

## Experimental validation of a new adaptable LCM mold filling software

Christof Obertscheider, Ewald Fauster & Simon Stieber

**To cite this article:** Christof Obertscheider, Ewald Fauster & Simon Stieber (2023) Experimental validation of a new adaptable LCM mold filling software, *Advanced Manufacturing: Polymer & Composites Science*, 9:1, 2282310, DOI: [10.1080/20550340.2023.2282310](https://doi.org/10.1080/20550340.2023.2282310)

**To link to this article:** <https://doi.org/10.1080/20550340.2023.2282310>



© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.



Published online: 12 Dec 2023.



Submit your article to this journal [↗](#)



Article views: 152



View related articles [↗](#)



View Crossmark data [↗](#)

## Experimental validation of a new adaptable LCM mold filling software

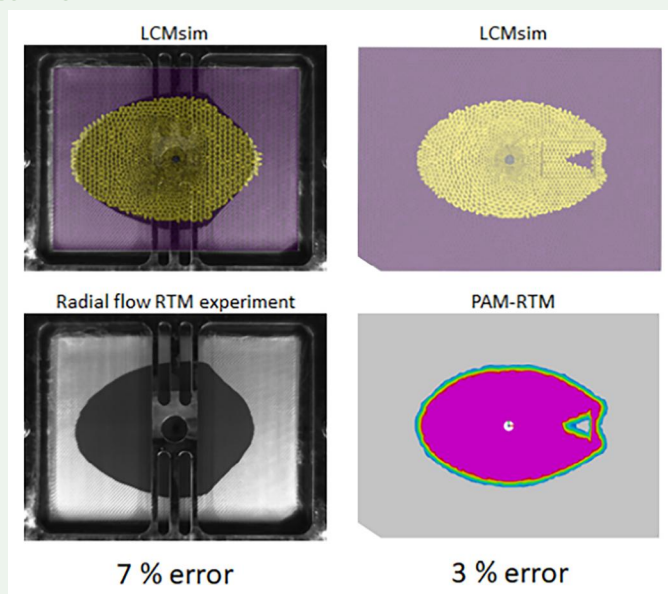
Christof Obertscheider<sup>a</sup>, Ewald Fauster<sup>b</sup> and Simon Stieber<sup>c</sup>

<sup>a</sup>Aerospace Engineering Department, University of Applied Sciences, Wiener Neustadt, Austria; <sup>b</sup>Department Polymer Engineering and Science, Montanuniversität Leoben, Leoben, Austria; <sup>c</sup>Institute for Software and Systems Engineering, University of Augsburg, Augsburg, Germany

### ABSTRACT

Resin Transfer Molding (RTM) is a manufacturing process for fiber reinforced polymer composites where dry fibers are placed inside a mold and resin is injected under pressure. During mold design, filling simulations can study different manufacturing concepts (i.e. placement of injection gates and vents) to guarantee complete filling of the part and avoid air entrapment where flow fronts converge. In this work, a novel software tool LCMsim, which was implemented by the authors, is benchmarked against other tools and real-world flow experiments. Its development was driven by two ideas: Easy-of-use for the mold engineer and maximum flexibility for the researcher. Two experiments were used for validation. In the first, zones with different preform properties were present and in the second, race-tracking was enforced. Flow fronts from LCMsim and experiment agree with 7% error and simulated flow fronts from LCMsim and the commercially available software PAM-RTM agree with 3% error.

### GRAPHICAL ABSTRACT



### ARTICLE HISTORY

Received 18 May 2023  
Accepted 30 October 2023

### KEYWORDS

Resin transfer molding (RTM); liquid composite molding (LCM); filling simulation; computational fluid dynamics (CFD); shell mesh; finite area method; flow front tracking; resin flow

## 1. Introduction

### 1.1. LCM process

In all liquid composite molding (LCM) technologies dry fiber preforms are placed into a mold which is subsequently closed or sealed with a flexible bag by drawing a vacuum. Resin then enters into the mold until the preform is fully wetted with resin. As soon

as the resin is cured the mold is opened and the part is extracted. In Resin Transfer Molding (RTM) resin is injected into a matched mold under pressure which gives excellent surface finish on both sides and high fiber volume fractions (55-65%) can be obtained. The mold walls are considered rigid and the preform is stationary during injection. In contrast to RTM, in VARI vacuum pulls resin

**CONTACT** Christof Obertscheider  [christof.obertscheider@fhwn.ac.at](mailto:christof.obertscheider@fhwn.ac.at)  Aerospace Engineering Department, University of Applied Sciences, Wiener Neustadt, Austria

© 2023 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

through the preform on a single-sided mold with vacuum bagging. For a complete process description see e.g. Strong [1] or Niu [2].

### 1.2. LCM filling simulations

During mold design, filling simulations can study different manufacturing concepts (i.e. placement of injection gates and vents) to guarantee complete filling of the part and avoid air entrapment where flow fronts converge.

Required input for filling simulations are the mold cavity geometry, preform properties (e.g. permeability, porosity and thickness values which describe the fibrous preform regions), process parameters (e.g. initial cavity pressure and injection pressure) and fluid properties (e.g. resin viscosity).

During pre-processing the physical domain (i.e. the mold cavity) is transferred to a numerical domain (i.e. the mesh) and preform properties and initial conditions for the flow variables are assigned to the cells of the mesh. The solver calculates how the values of the flow variables in the cells change in the course of the filling. Different physical models of the filling can be implemented in the solver. The physical models have to describe all relevant issues of the filling process. During post-processing the resin flow front at different time instances is visualized with contour plots.

Before a simulation is started, parameters which describe the flow through porous media (permeability and porosity) must be determined. In-plane permeability is characterized experimentally from linear or radial flow experiments where the flow front location is tracked over time and the entries of the in-plane permeability tensor are computed according to specifically developed mathematical algorithms, see May et al. [3]. Experiments for different porosity levels are performed. Porosity is the fraction of the volume of voids over the total volume. For fibrous preforms inside a mold, the volume averaged porosity can be calculated from cavity thickness  $t$ , number of plies  $n$ , mass density  $\rho_f$  and areal weight  $A_f$  of the preform material as  $\varepsilon = 1 - (nA_f)/(\rho_f t)$ . The second term is the ratio of the preform volume over the total volume, see for example Neitzel et al. [4].

### 1.3. Requirements for LCM software

General purpose CFD software packages such as ANSYS® Fluent® or OpenFOAM as well as specialized software packages such as PAM-RTM®, RTM-Worx®, LIMS® or myRTM can be used for filling simulations.

The models implemented in the different software packages mentioned above perform well for

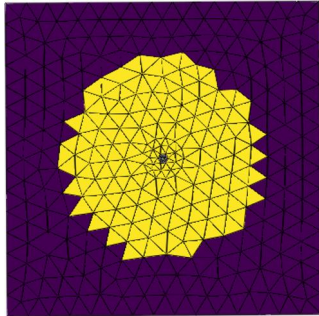
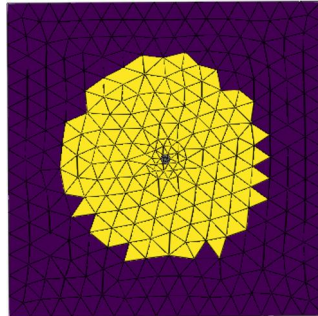
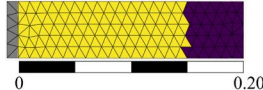
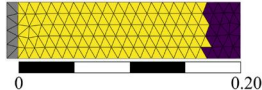
most scientific and engineering applications, but there are applications where no satisfactory solution is found, e.g. for Compression-RTM where an equation of state to model fluid compressibility must be considered. See e.g. Bickerton and Abdullah [5] for a process description.

Since the required physical model to describe all relevant issues of the filling process often is not known in advance, a functional requirement for a new filling software is maximum flexibility for the user. Maximum flexibility is only possible if the code is open-source and code modification is possible for people with basic CFD and programming knowledge, similar to creating new solvers in the open-source CFD package OpenFOAM. Typical code modifications are the implementation of different boundary conditions (control pressure level or control opening and closing of ports) or adding additional equations (e.g. temperature or degree-of-cure equations since the resin viscosity is known to depend on temperature and degree of cure) or extracting the solver section (i.e. the loop for the time evolution) and integrate in separate code. The latter was already done for a control study with three inlet ports to keep the flow front straight in a rectilinear flow with reinforcement patches, see Stieber et al. [6]. If different models can be implemented easily, as a side result, comparisons between different models are possible also for complex geometries and not only for linear and radial test cases where analytical formulas are available.

In addition, mold engineers need an easy-to-use software to determine the optimal location of gates and vents. This is possible, if a shell mesh of the cavity with specified sets for different preform regions is available and the software can be started from a GUI with only preform-, process- and ports-relevant input data.

Commercial software packages (with the advantage of a user-friendly GUI, an extensive documentation, a large number of industrial case studies, ... and with the obvious limitation in customizability) cannot fulfill these functional requirements. Non-commercial software packages from the list above are the open source CFD software package OpenFOAM and myRTM, a free industry standard for filling simulations. OpenFOAM was already used for implementing and testing new filling models, see for example Seuffert et al. [7] or Sebastian et al. [8], but OpenFOAM comes without GUI and requires both CFD and programming knowledge to perform appropriate LCM filling simulations. myRTM can only simulate RTM filling. The predicted filling pattern is correct but the predicted filling time differs considerably from that of the actual component, see Barandun et al. [9] and myRTM [10]. In contrast to RTMsim and

**Table 1.** Comparison of the new LCMsim module with the pre-existing RTMsim module.

	RTMsim module	LCMsim module
<b>Advantages</b>	<ul style="list-style-type: none"> <li>• GUI or text input file for assignment of preform and process parameters</li> <li>• Open source</li> <li>• Validation and verification cases for RTM available, see radial flow test case below and Obertscheider and Fauster [11]</li> </ul>	<ul style="list-style-type: none"> <li>• GUI or text input file for assignment of preform and process parameters</li> <li>• Open source</li> <li>• Prepared for code extensions/modifications for people with basic CFD or programming knowledge</li> <li>• Flow physics model allows for RTM and VARI simulations and also other process variants</li> </ul>
<b>Limitations</b>	<ul style="list-style-type: none"> <li>• Shell mesh required, no mesher for surfaces included</li> <li>• Flow front propagation too slow for high viscosity and small permeability, see linear flow test case below</li> <li>• Only RTM simulations</li> </ul>	<ul style="list-style-type: none"> <li>• Shell mesh required, no mesher for surfaces included</li> <li>• Not yet run-time optimized</li> </ul>
<b>Radial flow test case</b> from Isoldi et al. [12] to show similar filling pattern for typical RTM filling test case	Preform size $0.6 \times 0.6 \times 0.003 \text{ m}^3$ , central injection gate (grey) with 0.02 m diameter, 35,000 Pa difference between injection and initial cavity pressure, dynamic viscosity 0.06 Pas, porosity 0.7, isotropic permeability $3 \cdot 10^{-10} \text{ m}^2$ . Same mesh for all simulations. Contour plot of the filling fraction after 200 s (yellow/blue is filled/unfilled):	
		
