is the energy conservation equation, and Equation (3) is Darcy's law. Equation (4) is a general formulation of relations used for the fluid property prediction. It means that the phase equilibrium depends on the average fluid pressure P, temperature T, and bulk composition c_t. Equation (5) defines the relative permeability and capillary pressure as saturation functions. Equations in (6) are additional relations. For simplicity, we assume that ϕ, K, and ρ_r are constants and that $e_r(T)$ is a given function of temperature. If the isothermal flow is simulated, then Equation (2) should be excluded from the system of governing equations, and the condition T = const should be assumed.

In the case of mechanical dispersion, the fluxes $\boldsymbol{\psi}_{(j)}$ take the following form [12]:

$$\boldsymbol{\psi}_{(j)}=-\phi\sum_{i=1}^{np}s_i\boldsymbol{D}_i\nabla c_{i(j)}, \quad (7)$$

$$\boldsymbol{D}_i=\frac{1}{\phi s_i}\begin{pmatrix}\lambda_{L,i}u_{x,i}+\lambda_{T,i}(u_{y,i}+u_{z,i}) & 0 & 0\\ 0 & \lambda_{L,i}u_{y,i}+\lambda_{T,i}(u_{x,i}+u_{z,i}) & 0\\ 0 & 0 & \lambda_{L,i}u_{z,i}+\lambda_{T,i}(u_{x,i}+u_{y,i})\end{pmatrix}, \quad (8)$$

where $\boldsymbol{D}$ is the tensor of mechanical dispersion; λ_L and λ_T are the longitudinal and transverse mean free path of a small volume of the ith phase caused by mixing in the porous medium; and u_x, u_y, and u_z are the components of the Darcy velocity.

The governing equations can be distinctively split into two sets of equations. The first set describing the fluid transport includes Equations (1)–(3), (7), and (8). The second set of Equations (4) and (5) is responsible for predicting the phase equilibria and transport properties of the fluid. Often only the equations of the second set are different, and the equations of the first set are the same in different applications (Figure 1).

The described splitting of the governing equations is particularly useful in designing reservoir simulators. It results in a natural architecture of the software, involving two large program modules (Figure 1). One module that we further refer to as the "kernel" involves all procedures not related to predicting phase equilibria and all fluid properties in general. Such a module includes finite-difference schemes, linear and nonlinear solvers, and other procedures related to the solution of Equations (1)–(8). The second module that we further refer to is the "EoS-module" that contains only the procedures for calculating fluid properties. Obviously, the kernel is larger and more elaborate than an EoS-module. However, an EoS-module can also be quite elaborate if it is based on the iterative prediction of phase equilibria. For example, such complications arise in compositional reservoir simulations [13] or reactive transport modelling [14, 15].

The modular architecture is very convenient for extending the application area of reservoir simulators. The kernel can be linked with different EoS-modules. Thereby, the simulator can be used for modelling the transport of different types of fluids. If the development of an EoS-module is not very time-consuming, then such an extension is easy to achieve. Moreover, such an extension has all the modelling options already implemented in the kernel, e.g., the mechanical dispersion (see Equations (7) and (8)), coupled reservoir-wellbore flows, and other sophisticated options. The discussed modular

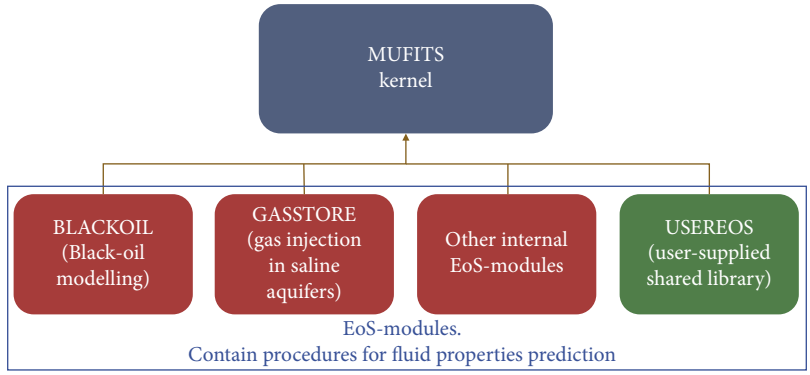

Figure 1: Architecture of MUFITS (modified after [9]).

architecture is explicitly implemented in TOUGH2 [6] and MUFITS [9] codes.

1.3. Goal of This Work. This article is aimed at presenting the recent extension of the MUFITS simulator that makes possible the development of user-supplied EoS-modules (Figure 1). MUFITS is an academic noncommercial software that has continued to develop over the past decade [9, 10, 12]. The software's kernel allows for parallel simulations, the coupled modelling of flows in porous media and wellbores, the modelling of the mechanical dispersion, and other options. For example, MUFITS has recently been extended for the automated prediction of and history matching to Earth's surface displacement and gravity changes associated with subsurface flows [16].

The simulator has several built-in EoS-modules (Figure 1). For example, it includes EoS-modules for the black-oil modelling of petroleum reservoirs and the subsurface storage of natural gas or CO_2 in saline aquifers. Although the software includes the standard modules for such applications, permanent demand exists in the development of new specific EoS-modules intended to be applied in nonstandard cases and the narrow area of research and engineering applications. To address such demand, we have implemented the possibility of linking the MUFITS kernel with user-supplied EoS-modules. Now, an EoS-module for a specific fluid can be developed by the user and linked with the software as an external shared library. Thus, from now on, the MUFITS extension for modelling the flows of other fluids can be done solely by the users. To ease and facilitate such extensions, we have reduced the number of procedures exported from the shared library to the kernel to just four.

Organising the development of an EoS-module as shared library with a convenient calling interface constitutes the advancement and innovation of the presented software's extension. Usually, the extension of reservoir simulator for the flows of a specific fluid requires significant programming efforts that often can be carried out based on only in-depth knowledge of software including its kernel. A rather complicated interface between the kernel and EoS-module requires that EoS-module is embedded in software as static library. For example, such static EoS-modules exist in MUFITS [9, 10] (Figure 1), TOUGH2 [6], and other codes. Usually, extending a reservoir simulator to reactive transport also assumes static compilation [7, 8], which limits the flexibility of software development and modification. To some degree, the calling interface and the dynamic linking proposed in this study solve the problem by allowing users to develop their own EoS-modules without in-depth knowledge on the whole software's architecture.

