Available online at www.sciencedirect.com

ScienceDirect

Energy Procedia 132 (2017) 285–290

Energy Procedia

www.elsevier.com/locate/procedia

11th Nordic Symposium on Building Physics, NSB2017, 11-14 June 2017, Trondheim, Norway

Implementation of salt transport modules in a solver framework for heat and mass transport in porous materials

Andreas Nicolai*

Institut für Bauklimatik, TU Dresden, Zellescher Web 17, 01069 Dresden, Germany

Abstract

The modeling framework and simulation program DELPHIN 6 for heat and mass transport in porous media is implemented as set of modules for balance equations, transport functions, sources and sinks, boundary and contact conditions. The article starts with an overview of the software architecture, modeling assumptions and numerical considerations. Then, the extension of the framework towards salt transport is presented from the perspective of a physical modeler without in-depth numerical programming experience. It will be shown, that the current modeling framework is sufficient to enable researchers and modelers to make extensions without investing excessive time on core solver and data structure development.

Peer-review under responsibility of the organizing committee of the 11th Nordic Symposium on Building Physics.

Keywords: Simulation, heat and mass transport, salt transport, phase transitions, module, implementation, framework, solver

1. Introduction

Development of sophisticated simulation software for heat and mass transport in porous media is far from trivial, especially when 2D/3D geometries are to be considered. Also, with an increasing number of model components and physical effects implemented, source code becomes quite complex. Typically, further model extensions require detailed knowledge of numerical solution methods, algorithms, underlying data structures and relevant software libraries. For researchers, doctoral and master students, or engineers the development of new solvers from scratch is typically not an option and most likely not economically feasible.

* Corresponding author.
E-mail address: andreas.nicolai@tu-dresden.de

1876-6102
Peer-review under responsibility of the organizing committee of the 11th Nordic Symposium on Building Physics
10.1016/j.egypro.2017.09.728

Most researchers would like to concentrate on the physical model and parametrization, alone, and not deal with details of the numerical solution method. This lead to a design concept for an abstract framework for modeling and simulating heat and mass transfer in porous media [1]. Since the presentation of the first design concept, it has been implemented and forms now the core of the new DELPHIN 6 simulation engine. In this article, the use of the framework will be shown while it is being extended to include a salt transport model.

2. Basic calculation procedure and implementation

Before going into detail of the salt transport extension, a short overview will be given of the numerical solution algorithms, the principle software architecture and some key modeling assumptions.

2.1. Spatial discretization and time integration

Transport processes in porous media are described by partial differential equations (PDE). First, a suitable spatial discretization technique is applied and the system of PDEs is transformed into a system of ordinary differential equations (ODE). In the DELPHIN code the Finite-Volume-Method is used which ensures strictly mass and energy conservative solutions. Additionally, convective transport quantities that arise in case of air flow or salt transport, can be handled very accurately.

This sparsely coupled system of ODE, which may become quite large in the case of 2D/3D problems, expresses the time evolution of the conserved quantities, i.e. the energy and mass densities. The time integration is performed using an implicit method, specifically a backward differentiation formula (BDF), implemented as variable-order multi-step method with variable time step sizes [2]. The method order and time step sizes are adjusted based on local truncation error estimates, thus ensuring time-accurate evolution of our conserved quantities.

The non-linear systems of equations arising from the implicit scheme are solved with a modified Newton method. The linear equation systems solved within each Newton iteration tend to be large and sparse. The banded structure of corresponding Jacobian matrices is considered through usage of dedicated factorization and backsolving functions.

The interface to the physical model developer is the so-called *system function*, essentially the time derivatives for all balance equations in all elements. Key task for the model developer is to compute said derivatives, and hence essentially compute the divergences of fluxes across element's sides, and the source/sink rates. This system function is also used in a semi-generic way to approximate the Jacobian matrix, thus eliminating the need for implementing physics a second time for the purpose of Jacobian matrix assembly.

2.2. Evaluation of the physical model

The implementation of the physical model (evaluation of the system function) remains the core task when developing the numerical simulation program. The system function is given a time point and a vector of variables of states (conserved quantities), comprising the current estimate of the solution at this time point. If both heat and moisture balances are used, the vector with conserved variables holds energy densities and moisture mass densities in all Finite Volumes (also termed *elements*).

When implementing the physical model, essentially four steps are needed:

1. Calculation of element-based properties from conserved variables, for example temperatures, pressures, densities, and transport coefficients, etc. in all elements,
2. Calculation of energy and mass fluxes across sides/interfaces between elements and at the boundary,
3. Compute source and sink rates in all elements,
4. Compute divergences of fluxes and add source rates to obtain time derivatives of all balance equations in all elements

The computed time-derivatives are returned to the numerical integration engine and integrated in time.