<b>Linear flow test case</b> to show differences for increased viscosity and decreased permeability	Preform size $0.2 \times 0.06 \times 0.003 \text{ m}^3$ , line injection at the left (grey), 35,000 Pa difference between injection and initial cavity pressure, dynamic viscosity 0.1 Pas, porosity 0.7, isotropic permeability $3 \cdot 10^{-11} \text{ m}^2$ . Same mesh for all simulations. Flow front position at 0.17 m calculated with analytical formula, see for example Isoldi et al. [12]. Contour plot of the filling fraction after 1000s:	
		

*Notes:* The first two lines list advantages and limitations. The third line shows the results of a typical radial RTM flow test case. The fourth line shows the results of a linear RTM flow test case with increased viscosity and decreased permeability. RTMsim gives reliable results for most RTM test cases, particularly if only the filling pattern is considered. LCMsim in contrast to RTMsim performs well for all relevant viscosity and permeability ranges (see the difference in the linear flow) and can simulate RTM, VARI and other process variants.

LCMsim, myRTM includes a mesher for surface geometries and preform properties can be assigned graphically. The main differences between LCMsim and the pre-existing RTMsim is the capability to perform well for all relevant viscosity and permeability ranges, the capability to simulate VARI filling in addition to RTM and the customizability for other process variants. Table 1 compares the new LCMsim module with the pre-existing RTMsim module.

#### 1.4. Description of new LCM software

The existing Julia module RTMsim described in Obertscheider and Fauster [11] for iso-thermal RTM filling simulations was extended with a new fluid model and with new functionalities. The extended Julia module LCMsim can be downloaded from the GitHub repository <https://github.com/obertscheiderfhw/LCMsim>. Installation instructions and instructions on how to run a simulation are given in a README file on the repository. Figure 1 shows the GUI with explanations and typical values.

The work at hand describes the flow physics model which is implemented in LCMsim in detail and presents a validation of LCMsim for RTM. Real-world RTM experiments under real process conditions, i.e. permeameter experiments with zones with different in-plane permeability and porosity levels, are the basis for the RTM assessment.

## 2. Materials and methods

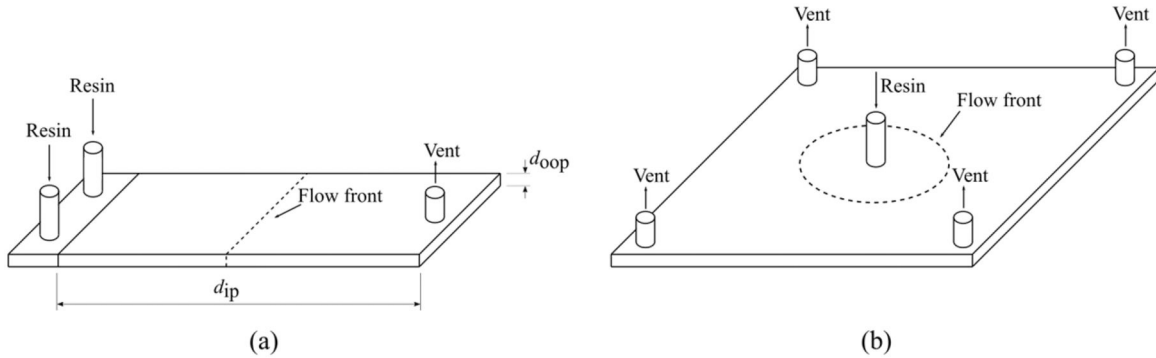
### 2.1. Physical model

RTM is a manufacturing process suitable for producing thin-walled fiber reinforced polymer composites (FRPC) parts. The thickness of the part is much smaller than the overall dimensions as depicted in Figure 2. Dry reinforcement fibers are placed inside a mold and resin is injected under pressure into the fibrous preform. On a macroscopic level, the physical quantities porosity and permeability are used to describe resin flow through the fibrous media.

For usual textiles the out-of-plane permeability  $k_{oop}$  is one or two orders of magnitude smaller than the in-



**Figure 1.** LCMsim GUI filled with parameters for the radial flow test case from Table 1. All parameters in SI units. The GUI is divided into four major sections as highlighted, with the most important fields of input data mentioned in the describing text.



**Figure 2.** A schematic of a thin-shell RTM geometry (in-plane dimension much greater than the out-of-plane dimensions,  $d_{ip} \gg d_{oop}$ ) for manufacturing a flat plate with linear flow and line injection gate (a) and with radial flow and single point injection (b).

plane permeability  $k_{ip}$ , see for example Simacek and Advani [13] or Sirtautas et al. [14]. For thin-walled parts considered in this study the thickness  $d_{oop}$  is two orders of magnitude smaller than the characteristic in-plane dimension  $d_{ip}$ . This gives an aspect ratio

$$AR = \frac{d_{oop}}{d_{ip}} \sqrt{\frac{k_{ip}}{k_{oop}}} \quad (1)$$

between 0.01 and 0.1. Only for higher aspect ratio (for example thick laminates or a flow distribution

medium on top of the preform) three-dimensional modeling of the flow is desirable [13]. Consequently, transverse flow is neglected and the flow is modeled as two-dimensional, considering in-plane flow contributions only.

In addition one assumes slip boundary conditions at the walls of the cavity since the flow is described on a macroscopic level. In the absence of a porous preform a flow profile with zero velocity at the top and bottom cavity walls and maximum velocity in



**Figure 3.** Flow velocity profile through an empty cavity (a) and assumed volume-averaged (superficial) flow velocity profile through a porous cavity (b).

the middle would develop. Since the cavity is filled with the preform, flow takes place in small flow channels with potential cross-flow but is described by a superficial velocity and the assumption of a plain flow front is valid. Figure 3 illustrates this difference. The physical velocity through the small channels is related to the superficial velocity via the porosity (fraction of the void volume over the total volume). The superficial velocity is smaller than the physical such that the same volumetric flow rate moves through a cross sectional area, see e.g. Tucker [15] or Nield and Bejan [16].

In contrast to the original model from Obertscheider and Fauster [11], a more realistic model was implemented in the novel LCMsim software module. The implemented model results in an algorithm which (a) is explicit in time and (b) avoids pressure-velocity coupling iteration during one time step.

Initially the porous cavity is either evacuated or filled with air (air density according to cavity pressure) and pressurized resin is injected into the fibrous preform.

Fluid flow is described by (i) a compressible continuity equation for the mixture density, (ii) an equation-of-state which describes the pressure build-up with increasing mixture density and (iii) a momentum equation with dynamic viscosity of the resin in the Darcy term. The in-plane velocity vector  $\mathbf{u} = (u, v)$  is described in the system of the orthotropic in-plane permeability. For tracking the flow front a volume-of-fluid equation is solved.

The physical quantities in the governing equations are functions of space  $\mathbf{x}$  and time  $t$  where  $\mathbf{x} \in \Omega \subset \mathbb{R}^3$  and  $\Omega$  is the fluid body also called the fluid cavity. The governing equations<sup>1</sup> are

$$\frac{\partial \varepsilon \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (2)$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u}) = -\nabla p + S_u \quad (3)$$

where

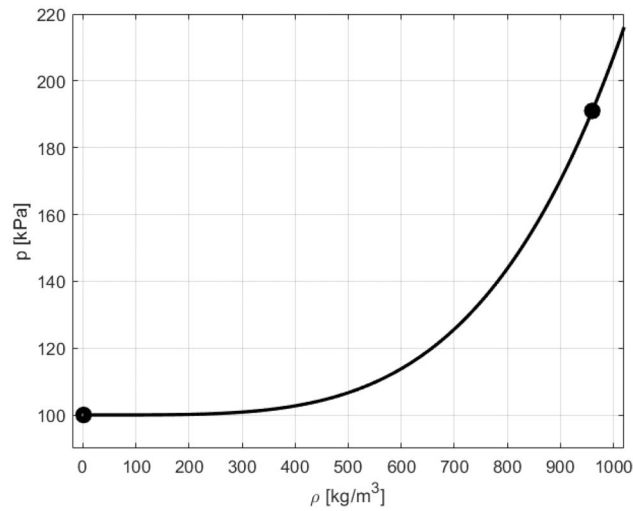
- $\rho = c\rho_{resin} + (1-c)\rho_{air}$  is the mass density of the mixture,

- $\varepsilon$  is the porosity of the preform,
- $\rho_{resin}$  is resin mass density at injection pressure,
- $\rho_{air}$  is the air mass density at initial cavity pressure,
- the binary fraction function  $c$  is equal to  $c=0$  (i.e. empty) if  $\rho < \bar{\rho}$  and  $c=1$  (i.e. completely filled) if  $\rho \geq \bar{\rho}$  and the threshold value is  $\bar{\rho} = (\rho_{resin} + \rho_{air})/2$ ,
- $\mathbf{u}$  is the superficial velocity in a local coordinate system which is related to the physical velocity via the porosity  $\varepsilon$  given by the stationary fibrous media,
- $\mathbf{u}\mathbf{u}$  is the dyadic product,
- $p$  is the fluid pressure,
- $S_u$  is a source term which contains the pressure loss from flow through a porous medium according to Darcy's law, i.e.  $S_u = -\mu\mathbf{K}^{-1}\mathbf{u}$  with dynamic viscosity  $\mu$  and permeability tensor  $\mathbf{K}$ ,
- the contribution  $\rho\mathbf{g}$  of the gravitational acceleration in the momentum equation is neglected, since in the currently considered applications it is assumed that gravity pressure effects are negligible compared to the pressure differences resulting from injection and cavity pressure,
- the contributions of the viscous stress tensor in the momentum equation are neglected, since in many porous media flow problems the viscous stress is much smaller than the solid-fluid drag, see Advani [17],
- the equation of state (see Figure 4) models the pressure rise as the mixture mass density increases. It is assumed that the pressure build up is slow at small mixture density (i.e. zero slope  $\partial p/\partial \rho$ ). Alternatively, the real compressibility of the resin can be used for the calculation of the slope at injection pressure. A more realistic, reduced resin compressibility will result in much smaller time steps.