We supplement the article with a few simple open-source examples of the dynamically linked EoS-modules and benchmark them against 1-D analytical solutions. In the future, these examples can serve as templates for other more complicated modules. The discussed development is also useful for coupling MUFITS with other elaborate software for fluid properties prediction. Here, we present the result of MUFITS coupling with the geochemical software GEMS3K [17, 18], which extends the simulator for reactive transport modelling [1, 2, 14]. The software is based on the Gibbs energy minimisation (GEM) approach, which allows for finding unknown phase equilibria containing a considerable number of fluid and solid phases. We supplement the article with the source code of the EoS-module that dynamically links the MUFITS kernel with GEMS3K, compiled as a shared library. The coupling is validated against the 1-D benchmark study of reactive transport.

This article is organised as follows. In Section 2, we describe the architecture of a user-supplied EoS-module, provide an overview of the primary data flows, and discuss the application program interface (API) for all the procedures exported by the shared library. In Section 3, we present two simple EoS-modules, one for a mixture of ideal gases and another for a two-phase fluid. Section 4 is devoted to MUFITS coupling with GEMS3K. We end the article with Conclusions, where we briefly outline possible further extensions of development.

2. Architecture of an EoS-Module

2.1. Overview

2.1.1. Data Flows. A sketch of data flows associated with any user-supplied EoS-module is shown in Figure 2. Further, we refer to such a module as USEREOS. Figure 2(a) outlines the four mandatory program procedures of the module. Three of them are the preprocessing routines called to choose the modelling options and specify the arrays' sizes and the general (i.e., global) parameters of a fluid model. The fourth

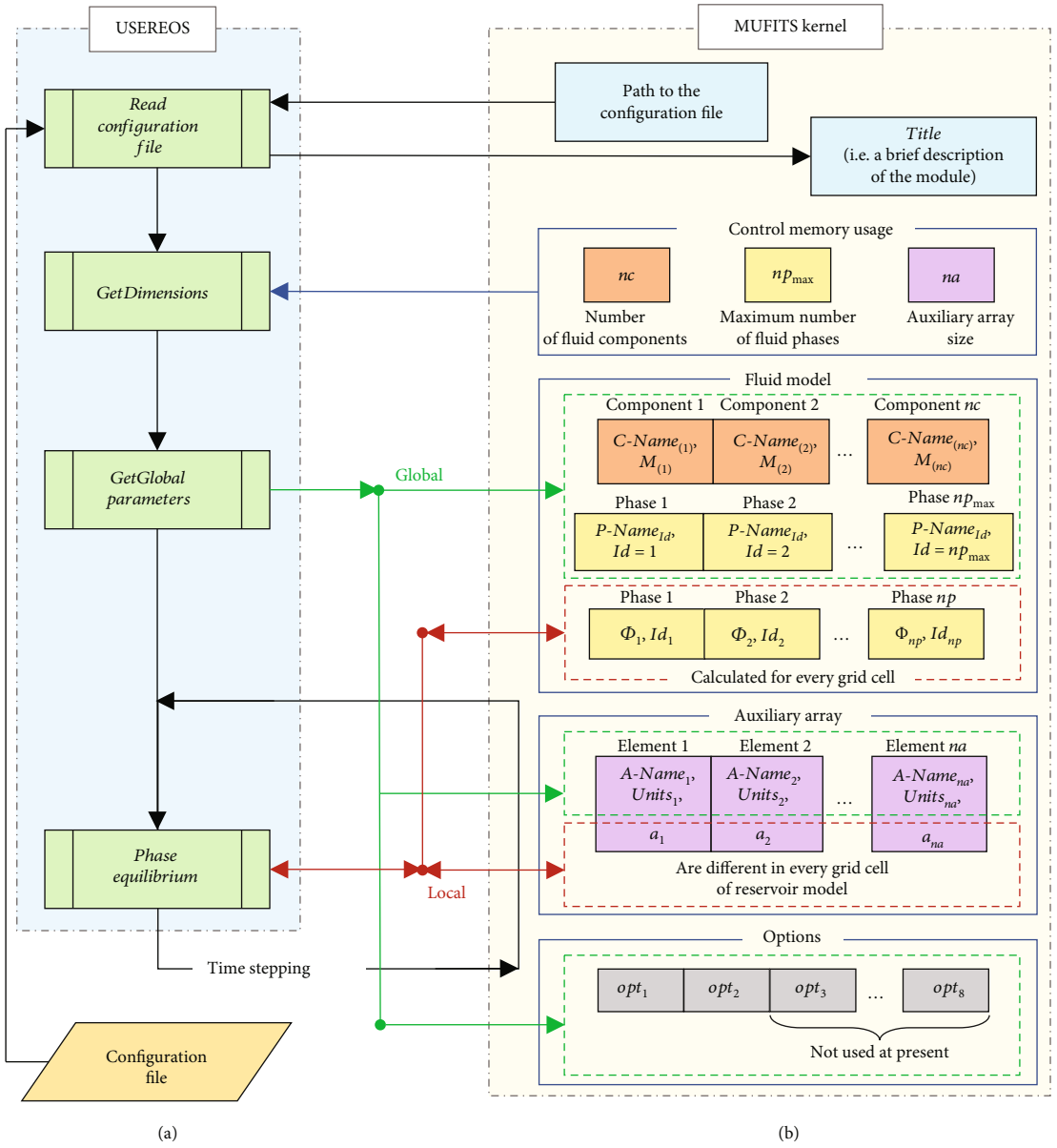

Figure 2: Sketch of data flows associated with USEREOS. The green and red dashed boxes in the right panel show the global and local parameters, respectively.

routine, named *PhaseEquilibrium*, supplies the kernel with fluid parameters for a given P, T, and c_t. This routine is called many times for any cell (i.e., grid block) of the reservoir model during time stepping. The coefficients of the fluid model are loaded directly into USEREOS by reading the configuration file.

Figure 2(b) summarises the parameters and arrays in the kernel that are changed by or loaded into USEREOS. The data are gathered in four main blocks. The first block contains all the constants controlling memory usage. The second block, "Fluid model," contains the parameters characterising the fluid and its phase and chemical equilibria. Some of these parameters that are shown within the green dashed box are global, i.e., they are identical for all the grid cells. Other parameters of the fluid that are within the red dashed box are local, i.e. they may be different in different cells. The third data block is for an auxiliary array that can be used as additional memory provided in the kernel for every cell. Again, this data block has global and local parameters. The fourth data block contains modelling options.

