Integrating Materials and Manufacturing Innovation (2022) 11:363–381
https://doi.org/10.1007/s40192-022-00268-1

TECHNICAL ARTICLE

Coupled Process Modeling of Flow and Transport Phenomena in LCM Processing

Pavel Simacek[1] · Navid Niknafs Kermani[1] · Suresh G. Advani[1]

Received: 9 April 2022 / Accepted: 24 June 2022 / Published online: 28 July 2022

Abstract
In Liquid Composite Molding (LCM) processes, dry fabric preforms are impregnated with a thermoset resin in a closed mold to fabricate a composite. The resin impregnation process is usually accompanied by other phenomena. In this work, we concentrate on transport phenomena such as convection of heat, cure and volatiles that impact the filling process and/or the quality of the manufactured part. Conventional approach to modeling such a flow is to integrate all the involved physics into the numerical solver. The complexity of integrated computational models will increase the computational time and complicate modification of transport and retention models once implemented. The latter particularly complicates the exploratory modeling attempts to uncover additional physics that require fast and easy code modification. We propose and provide an implementation methodology to separately couple transport and flow models. With this approach, one can couple flow simulation using highly specialized simulation tools with associated transport phenomena that use separate implementation. The emphasis is on transport of volatiles, both discrete bubbles and dissolved solvents, but it is equally applicable to other problems, such as particle transport and filtration or cure propagation. The challenge of this approach is to (1) formulate proper models and implement them and (2) solve the communication between models efficiently. In our case, the distinct models can be coupled through data exchange, via standardized message passing interface (MPI). A well-tested and optimized flow simulation tool LIMS (Liquid Injection Molding Simulation) is used to implement the coupled simulation processes and transfer the simulation state in an efficient manner to model other transport phenomena. Coupled models to track distinct particles and/or bubbles of volatiles and implement convected/diffused dissolved volatiles. The results are presented highlighting the feasibility and utility of this methodology.

Keywords Liquid composite molding · Resin infusion · Numerical simulation · Volatiles · Porosity

Introduction

Liquid Composite Molding (LCM) encompasses a family of composite manufacturing processes in which the fibrous reinforcement (preforms) is placed in a closed mold and infused or injected with a low viscosity resin usually a thermoset. The resin penetrates the spaces between the fibers occupying them. Once the resin has saturated all the empty spaces, the curing of the resin is completed and the composite part is demolded. It is imperative to saturate the preform leaving no or very limited porosity in order to avoid knockdown of composite mechanical properties.

This process is well suited to make large complex net shape composite structures. However, as the fabric within the mold is anisotropic and there may be race tracking along the edges of the mold and many inserts present within the mold, the resin flow patterns are non-trivial (and non-intuitive in many cases) hence numerical analysis of the flow process has been developed and applied for some time [1–12]. Such simulations predict the flow patterns as a function of resin and fiber properties, resin injection scheme and part geometry.

There are multiple issues complicating the resin infiltration. Obviously, the resin properties depend greatly on temperature and degree of resin conversion [9, 10]. In addition to resin flow, other quantities such as volatiles may be transported with the resin in the process. Volatiles enter the mold

✉ Pavel Simacek
psimacek@udel.edu

[1] Department of Mechanical Engineering, Center for Composite Materials, University of Delaware, Newark, DE 19716, USA

with resin in various forms. These are also trapped due to uneven flow front progression. Particles may also be added to the resin [13], either intentionally or accidentally or they may be already present in the preform as binder particles that are added to the fiber preform to maintain its shape and or the composite toughness. These could detach and flow with the resin. As the simulation software and computational power improve, description of these phenomena is included in the model [13, 14].

On the other hand, the necessity to optimize and actively control the process leads to optimizing the simulations for performance and particular properties of the process, while trimming down the "secondary" physics to allow fast execution and full simulation for optimization and process control [12, 15, 16].

Modeling the Liquid Composite Molding

To predict the resin flow behavior in the LCM process, the theory of flow through anisotropic porous media is used. This has been experimentally validated and forms the backbone to describe the resin flow as it impregnates the fibrous porous media. It uses Darcy's law to relate volume averaged velocity, $<\boldsymbol{v}_f>$ to the resin pressure gradient $\nabla \boldsymbol{p}$ through the preform permeability tensor $\boldsymbol{K}$, and resin viscosity η:

$$\langle \boldsymbol{v}_f \rangle = -\frac{\boldsymbol{K}}{\eta} \cdot \nabla \boldsymbol{p} \tag{1}$$

The governing equation is obtained by applying the mass conservation principle over the filled domain and, conveniently, over the resin distribution system if the dynamics of infusion lines should be modeled (say, for large structures). This is solved (quasi-statically) to obtain resin pressure within the filled domain:

$$\nabla \cdot \left(\frac{\boldsymbol{K}}{\eta} \cdot \nabla \boldsymbol{p} \right) = 0 \tag{2}$$

The pressure outside the domain is equal to the pressure at the vent. Thus, Eq. (2) is straightforward to solve and the model advances the flow boundary using Eq. (1). For linear systems, this can be implemented quite efficiently [16] and without requirements for too small a time step, apart from those stemming from required accuracy. The model has been incorporated in a numerical simulation of resin flow in LCM and has been utilized for the last three decades [1–12]. The initial aim of the simulation was to predict flow patterns (to establish injection gate and vent locations) or to quantify the pressure/time requirements. Once this capability was achieved, efficient numerical methods were introduced in the simulation to reduce the solution time drastically making it possible to apply it for process improvement through optimization and control [6–8, 11, 15, 16].

Additionally, the physics, originally neglected in the process has been gradually incorporated into the model to improve the predictions and address other important phenomena encountered during this manufacturing process. This includes features such as dual scale flow which represents textile-like stitched or woven fabrics that have different fiber tow permeability as compared to permeability in between the tows, accounting for acceleration (gravity) forces and fabric deformation [17, 18]. Variable temperature and/or resin reaction have been addressed by coupling the governing equation with energy conservation during the process [4, 9, 10, 19].

These two lines of development, namely the optimization and process control on one hand and introduction of additional physical phenomena on the other, require one to focus on the desired objective as there are some limitations to the model perfection. One of these is the determination of necessary process parameters and their *variability*. This is beyond the scope of the current treatise though deserving significant attention. In many cases, the availability of data may limit what can be accurately modeled. Secondly, the process optimization and control in real time call for simulating *huge* troves of scenarios as necessitated by the variability of the process.

The complexity of integrated computational model that includes multitude of phenomena will increase the computational time. It also "locks" in the constitutive models. The former makes optimization challenging, the latter complicates particularly the exploratory modeling attempts that require fast and easy code modification to deal with uncertain constitutive relations and transport formulas. Moreover, most of the additional phenomena to be considered in modeling will not preserve this simplicity. Dual scale phenomena tend to introduce pressure-dependent sink. So does the preform deformation, which will additionally make permeability $\boldsymbol{K}$ dependent on pressure and possibly external forces and stiffness of mold.

In this work, we will focus on numerical implementation of additional transport phenomena models such as heat and cure, particles and their filtration and volatile and gas dissolution that occur during the process. We will utilize the flow simulation and demonstrate how other models can be intrinsically coupled within the same framework while being separate. The objectives are:

- To preserve the functionality of the flow model. Many significant algorithms were introduced in the code to simulate "practical" process issues such as infusion control, injection hardware, flow disturbances [15].
- To implement the transport models independently to facilitate the simulation of transport within easily mod-

ifiable platforms that use the flow simulation engine for flow parameters and provide it with parameter corrections if needed.
- To create a simple way to communicate between the models and share data intrinsically, which is amenable to multiple cores or processors.

Transport Phenomena in LCM

Heat and Cure Transport

The most common physical transport phenomena considered in LCM modeling are energy (heat transport) and cure (species) transport. These were modeled in many existing implementations but with certain limitations [4, 9, 10, 19] and are usually integrated directly within the flow model. Note that local temperature and resin reaction *are* coupled with Eq. (2) as the resin viscosity η will depend on both the temperature and the degree of cure.

To evaluate the degree of cure α and temperature T throughout the solution domain one needs to couple the energy and the reaction equation with Eq. (2). This also requires one to specify additional parameters to describe the temperature and reaction fields.

The energy equation within the flow domain can be written as [20]:

$$\begin{aligned}\left(\rho C_p\right)_{ef}\frac{\partial T}{\partial t}+\left(\rho C_p\right)_f\langle \boldsymbol{v_f}\rangle\cdot\nabla \boldsymbol{T}\\ =\nabla\cdot\left(\boldsymbol{k}+\boldsymbol{K_D}\right)\cdot\nabla \boldsymbol{T}+\eta\langle \boldsymbol{v_f}\rangle\cdot\boldsymbol{K}^{-1}\cdot\langle \boldsymbol{v_f}\rangle+\phi RG\end{aligned} \tag{3}$$

It equates the internal energy change and heat convection with energy transport by heat conduction and dispersion, dissipation and heat generated due to the reaction (Reaction heat R and reaction rate G which depend on T and α respectively). Other necessary data needed to solve this equation include porosity ϕ, density and heat capacity (effective $(\rho c_p)_{ef}$ and fluid$(\rho c_p)_f$) as well as heat conductivity and dispersion tensors ($\mathbf{k}$ and $\mathbf{K}_D$). The reaction equation presents the species preservation in the form:

$$\phi\frac{\partial \alpha}{\partial t}+\langle \boldsymbol{v_f}\rangle\cdot\nabla\boldsymbol{\alpha}=\nabla\cdot\phi\left(\boldsymbol{D}+\boldsymbol{D_D}\right)\cdot\nabla\boldsymbol{\alpha}+\phi G \tag{4}$$

This equates the species accumulated within the volume and convected with the resin to the species diffused and dispersed (diffusion tensor $\boldsymbol{D}$ with dispersion component $\boldsymbol{D_D}$).

For rigorous solution, Eqs. (3) and (4) should be solved coupled with the Eqs. (1) and (2). However, this by itself is a costly numerical exercise, as the system will be non-linear and an iterative solver will be necessary with criteria for stability and convergence. However, the accuracy gained from rigorous solution may not be justified as the uncertainty in experimentally determining some of the material parameters in Eqs. (3) and (4) can lead to larger errors. Also, Eqs. (3) and (4) are convection dominated and will impact the allowable time step. In some formulations [4, 10], the time step can be adjusted. For CV/FEM-based approaches [2, 19] the time step is determined by flow and is resolved by partial decoupling and staggering the solutions of Eqs. (1) and (2), followed by Eqs. (3) and (4), with different time steps. Thus, the temperature and cure values are calculated after obtaining the flow solution using the viscosity from the previous time step temperature and cure values.