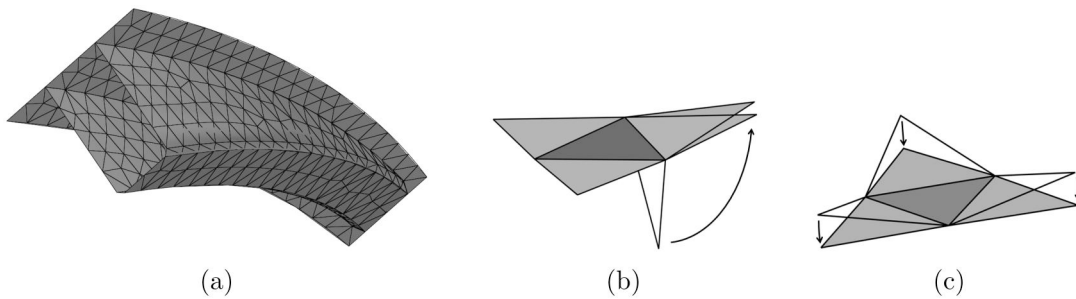
## 2.2. Implementation

In order to numerically solve the flow model described by Equation (2) and (3), respectively, with the equation of state from Figure 4, the computational domain (time and space) is discretized. Discretization of the governing equations on an

<sup>1</sup>In this paper, mathematical symbols and units are noted according to the ISO 80000 standard.



**Figure 4.** The equation of state models the pressure build-up for the air-resin mixture. A quadratic polynomial fit through the two mass density/pressure pairs for air at initial cavity pressure and resin at injection pressure is used. The shown curve is for air density  $1.225 \text{ kg/m}^3$  at  $100,000 \text{ Pa}$  and resin density  $960 \text{ kg/m}^3$  at  $191,000 \text{ Pa}$  which is used in the validation cases.



**Figure 5.** (a) Annulus filler mesh with T-sections and curved regions. (b) and (c) Locally flat mesh.

unstructured mesh follows the ideas presented in Versteeg and Malalasekera [18].

The temporal domain (i.e. the simulation time) is split into a finite number of time steps where values for the physical quantities on the spatial domain are calculated in a time-marching manner. An explicit method with adaptive time stepping is used.

The spatial domain is defined on the cavity's mid-surface and described by a shell mesh in a format similar to the NASTRAN bulk data format, see NASTRAN [19]. The mid-surface model can be curved and cells can have edges where more than two cells are connected to each other such as for handling T-junctions. A general mesh is illustrated in Figure 5(a). The injection gate is part of the computational domain and injection pressure and resin fraction is assigned to the cells belonging to this region. Vents can be specified optionally. If no vents are specified (similar to software myRTM) the simulated filling is not influenced as long as the flow front does not reach the fictitious vent position. Initial and boundary conditions are assigned. In all cells except in those belonging to the injection gates and vents the initial velocity is zero, the resin fraction is zero and the pressure is equal to the initial cavity pressure which is an input parameter. In all

cells belonging to the injection gates the fluid pressure is equal to injection pressure and resin fraction is equal to one. In all cells belonging to the optional vents the fluid pressure is equal to initial cavity pressure and resin fraction is zero. The values in the cells for the injection gates and vents are not changed during a simulation. If injection pressure changes during the filling process or injection gates are added or removed, the simulation can be continued from the previous filling state with changed boundary conditions.

For discretizing the equations on the shell mesh of the part's mid-surface it is assumed that the geometry is locally flat. The neighboring cells are rotated about the common edges to lie in the plane of the considered cell (see Figure 5(b,c)) and the velocity vectors of the neighboring cells are transformed into the same coordinate system.

The solver section (which is the relevant section if a different physical model is implemented) consists of a for-loop over all interior and wall cells inside a while-loop for the time evolution. For every interior and wall cell, the following steps are performed:

1. The pressure and filling fraction gradients are evaluated.

2. The time evolution for new mass density,  $x$ - and  $y$ -velocities with the discretized continuum,  $x$ - and  $y$ -momentum equations in the cell coordinate systems is performed.
3. The pressure is evaluated according to the equation of state.
4. The filling fraction is evaluated.

Boundary conditions need not be updated since in the inlet and outlet cells pressure and mass density remain unchanged and the velocity values at the cell boundaries are evaluated in the numerical flux evaluation.

The discretized versions of the governing Equations (2) and (3), respectively, give the numerical scheme

$$\rho_i^{n+1} = \left( \varepsilon_i \rho_i^n - \frac{\Delta t}{V_i} \sum_j \mathbf{n}_{i,j} \cdot \underbrace{\frac{1}{2} (\rho_i^n + \rho_j^n)}_{(\rho \mathbf{u})_{i,j}^n} \frac{1}{2} (\mathbf{u}_i^n + \mathbf{u}_j^n) A_{i,j} \right) / \varepsilon_i \quad (4)$$

$$\mathbf{u}_i^{n+1} = \left( \rho_i^n \mathbf{u}_i^n - \frac{\Delta t}{V_i} \sum_j \mathbf{n}_{i,j} \cdot (\rho \mathbf{u})_{i,j}^n \mathbf{u}_{i,j}^n A_{i,j} - \left( \frac{\partial p}{\partial x} \right)_i \Delta t \right) / (\rho_i^{n+1} + (\mu_i/k_1) \Delta t) \quad (5)$$

$$\mathbf{v}_i^{n+1} = \left( \rho_i^n \mathbf{v}_i^n - \frac{\Delta t}{V_i} \sum_j \mathbf{n}_{i,j} \cdot (\rho \mathbf{u})_{i,j}^n \mathbf{v}_{i,j}^n A_{i,j} - \left( \frac{\partial p}{\partial y} \right)_i \Delta t \right) / (\rho_i^{n+1} + (\mu_i/k_2) \Delta t) \quad (6)$$

for cell  $i$  where  $j \in S_i$  represents the index of a cell neighbor and the combined index  $i, j$  represents the index of the cell interface between cell  $i$  and cell  $j$ . The Darcy source term is treated implicitly. Otherwise a much smaller time step than given by the CFL condition introduced in Courant et al. [20] would be required. The physical quantities  $u_{i,j}^n$  and  $v_{i,j}^n$  at the cell interface are evaluated using first-order upwinding. For flow from cell  $i$  to cell  $j$ , first order upwinding is e.g.  $u_{i,j}^n = u_i^n$ . The  $j$ -sum runs over all neighboring cells. When the neighboring cell is a pressure inlet cell, the velocity in the factor without upwinding in the convective term is evaluated according to Darcy's law.

Code snippets shown in the appendix illustrate how the different flow physics models are implemented and how an additional equation can be added.

### 3. Results and discussion

#### 3.1. Running a simulation

The simulation is started with a set of process parameters and a set of material properties, see the README file on the GitHub repository <https://github.com/obertscheiderfhnw/LCMSim>, either from

the command line or with the GUI. Figure 1 shows the GUI with the relevant input parameters. Contour plots for the filling fraction and the pressure at different time instances are generated during post-processing.

#### 3.2. Simulation overview

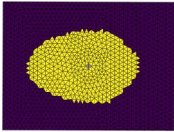
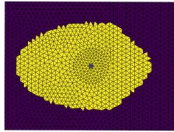
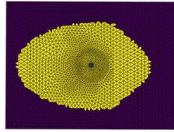
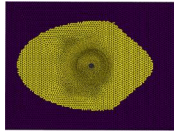
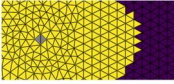
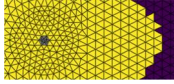
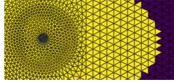
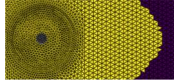
The RTMsim module was developed for fast filling simulations and implements a simple fluid flow model, whereas LCMsim implements a more complex flow physics model which can be used for the simulation of all LCM processes. The RTMsim module was verified with three RTM test cases as reported in Obertscheider and Fauster [11]. The new model implementation provided in LCMsim was initially tested against these test cases. Table 1 shows the results of a radial and a linear flow experiment run with RTMsim and LCMsim. The radial flow is a typical RTM filling test case from Isoldi et al. [12] and gives well comparable contour plots of the filling fraction after 200 s. Significant differences between LCMsim and the RTMsim occur in rectilinear flow with high viscosity and small permeability as shown in the linear flow experiment. This is a result of modeling the resin as compressible in RTMsim, whereas the resin in LCMsim is only slightly compressible. If the resin is compressible, the pressure build-up in the cells requires more material and this slows down the flow. The slower the flow, the stronger is this effect. Slower flow is a result of higher viscosity and lower permeability.

Table 2 shows the validity of the used mesh (mesh 3) for the first validation case. The flow front position after 120 s filling time shows numerical convergence. The flow front position is comparable for meshes 2, 3 and 4. This is quantified by the calculation of the filled area. For the second validation case a finer mesh away from the pressure injection gate is used because the flow around a patch is investigated there.

The following general conclusions for simulations with LCMsim can be drawn from this mesh refinement study. If the mesh at the injection gates is too coarse (see detail view of mesh 1) the flow front propagation speed is influenced. In general, the mesh size must be finer around the pressure injection gates and can increase with the distance from the gates (see mesh 2). Depending on the purpose of a filling simulation, a coarser or a finer mesh must be chosen. If the actual filling time is important a fine mesh must be chosen and a numerical convergence study similar to the one shown in Table 2 must be performed. For simulations during mold design (e.g. to find locations for injection gates and vents) a coarser mesh (for example mesh



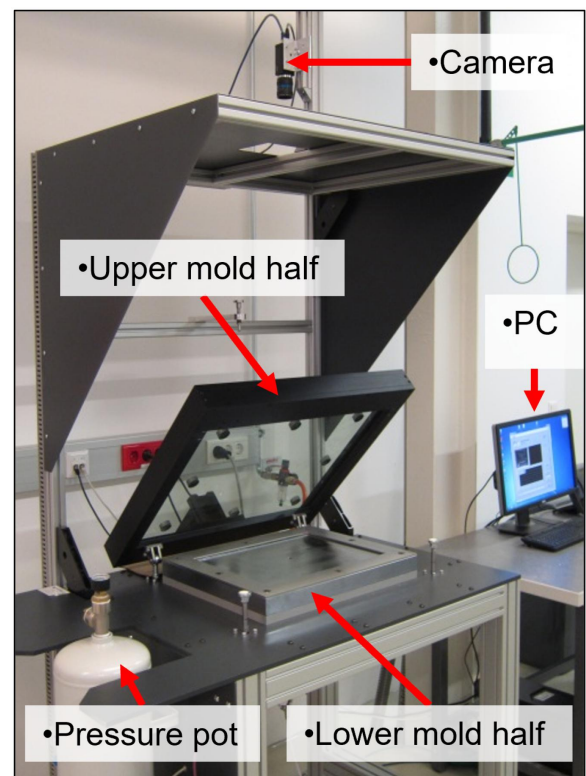
**Table 2.** Results of a mesh refinement study for validation case 1 in terms of well comparable contour plots of the filling fraction after 120 s (yellow/blue is filled/unfilled).