2.1.2. Constants Controlling Memory Usage. There are just three constants controlling memory allocation in the kernel.

(i) nc is the number of fluid components (see also Equations (1)–(8))

(ii) $np_{\max}$ is the maximum number of fluid phases. The number of fluid phases np depends on P, T, and c_t. Thus, it can be different in different grid cells. However, for any P, T, and c_t, the number of phases exported by USEREOS must not be larger than $np_{\max}$, i.e., $np \le np_{\max}$

(iii) *na* is the size of the auxiliary array. It specifies the number of additional 8-byte real numbers allocated in the kernel for every grid cell

2.1.3. Parameters of the Fluid Model. Every fluid component $j = 1, \cdots, nc$ is characterised by two global parameters:

(i) $C\text{-}Name_{(j)}$ is a character name of the component (e.g., 'H_2O' or 'CO_2')

(ii) $M_{(j)}$ is the molar weight of the component, which is used in the kernel for calculating molar quantities

Every fluid phase $i = 1, \ldots, np$ is characterised by one global and $6 + nc$ local parameters. The global parameter $P\text{-}Name_i$ is a character phase name (e.g., "GAS" or "LIQ"). The local parameters correspond to the vector

$$\Phi_i = \left\{ \rho_i, h_i, \mu_i, s_i, K_{ri}, \Delta P_i, c_{i(1)}, \cdots, c_{i(nc)} \right\}, \quad (9)$$

where

$$\Delta P_i = P_i - P, \quad (10)$$

is the difference between the phase fluid pressure P_i and the average pressure P. The variable ΔP_i is used for modelling the influence of capillary pressure P_c. For example, if the fluid can split into just two phases, liquid ($i = 1$) and gas ($i = 2$), then one can choose $P = P_1$, $\Delta P_1 = 0$, and $\Delta P_2 = P_2 - P_1 = P_{c,21}$. For every Φ_i, the specific energy of every phase is calculated in the kernel as

$$e_i = h_i - \frac{P_i}{\rho_i}. \quad (11)$$

2.1.4. Auxiliary Array. The auxiliary array of size *na* is allocated in the kernel for every grid cell. This array can be used as additional memory provided for USEREOS. Some parameters can be stored in this array for later usage in the following time steps. For example, the auxiliary array can be used for modelling the reaction kinetics, the hysteresis phenomena, or calculating additional (secondary) parameters of the fluid not belonging to Φ_i (see Equation (9)). The latter is used when coupling MUFITS with GEMS (see Section 4).

The elements of the auxiliary array can also be saved in the summary files in every report time period for later usage in the postprocessing of the simulation results. To make such reporting possible, every element $k = 1, \cdots, na$ of the array is characterised by two global parameters:

(i) $A\text{-}Name_k$ is a character name of the element that must begin with the symbol "#." One can refer to the element by its name in the input data to MUFITS

(ii) $Units_k$ is a string specifying the units of the physical quantity stored in the element. These units are reported into the summary files. For dimensionless quantities, $Units_k$ should be an empty string or "NODIM."

Every element k of the array is also characterised by one numeric quantity a_k of type "double" (8-byte real number). The variable a_k is local, i.e., it can be changed in USEREOS in every grid cell and at every time step.

2.1.5. The Option Array. The option array is an integer array of length 8, opt_l, where $l = 1, \cdots, 8$. If $opt_l = 1$, then the corresponding option is enabled (on). If $opt_l = 0$, then the option is disabled (off). By default, all the options are disabled. At present, only opt_1 and opt_2 can be used, whereas the other elements of the option array, opt_l, $l = 3, \cdots, 8$, are reserved for future developments.

The option opt_1 controls the calculation of the relative permeability K_{ri}. If $opt_1 = 0$, then the relative permeability is calculated in the kernel using standard input data to the simulator (e.g., specified with the SATTAB keyword). In this case, there is no need to export K_{ri} from USEREOS, i.e., the 5th element of Φ_i can be an undefined quantity. If $opt_1 = 1$, then the default treatment of relative permeability is overridden by the value K_{ri} exported from USEREOS.

The option opt_2 controls the calculation of the capillary pressure P_c or, to be strict, the relative phase pressure ΔP_i (see Equation (10)). If $opt_2 = 0$, then the relative pressure is calculated in the kernel using standard input data to the simulator (e.g., provided with the SATTAB keyword). In this case, there is no need to export ΔP_i from USEREOS, i.e., the 6th element of Φ_i can be an undefined quantity. If $opt_2 = 1$, then the default treatment of capillary pressure is overridden by the value ΔP_i exported from USEREOS.

2.1.6. Configuration File. The configuration file is an input data file to USEREOS. The path to this file is imported from the kernel. It is specified as the second argument of the USEREOS keyword that activates the presented modelling option (see Section 2.3).

The format of the configuration file can be specific for every particular shared library, because the kernel does not read this file. Every new USEREOS module can be supplemented with its own format of the file.

It is implied that the configuration file contains coefficients of the fluid model that are not explicitly defined as static variables in USEREOS. The configuration file can contain paths to other external files loaded into USEREOS.

2.2. Subroutines Exported from USEREOS

2.2.1. General Description. In this section, we give a brief overview of the main subroutines exported from USEREOS. The four subroutines placed in the order of their invoking by the kernel are as follows:

(i) *ReadConfigurationFile*. This routine is called by the kernel at the initialisation stage, immediately after the USEREOS keyword is encountered in the input data file to MUFITS (Section 2.3). The path to the configuration file is the second argument of the keyword that is imported by USEREOS as an argument

of the subroutine. It is implied that the shared library reads the file and then exports the title of the module to the kernel. The title may contain a brief description of the module and its version. The title is reported to the LOG-file of MUFITS.

(ii) *GetDimensions*. This subroutine is called at the end of initialisation stage. It is aimed at importing the sizes of arrays that need to be allocated into the kernel. The subroutine exports the number of fluid components nc, the maximum number of fluid phases ($np_{\max}$), and the auxiliary array size (na).

(iii) *GetGlobalParameters*. The procedure is called immediately after the previous one. It is aimed at loading the global parameters of the fluid model into the kernel. First, the 3-byte character name ($C\text{-}Name_{(j)}$) and the molar weight ($M_{(j)}$) of every component $j = 1, \ldots, nc$ must be specified. Second, the 3-byte character name ($P\text{-}Name_i$) for each fluid phase $i = 1, \ldots, np_{\max}$ is exported by USEREOS. Third, the 8-byte character name ($A\text{-}Name_k$) and units ($Units_k$) for every element $k = 1, \ldots, na$ of the auxiliary array must be specified. Any $A\text{-}Name_k$ must begin with the "#" character. Fourth, the options array, opt_l, is exported.