The non-linearity and stability issues can be overcome. Other specifics of the process to address are.

- If solved in the flow modeler, Eq. (3) will cover only the resin-filled domain within the mold cavity. The heat transfer is by no means bound by this domain and in many cases, it needs to be modeled over the entire mold and its environment with appropriate boundary conditions.
- Eq. (4) assumes that the converted species move with the resin flow. There are some indications that the fibrous reinforcement may filter the longer molecules [21], rendering the convection term in Eq. (4) inaccurate. Thus, we may need not only a modified equation but also additional material parameter(s) to describe cure transport during the filling process. Similarly, tracking particles will require a model for filtration/retention and updating the change in permeability [14].

In summary, heat transfer and cure models are available in integrated form, but the case can still be made to decouple them from flow model solution on basis of their different solution domains and uncertainty in material parameter values in the heat and species transport equations, or even ease of implementation so one can explore different conversion models in a fast and reliable way.

Solid Filler Particles and Filtration

From nano-particles to elastomer spheres, addition of particles into the resin before injecting into a mold containing the stationary fibrous media to manufacture a composite using the LCM process has been considered to improve the toughness or to introduce multi-functionality [22]. Tracking particles during suspension flow is well established in injection molding simulations to determine location, concentration and orientation of fibers and fillers. Simplest approach consists of integrating the velocity field and using force and torque balance around the particle to predict its orientation during flow. In LCM modeling the former is straightforward. Equation (1) provides the volume-averaged velocity at any

location. The speed of a particle $\boldsymbol{v}$ is given by the apparent velocity:

$$\boldsymbol{v} = \boldsymbol{v}_f = \frac{1}{\phi}\langle \boldsymbol{v}_f \rangle = \frac{-\boldsymbol{K}}{\phi\eta} \cdot \nabla \boldsymbol{p} \tag{5}$$

which can be easily integrated to provide the particle location. The particle orientation, if non-spherical, may be far more complex as it requires velocity gradients that are not available due to the flow averaging. Also, unlike injection or compression molding, which has an empty mold, in LCM we have a mold containing fiber preform which is likely to filter the solid particles if the particle sizes are of the order or larger than the gaps between the fibers [14]. The physics of such filtration has been addressed only for spherical particles. In this paper, only spherical particles and gas bubbles are considered.

In the simplest case, low concentration of particles does not influence the flow and Eq. (5) can be solved as partially (or simply) coupled problem: The solution of Eqs. (1) and (2) influences the solution of Eq. (5) and have to be carried out beforehand. However, whatever was obtained from Eq. (5) has no influence on the flow solution in the subsequent time steps. The challenge stems from the ability of the stationary reinforcement to slow down and potentially capture the particle or accelerate its motion. To deal with this we may augment Eq. (5) with the "mobility factor" $\boldsymbol{U}$, tensorial value that modifies the particle velocity in Eq. (1) due to the interaction with the fibrous reinforcement and Eq. (2) that can increase the velocity due to particle buoyancy. Thus particle velocity can be written as

$$\boldsymbol{v} = -\boldsymbol{U} \cdot \frac{\boldsymbol{K}}{\phi\eta} \cdot \nabla \boldsymbol{p} \tag{6}$$

In the case of tracking gaseous bubbles, $\boldsymbol{U}$ is called the "bubble mobility" [23].

If particles get slowed down or captured, they may reduce flow channel(s) and thus modify the permeability $\boldsymbol{K}$. Then, the problem becomes "fully" coupled just as in the case of energy transport and outcome of Eq. (6) will influence the system of Eqs. (1) and (2).

Dissolution, Voids, and Volatiles

The motion of porosity during the resin infusion combines the transport of discrete entities (gaseous bubbles) with the convective transport of dissolved gases. Once the volatile bubbles nucleate, they behave just like particles that can expand or contract and can be represented by Eq. (6). The mass transfer of dissolved gases (such as water vapor) can be described by the concentration of the dissolved gas c which can be described by transport equation similar to Eq. (4):

$$\phi\frac{\partial c}{\partial t} + \langle \boldsymbol{v}_f \rangle \cdot \nabla \boldsymbol{c} = \nabla \cdot \left(\phi \boldsymbol{D}_{\boldsymbol{c}} \cdot \nabla \boldsymbol{c}\right) + \phi \mathrm{S} \tag{7}$$

This equates to the concentration of species accumulated and convected with the diffused and generated. The generation rate S is not related to the reaction rate but to the ability of discrete bubbles of the gas to be dissolved in (and out of) the resin. The discrete bubbles *are not* described by Eq. (7). However, they will be coupled to this equation through the generation/dissolution rate S. The diffusion coefficient should contain dispersion (i.e., be flow velocity dependent) although experimental characterization of this phenomenon seems elusive. To obtain proper form of S, one needs to introduce some bubble growth description, based on Henry's law or some other critical concentration constitutive relationships.

As the resin saturation with volatiles may drop below the critical value, numerical implementation of such models requires significant "bookkeeping" to execute. On the other hand, despite the aforementioned challenges, the modeling of bubble motion, growth and reduction can be simply coupled with the LCM flow model as long as the reinforcement permeability $\boldsymbol{K}$ is not significantly affected by the presence of the bubbles. This allows one to compute resin flow and solve Eqs. (6) and (7) based on its results without influencing the resin flow. Incorporation of these additional transport phenomena during the LCM process requires re-organization of the structure of flow modeling to accommodate description of other variables such as temperature, cure and bubble motion and growth during the resin flow.

Other Simulation Applications and Computational Resources

In the previous sections, we presented governing equations for several physical phenomena that depend on resin flow and are modeled along with this flow. The heat and mass transfer are strongly coupled with the flow model in the sense that temperature and cure impact the flow through resin viscosity. Other transport phenomena (particles, volatiles) do not necessarily exhibit such strong impact and may be in many cases treated as dependent on flow but not impacting the flow itself. If the filtration influences permeability [14] this coupling must be added.

Similarly, different types of volatiles may be transported in various forms. This will result in multiple Eqs. (4), (6) or (7), possibly coupled together. Embedding these directly into a flow model would be challenging and detrimental to the computational performance. Coupling these simulations with flow model as independent codes will at least reduce the complexity and allow for easy exploration of the importance of these phenomena.

There are additional models that may depend on flow simulation, require significant computational effort and result in changes to the flow model. These may, for example, include the flow modeling with deformable mold. The deformation depends on pressure field in which case the model requires elastic solution of the mold deformation and these dimensional changes will affect the flow model.

For process control one needs to obtain a solution as the process is evolving which could be in the order of seconds or a few minutes to be able to decide on a control action, hence the simulation execution needs to be quick, reasonably accurate and efficient. Optimization to obtain a processing window may require executing the simulation several times with varying parameters which will require significant resources. The objective and application will guide to what extent this requires executing the simulation with significant processing at each time step. Hence a structure of the simulation that allows one to independently include or exclude a phenomenon will be useful when considering such manufacturing simulations that involve a multitude of physical phenomena.

Integrated and Coupled Models

There are two approaches to expand the existing modeling capability. These fundamentally deal with the organization of the computational process, not the needed modeling. The first approach is to embed the additional physics to be modeled and its governing equations directly into the existing numerical implementation. Then, all the solutions are executed within a single executable framework. The framework grows with the added functionality. To fall back to the original, simpler model one either uses the older version or some input that switches the new additions off. We will call this approach integrated modeling. Currently, most LCM models follow integrated flow modeling with select cure and energy transport models embedded in them [4, 9, 10, 19]. This encapsulation of all models that run in parallel within a single implementation package is a natural approach and allows for optimization of execution speed, particularly for the most complex and demanding models. It has a few drawbacks as well, for example, it cannot handle different domains for energy and mass transfer gracefully.

The second approach is to let each model do its specialty and synchronize data as needed. In this case, we implement the additional functionality as a separate model and resolve the communication of results between the models. We will call this approach coupled modeling. There is nothing particularly new about this approach. It has been used extensively in commercial packages for example to couple thermal and plastic deformation solutions.

The novel proposal herein is to apply this approach to modeling transport phenomena in LCM. Then, the simulation (Fig. 1) consists of executing the flow model that provides the resin flow data to other coupled models. Multiple coupled models can be added as needed, each in its own separate code, keeping the models simple and easily modifiable and upgradable. The benefits are simplicity, ease of modification and addition of more modules. The performance can gain as one can add or remove models and keep the number of those executed to a minimum and allowing

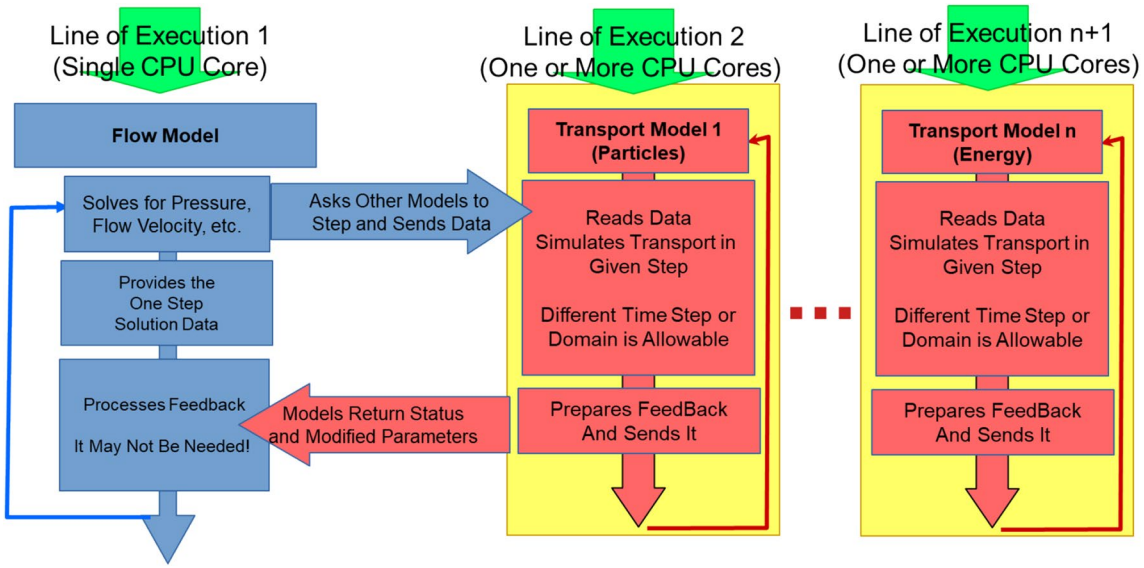

Fig. 1 Coupled modeling of LCM process with transport phenomena. Each model runs in its own solver (executable), with the option to use its own CPU core(s)

for multi-core or multiple CPU execution without the need to parallelize individual models.