	Mesh 1 Coarse mesh	Mesh 2 Finer at inlet, coarser outside	Mesh 3 Medium mesh	Mesh 4 Fine mesh
Filling pattern (whole domain)				
Filling pattern (detail)				
Number of cells	2516	2668	5150	12,296
CPU time	3.6%	8.2%	100%	323%
Filled area	0.282	0.401	0.405	0.411

Notes: The pictures in the first line show the whole domain, the pictures in the second line show the mesh around the pressure injection gate. The cell size is reduced by a factor 1.5 between meshes 2 and 3 and between meshes 3 and 4. Furthermore, the number of cells, the computational time (relative to the reference of mesh 3) and the filled area as fraction of filled by total domain area are given.

2) can be chosen to keep the simulation time short. Furthermore, it can be seen that the required computational time does not only depend on the number of cells, also on the size of the smallest cells. This is a result of the adaptive time step calculated with the CFL condition [20]. According to the CFL condition, the maximum time step is proportional to the cell size and inversely proportional to the flow speed in the cell. For every cell the maximum time step must be calculated and the minimum over all cells is then taken as next time step. Summing up, the smaller the cell size for the same flow speed (i.e. for the same considered case), the smaller the time steps for the simulation.

The need for fine resolution around the injection gate comes from the implemented non-linear equation of state shown in Figure 4. This is illustrated by the following thought experiment: Consider a linear flow through a sequence of equidistant cells, the first belonging to the inlet gate, all others belonging to the preform. At the beginning of the filling the pressure gradient between the inlet and the first cell inside the preform remains unchanged until the cell is almost completely filled. Consequently there is no flow out of the preform cell into neighboring preform cells (which are still at initial cavity pressure) for some time. Then, the first interior cell of the preform is split into several small cells and the cell belonging to the inlet is also replaced by a small cell. At the beginning of the filling the pressure gradient between the small inlet cell and the first small cell inside the preform is higher compared to the first setup with large cells resulting from the smaller spatial distance. If the flow reaches the last small cell the pressure gradient is equal to the pressure gradient between the two large cells from the first setup. Since the fluid velocity is proportional to the pressure gradient according to Darcy's law, this thought experiment shows that the filling for finer cells is faster. If the

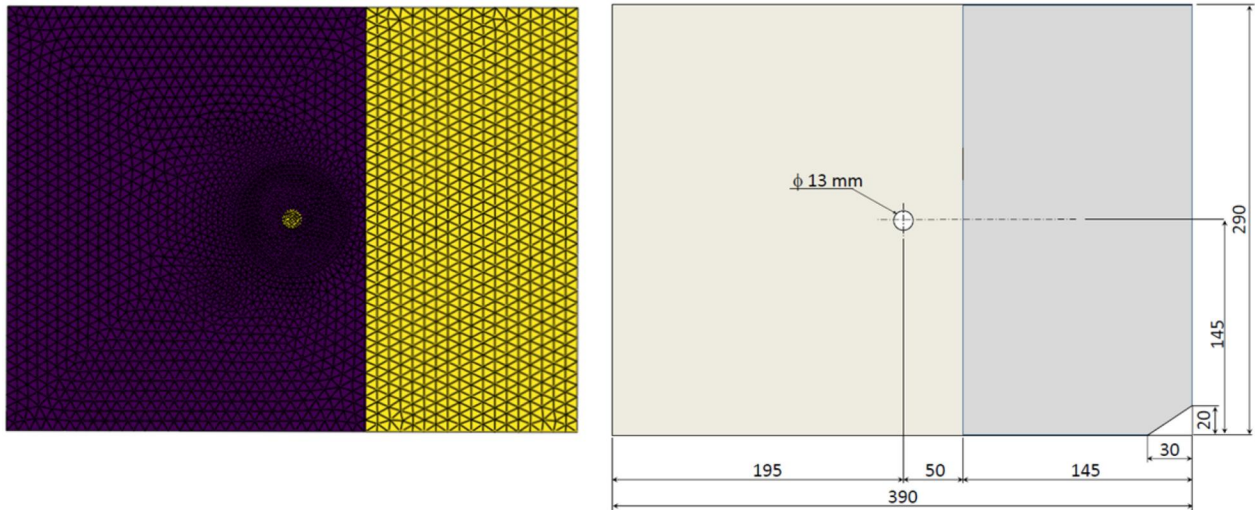
**Figure 6.** Optical permeameter for radial flow experiments.

equidistant cell size is below a certain limit (which must be shown by a numerical convergence study) the filling is independent of the cell size.

### 3.3. Flow experiments

The verification and validation benchmark presented in this study consists of two real-world flow experiments. All test cases are derived from an optical permeameter available at Montanuniversität Leoben, see Grössing et al. [21] and Fauster et al. [22]. For one of the test cases also a verification with a different software tool was performed.

Figure 6 shows the test rig which is used to run radial flow experiments. Directly on top of the



**Figure 7.** Mesh with highlighted central injection gate and section with different properties than the main preform (left) and dimensions (right) for the first validation test case.

working table the metal mold half is mounted. The upper mold half is made of glass and is framed with metal profiles. The actual mold cavity is specified through the cavity frame showing an inner dimension of  $300 \times 400$  mm. After placing the stack of reinforcing materials of dimension  $290 \times 390$  mm inside the cavity, the mold is closed by flapping the glass plate into a horizontal position. The radial flow experiment is then executed by injecting the test fluid into the cavity through a central injection point in the metal bottom mold half. Due to the thickness of the glass plate and the clamping frame used to tighten the glass plate, a maximum injection pressure of 4 bar can be applied. A camera system, mounted 1 m above the mold, is used to acquire an image sequence during the radial flow experiment. If the experiment is used for permeability characterization, the sequence images are evaluated by means of a digital image processing algorithm specifically developed for this application. The flow front is described by an elliptical geometry model. Thus, the radially advancing flow front is finally obtained in terms of the major and minor axes length characteristics.

### 3.4. Validation with real-world flow experiments

The preforms for the validation experiments consist of layers of woven fabric Hexcel 1202. The preform is positioned such that the first principal flow direction is aligned with the long edge of the preform. The fabric was characterized beforehand for different numbers of layers in the described optical permeameter. Also mass density and dynamic viscosity of the used test fluid at lab temperature were determined before the experiments were conducted.

Initially the cavity is filled with air and plant oil is injected as test fluid. The fluid properties are air density  $1.205 \text{ kg/m}^3$  at initial cavity pressure and oil density

**Table 3.** Preform properties (number of layers  $N$ , porosity  $\varepsilon$ , orthotropic permeability  $k_1$  and  $k_2$ ) for the first validation case (variation of preform properties).

	$N$ [-]	$\varepsilon$ [-]	$k_1$ [ $10^{-12} \text{ m}^2$ ]	$k_2$ [ $10^{-12} \text{ m}^2$ ]
Main section	11	0.604	163	50.3
Manipulated section	14	0.468	28.6	3.4

$960 \text{ kg/m}^3$  at injection pressure. The measured dynamic viscosity of the oil is 0.071 Pas in the first test case and 0.062 Pas in the second test case. The differences result from temperature variations in the lab.

#### 3.4.1. Variation of preform properties

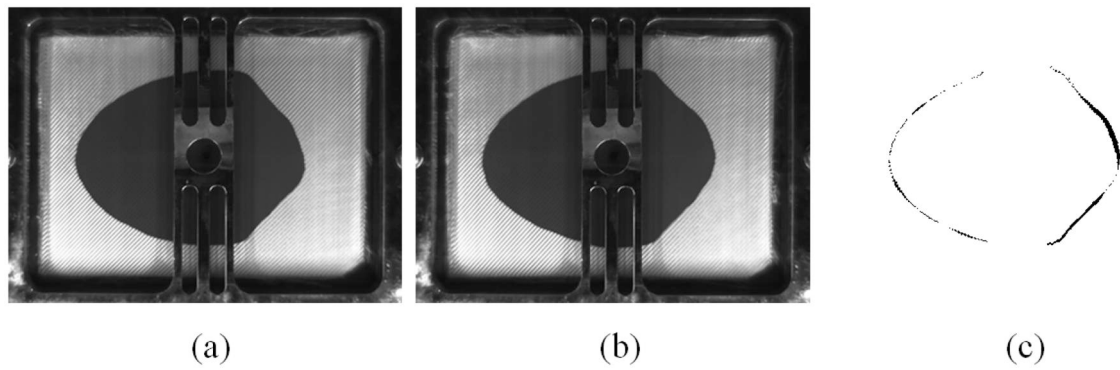
The setup is a model of the cavity in the optical permeameter (see Figure 6), i.e. a flat plate with  $390 \times 290 \times 3.14 \text{ mm}^3$ . The main preform covers the leftmost section (245 mm wide) with the central injection port with 13 mm diameter. The remaining section (145 mm wide) is manipulated to show lower permeability and porosity. Figure 7 shows the top view of the preform model (mesh) with the central injection port and the reinforced section.

The preform consists of 11 layers of fabric Hexcel 1202 in the main section and 14 layers in the section in the right end. The fabric properties are given in Table 3.