(iv) *PhaseEquilibrium*. Normally, this routine is called multiple times for every grid cell during linearisation of the governing Equations (1)–(8) and time stepping. It is aimed at reporting the phase equilibrium for a given P, T, and c_t. It reports the number of phases $np \le np_{\max}$, the vector Φ_i, and the integer variable Id_i for every phase $i = 1, \ldots, np$. The quantity Id_i (the "phase identifier") is aimed at associating the ith phase with the phase names (i.e., phase types) specified by *GetGlobalParameters*. The value of Id_i must be equal to the serial number of the $P\text{-}Name_k$, $k = 1, \ldots, np_{\max}$ specified by *GetGlobalParameters*.

A very important argument of *PhaseEquilibrium* is $Mode$. If $Mode = 0$, then the subroutine is supplied by the kernel with the initial guess for the phase equilibrium. The variables np, Id_i, and Φ_i, $i = 1, \ldots, np$, are used to specify the initial guess. In the case of $Mode = 0$, *PhaseEquilibrium* must calculate the equilibrium using the provided initial guess. The procedure must not change np and Id_i. It can only change Φ_i for a given P, T, and $c_{t(j)}$. Since the number of phases np is not changed, the reported saturations s_i and concentrations $c_{i(j)}$ and $c_{t(j)}$ are allowed to take negative or larger-than-one values [19]. If $Mode = 1$, then the initial guess is not specified. In this case, the subroutine must calculate the phase equilibrium—i.e., np, Id_i, and Φ_i—from scratch. If $Mode = 1$, then the reported s_i and $c_{i(j)}$ must satisfy the inequalities $0 \le s_i \le 1$ and $0 \le c_{i(j)} \le 1$, respectively.

2.2.2. Calling Interface for the USEREOS Subroutines. The USEREOS API is provided in the Fortran programming language. However, USEREOS can be written in any language that supports compilation to a shared library. To export the subroutines described in Section 2.2.1, it may be necessary to change these subroutines' signatures to accommodate differences between programming languages. Since GEMS3K is written in C++, the USEREOS module for coupling GEMS3K and MUFITS was written in C++ as well (Section 4).

The Fortran API for the four primary subroutines exported by USEREOS is given in Tables 1–4. In the tables, every argument of each subroutine is supplemented with a brief description, its datatype, and the intent. The intent values "in" or "out" mean that the corresponding variable is imported to or exported from USEREOS, respectively. The value "in/out" means that variable is used as both an input and output parameter. The datatypes are shown in a standard format used in Fortran. For example, the datatypes "integer(4)" and "real(8)" correspond to 4-byte integer and 8-byte real numbers, respectively. All physical quantities must be given in SI units. For example, the units of ρ_i, h_i, μ_i, and ΔP_i in the vector Φ_i must be kg/m^3, J/kg, Pa•s, and Pa, respectively.

2.3. Keywords and Mnemonics Associated with USEREOS. The input data to MUFITS are specified by the keywords, i.e., the commands to the simulator entered in the input data file. There is just one additional keyword USEREOS associated with the developed modelling option. This keyword activates the developed option. Its syntax is given in Table 5.

The keyword USEREOS must be specified in the first section, RUNSPEC, of the input file. It takes just two arguments: the paths to the shared library and the configuration file. By default, the path to the library is USEREOS.DLL on Windows and USEREOS.DYLIB on Mac OS. The default path to the configuration file is an empty string, i.e., the library does not require such a file. When the simulator encounters the keyword, it searches for the compiled shared library and checks that it exports the four subroutines. If the library or the subroutines cannot be found, then the simulation is terminated with error. Otherwise, the simulator calls the *ReadConfigurationFile* procedure, passing the path to the configuration file.

A mnemonic is a short abbreviation of a reservoir model parameter. In the input data file, one can refer to a parameter by its mnemonic. There are eight additional mnemonics associated with USEREOS. These mnemonics are listed in Table 6, where "$C\text{-}Name_{(j)}$" and "$P\text{-}Name_i$" are the names of the jth component and ith phase specified by the subroutine *GetGlobalParameters*. For example, if a component name is "CO_2" and a phase name is "LIQ," then CCO_2-LIQ is the mass fraction of CO_2 in the phase LIQ, DLIQ is the density of LIQ, and $ZFCO_2$ is the bulk mass fraction of CO_2.

3. Simple Application Examples

To demonstrate the software development and ease its usage, we further present two simple examples of USEREOS and their validation. The examples are supplemented with the source code and compiled executables [20, 21].

Table 1: API for *ReadConfigurationFile*.

Interface: call ReadConfigurationFile (*filename* , i_{err} , *title*)			
Variable	Intent	Datatype	Description
filename	in	character	Configuration file name (array of size 256)
i_{err}	out	integer(4)	Error specifier (if $i_{err} = 0$, then no error occurred during subroutine execution; if $i_{err} = -1$, then the configuration file is not found)
title	out	character	EoS-module title, i.e., a brief description (array of size 80)

Table 2: API for *GetDimensions*.

Interface: call GetDimensions(*nc*, $np_{\max}$, *na*)			
Variable	Intent	Datatype	Description
nc	out	integer(4)	Number of fluid components
$np_{\max}$	out	integer(4)	Maximum number of fluid phases
na	out	integer(4)	Auxiliary array size (can be 0)

3.1. Modelling Ideal Gas Flow through a Porous Medium. The first example concerns the numerical modelling of non-isothermal single-phase flows of ideal gas. The corresponding module can account for gases consisting of an arbitrary number of components. The fluid properties are calculated using the following equations:

$$\rho = \frac{PM}{RT},\ M = \sum_{j=1}^{nc} M_{(j)} x_{(j)}, \tag{12}$$

$$h' = \left(\sum_{j=1}^{nc} C_{(j)}' x_{(j)} \right) T,\ \mu = \sum_{j=1}^{nc} \mu_{(j)} x_{(j)}, \tag{13}$$

$$s = 1,\ c_{(j)} = \frac{M_{(j)} x_{(j)}}{\sum_{k=1}^{nc} M_{(k)} x_{(k)}}, \tag{14}$$

where R is the universal gas constant, x is the molar concentration, h' and C' are the molar enthalpy, and heat capacity at $P = \text{const}$, respectively. For simplicity, we omit the subscript $i \equiv 1$ of the only phase of the fluid.