Thus, for isothermal single-scale mold filling, only the flow model would be executed, while for reacting resin with porosity and bubble transport multiple models are needed. Whenever these models influence the flow (say reaction and temperature field influences viscosity in elements) the execution of both models needs to be synchronized during each step, otherwise, the flow model can theoretically "run ahead." Each model can utilize a separate CPU core or cores, allowing it to benefit from some level of parallelization even if the model itself is not parallelized.

By the nature of coupled models, these tend to simulate the world in staggered fashion. In Fig. 1, the transport model(s) are a time step behind and this delay is kept through the execution of the entire simulation. Whatever data modification transport models impart to flow model will be one step behind as well.

The model separation with individual solvers, as suggested above offers some advantages over the integrated code:

1. It is possible to have different solution domains. Most importantly, for energy transfer, the transport model can address heating and cooling of the mold parts that are not considered in the flow domain or volatile transport as the energy transfer is not bound by the mold cavity.
2. The individual models are much simpler and constitutive laws can be tested and developed without the need to consider the impact on other unknowns in the solver. Stability, for example, can be addressed within each model.
3. Fig. 1 schematically depicts two processes running in parallel. If multiple CPU cores are available, this allows one to use them even if implementation of solvers is unfriendly to parallelization. Use of highly parallel systems, for example, if we are tracking discrete entities each modeled independently, is straightforward.
4. If the interfacing is properly defined, the individual solvers may be written in different programming languages or even executed on different platforms.

Implementation

To examine transport phenomena we need a code to simulate the mold filling flows in LCM. We will use Liquid Injection Molding Simulation (LIMS) program [15] that was developed at UD several decades. LIMS can be executed through a script which allows one to access and change material properties, boundary conditions, values of the variables at each time step. The coupled solution as shown in Fig. 1 can be implemented on a step-by-step basis using this scripting. We implemented this approach previously to capture the dual scale filling phenomena [24] and for the case of a deforming preform [18]. This is possible as the nature of LCM process calls for significant flexibility within the simulation. In case of LIMS, this is achieved by the scripting control of all the aspects of the filling process [15]. The program can process scripts that are based on the state (current results) of the simulation. This is used even when LIMS models very simple resin impregnation into the preform and it can be used to control repetitive simulations.

Figure 1 shows the LCM model controlling the transport modeling. We can use this approach in two different ways to control the coupled simulation. Either, as shown, LIMS simulation of the flow controls stepping and data transfer to and from the additional models in the system, or the additional programs in the system control LIMS by issuing it commands. The latter is of limited use herein as we will communicate with simple transport models. However, if the other program offers better flow control or data processing capabilities than the LIMS script, this may be very useful approach.

We define and implement the communication mechanism in such a way that either option is available, but in the examples presented we will use the flow simulation to control the other models. In either case, we need to provide ways to pass commands and simulation data between LIMS and other interfaced program(s). This can be in the form of short, simple messages for commands and synchronization or it may involve passing large messages containing blocks of binary data with entire model state. The binary format is necessary as it will not slow down the simulation performance. The memory sharing scheme implements the communication routines using standard MPI (Message Passing Interface) runtime. The necessary functionality to implement coupling can utilize MPI with shared buffers. Appendix A provides the implementation details.

Examples to Simulate Transport Equations with Coupled Approach

In this section, we explore modeling of two transport phenomena, one tracking discrete entities inflow and one being continuum transport through convection. The implementations are coupled with LIMS as a separate executable, instead of embedding it directly in the code or the script. As stated, this allows simpler and more flexible coding. In both examples, the solution of transport equations requires knowledge of LIMS flow solution but it does not affect the subsequent LIMS flow modeling.

In this study, a 2D planar, 0.5 m × 0.5 m preform is considered for the simulations. The preform is discretized by a

50×50 uniform mesh (Fig. 2). The inlet gate for injection process is at the bottom-left corner (node 1) of the preform.

Bubbles are introduced in the same area, but in the bubble distribution model, the initial location of the bubbles adjacent to the inlet gate highly influences the bubbles trajectory in the simulation. The velocity representation in the single element is poor (constant) and we will introduce the bubbles in a slightly wider area. The 25 (5×5) elements adjacent to the inlet gate are assumed to be the bubble entry domain from which the initial location of the bubbles is selected randomly.

The first example investigates how LIMS can be coupled with an external model to predict the discrete particle (or bubbles) distribution within the preform during the LCM process. In the second study, the external model predicts the convection of dissolved volatile concentration during the process. As stated above, we will limit this work to one-directional coupling. It is assumed that the presence of bubbles does not influence the permeability of the mold domain.

To implement all the transport models, a simulation template was created which:

- Performs all the communication with LIMS over MPI – it reads and parses commands and can read the entire simulation state.
- Pre-computes the expected data necessary for any generic transport modeling, such as velocities and flow rates between elements.
- Creates the map of element-to-element connectivity needed to follow the transport source or destination as the models are based on individual elements.

This can be considered as a common underlying framework code to model any transport phenomena. The approach may be further optimized for individual tasks as some computation may be unnecessary.

Bubble Dynamics During Resin Flow

The preform, depicted in Fig. 2, with the permeability of $10^{-9}m^2$ is selected for the simulation examples with an inlet gate at the bottom left corner. Figure 3 presents the flow development simulation during the injection process.

As mentioned earlier, the transport of particles, in forms of either solid filler or gaseous bubbles, is one of the challenges in LCM processes. To track a single particle, we need only the velocity field. As LIMS provides the pressure field across the preform, the particle tracking simulation solves Eq. (5) within the filled domain based on the velocity vector within each simulation element. This can use different discretization approach than LIMS uses internally. For the convection description, it is advantageous to compute best fit constant velocity over the entire element instead of using continuously or piecewise varying value.

Using the best velocity value will exactly match LIMS internal solution only for linear elements. This approach should be convergent for other elements with mesh refinement *as long as inlets are properly modeled*. For singular (but commonly used) description of inlets, such as single node prescribed pressure, the convergence will become an issue even in a simple flow model [25]. Also note that the particle tracking *usually uses* apparent velocity, not volume averaged one as usually used in flow simulation and convection. In Eq. (5) this is demonstrated by the presence of porosity in the denominator; Generally, care is needed in how the permeability $\boldsymbol{K}$ and bubble mobility U is defined as those definitions change between various fields of science. The computed velocity vectors within the

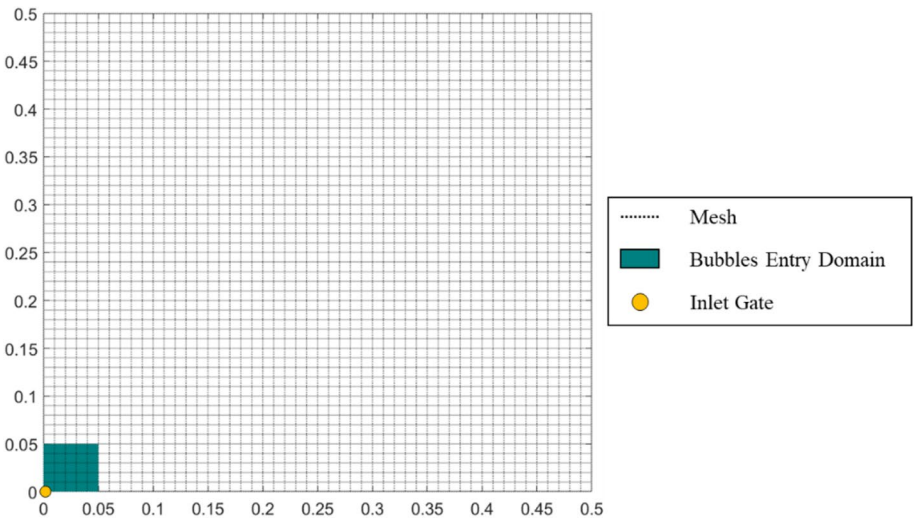

Fig. 2 Mold geometry and the mesh. Resin inlet and the bubble entry region are depicted