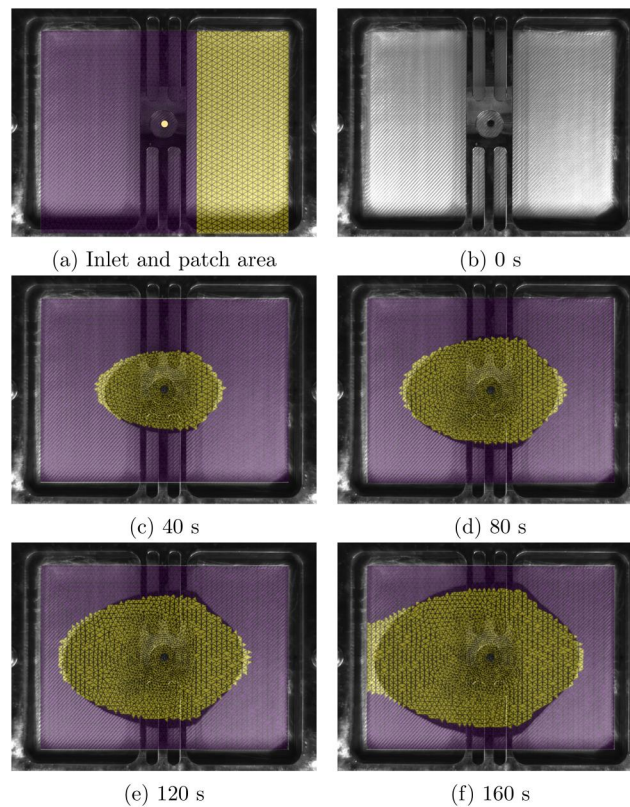
Initial cavity pressure is ambient pressure and injection is with 100,000 Pa over-pressure. Due to pressure losses in the feed line the measured pressure difference between injection gate and cavity is 91,000 Pa.

Different propagation speeds occur in the different preform zones. This can be inspected if the flow front is smooth. Consequently, for this case a fine mesh (5150 cells) was chosen.

For the validation of the simulation the flow front propagation is compared with that of the flow experiments. Three experiments were conducted and all three showed a similar filling behaviour. A



**Figure 8.** Filling after 120 s for two different experiments with the same process, permeability and fluid parameters is shown in subplots (a) and (b). The difference in the filling is shown in (c).



**Figure 9.** Inlet and patch areas (a), picture from the start of the flow experiment (b) and overlaid filling from simulation and experiment for different time instances (c–f) for the first validation case (variation of preform properties). The flow front propagation slows down in the zone of reduced porosity and in-plane permeability. Simulation and experiment show very good agreement.

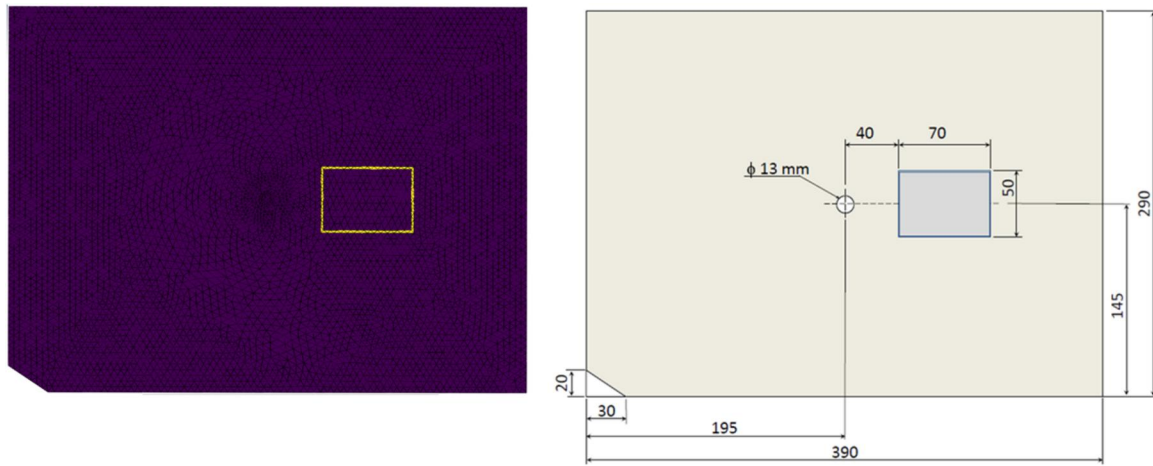
visual comparison of the filling after 120 s filling time for two different experiments is used to show the reproducibility of the experiments. [Figure 8](#) shows the difference in filling after 120 s for two experiments. The relative error in the filling areas is 2%. This difference must be considered in the interpretation where flow fronts from experiment and simulation are compared.

[Figure 9\(c–f\)](#) compares the filling of one representative flow experiment and simulation. Pictures from similar time instances are transparently overlaid for visual comparison. Due to the reinforcement with lower permeability and lower porosity in the rightmost section the flow front propagation slows down there. In general, if propagation speed and

filling pattern are compared, experiment and simulation show good agreement. The simulated filling leads the experimental filling in horizontal direction and trails in vertical direction. This can be explained with small discrepancies between the experimentally determined volume-averaged in-plane permeability values from those of the actual preform.

Race tracking was expected in a vertical channel at the boundary between the two preform zones but was not observed in the experiments.<sup>2</sup> Since race-

<sup>2</sup>If race-tracking is present it is included in the simulation with a small zone with increased permeability and porosity. Typically, the permeability in the race tracking zone is set one or two orders of magnitude higher than the permeability in the main preform and the porosity is close to 1.



**Figure 10.** Mesh (left) and dimensions (right) for the second validation test case. A 2 mm wide gap was realized around the edges of the patch, where race-tracking is enforced. This zone is highlighted in the mesh.

**Table 4.** Preform properties (number of layers  $N$ , porosity  $\varepsilon$ , orthotropic permeability  $k_1$  and  $k_2$ ) for the second validation case (flow manipulation by local variation of preform properties).

	$N$ [-]	$\varepsilon$ [-]	$k_1$ [ $10^{-12}$ m <sup>2</sup> ]	$k_2$ [ $10^{-12}$ m <sup>2</sup> ]
Main section	11	0.583	96.6	38.6
Patch	16	0.393	1.2	15.3
Race-tracking zone	0	0.960	1500	1500

Note: The permeability values in the main preform were slightly increased compared to the values from the preform characterization to match the flow front propagation in the main preform.

tracking is omnipresent in RTM, an additional experiment with enforced race-tracking was used for validation.

### 3.4.2. Flow manipulation by local variation of preform properties

The setup is again a model of the cavity in the optical permeameter, but with cavity thickness 3.0 mm. A 2.0 mm wide gap was realized around the edges of the patch, where race-tracking is enforced. The setup is illustrated in Figure 10.

The preform consists of 11 layers of fabric Hexcel 1202 in the main section and 16 layers in the patch. The fabric properties are given in Table 4.

Initial cavity pressure is ambient pressure and injection is with 100,000 Pa over-pressure. Due to pressure losses in the feed line the measured pressure difference between injection gate and cavity is 91,000 Pa.

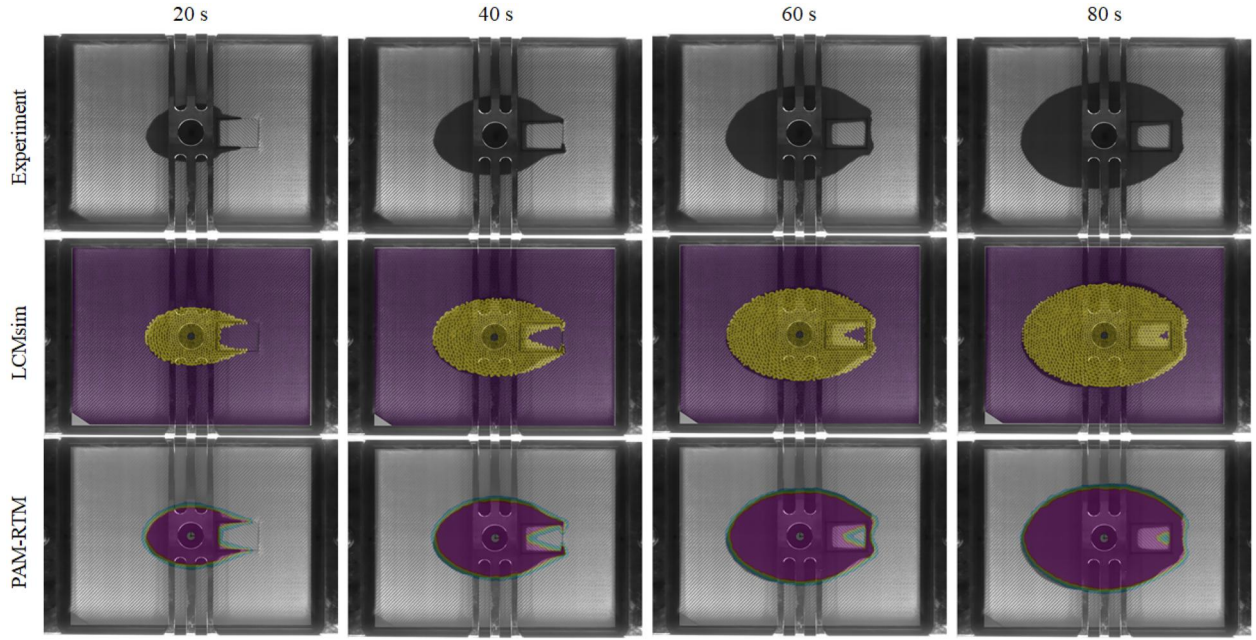
The expected flow front propagation is an ellipse in the main preform and fast flow through the race-tracking channel to create an entrapment in the patch. A similar mesh as for the first test case was chosen (5308 cells).

One of the best-known software tools for predicting the flow front advancement in RTM is PAM-RTM, see PAM-RTM [23]. Therefore, the predicted flow front advancements of LCMsim are compared to the results of PAM-RTM using the same shell

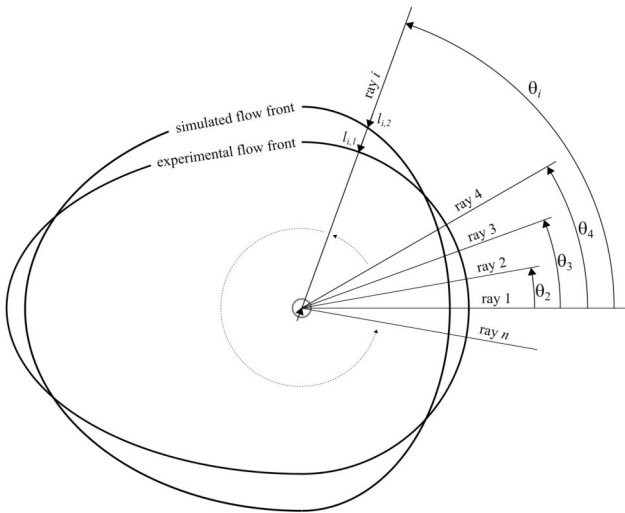
mesh as well as identical material properties and process parameters.