Using the developed module, we present a 1-D study for a three-component gas composed of N_2, O_2, and CH_4. We consider a 1-D flow in a horizontal homogeneous reservoir $X \in [0,100]$ m, where X is the length. The porosity is $\phi = 0.2$, and the permeability is $K = 100$ mD. Other relevant parameters are the rock density, $\rho_r = 2500\ \text{kg/m}^3$, and the specific rock heat capacity, $C_r = 1\ \text{kJ/(kg•K)}$. At $t = 0$, the porous medium is saturated with the gas of composition $c_t = \{0.1, 0.0, 0.9\}$, i.e., it consists of 10% of N_2 and 90% of CH_4. The initial reservoir pressure and temperature are 200 bar and 50°C, respectively. The Dirichlet boundary conditions are imposed at $X = 100$ m, i.e., the initial pressure (and temperature) are kept constant at the open boundary $X = 100$ m. A gas of different composition $c_t = \{0.77, 0.23, 0.0\}$, i.e., air, is injected into the reservoir through the boundary $X = 0$ m. The injected gas temperature is 30°C at $P = 200$ bar. The injection rate is kept constant at 0.2 m/day at reservoir pressure. The permeability K is so large that the pressure deviation from its initial value does not exceed 0.5 bar. Thus, the volume injection rate can be assumed at the initial reservoir pressure. Other relevant parameters of the fluid are summarised in Table 7.

Assuming that all dispersion effects can be neglected, i.e., $\boldsymbol{\psi} = 0$, the formulated problem admits a simple analytical solution that includes two shocks S_c and S_T (Figure 3). The concentrations c_t are discontinuous at the leading shock S_c. The initial gas composition is preserved ahead of S_c, whereas the porous medium is saturated with the injected gas behind S_c. The reservoir is at the initial temperature $T = 50$°C ahead of the trailing shock S_T and at the injection temperature of 30°C behind S_T. When constructing the analytical solution, one can derive that S_c and S_T are propagating at constant velocities of 0.709 m/day and 0.0254 m/day, respectively. This piecewise constant distribution can be obtained as the solution to Equations (1)–(6) where $\boldsymbol{\psi} = 0$ and the fluid properties in Equation (4) are given by Equations (12)–(14). The velocities of the shocks are given by the Rankine-Hugoniot conditions corresponding to the balance Equations (1) and (2). The initial and boundary conditions are discussed in the previous paragraph.

As shown in Figure 3, the simulated distributions agree with the analytical solution. However, because of numerical dispersion, the shocks are zones of continuous transition of finite extent. The results are obtained with a regular grid consisting of 1000 elements and the maximum time step of 0.25 days.

3.2. Modelling Two-Phase Immiscible Displacement. This article is also supplemented with an example of an external library designed for modelling two-phase immiscible displacement. Although this USEREOS module allows for non-isothermal modelling, we consider a simpler case of an isothermal flow to validate the module.

In the module, the fluid properties are predicted using the following relations:

$$\rho_i = \rho_{i,\text{ref}}(1 + \alpha_i(P - P_{\text{ref}}) - \beta_i(T - T_{\text{ref}})), \tag{15}$$

$$h_i = C_i(T - T_{\text{ref}}),\ \mu_i = \text{const}, \tag{16}$$

$$s_i = \frac{\rho_i c_{t(i)}}{\sum_{k=1}^{2} \rho_k c_{t(k)}}, \tag{17}$$

$$c_{1(1)} = 1,\ c_{1(2)} = 0, c_{2(1)} = 1, c_{2(2)} = 0, \tag{18}$$

where α is the isothermal compressibility, β is the coefficient of

Table 3: API for *GetGlobalParameters*.

Interface: call GetGlobalParameters(*C-Name, M, P-Name, A-Name, units, opt*)			
Variable	Intent	Datatype	Description
C-Name	out	character	3-byte names of the fluid components (array of size $3 \times nc$)
M	out	real(8)	Molar weights of the components (array of size *nc*), kg/mol
P-Name	out	character	3-byte names of the fluid phases (array of size $3 \times np_{\max}$)
A-Name	out	character	8-byte names of the auxiliary arrays (array of size $8 \times na$); any such name must begin with the "#" character
units	out	character	Units of the auxiliary arrays (array of size $8 \times na$)
opt	out	integer(1)	Options (array of size 8)

Table 4: API for *PhaseEquilibrium*.

Interface call PhaseEquilibrium(P, T, c_t, np, Φ, Id, a, $Mode$)			
Variable	Intent	Datatype	Description
P	in	real(8)	Average fluid pressure, Pa
T	in	real(8)	Fluid temperature, degree K
c_t	in	real(8)	Bulk mass composition (array of size nc)
np	in/out	integer(4)	Number of fluid phases
Φ	in/out	real(8)	Parameters of the phases (2-D array of size $(6 + \text{nc}) \times np_{\max}$); only the first np columns of the array are relevant, i.e. Φ_i, $i = 1, \ldots, np$
Id	in/out	integer(1)	Phase identifier (array of size $np_{\max}$); only the first np elements of the array are relevant, i.e. Id_i, $i = 1, \ldots, np$
a	in/out	real(8)	Auxiliary variables (array of size na)
$Mode$	in	integer(1)	Calculation mode

Table 5: The USEREOS keyword syntax. It takes two arguments: the paths to the library and configuration files.

USEREOS
'Shared library' 'Configuration file' /

Table 6: Additional MUFITS mnemonics associated with the USEREOS option.

Parameter	Mnemonic
$c_{i(j)}$	C '$C\text{-}Name_{(j)}$'-'$P\text{-}Name_i$'
ρ_i	D'$P\text{-}Name_i$'
h_i	ENTHM'$P\text{-}Name_i$'
P_i	P'$P\text{-}Name_i$'
K_{ri}	REL'$P\text{-}Name_i$'
s_i	S'$P\text{-}Name_i$'
μ_i	VIS'$P\text{-}Name_i$'
$c_{t(j)}$	ZF'$C\text{-}Name_{(j)}$'

thermal expansion, P_{ref} and T_{ref} are the reference values of pressure and temperature, respectively, and C is the specific heat capacity at $P = \text{const}$. According to Equation (18), the component $j = k$ of the mixture forms the phase $i = k$, where $k = 1, 2$. For example, the phase (component) $i = 1$ is water, and the phase $i = 2$ is oil.