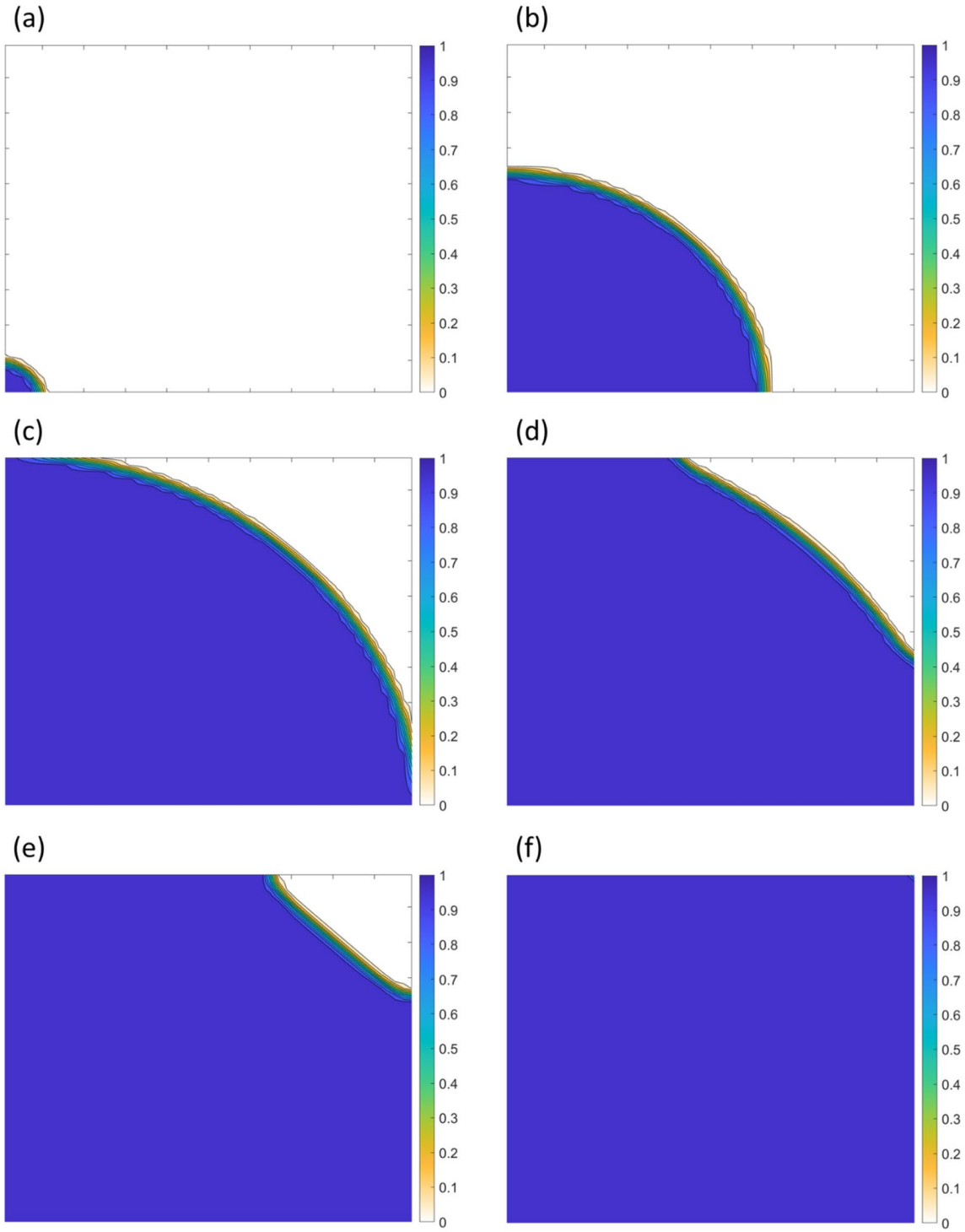

Fig. 3 LIMS simulation results showing the injection molding process, for a $0.5 \times 0.5\ m^2$ preform with an inlet gate at the bottom left corner at different times: **a** 14.2, **b** 1126.2, **c** 3141.8, **d** 3433.8, **e** 3787.2, and **f** 4183.5 s. The color bar shows the fill factors of elements. The fill factor 1 represents the fully saturated region while the fill factor 0 corresponds to the dry region

saturated elements at 89.3 s and at the end of the simulation are shown in Fig. 4.

As expected, Fig. 4a illustrates a radial flow around the inlet gate as it approaches the outlet. Once the flow loses the radial symmetry after the resin contacts one of the walls (Fig. 3d and e) and the flow pattern changes (Fig. 4b) with the diagonal direction becoming more dominant.

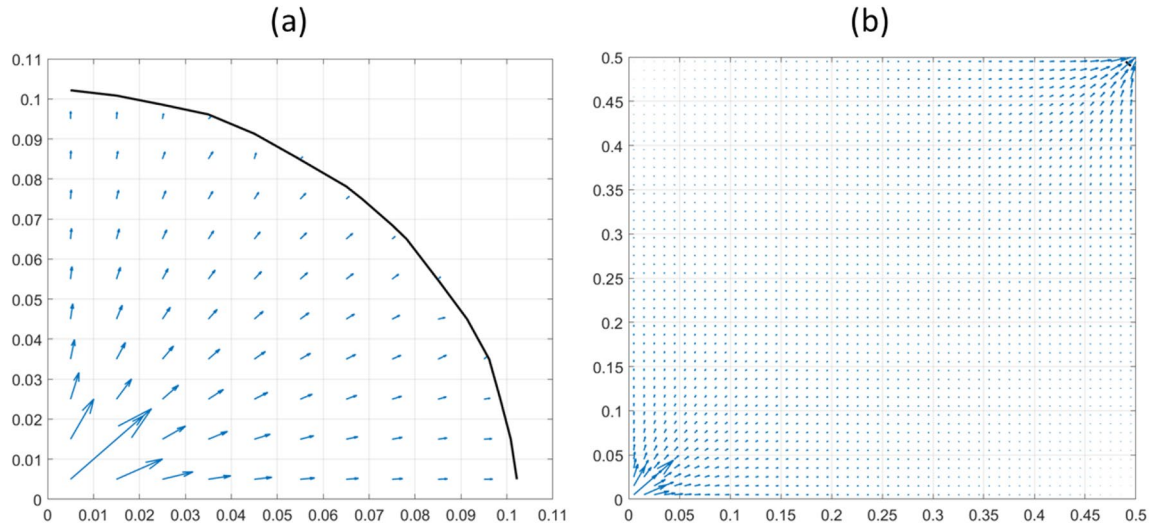

Fig. 4 Velocity vectors within the elements describing the velocity field across the preform at **a** 89.3 and **b** 4183.5 s

By multiplying the velocity vector of the element with the mobility factor of the particles located in that specific element (as in Eq. (6)), the particle's apparent velocity vector is computed. Therefore, to move the particle we need to (i) identify the element in which the particle is located (ii) compute the flow velocity in that element and (iii) multiply it by the mobility factor and move the particle using the time step of LIMS simulation.

This simplistic approach will work well for two and three-dimensional geometries when tracking a single particle. To track multiple particles, it will be better, for the sake of efficiency, to pre-compute velocities in all elements and to track the elements in which the particles are located, following them into neighboring elements and changing the velocity vector mid-step as they cross the boundary. This is a general approach which will work well for curved shell (2.5 D geometry) to find and track the bubble or the particle. For the current 2D case, the template precomputes the velocity and to find the host element of a particle, sum of angles between the vectors connecting the particle to the element's vertices must be computed. The particle is in an element if the sum of the angles is equal to 2π.

The mobility factor of a bubble of a specific size flowing in a porous medium is a difficult quantity to estimate [23]. In flow of a single-phase fluid, it can be estimated using buoyancy, but the presence of porous reinforcement leads to significant irregularities. There is obviously some dependence on the bubble size and the presence of reinforcement magnifies it. We have chosen to simulate it with a range of these factors being between 0.5 and 2.0 relative to the apparent fluid velocity. For the bubbles to escape at the flow front, the bubble mobility factor, *U,* must be greater than unity. If a bubble arrives at the flow front it will either burst or escape and is no longer present within the domain [26]. We keep the factor *U* constant through individual simulations as, in this case, we do not consider bubble growth. However, this one-way coupling approach can easily allow for inclusion of further physics of bubble dynamics without any changes being incorporated into the flow simulation.

In the following simulations, it is considered that one bubble is injected into the preform per LIMS time step. The initial location of the bubbles is randomly selected within the 25 elements adjacent to the inlet gate (Fig. 2). This is due to the constant velocity vector within each of the elements at that time step. If the initial location of the bubbles were only the element next to the inlet, due to the constant diagonal flow around the inlet gate, all bubbles would move in the diagonal direction. The alternative solution to this issue would be to model the gate more rigorously, as a semi-circle with multiple adjacent elements. LIMS can identify the elements that contain the flow front and bubbles in those elements are eliminated. Figures 5 and 6 depict the simulation results of the LIMS connected with the bubble mobility through MPI programming approach.

Figure 5 shows the "raw" simulation tracking individual bubbles and the averaging of the bubble volume to obtain the resulting porosity. The bubble size *is not to* scale. The mobility factor is constant and equal to 1.5. Therefore, it is expected that the bubbles arrive at the flow front and escape. This phenomenon results in non-uniform porosity distribution in the part. Figure 6a and (6b) clearly show that porosity close to the flow front is much lower than the middle of the part where the bubbles were not able to escape and were trapped.

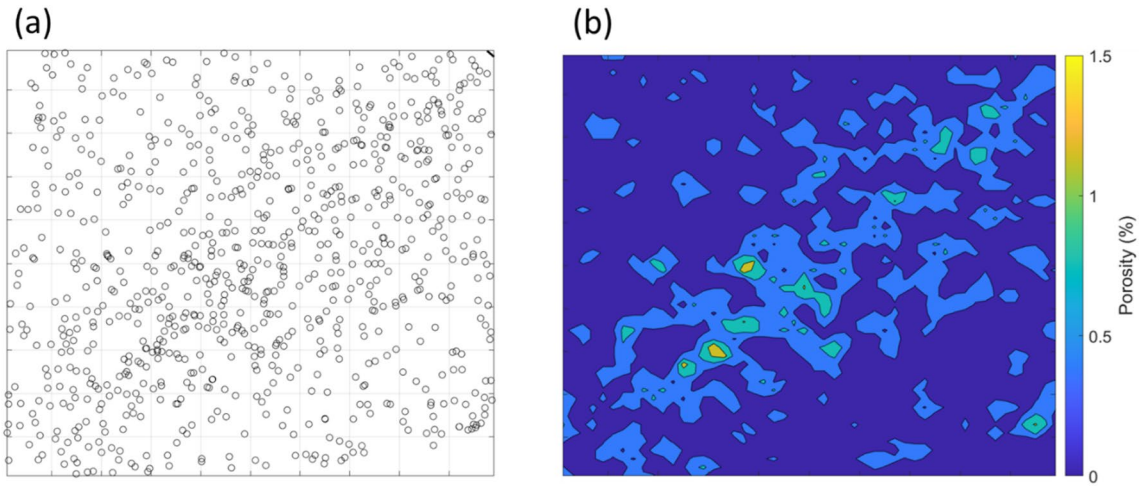

Fig. 5 Bubble location and the porosity map. The porosity of elements is smoothened by averaging the porosity of the element with 4 adjacent elements. The bubble size is not scaled. The radius of bubbles is 0.001 m and constant during the entire simulation