The simulated flow front from both simulation tools are compared with that of the flow experiment. Again, three experiments were conducted and all three showed a similar filling behavior. Simulation results are compared with pictures of one representative flow experiment. Pictures from similar time instances are transparently overlaid for visual comparison in Figure 11. Both simulation tools render very similar filling patterns for the shown time instances, and simulations and experiment are in very good agreement. The overall flow front propagation is elliptical. As soon as the race-tracking channel is completely filled, air entrapment is formed in the patch area. However, the filling of the patch is much faster in both simulations compared with the experiment. From a mold engineering point of view it is sufficient to see that an entrapment occurs for a certain configuration and then the injection strategy (i.e. the location of gates and vents) must be changed. From a research point of view this phenomenon must be studied in more detail.

The two simulation tools perform well for applications with flow through preforms with different permeability and porosity values. This is demonstrated for example in validation case 1 of this study and in Grössing et al. [21]. Only in case of race-tracking (which is modeled by patches with much higher permeability and porosity), the filling from the experiment is different than the filling from both simulations tools. If the preform values are adjusted to match the flow in race-tracking channel direction, there is too much cross flow in both simulation tools. Since flow through porous media is described by Darcy's law in both simulations tools, the results of validation case 2 suggest that there is an additional flow resistance between regions with significantly different porosity and/or permeability values. Summing up, both filling



**Figure 11.** Flow front from experiment (first row), overlaid filling from LCMsim simulation and experiment (second row) and overlaid filling from PAM-RTM simulation and experiment (third row) for different time instances for the second validation case (flow manipulation by local variation of preform properties).



**Figure 12.** Sketch explaining the used variables for calculating the relative error  $e_i$  along ray  $i$ .

simulations describe the flow through porous media based on Darcy's law, which is used because of its simplicity. However, it is only applicable under certain conditions (isothermal, steady and laminar flow of a Newtonian fluid through a homogeneous-porous medium). These assumptions are apparently violated in this test case at the transition of the preform to the patch (with significant differences in fiber volume content, porosity and permeability) and consequently simulation and experiment are no longer in good agreement there.

### 3.5. Discussion

In order to compare the simulated and experimentally determined flow fronts quantitatively, the filling after

120 s for the first test case and the filling after 60 s for the second test case are analyzed in more detail.

The flow front position is characterized by  $n$  equally distributed rays starting at the inlet center and ending at the intersection with the flow front. The relative error along ray  $i$  in direction  $\theta_i$  is calculated by:

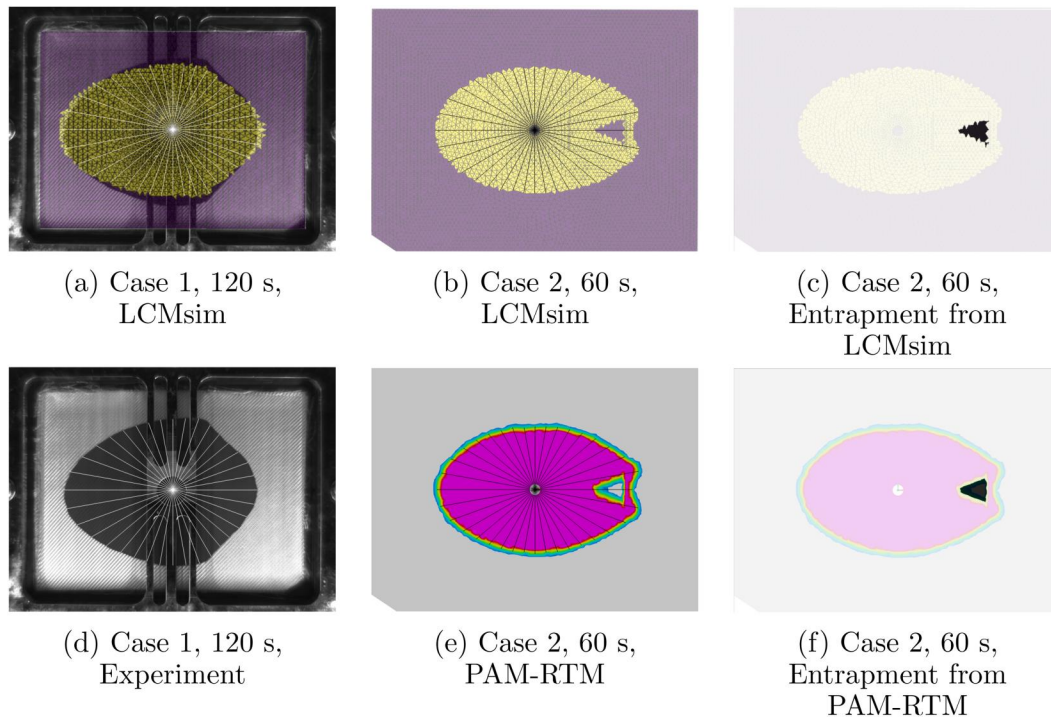
$$e_i = \frac{l_{i,1} - l_{i,2}}{l_{i,2}}, \quad (7)$$

where  $l_{i,1}$  and  $l_{i,2}$  denote the length of ray  $i$  with respect to the first and second type of flow front, respectively. Figure 12 shows how the lengths along ray  $i$  are measured.

In the first test case, the flow front predicted by LCMsim is compared with the flow experiment. In the second test case, the flow front predicted by LCMsim is compared with that predicted by PAM-RTM. In both cases,  $n=37$  was chosen, i.e. the flow fronts are sampled with a spacing of  $10^\circ$ . The root-mean-square (RMS) error in the flow front determined by the two different methods is then:

$$e = \sqrt{\frac{1}{n} \sum_{i=1}^n e_i^2} \quad (8)$$

Since in case 1 the simulated filling leads the experimental filling in horizontal direction and trails in vertical direction, the flow fronts from experiment and LCMsim have a RMS error of 7%. The flow fronts from case 2 simulated with PAM-RTM and LCMsim exhibit a deviation of 3% in terms of the RMS error. The entrapment in case 2 is characterized by the enclosed area. Black/white pictures of equal size where the black area covers the entrapment are



**Figure 13.** The flow front position is characterized by 36 equally distributed rays starting at the inlet center and ending at the intersection with the flow front. (a) and (d) describe the flow fronts from case 1 after 120 s simulated with LCMsim and measured in the experiment. (b) and (e) describe the flow fronts from case 2 after 60 s simulated with LCMsim and PAM-RTM. The entrapment is characterized by the enclosed area. (c) and (f) describe the entrapment in the patch from case 2 after 60 s simulated with LCMsim and PAM-RTM.

analyzed. The numbers of black pixels are used to calculate a relative error in the entrapment area. The entrapment area is nearly identical (0.5% difference).

Figure 13 shows the analyzed pictures. LCMsim and experiment have a sharp boundary for the flow front. For PAM-RTM the position of the flow front was taken between light and dark green in the contour plot which corresponds to a filling of 50%.

It has been confirmed that the LCMsim software is appropriate for performing RTM filling simulations under real-world conditions. Case 1 shows that the predicted flow front advancement simulated with LCMsim agrees with those from the experiment. Case 2 shows that both simulation tools, LCMsim and the commercially available PAM-RTM, render similar results, although different modeling equations are solved.

#### 4. Conclusion and outlook

The work at hand presents a benchmark of the software LCMsim with a new flow physics model for filling simulations in RTM and how this model was implemented in the programming language Julia. The Julia module can be downloaded from a GitHub repository <https://github.com/obertscheiderfhnw/LCMsim>. In general, the software is easy-to-use for the tool engineer and gives maximum flexibility for scientific investigations. The new software tool can be customized for non-standard filling applications. For

example, by adding new or customizing existing equations or boundary conditions. Furthermore, since the code is open source, code snippets can be extracted and integrated in separate codes, all intermediate results are available by creating new interfaces, and the simulation parameters can be modified during a run (for example, changing inlet pressure levels as function of flow front shape).

The software LCMsim implements a new fluid mixture model (continuity equation for the mixture density, momentum equations in a local cell coordinate system, equation of state which models the pressure build-up as function of mixture density, filling fraction determined from mixture density). The implementation results in a robust algorithm without pressure-velocity coupling during one time step, only takes a shell mesh as input and can be extended easily. In order to solve the governing equations on a shell mesh, the finite area method is generalized for curved surfaces with junctions.

Verification of the presented filling model was previously done with radial flow cases and by comparison with filling results from other software tools [11]. The study at hand successfully validates the new software with real world experiments under real process conditions for RTM. The software can deal with patches with different in-plane permeability and porosity levels and with race-tracking. Furthermore, it has been shown that LCMsim and PAM-RTM yield very similar results.

Based on the already presented verification and validation cases, the next step will be an extension of the software to cover particularities of the vacuum-assisted resin infusion (VARI) process. There, transverse flow from the flow distribution media into the fibrous preform will be implemented on the shell mesh. If needed, cells with two layers representing the flow distribution medium and the preform will be provided. Moreover, models for resin temperature and viscosity as well as cure kinetics will be added to the software. Apart from improving the physical model, the comparison in Section 1.3 and Table 1 shows the need of including a mesher and improving the computational performance.

### Acknowledgments

The authors thank Oliver Rausch-Schott for running the flow experiments on the optical permeameter test rig at Montanuniversität Leoben.

### Disclosure statement

No potential conflict of interest was reported by the author(s).

### Data availability statement

The novel software tool LCMsim is openly available on GitHub at <https://github.com/obertscheiderfhwn/lcmsim>. Installation instructions and instructions on how to run a simulation are given in a README file on the repository. The input files `input_case4a.txt` and `input_case4b.txt` can be used to derive the data presented in the two test cases.