Table 7: Parameters of the ideal gas mixture.

	N_2	O_2	CH_4
$M_{(j)}$, g/mol	28	32	16
$C_{(j)}'$, J/(mol•K)	29.12	29.44	35.2
$\mu_{(j)}$, cP	0.016	0.022	0.012

We consider a 1-D flow in a horizontal homogeneous reservoir at $X \in [0,100]$ m. The porosity and permeability of the medium are $\phi = 0.2$ and $K = 200$ mD, respectively. The relative permeability is given by

$$K_{r1} = s_1^2,\ K_{r2} = s_2^3. \tag{19}$$

The capillary pressure is assumed to be 0. At $t = 0$, the reservoir is saturated by the oil phase $i = 2$, i.e., $c_{t(1)} = 0$, at $P = P_{\text{ref}} = 200$ bar and $T = T_{\text{ref}}$. The reservoir temperature T_{ref} is kept constant at all $t > 0$. Therefore, its value as well as the values of β_i and C_i does not influence the flow. The initial pressure is kept constant at the open boundary $X = 100$ m. The water phase ($i = 1$) is injected through the boundary $X = 0$ m at constant injection rate of 0.1 m/day.

The compressibilities α_i are assumed to be small such that the phase densities ρ_i can be considered constant.

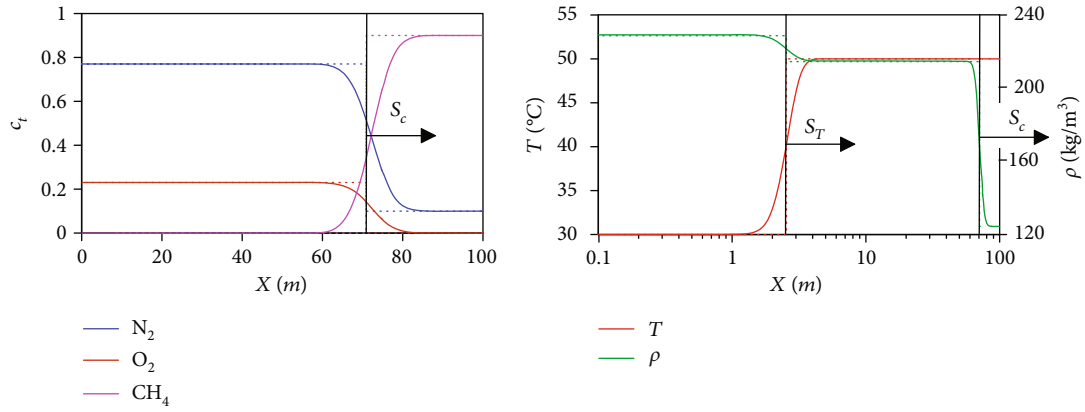

Figure 3: Distributions of (a) c_t and (b) ρ and T at $t = 100$ days. The solid and dotted curves show the numerical and analytical solutions, respectively.

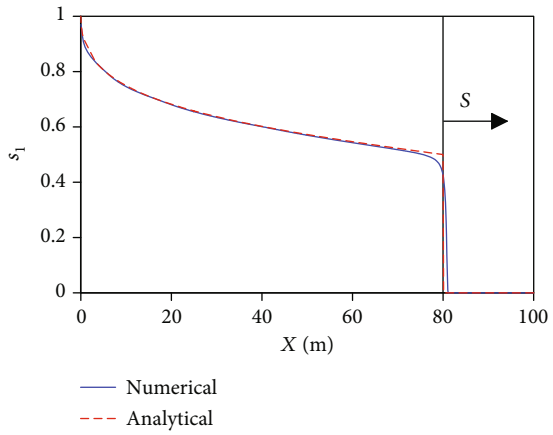

Figure 4: Distribution of the water saturation at $t = 100$ days. The solid and dashed curves correspond to the numerical and analytical solutions, respectively.

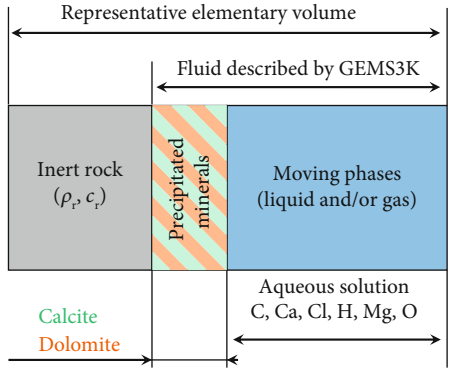

Figure 5: Sketch of the representative elementary volume. The fluid, i.e., the moving phases and precipitated minerals, occupies the pore space. The rest of the volume is regarded as inert rock.

Table 8: Fluid composition (mass fractions) at the inflow boundary $X = 0$ and at $t = 0$.

	Boundary	Initial
C	1.20*e*-12	3.96*e*-06
Ca	4.01*e*-12	1.32*e*-05
Cl	7.09*e*-05	7.09*e*-09
H	1.12*e*-01	1.12*e*-01
Mg	2.43*e*-05	2.43*e*-09
O	8.88*e*-01	8.88*e*-01

Therefore, only the phase viscosities, $\mu_1 = 0.75$ cP and $\mu_2 = 1.5$ cP, are relevant parameters.

The water injection results in the immiscible displacement of oil from the boundary $X = 0$ m to $X = 100$ m (Figure 4). The saturation distribution is governed by the Buckley–Leverett solution [22]. This analytical solution consists of the leading displacement front S, at which s_1 is discontinuous, and the trailing Riemann wave, in which s_1 changes continuously, reaching $s_1 = 1$ at $X = 0$ m. One can show that for Equation (19) and specified phase viscosities μ_1 and μ_2, the shock propagates at a velocity of 0.8 m/day. Thus, S is exactly at $X = 80$ m at $t = 100$ days.

As shown in Figure 4, the numerical solution agrees with the analytical solution to the Buckley–Leverett problem. The simulation was conducted with a regular grid consisting of 1000 cells. The time step was limited by 0.25 days to reduce truncation errors.