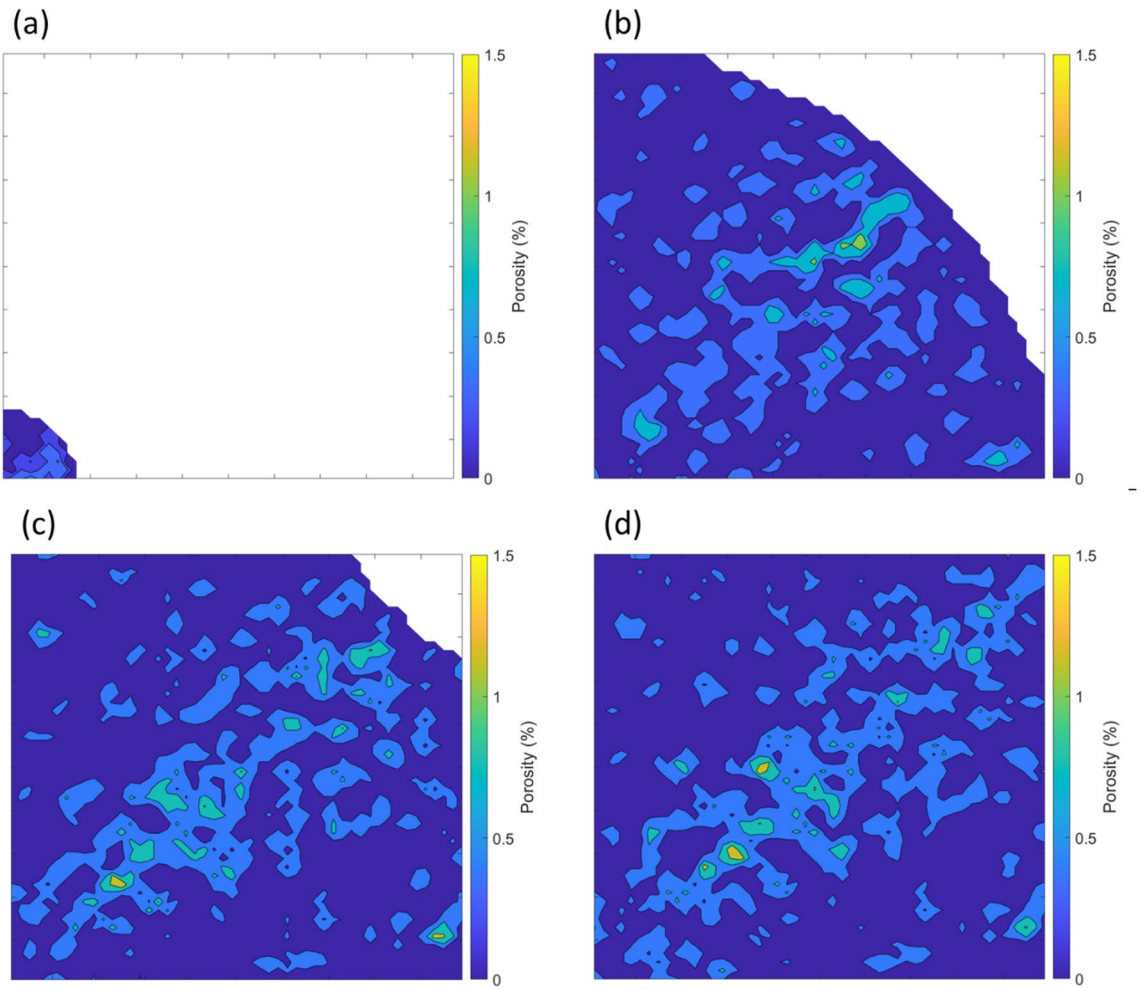

Fig. 6 Flow front location and porosity distribution due to bubble transport during the liquid injection molding of a square fabric at different times **a** 64.5, **b** 3141.8, **c** 3962.9, **d** 4184.4 s. One bubble per LIMS time step is injected at random location within the 25 elements adjacent to the inlet gate into the preform. The results correspond to 1.5 mobility factor (U) in the simulation

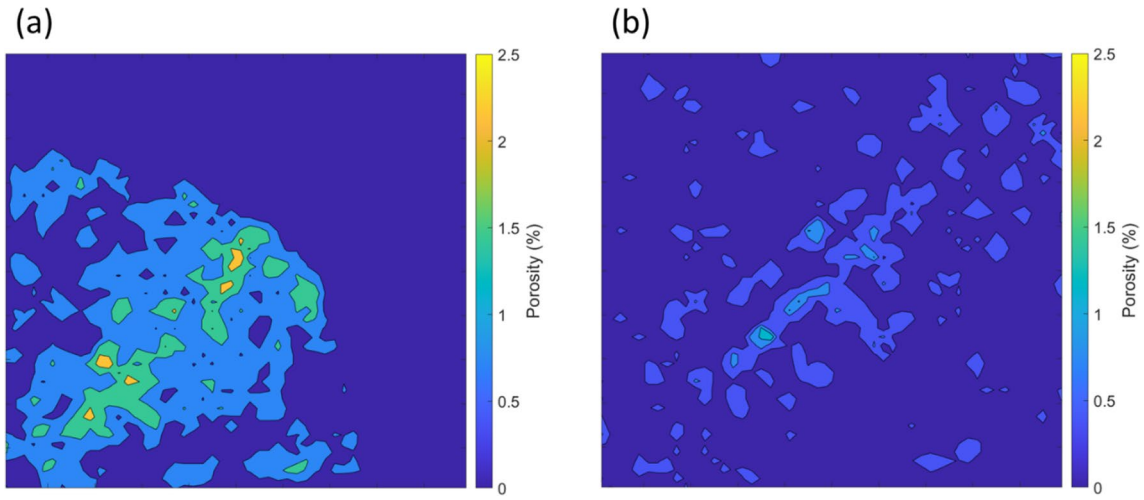

Fig. 7 Porosity distribution at the end of liquid injection molding simulation with two different mobility factors **a** U = 0.5, and **b** U = 2. The porosity is much higher for slower bubble motion entrapping them within the limited subdomain

Figure 7 shows the porosity distribution for the same flow simulations, but with slower ($U = 0.5$) and faster ($U = 2.0$) mobility factors respectively.

In the simulation with lower mobility factor (0.5) (Fig. 7a), bubbles move slower than the flow front and are entrapped while bubbles with higher mobility factor (2.0) move faster than the flow front velocity and escape as soon as they reach the flow front. Hence, the number of bubbles, remaining in the preform at the end of the simulation, in the former case is significantly higher than in the simulation with the higher mobility factor. The bubble escape rate, out of 1313 introduced bubbles during the simulation with the slower mobility factor (0.5) was 0.8% while it is 51.7% when the mobility factor was 2.

The presented example for bubble motion and distribution illustrates modeling of such coupled phenomena in LCM. However, the example was grossly simplified: (i) it did not address the bubble growth (ii) it assumed a constant mobility factor and (iii) it assumed that bubble transport does not affect the flow development. From practical viewpoint, the first two simplifications will not be true for bubbles while the last one is invalid if filler particles are considered, and filtration occurs. Coupling more detailed models for particle transport and adding the effect of particle distribution on flow development may address the current challenges of particle infiltration simulation.

In RTM, when the fabric is not flush along the edges of the mold, the higher permeability along the boundaries will cause the resin to go faster along the edges and this

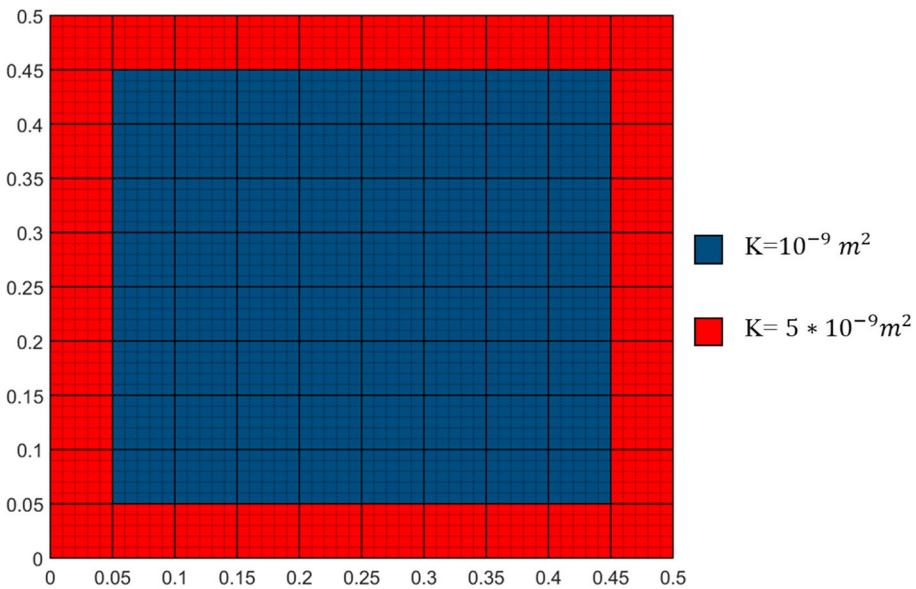

Fig. 8 Racetracking effect is introduced by increasing the permeability of the elements adjacent to the edges. The inlet gate and the bubble entry domain are similar to Fig. 2

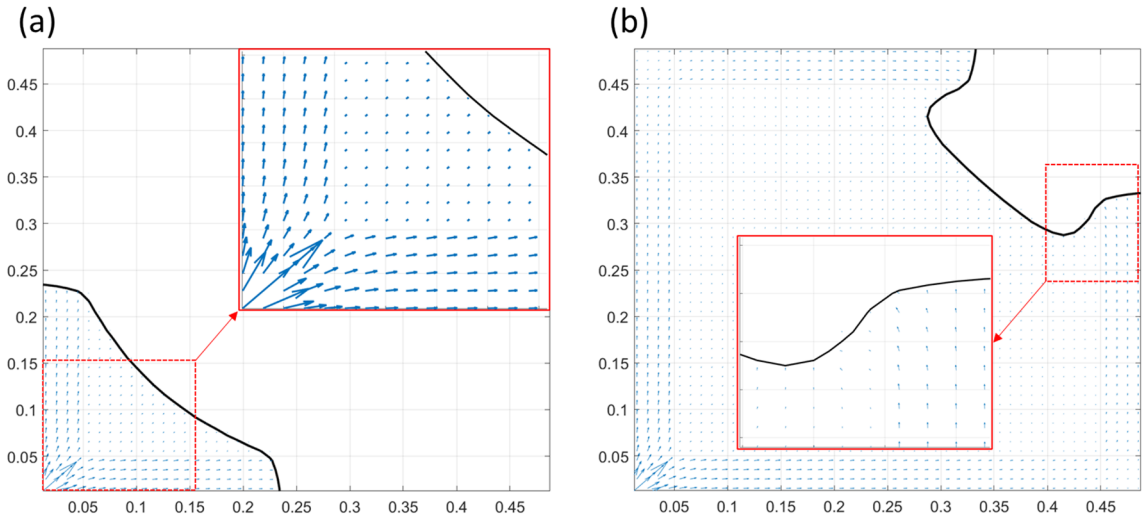

Fig. 9 The resin flow front and the velocity vector at **a** 98 and **b** 930 s after the beginning of the injection. In each figure, the velocity field next to the flow front is magnified. The inlet pressure is 100 kPa

phenomenon is called racetracking. This could have an influence on particle and bubble transport and final porosity distribution. In the following examples, the effect of racetrack phenomenon on porosity distribution due to bubble transport is demonstrated by increasing the permeability of the elements along the edges of the mold. (Fig. 8).