### References

- Strong AB. Fundamentals of composites manufacturing, second edition: materials, methods and applications. Dearborn, Michigan: Society of Manufacturing Engineers; 2008.
- Niu MCY. Composite airframe structures: practical design information and data. Hong Kong: Conmilit Press Ltd.; 1992.
- May D, Aktas A, Advani SG, et al. In-plane permeability characterization of engineering textiles based on radial flow experiments: a benchmark exercise. *Compos Part A Appl Sci Manuf.* 2019;121:100–114. doi: [10.1016/j.compositesa.2019.03.006](https://doi.org/10.1016/j.compositesa.2019.03.006).
- Neitzel N, Mitschang P, Breuer U. Handbuch verbundwerkstoffe, 2., aktualisierte und erweiterte auflage. München: Carl Hanser Verlag; 2014.
- Bickerton S, Abdullah MZ. Modeling and evaluation of the filling stage of injection/compression moulding. *Compos Sci Technol.* 2003;63(10):1359–1375. doi: [10.1016/S0266-3538\(03\)00022-8](https://doi.org/10.1016/S0266-3538(03)00022-8).
- Stieber S, Heber L, Obertscheider C, et al. Control of RTM processes via deep reinforcement learning. [accessed 2023 Nov 16]. Available from: <https://www.researchgate.net/publication/367191417>
- Seuffert J, Kärger L, Henning F. Simulating mold filling in compression resin transfer molding (CRTM)

- using a three-dimensional finite-volume formulation. *J Compos Sci.* 2018;2(2):23. doi: [10.3390/jcs2020023](https://doi.org/10.3390/jcs2020023).
- Sebastian RG, Obertscheider C, Fauster E, et al. Equation for modelling energy transfers in multi-phase flows through porous media, optimised for liquid composite moulding processes. *Int J Heat Mass Transf.* 2021;181:121856. doi: [10.1016/j.ijheat-masstransfer.2021.121856](https://doi.org/10.1016/j.ijheat-masstransfer.2021.121856).
- Barandun G, Henne M, Giger E, et al. MyRTM: an approach for the simulation of resin transfer moulding (RTM) processes based on cellular automata. In: Eberhardsteiner J, et al., editors, ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering; 2012 Sep 10–14; Vienna; 2012.
- myRTM. myRTM - Software for simulating the RTM process. [accessed 2023 Nov 16]. Available from: <https://www.myrtm.ch>
- Obertscheider C, Fauster E. RTMsim - a Julia module for filling simulations in resin transfer moulding with the finite area method. *JOSS.* 2023;8(84):4763. doi: [10.21105/joss.04763](https://doi.org/10.21105/joss.04763).
- Isoldi LA, Oliveira CP, Rocha LAO, et al. Three-dimensional numerical modeling of RTM and LRTM processes. *J Braz Soc Mech Sci Eng.* 2012;34(2):105–111. doi: [10.1590/S1678-58782012000200001](https://doi.org/10.1590/S1678-58782012000200001).
- Simacek P, Advani SG. Desirable features in mold filling simulations for liquid composite molding processes. *Polym Compos.* 2004;25(4):355–367. doi: [10.1002/pc.20029](https://doi.org/10.1002/pc.20029).
- Sirtautas J, Pickett AK, George A. Materials characterisation and analysis for flow simulation of liquid resin infusion. *Appl Compos Mater.* 2015;22(3):323–341. doi: [10.1007/s10443-014-9411-6](https://doi.org/10.1007/s10443-014-9411-6).
- Tucker C. Governing equations for flow and heat transfer in stationary fiber beds. In: Advani SG, editor. *Flow and rheology in polymer composites manufacturing.* Amsterdam: Elsevier; 1994. p. 257–323.
- Nield DA, Bejan A. 2006. *Convection in porous media.* New York: Springer.
- Advani SG. 1994. *Flow and rheology in polymer composites manufacturing.* Amsterdam: Elsevier.
- Versteeg HK, Malalasekera W. 2007. *An introduction to computational fluid dynamics: the finite volume method.* Harlow, England: Pearson Education Limited.
- NASTRAN. NASA STRuctural ANalysis (NASTRAN). [accessed 2023 Nov 16]. Available from: <https://software.nasa.gov/software/LAR-16804-GS>
- Courant R, Friedrichs K, Lewy H. Über die partiellen differenzengleichungen der mathematischen physik. *Math Ann.* 1928;100(1):32–74. doi: [10.1007/BF01448839](https://doi.org/10.1007/BF01448839).
- Grössing H, Stadlmajer N, Fauster E, et al. Flow front advancement during composite processing: predictions from numerical filling simulation tools in comparison with real-world experiments. *Polym Compos.* 2016;37(9):2782–2793. doi: [10.1002/pc.23474](https://doi.org/10.1002/pc.23474).
- Fauster E, Berg DC, Abliz D, et al. Image processing and data evaluation algorithms for reproducible optical in-plane permeability characterization by radial flow experiments. *J Compos Mater.* 2019;53(1):45–63. doi: [10.1177/0021998318780209](https://doi.org/10.1177/0021998318780209).
- PAM-RTM. PAM-RTM - liquid composites molding and curing simulation. [accessed 2023 Nov 16]. Available from: <https://www.esi.com.au/software/pamrtm/>

## Appendix A: code modifications

The code snippet in A1 shows the implementation of the numerical scheme as described by Equations (4), (5) and (6), respectively. During the time evolution (line 8) for every internal cell (index ind in line 12) the main steps are:

- The pressure gradient function (line 15) is called to evaluate  $(\frac{\partial p}{\partial x})_i$  and  $(\frac{\partial p}{\partial y})_i$ .
- The numerical flow (e.g.  $\mathbf{n}_{i,j} \cdot (\rho \mathbf{u})_{i,j}^n u_{i,j}^n A_{i,j}$  for the  $x$ -velocity) at the boundaries of the considered cell  $i$  to all neighboring cells  $j$  (lines 26–67) is evaluated.
- The mass density and velocity components at the new time step (lines 69–78) are calculated according to the numerical schemes. Depending on the flow physics model, a different continuity equation is used. The original flow physics model corresponds to the blocks with `if i_model==1` and the current flow physics model to the blocks with `elseif i_model==2`. The factors for porosity, permeability, ... are equal to 1 and prepared the code for VARI with `elseif i_model==3`
- Depending on the flow physics model, the filling fraction and the fluid pressure are calculated (lines 80–105).
- In the evaluation of the cell boundary flow, first-order upwinding is used (e.g. lines 132–137 for the calculation of  $u_{i,j}^n$  in the  $x$ -velocity flow  $\mathbf{n}_{i,j} \cdot (\rho \mathbf{u})_{i,j}^n u_{i,j}^n A_{i,j}$ ) depending on flow out of (with  $\geq$ ) or into the considered cell. The contribution `n_dot_rho` at boundaries to pressure inlet and outlet cells is calculated differently (lines 129 and 171) with the Darcy velocity (line 56) or the outflow velocity (line 58). This difference is the reason for two different functions. Different (higher-order) methods for the evaluation of the cell boundary flow can be added with the function argument `i_method`.

If an additional equation, e.g. for the degree-of-cure  $\alpha$ , is added, the following steps must be performed:

- Additional parameters for the initial and inlet degree of cure (`alpha_init = 0` and `alpha_a`) and for the cure kinetics model (e.g. `k` and `n` for the  $n$ -th order reaction model  $S_x = k(1 - \alpha)^n$ ) are defined. Additional arrays `alpha_new` and `alpha_old` must be initialized similar to `rho_new` and `rho_old`. Instead of hard-coded parameters, other sections in the code can be modified to take these as input parameter. Directly before the time evolution section begins, the additional arrays and parameters are prepared:

```

1         #Assign and prepare physical parameters
2         alpha_a=0;
3         alpha_init=0;
4         k=0;
5         n=1;
6
7         # Array initialization
8         alpha_old=Vector{Float64}(undef, N);
9         alpha_new=Vector{Float64}(undef, N);
10        for ind in 1:N;
11            alpha_old[ind]=alpha_init;
12        end
13        for ind in 1:N
14            alpha_new[ind]=-9e9;
15        end
16
17        # Boundary conditions
18        for ind in 1:N;
19            if celltype[ind]==-1; #pressure boundary
20                alpha_old[ind]=alpha_a;
21            elseif celltype[ind]==-2; #pressure outlet
22                alpha_old[ind]=alpha_init;
23            end
24        end
    
```

- If the degree-of-cure equation is added to the new flow physics model, replace all occurrences of `if i_model==2` by `if i_model==2 || i_model==4` since the new flow physics model with degree-of-cure is identified with `i_model = 4`.
- After the equation of state section (after line 105) add a section inside an `if i_model==4` block, where the degree-of-cure equation

$$\frac{\partial \varepsilon \alpha}{\partial t} + \mathbf{u} \cdot \nabla \alpha = S_x \quad (\text{A1})$$

in a conservative form

$$\frac{\partial \varepsilon \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) - \alpha (\nabla \cdot \mathbf{u}) = S_x \quad (\text{A2})$$

is solved. Therefore, the code snippet of the filling fraction equation of `i_model = 1` which is also a transport equation is copied (lines 81–83). `Gamma` is replaced by `alpha` and a source function is added:

```

1         S_alpha=k*(1-alpha_old[ind])^n;
2         alpha_new[ind]=((cellporosity[ind]*porosity_factor[ind])*alpha_old[ind]-deltat*(
3             F_alpha_num-alpha_old[ind]*F_alpha_num1+S_alpha*deltat)/(cellvolume[ind]*
4             volume_factor[ind]))/(cellporosity[ind]*porosity_factor[ind]);
5         alpha_new[ind]=min(1,alpha_new[ind]);
6         alpha_new[ind]=max(0,alpha_new[ind]);
    
```



- Also in the functions for the calculation of the flow at the cell boundaries, the sections for gamma (lines 145–152 and 186–193) are copied and all gamma occurrences are replaced by alpha. In addition, alpha\_P and alpha\_A must be added to the vector arguments vars\_P and vars\_A. alpha\_P = gamma\_old[i\_P]; and alpha\_A = gamma\_old[i\_A]; are assigned just before the vectors are formed (before line 41). In the flow functions they must be assigned by alpha\_P = vars\_P[5]; and alpha\_A = vars\_A[5];.
- At the end of the flow definition (after line 25) the variables for alpha must be defined:

```
F_alpha_num = Float64(0.0); F_alpha_num_add = Float64(0.0); and
F_alpha_num1 = Float64(0.0); F_alpha_num1_add = Float64(0.0);
```