4. Coupling with GEMS3K

This more elaborate example concerns an EoS-module that can be used as a template for MUFITS coupling with a geochemical software. We couple the simulator with the numerical code GEMS3K to simulate reactive transport in porous media. GEMS3K is a C++ library implementing the efficient IPM-3 algorithm for GEM [17]. Previously, it was coupled with various

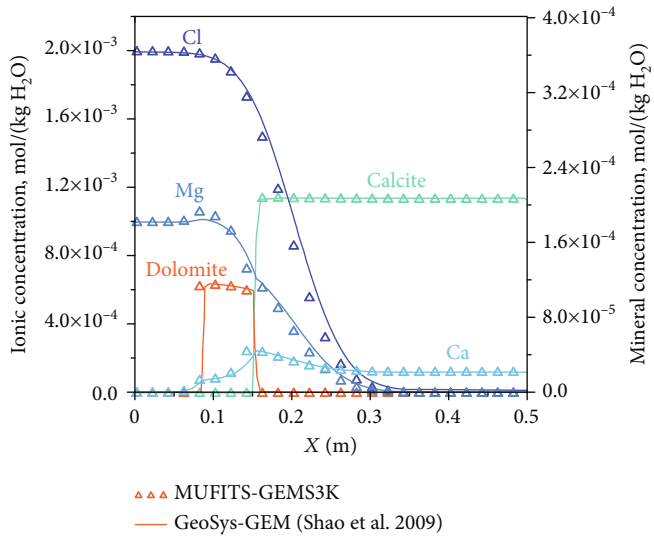

FIGURE 6: Concentrations of the aqueous components and minerals at $t = 21000$ s. The solid curves show the results of Shao et al. [24], and the symbols are the results of the MUFITS–GEMS3K coupled simulation.

transport codes such as CSMP++, OpenGeoSys, COMSOL, and other software packages [23–25].

In the developed USEREOS, the representative elementary volume of a porous medium consists of the inert rock, whose properties are calculated in the kernel, and the pore fluid described by GEMS3K (Figure 5). The fluid can split into different phases, including liquid and gas, that can move through the porous medium and several precipitated minerals, i.e., solid phases. Thus, the rock consists of the inert part, which does not participate in chemical reactions, and several minerals that can precipitate from or dissolve in the moving phases. We assume that the reactive transport occurs under a local chemical equilibrium between fluid and rock. Thus, the equilibrium is characterised by P, T, and the bulk concentrations c_t of the primary chemical elements that the fluid is composed of. For a given P, T, and c_t, GEMS3K calculates the phase equilibrium, including the number of phases np and the mole amounts of the chemical species in the phases. The latter quantities are translated by USEREOS to the mass concentrations $c_{i(j)}$. The chemical speciation is stored in the auxiliary array discussed in Section 2.1.4.

The configuration files (see Section 2.1.6) for the developed USEREOS are identical to those for GEMS3K. They can be prepared in GEM-Selektor, a program for geochemical modelling with a graphical user interface, which provides access to various thermodynamic databases and allows for exporting parameters of the fluid model and numerical controls in a format directly compatible with GEMS3K.

To validate and demonstrate the developed USEREOS, we consider a 1-D benchmark problem of mineral dissolution–precipitation that is often used for the validation of reactive transport modelling [8, 23, 24, 26]. Here, we follow the problem statement presented in [23, 24]. We consider a 1-D porous column located at $X \in [0, 0.5]$ m, where $\phi = 0.32$. At $t = 0$, the column is saturated with an aqueous solution in equilibrium with calcite at $P = 1$ bar and $T = 25$°C. The initial pressure is kept constant at the open boundary $X = 0.5$ m. The column is flushed through the boundary $X = 0$ with a $MgCl_2$ solution at a constant flow rate $q = 3 \bullet 10^{-6}$ m/s. The initial and boundary elemental concentrations are listed in Table 8. The flow is assumed to be isothermal at $T = 25$°C. The pressure in [23, 24] is also fixed at 1 bar. In MUFTIS, the presence of a pressure gradient is necessary to produce a flow according to Equation (3), but for the thermodynamic equilibria calculations in GEMS3K, the pressure is set to 1 bar to achieve a better match to the reference results.

The mechanical dispersion in this problem is modelled by Equations (7) and (8). Since the study is 1-D, only the longitudinal dispersion length, $\lambda_L = 0.00214$ m, must be specified. This length corresponds to that used by Shao et al. [24]. However, because Shao et al. [24] use a different definition for the dispersion length, the quantity λ_L scales to that in [24] by the factor ϕ.

To simulate the fluid flow, we use a fluid consisting of the six components listed in Table 8. The simulated distributions of the concentrations of Cl, Mg, and Ca in the fluid and the concentrations of calcite and dolomite after 21000 s of $MgCl_2$ injection are shown in Figure 6. The simulated distributions are in good overall agreement with the results presented in [24]. Herewith, dolomite is formed in a finite moving zone, and calcite dissolves at the leading reaction front. Minor differences between the concentration profiles can be the result of using different equations for the hydrodynamic part of the model and different coupling schemes. In [23, 24], the sequential noniterative approach (SNIA) is employed. This approach assumes that the solution of the transport equations is conducted separately from the chemical equilibrium calculations at each simulation time step. In this work, the transport and the chemical equilibria are calculated simultaneously at every time step (the global implicit

approach). This ensures that both the mass balance and the assumption of the chemical equilibrium are satisfied at the end of each time step.

5. Conclusions

We demonstrate that the developed extension is robust and useful because it can be applied in various studies ranging from the petroleum reservoir simulations to nonisothermal flows and reactive transport modelling. By using the 1-D studies, we validate the correctness of the program implementation of numerical algorithms associated with USEREOS. Although the considered benchmarks are rather simple, USEREOS can be applied in more sophisticated 2-D and 3-D scenarios involving complicated phenomena and processes. These can include coupled reservoir-wellbore flows or transport in fractured–porous media, which can be modelled with the available capabilities of the software implemented in its kernel.

The created capabilities for MUFITS coupling with other software for fluid property prediction deserve special attention. Aiming at reactive transport modelling, we consider only an example of coupling with GEMS3K. However, in the same way, the simulator can be coupled with other geochemical software, e.g., PHREEQC [27] or those implementing accelerated approaches [15, 28]. It is important that from now on, such coupling can be done independently by the software's users. This provides the scientific community with an advanced and innovative opportunity for modelling subsurface flows, given that the developed API between MUFITS and USEREOS is relatively simple and easy to use.

Data Availability

The source code of the developed USEREOS (v1.0) modules, the compiled executables, and all input data associated with the presented development and benchmark studies can be downloaded at [21, 22]. The source code of GEMS3K can be downloaded at http://gems.web.psi.ch/GEMS3K/. MUFITS executable files are freely available at [21] and http://www.mufits.imec.msu.ru/download.html.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The authors acknowledge funding from the Russian Science Foundation under Grant # 20-17-00184.