Figure 9 presents the flow front and the velocity vector within the mold at different time steps.

The resulting flow due to race tracking is shown in Fig. 9 in which the velocity within the elements with higher permeability is an order of magnitude greater than the velocity within the interior elements. Moreover, the direction of the flow within the racetracking elements is aligned with the adjacent edge of the mold. Thus, as racetracking changes the flow pattern it also influences the bubble motion and hence the final porosity distribution in the composite part.

In this example, the bubbles mobility factor of 2 was used. The porosity distribution at the end of the simulations for cases with and without racetracking is compared in Fig. 10.

By comparing the results presented in Fig. 10, it can be observed that when the racetracking effect is present, there is higher porosity along the diagonal axis of the preform. This is due to the symmetric velocity field around the diagonal axis and the fountain flow effect that convects the bubbles towards the center of the performance. This phenomenon is even more pronounced when the permeability

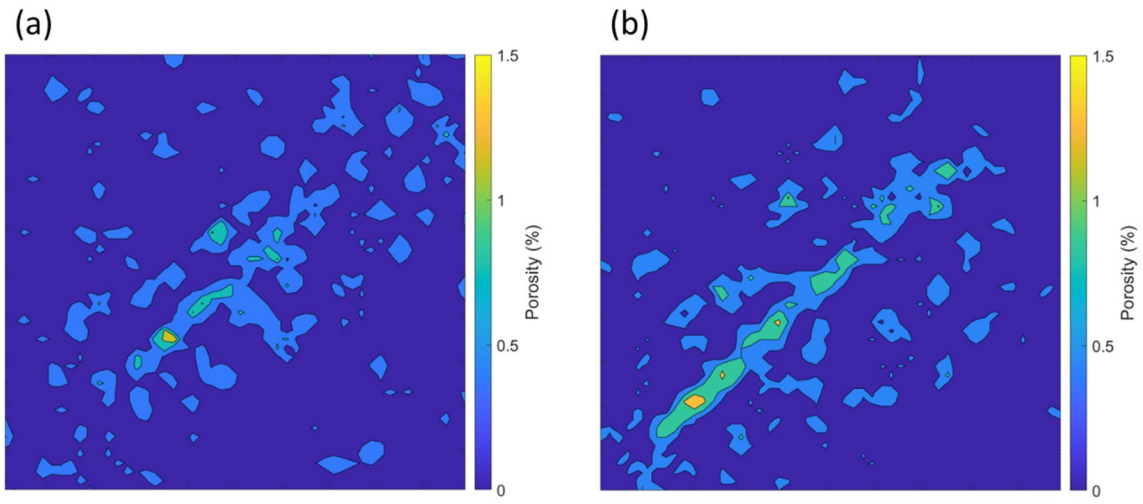

Fig. 10 Final porosity distribution for a 0.5×0.5 preform **a** without racetracking and **b** with racetracking effect. The permeability of the racetrack elements along the edge was five times that of the interior elements

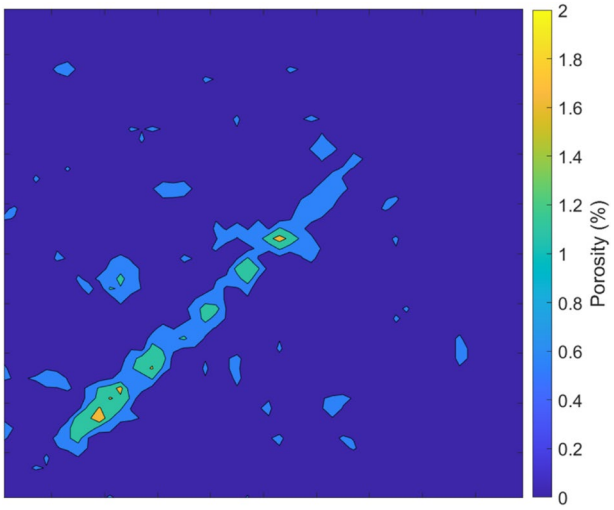

Fig. 11 Final porosity distribution with the permeability of the race-tracking elements equal to 1e-8 m^2. The permeability of the interior nodes is 1e-9 m^2

of the racetracking elements is higher. Figure 11 presents the final porosity distribution in which the permeability of the racetracking elements is increased by an order of magnitude relative to the rest of domain.

Example of Dissolved Volatile Concentration Convection

The other type of phenomenon that can be coupled with LCM simulations is the convection-based transport of a fluid property. This property can be for example the dissolved volatile/solvent/moisture concentration (Eq. (7)) or the degree of cure (Eq. (4)) and the related species concentration. In the former case, volatile transport can be simulated by a simple one-way coupled model between LIMS (modeling flow) and the program that simulates concentration convection. It is unlikely that this transport property affects the flow significantly. In the latter case, a full coupling between models is needed as fluid viscosity changes with the degree of cure. This in turn affects the flow dynamics. Moreover, several reaction stages can be considered resulting in modeling of several concentration species. A simple example related to a one-way coupled model for the convection of the volatile concentration is implemented and presented. Note that this value *is necessary* to include any bubble growth model in the future work.

The concentration of volatile or solvent is important in LCM due to the diffusion-induced growth of bubbles which is dependent on the fluid pressure and temperature. Therefore, to simulate the diffusion-induced growth in LCM, it is necessary to compute the volatile concentration within the fluid domain as the resin fills the mold. The diffusion-induced bubble growth behaves as a sink term in the volatile convection equation (Eq. 7) while the bubble shrinkage and diffusion of volatiles from the bubbles into the fluid behave as a source term in that equation. The coupling of the diffusion-induced growth model with the concentration transport is straightforward but beyond the scope of this work and will be treated in future work. Here we illustrate the coupling of the flow model solved by LIMS with the simple model of volatile convection (with no diffusion) during LCM. The purpose of this is to evaluate modeling performance and behavior for exploration of more complex future work.

Similar to the bubble distribution model, at each time step, LIMS simulation provides the pressure field across the fluid-covered domain. Consequently, the template code obtains this data and evaluates average flowrates at the boundaries of each element within the fluid domain. Together with the current volatile concentration in each element is stored in the transport simulation from the last time step, this determines the amount of volatile flowing in or out of the element. This allows a very simple control volume solution for concentration c in each element i:

$$V_i \Delta C_{i,t+dt} = dt.C_{in,t}\dot{V}_{in} - dt.C_{out,t}\dot{V}_{out} \tag{8}$$

Here, V_i is the element porous volume and $\dot{V}_{in}$ and $\dot{V}_{out}$ are volumetric flow rates across the boundaries and $C_{in,t}$ and $C_{out,t}$ are the current volatile concentrations in proper element for upstreaming formulation. The inflow is considered at the source element concentration. The time step of LIMS is known but it is determined based on filling of the next additional node. For convection of the concentration, sub-steps may be needed to achieve numerical stability of the solution scheme. In this example, fixed number of sub-steps were used but they can be evaluated adaptively based on element volumes and flow rates through their boundaries if CPU performance is necessary. The volume, volumetric flow rates and the element connectivity is precomputed in the "template" code as necessary. Note that solving Eq. (8) on element by element basis will provide a textbook case for parallelization or GPU processing.

For the volatile convection simulation example, the injection simulation is identical to the previous cases shown in Fig. 2 and Fig. 8. In this example, to illustrate the volatile convection simulation, the volatile concentration at the inlet changes from 0.4 to 0.08 after 300 s. Therefore, the concentration of the volatile within the resin is higher during the first 300 s compared to later times of filling. Figure 12 presents the contour plot of the volatile concentration within the injected resin at different times of the filling simulation.

The results in Fig. 12 illustrate how the change in volatile concentration at the inlet after 300 s affects the concentration over the entire resin domain. Due to the convection of the