## A1. Code snippets

```

1      #
2      # Time evolution
3      #
4      n_progressbar=20;
5      deltat_progressbar=tmax/n_progressbar;
6      p=Progress(n_progressbar);
7      iter=1;
8      while t<=tmax;
9
10     [...]
11
12     for ind in 1:N
13         if celltype[ind]==1 || celltype[ind]==-3;
14             #Pressure gradient calculation
15             dpdx,dpdy=numerical_gradient(3,ind,p_old,cellneighboursarray,
16                                     cellcentertocellcenterx,cellcentertocellcentery);
17
18             #FV scheme for rho,u,v,vof conservation laws
19             cellneighboursline=cellneighboursarray[ind,:];
20             cellneighboursline=cellneighboursline[cellneighboursline.>0]
21             len_cellneighboursline=length(cellneighboursline)
22             F_rho_num=Float64(0.0); F_rho_num_add=Float64(0.0);
23             F_u_num=Float64(0.0); F_u_num_add=Float64(0.0);
24             F_v_num=Float64(0.0); F_v_num_add=Float64(0.0);
25             F_gamma_num=Float64(0.0); F_gamma_num_add=Float64(0.0);
26             F_gamma_num1=Float64(0.0); F_gamma_num1_add=Float64(0.0);
27             for i_neighbour=1:len_cellneighboursline;
                i_P=ind;
```

```

28     i_A=cellneighboursarray[ind,i_neighbour];
29     rho_P=rho_old[i_P];
30     rho_A=rho_old[i_A];
31     u_P=u_old[i_P];
32     v_P=v_old[i_P];
33     uvec=[T11[ind,i_neighbour] T12[ind,i_neighbour]; T21[ind,i_neighbour]
          T22[ind,i_neighbour]]*[u_old[i_A];v_old[i_A]];
34     u_A=uvec[1];
35     v_A=uvec[2];
36     gamma_P=gamma_old[i_P];
37     gamma_A=gamma_old[i_A];
38     A=cellfacearea[i_P,i_neighbour]*face_factor[i_P,i_neighbour];
39     n_x=cellfacenormalx[i_P,i_neighbour];
40     n_y=cellfacenormaly[i_P,i_neighbour];
41     vars_P=[rho_P,u_P,v_P,gamma_P];
42     vars_A=[rho_A,u_A,v_A,gamma_A];
43     meshparameters=[n_x,n_y,A];
44     if i_A>0 && (celltype[i_A]==1 || celltype[i_A]==-3); #neighbour is
45         inner or wall cell
46         F_rho_num_add,F_u_num_add,F_v_num_add,F_gamma_num_add,
47             F_gamma_num1_add=numerical_flux_function(1,vars_P,vars_A,
48                 meshparameters);
49         F_rho_num=F_rho_num+F_rho_num_add;
50         F_u_num=F_u_num+F_u_num_add;
51         F_v_num=F_v_num+F_v_num_add;
52         F_gamma_num=F_gamma_num+F_gamma_num_add;
53         F_gamma_num1=F_gamma_num1+F_gamma_num1_add;
54     end
55     if i_A>0 && (celltype[i_A]==-1 || celltype[i_A]==-2); #neighbour is
56         pressure inlet or outlet
57         A=A*cellthickness[i_P]/(0.5*(cellthickness[i_P]+cellthickness[
58             i_A]));
59         meshparameters=[n_x,n_y,A];
60         if celltype[i_A]==-2; #pressure outlet
61             n_dot_u=dot([n_x,n_y],[u_P;v_P]);
62         elseif celltype[i_A]==-1; #pressure inlet
63             n_dot_u=min(0,-1/(cellviscosity[i_P]*viscosity_factor[i_P])*
64                 dot([cellpermeability[i_P]*permeability_factor[i_P] 0;
65                     cellalpha[i_P]*cellpermeability[i_P]*
66                     permeability_factor[ind]]*[dpdx;dpdy],[cellfacenormalx[
67                         i_P,i_neighbour];cellfacenormaly[i_P,i_neighbour]])); #
68                 inflow according to Darcy's law and no backflow possible
69         end
70         F_rho_num_add,F_u_num_add,F_v_num_add,F_gamma_num_add,
71             F_gamma_num1_add=numerical_flux_function-boundary(1,vars_P,
72                 vars_A,meshparameters,n_dot_u);
73         F_rho_num=F_rho_num+F_rho_num_add;
74         F_u_num=F_u_num+F_u_num_add;
75         F_v_num=F_v_num+F_v_num_add;
76         F_gamma_num=F_gamma_num+F_gamma_num_add;
77         F_gamma_num1=F_gamma_num1+F_gamma_num1_add;
78     end
79     end
80     if i_model==1;
81         rho_new[ind]=rho_old[ind]-deltat*F_rho_num/(cellvolume[ind]*
82             volume_factor[ind]);
83     elseif i_model==2;
84         rho_new[ind]=((cellporosity[ind]*porosity_factor[ind])*rho_old[ind]-
85             deltat*F_rho_num/(cellvolume[ind]*volume_factor[ind]))/(
86             cellporosity[ind]*porosity_factor[ind]);
87     end
88     rho_new[ind]=max(rho_new[ind],0.0)
89     S_u=dpdx;
90     u_new[ind]=(rho_old[ind]*u_old[ind]-deltat*F_u_num/(cellvolume[ind]*
91         volume_factor[ind])+S_u*deltat)/(rho_new[ind]+(cellviscosity[ind]*
92         viscosity_factor[ind])/(cellpermeability[ind]*permeability_factor[
93         ind])*deltat);
94     S_v=dpdy;
95     v_new[ind]=(rho_old[ind]*v_old[ind]-deltat*F_v_num/(cellvolume[ind]*
96         volume_factor[ind])+S_v*deltat)/(rho_new[ind]+(cellviscosity[ind]*
97         viscosity_factor[ind])/(cellalpha[ind]*cellpermeability[ind]*
98         permeability_factor[ind])*deltat);
99
100     if i_model==1;
101         gamma_new[ind]=((cellporosity[ind]*porosity_factor[ind])*gamma_old[
102             ind]-deltat*(F_gamma_num-gamma_old[ind]*F_gamma_num1)/(
103             cellvolume[ind]*volume_factor[ind]))/(cellporosity[ind]*
104             porosity_factor[ind]);
105         gamma_new[ind]=min(1,gamma_new[ind]);
106         gamma_new[ind]=max(0,gamma_new[ind]);
107         #EOS:
108         if gamma>1.01;
109             p_new[ind]=ap1*rho_new[ind]^2+ap2*rho_new[ind]+ap3;
110         else
111             p_new[ind]=kappa*rho_new[ind]^gamma;
112         end
113     elseif i_model==2;
114         if rho_new[ind]>=0.5*rho_0_oil
115             gamma_new[ind]=1;
116         else
117             gamma_new[ind]=0;
118         end
119     end

```

```

95                                     end
96
97                                     #EOS:
98                                     betat2_fac=1 #0.1 #
99                                     exp_val=4;
100                                    a_val=p_init ;
101                                    c_val=(p-a-p_init)/(rho_0_oil-rho_0_air)^exp_val;
102                                    p_new[ind]=a_val+c_val*(rho_new[ind]-rho_0_air)^exp_val;
103                                    p_new[ind]=min(p-a , p_new[ind]);
104                                    p_new[ind]=max(p_init , p_new[ind]);
105
106                                end
107                            end
108
109                            [...]
110
111                            t=t+deltat;
112                        end
113                    end
114
115                    function numerical_flux_function(i_method , vars_P , vars_A , meshparameters);
116                    if i_method==1;
117                        #first order upwinding
118                        rho_P=vars_P [1];
119                        u_P=vars_P [2];
120                        v_P=vars_P [3];
121                        gamma_P=vars_P [4];
122                        rho_A=vars_A [1];
123                        u_A=vars_A [2];
124                        v_A=vars_A [3];
125                        gamma_A=vars_A [4];
126                        n_x=meshparameters [1];
127                        n_y=meshparameters [2];
128                        A=meshparameters [3];
129                        n_dot_rhou=dot([n_x; n_y] , 0.5*(rho_P+rho_A) * [0.5*(u_P+u_A); 0.5*(v_P+v_A)]);
130                        phi=1;
131                        F_rho_num_add=n_dot_rhou*phi*A;
132                        if n_dot_rhou >=0;
133                            phi=u_P;
134                        else
135                            phi=u_A;
136                        end
137                        F_u_num_add=n_dot_rhou*phi*A;
138                        if n_dot_rhou >=0;
139                            phi=v_P;
140                        else
141                            phi=v_A;
142                        end
143                        F_v_num_add=n_dot_rhou*phi*A;
144                        n_dot_u=dot([n_x; n_y] , [0.5*(u_P+u_A); 0.5*(v_P+v_A)]);
145                        if n_dot_u >=0;
146                            phi=gamma_P;
147                        else
148                            phi=gamma_A;
149                        end
150                        F_gamma_num_add=n_dot_u*phi*A;
151                        phi=1;
152                        F_gamma_num1_add=n_dot_u*phi*A;
153                    end
154                    return F_rho_num_add , F_u_num_add , F_v_num_add , F_gamma_num_add , F_gamma_num1_add
155                end
156
157                function numerical_flux_function_boundary(i_method , vars_P , vars_A , meshparameters , n_dot_u)
158                ;
159                if i_method==1;
160                    #first order upwinding
161                    rho_P=vars_P [1];
162                    u_P=vars_P [2];
163                    v_P=vars_P [3];
164                    gamma_P=vars_P [4];
165                    rho_A=vars_A [1];
166                    u_A=vars_A [2];
167                    v_A=vars_A [3];
168                    gamma_A=vars_A [4];
169                    n_x=meshparameters [1];
170                    n_y=meshparameters [2];
171                    A=meshparameters [3];
172                    n_dot_rhou=n_dot_u*0.5*(rho_A+rho_P);
173                    phi=1;
174                    F_rho_num_add=n_dot_rhou*phi*A;
175                    if n_dot_u <=0
176                        phi=u_A;
177                    else
178                        phi=u_P;
179                    end
180                    F_u_num_add=n_dot_rhou*phi*A;
181                    if n_dot_u <=0
182                        phi=v_A;
183                    else
184                        phi=v_P;
185                    end
186                    F_v_num_add=n_dot_rhou*phi*A;

```

```
186         if n_dot_u <= 0
187             phi = gamma_A;
188         else
189             phi = gamma_P;
190         end
191         F_gamma_num_add = n_dot_u * phi * A;
192         phi = 1;
193         F_gamma_num1_add = n_dot_u * phi * A;
194     end
195     return F_rho_num_add, F_u_num_add, F_v_num_add, F_gamma_num_add, F_gamma_num1_add
196 end
```