References

[1] S. Geiger, R. Haggerty, J. H. Dilles, M. H. Reed, and S. K. Matthäi, "New insights from reactive transport modelling: the formation of the sericitic vein envelopes during early hydrothermal alteration at Butte, Montana," *Geofluids*, vol. 2, no. 3, p. 201, 2002.

[2] S. A. Bea, U. K. Mayer, and K. T. B. MacQuarrie, "Reactive transport and thermo-hydro-mechanical coupling in deep sedimentary basins affected by glaciation cycles: model development, verification, and illustrative example," *Geofluids*, vol. 16, no. 2, p. 300, 2016.

[3] J. Bear, "Porous media," in *Theory and Applications of Transport in Porous Media*, pp. 1–98, Springer, 2018.

[4] J. R. Fanchi, *Principles of Applied Reservoir Simulation*, Elsevier, 2006.

[5] K. Aziz and A. Settari, *Petroleum Reservoir Simulation*, Applied Science Publishers, 2002.

[6] K. Pruess, C. Oldenburg, and G. Moridic, "TOUGH2 User's Guide Version 2.0," Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 1999.

[7] T. Xu, N. Spycher, E. Sonnenthal, G. Zhang, L. Zheng, and K. Pruess, "TOUGHREACT Version 2.0: a simulator for subsurface reactive transport under non-isothermal multiphase flow conditions," *Computational Geosciences*, vol. 37, no. 6, pp. 763–774, 2011.

[8] H. Prommer, D. A. Barry, and C. Zheng, "MODFLOW/MT3DMS-based reactive multicomponent transport modeling," *Ground Water*, vol. 41, no. 2, pp. 247–257, 2003.

[9] A. Afanasyev, "Hydrodynamic modelling of petroleum reservoirs using simulator MUFITS," *Energy Procedia*, vol. 76, pp. 427–435, 2015.

[10] A. Afanasyev, T. Kempka, M. Kühn, and O. Melnik, "Validation of the MUFITS reservoir simulator against standard CO_2 storage benchmarks and history-matched models of the Ketzin pilot site," *Energy Procedia*, vol. 97, pp. 395–402, 2016.

[11] Y. Wang, D. Voskov, M. Khait, and D. Bruhn, "An efficient numerical simulator for geothermal simulation: a benchmark study," *Applied Energy*, vol. 264, article 114693, 2020.

[12] A. Afanasyev, "Numerical modelling of solute flow dispersion in porous media using simulator MUFITS," *Journal of Physics Conference Series*, vol. 1129, article 012002, 2018.

[13] K. H. Coats, "An equation of state compositional model," *Society of Petroleum Engineers Journal*, vol. 20, no. 5, pp. 363–376, 1980.

[14] C. I. Steefel, C. A. J. Appelo, B. Arora et al., "Reactive transport codes for subsurface environmental simulation," *Computational Geosciences*, vol. 19, no. 3, pp. 445–478, 2015.

[15] M. De Lucia, T. Kempka, and M. Kühn, "A coupling alternative to reactive transport simulations for long-term prediction of chemical reactions in heterogeneous CO_2 storage systems," *Geoscientific Model Development*, vol. 8, no. 2, pp. 279–294, 2015.

[16] A. Afanasyev and I. Utkin, "Modelling ground displacement and gravity changes with the MUFITS simulator," *Advances in Geosciences*, vol. 54, pp. 89–98, 2020.

[17] D. A. Kulik, T. Wagner, S. V. Dmytrieva et al., "GEM-Selektor geochemical modeling package: revised algorithm and GEMS3K numerical kernel for coupled simulation codes," *Computational Geosciences*, vol. 17, pp. 1–24, 2013.

[18] A. Gysi, Y. Mei, and T. Driesner, "Advances in numerical simulations of hydrothermal ore forming processes," *Geofluids*, vol. 2020, Article ID 7649713, 4 pages, 2020.

[19] H. Salimi, K.-H. Wolf, and J. Bruining, "Negative saturation approach for non-isothermal compositional two-phase flow simulations," *Transport in Porous Media*, vol. 91, no. 2, pp. 391–422, 2011.

[20] A. Afanasyev and I. Utkin, "USEREOS v1.0," 2021, https://github.com/utkinis/USEREOS/.

[21] A. Afanasyev and I. Utkin, "MUFITS. User-supplied EoS-modules," 2021, http://www.mufits.imec.msu.ru/example-usereos.html/.

[22] S. E. Buckley and M. C. Leverett, "Mechanism of fluid displacement in sands," *Transactions of AIME*, vol. 146, no. 1, pp. 107–116, 1942.

[23] A. Yapparova, T. Gabellone, F. Whitaker, D. A. Kulik, and S. K. Matthäi, "Reactive transport modelling of dolomitisation using the new CSMP++GEM coupled code: governing equations, solution method and benchmarking results," *Transport in Porous Media*, vol. 117, no. 3, pp. 385–413, 2017.

[24] H. Shao, S. V. Dmytrieva, O. Kolditz, D. A. Kulik, W. Pfingsten, and G. Kosakowski, "Modeling reactive transport in non-ideal aqueous-solid solution system," *Applied Geochemistry*, vol. 24, no. 7, pp. 1287–1300, 2009.

[25] V. J. Azad, C. Li, C. Verba, J. H. Ideker, and O. B. Isgor, "A COMSOL-GEMS interface for modeling coupled reactive-transport geochemical processes," *Computational Geosciences*, vol. 92, pp. 79–89, 2016.

[26] L. H. Damiani, G. Kosakowski, M. A. Glaus, and S. V. Churakov, "A framework for reactive transport modeling using FEniCS–Reaktoro: governing equations and benchmarking results," *Computational Geosciences*, vol. 24, no. 3, pp. 1071–1085, 2020.

[27] D. L. Parkhurst and C. A. J. Appelo, *PHREEQC (Version 3.0.4)–a computer program for speciation, batch speciation, one-dimensional transport, and inverse geochemical calculations*, U.S. Geological Survey Techniques and Methods, book, 2013.

[28] J. Jatnieks, M. De Lucia, D. Dransch, and M. Sips, "Data-driven surrogate model approach for improving the performance of reactive transport simulations," *Energy Procedia*, vol. 97, pp. 447–453, 2016.