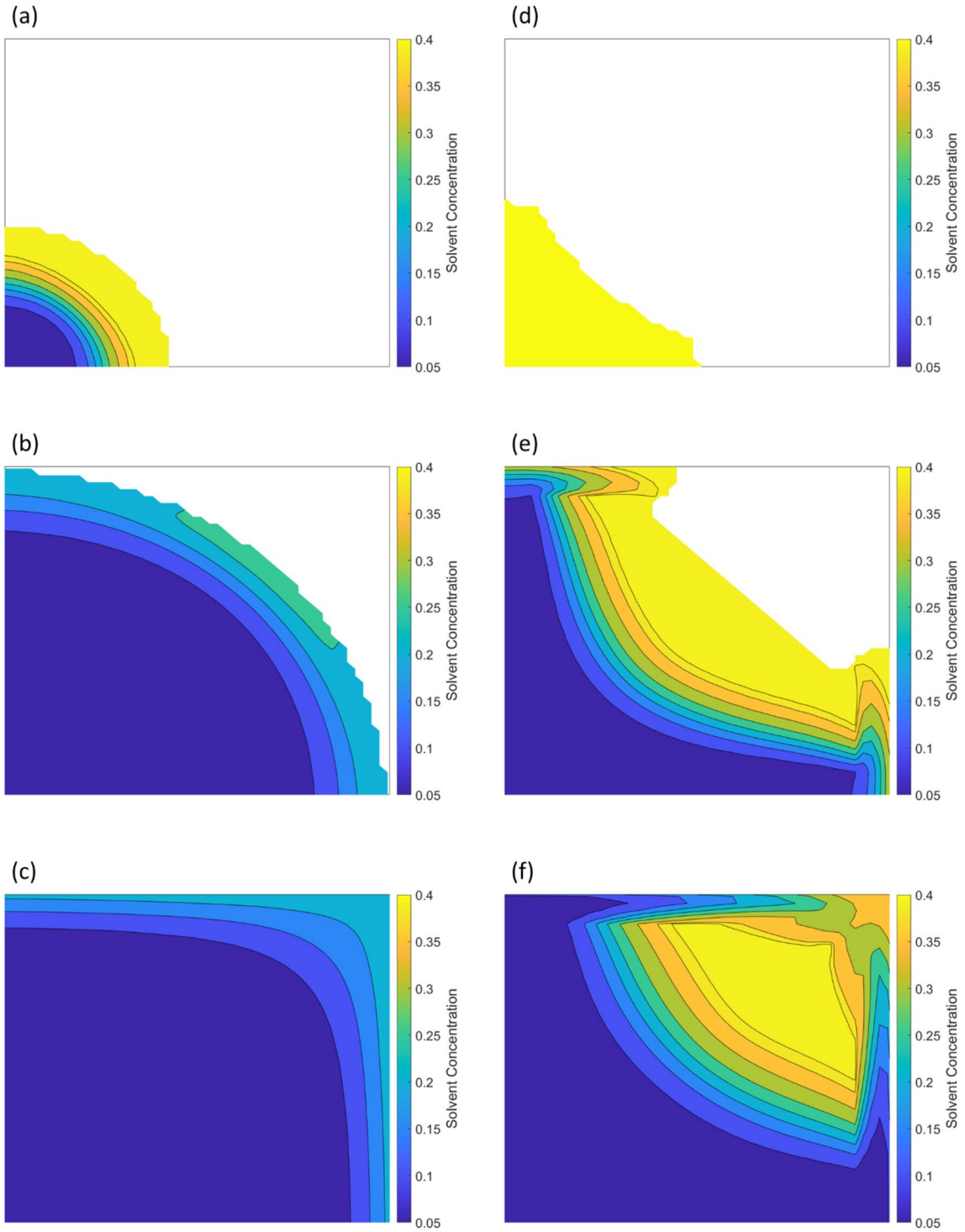

Fig. 12 Volatile concentration convection during the mold filling process. The mesh contains 2500 elements (50×50). The inlet gate is at (0,0). The inlet concentration is reset to 0.08 from 0.4 after 300 s. **a–c** show the simulation for a preform with 1e-9 m^2 permeability in all elements. The corresponding times are 507, 2959, 4183.5 s. **d–f** has racetracking along the edge elements with five times higher permeability than the bulk elements. The corresponding times are 110, 758, 1141

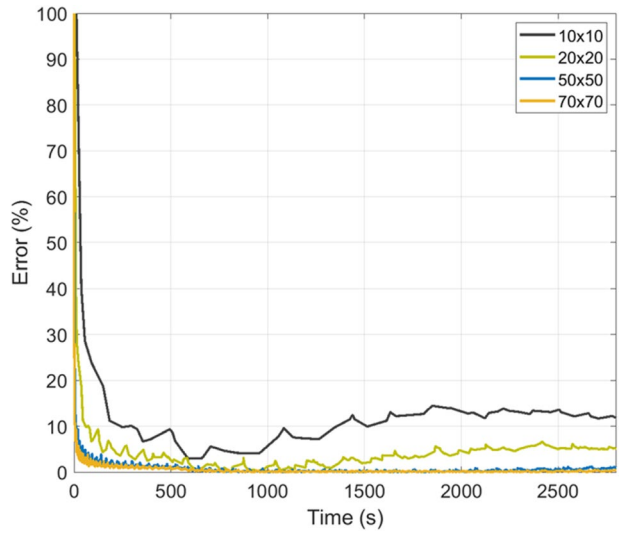

Fig. 13 Mass conservation errors in four different mesh sizes. The error represents the inaccuracy between the volatile mass injected from the inlet with the integration of volatile mass within the fluid

volatile concentration, the lower concentration at the inlet starts propagating within the resin. The simplistic evaluation scheme (Eq. (8)), results in sizeable numerical diffusion. Figure 12a–c shows the convection of volatiles within the preform with uniform permeability (1e-9 m^2) while (d)-(f) correspond to the simulation in which the racetrack effect is introduced due to five times higher permeability along the edge elements than the interior elements.

Numerical Implementation Evaluation

Several steps were taken to evaluate the numerical behavior of the implemented models.

First, to validate the results, it is important to check that the mass is conserved and converges as the mesh size is reduced. For the resin, this behavior is satisfied in LIMS. However, we also need to conserve the volatiles injected. Figure 13 compares error between the volatile mass injected from the inlet with the integration of volatile mass within the resin domain at different times for four gradually refined mesh sizes.

According to the error percentages of different mesh sizes, the 50×50 mesh selected for the simulations in this study provides the desired accuracy.

The stability of the time step is another field that was verified. For the flow simulation, there are no practical requirements and neither are there any for the motion of the discrete volatiles, though the accuracy calls for "reasonable" time step. In LIMS, this is adaptively determined by the need to fill one extra node at each time step. Thus, the flow cannot really advance more than one element size per time step.

For the convective transport models, like the volatile concentration, the situation is different. For the explicit solution based on elements (Eq. (8)), there is very simple upper time step limit dt_{max} based on the flow through each element compared to the volume of the element:

$$V_i \approx dt_{max} \dot{V}_{in} \tag{9}$$

The time step in LIMS is fixed and can be larger than the one in Eq. (9). The code template used evaluates the necessary values within each step. The explicit solutions to Eq. (8) can then be executed with multiple time sub-steps. The different models can do the same, each transport model executing with time steps optimal for its individual performance and stability.

The computational performance is at most preliminary, but the flow simulation as implemented in LIMS [16] has, for N nodes in the mesh, the order of solution between N^2 and N^3, assuming no changes are done to the mesh and material data. This is for the entire filling of N nodes. The exponent depends on banding of the matrix which depends on flow patterns, not node numbering.

By comparison, the single transport model evaluation is of the order $N\ x\ M\ x\ P$, the M being the number of substeps and P number of models. It is evaluated N times. As M tends to be limited number this seems to do equally well or better than LIMS (MPN^2) as long as number of models P is limited. In actual implementation the situation is better as all the inclusions are executed within a single model and the order is MN^2. Furthermore, this model is fairly easy to further parallelize or execute on a GPU.

Summary and Conclusions

This work provides a methodology to add the volatile tracking—and other transport phenomena—to full flow simulation in Liquid Composite Molding processes. The simulation processes for volatile transport are programmed separately to describe and track generic transport phenomena associated with resin. The emphasis is on transport of volatiles, and both the transport of discrete bubbles and dissolved solvent is addressed. The approach, with proper programming changes, is equally applicable to other problems, such as particle transport and filtration or cure propagation.

Instead of embedding increasing amount of specialized code directly in the flow simulation, the presented approach takes advantage of execution of multiple simulations coupled at each time step (one or both ways) through standardized message passing interface (MPI) and makes it intrinsic and is the execution is demonstrated with examples.

A flow simulation tool, LIMS, is used to simulate resin infusion over a generic geometry. With LIMS, the void

tracking inherits its ability to simulate infusion control, resin distribution networks and other physics. It is modified to implement the commands that allow the LCM simulation to control the coupled processes (or vice-versa) and transfer the simulation state in an efficient manner. A new and separate code is written to follow the bubble/particle transport and the transport of dissolved solvent on any geometry in which LIMS simulates the flow. The code allows one to track individual or multiple bubbles in general resin flow, evaluate the resulting porosity and simulate the convection of dissolved volatiles. There is a basic model template that evaluates velocity vectors, volumes and flow rates over each element (once for each LIMS step) to determine the transport terms and the time step. After that, the separate code is implemented to accomplish the transport of bubbles or volatiles.

This approach has *very wide applicability*, the two examples presented are just a simple demonstrations. The first one presents coupled simulation for transport of discrete bubbles or particles in the LCM filling flow. Multiple inclusions can be tracked and evaluated both individually and in aggregate. The second example presents the coupled simulation for convective transport of dissolved matter in the resin such as moisture or volatiles instead of embedding increasing amount of specialized code directly in the flow simulation. This demonstrates the ability to solve transport equations based on adaptive, reduced time steps to address the stability issue and to compute the values on element-by-element basis with no equation system assembly and solution, keeping the solution to a order low (N). Some issues were encountered with numerical diffusion and are being addressed, but the computation is stable, of low order and it preserves the mass of volatiles.

Future extension will add the connection between the dissolved volatiles and the discrete bubbles via growth and filtration models and carry out experimental studies to demonstrate the utility of this intrinsically coupled approach to add new physics to LCM processes.

Appendix A: Implementation Details

Previous LIMS Architecture for Communication with External Applications

In our previous work, the LIMS flow simulation was already coupled with additional programs. Primarily, this has been used for integrating the simulation into other software [19] and for optimization and control [19, 20, 27, 28]. The communication interface is shown in Fig. 14. This is used to pass the commands and communicate through textual printed lines through shared memory under Windows (used by LIMS interface (LimsUI) or, for example, Matlab through dynamic link library).

Additionally, there is available and tested extension that allowed passing of entire simulation state in the same way. It is not particularly efficient and is not currently supported in standard LIMS distribution. Note that this mechanism was used to couple the master program (LIMS or other programs) with other LIMS used as the worker process. There was no coupling with different simulation engine(s), although such steps were feasible. All the significant coupled physics model (dual scale flow [24], compliant reinforcement flow [18]) were implemented as LIMS scripts and ran within the simulation without a need for external communication.

Proposed Model Coupling Scheme(s)

The memory sharing scheme for commands was fairly successful in 1990s, but (1) It is based on low-level 32-bit windows implementation that is obsolete and (2) It never provided options to control other programs other than LIMS. The additional physics models run as scripts are feasible but inefficient hence the communication routines to standard MPI (Message Passing Interface) runtime were preferred and implemented. The necessary functionality to implement coupling as in Fig. 14 was obtained in simple, system-independent fashion by utilizing MPI to handle any shared buffers. For single CPU system it will likely be through shared memory as in the old implementation, but it should work over networks on clusters as well. New, more efficient

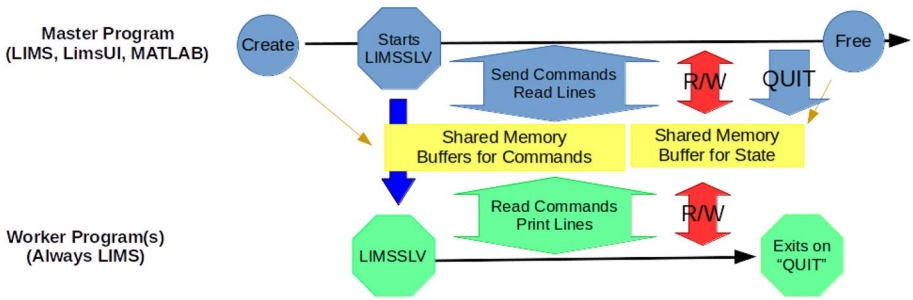

Fig. 14 LIMS communication interface through Windows shared memory

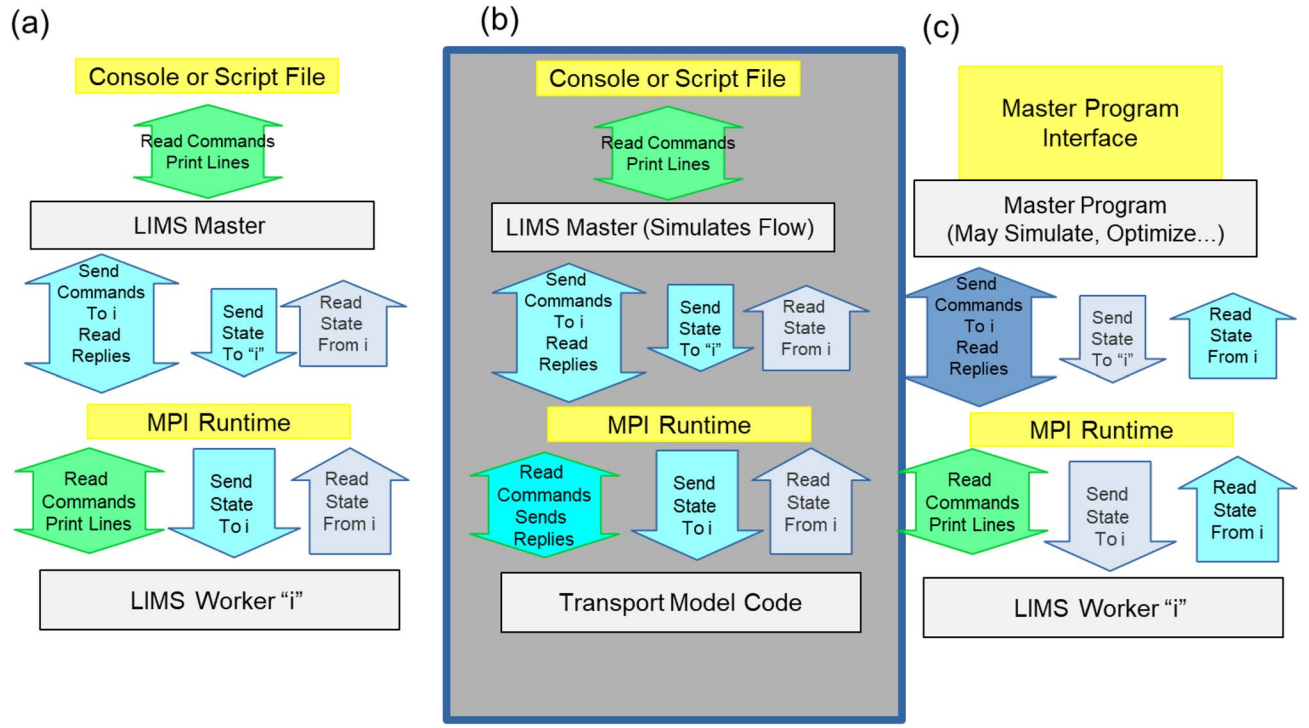

Fig. 15 LIMS communication interface through MPI. **a** LIMS used to control other LIMS programs(s), for example, for multi-scenario simulation and optimization. Either LIMS can run simulation and simulation state can be sent as needed. **b** Use of LIMS to simulate the flow and to control coupled simulation models, for example transport. This is the layout used in this paper. **c** Use other programs to control LIMS. LIMS simulates the flow, the master program may aggregate data from LIMS to simulate something else

way to pass the simulation state through the same interface was also implemented.

This approach allows LIMS to control other simulation processes including different instance(s) of LIMS, or to be executed and controlled by other executables. The messages passed through MPI simply replace the text typed on command line or in a script file.

Implementation Based on Basic MPI Functions

The implementation of coupling mechanism allows execution of all three possibilities in Fig. 15. Of these scenarios, (b) is the one we used to simulate additional transport phenomena. LIMS scripting is used to control the entire modeling and input/output sequence. The scenario (a) is more useful for process optimizations and development as it can run multiple flow simulations on multiple cores—option available on every modern workstation. Scenario (c) passes the control to a non-LIMS executable. It allows as much flexibility as the master program offers. As a part of its execution, master will request the flow simulation results as needed. This can work for both optimization and transport modeling, but it requires more coding unless the master program is MPI-enabled platform like Python.

Modified LIMS program can be executed as master, worker or both. It is executed through MPIrun or MPIexec and it is able to detect whether it runs as master (ID zero) or worker (ID 1 and higher).

When being executed as a master, LIMS reads its commands as self-standing and does print its output in the standard way. It can run as a self-standing flow simulation as well. However, it offers several commands in its scripting language to control the execution of workers. It can determine how many workers are available. It can communicate with its workers at any time with a line of text (as commands). Finally, it can determine if any worker sends a response and act accordingly.

If LIMS finds itself in worker's role, there are no additions for control. It receives commands transparently from the MPI runtime when the master issues them. The output is delivered to the master. For other worker programs, such as the transport simulations presented above, the control communication involves receiving and sending MPI messages in plain text format. For transport modeling, C++skeleton file (template) was built to:

- Take care of the communication and synchronization details

- Read LIMS simulation state at the beginning of each step
- Evaluate some additional resin flow data, such as velocities, which are not evaluated within LIMS and are missing in its state
- Evaluate geometry that is necessary for convective/conductive transfer
- Create connectivity links to allow solution of equations such as Eq. (4) or (7) on element-by-element basis

The simulations presented were built using this template.

The state data exchange is straightforward but (for the sake of efficiency) it requires binary compatibility between programs. Regardless of whether LIMS acts as a master or worker, the simulation state can be sent to other processes (master or worker) and retrieved from them by a single function call from the script or the command line. LIMS will use its current state to export and update its current state on import. This uses several large binary blocks of data and it is quite efficient compared to the older approaches that transferred data based on node by node, element by element fashion even if in binary form.

There are several MPI calls needed to import the state, as size information must precede the entity blocks. The other simulation programs need to read and write a proper sequence of messages starting with counts and followed by blocks of nodal, elemental and inlet data. They will need to understand LIMS data structures to decode these. This approach puts some limitations on executables and architecture—essentially, all executables should be built with the same compiler and settings. However, it avoids the need to reformat and re-code large data structures which may be extremely slow compared to the actual solution of Eq. (8). Note that the skeleton file built for the transport models already contains this code and the same skeleton may be used for other conceivable transport problems.

Acknowledgements The present work is funded by NSF Award No.2023323 which is gratefully acknowledged by the authors.

Declarations

Conflict of interest Authors declare that they have no financial interest or personal relationship that could appear to influence this work.

References

1. Hieber CA, Shen SF (1980) A finite element/finite difference simulation of the injection mold filling process. J Non-Newtonian Fluid Mech 7:1–31
2. Bruschke M, Advani SG (1990) Finite element/control volume approach to mold filling in anisotropic porous media. Polym Compos 11:398–405
3. Voller VR, Chen YF (1996) Prediction of filling times of porous cavities. Int J Numer Methods 23(7):661–672
4. Trochu F, Gauvin R, Gao D-M (1993) Numerical analysis of the resin transfer molding process by the finite element method. Adv Polym Technol 12(4):329–342
5. Phelan FR (1997) Simulation of the injection process in resin transfer molding. Process Polymer Composites 18:460–476
6. Mathur R, Fink BK, Advani SG (1999) Use of genetic algorithms to optimize gate and vent locations for the resin transfer molding process. Polym Compos 2:167–178
7. Minaie B, Chen YF, Mescher MA (2002) A methodology to obtaina desired pattern during resin transfer molding. J Comp Mater 14:1677–1692
8. Nielsen DR, Pitchumani R (2002) Closed-loop flow control in resin transfer molding using real-time numerical process simulation. Comp Sci Technol 2:283–298
9. Ngo NG, Mohan RV, Chung PW, Tamma KK (1998) Recent developments encompassing non-isothermal/isothermal liquid composite molding process modeling/analysis: physically accurate, computationally effective, and affordable simulations and validations. J Thermoplast Compos Mater 6:493–532
10. Trochu F, Ruiz E, Achim V, Soukane S (2006) Advanced numerical simulation of liquid composite molding for process analysis and optimization. Composites Part A 37:890–902
11. Chebil N, Deléglise-Lagardère M, Park CH (2019) Efficient numerical simulation method for three dimensional resin flow in laminated preform during liquid composite molding processes. Composites Part A 125:2154–2163. https://doi.org/10.1016/j.compositesa.2019.105519
12. Park CH, Lebel A, Saouab A, Bréard J, Lee WI (2011) Modeling and simulation of voids and saturation in liquid composite molding processes. Composites Part A 42:658–668
13. Lefevre D, Comas-Cardona S, Binétruy C, Krawczak P (2007) Modelling the flow of particle-filled resin through a fibrous preform in liquid composite molding technologies. Composites Part A 38:2154–2163
14. Abliz D, Berg DC, Ziegmann G (2019) Flow of quasi-spherical nanoparticles in liquid composite molding processes. Part II: Modeling and simulation. Composites Part A 125:105562. https://doi.org/10.1016/j.compositesa.2019.105562
15. Simacek P, Advani SG (2004) Desirable features in mold filling simulations for liquid composite molding processes. Polymer Comp 25(4):355–367. https://doi.org/10.1002/pc.20029
16. Maier RS, Rohaly TF, Advani SG, Fickie KD (1996) A fast numerical method for isothermal resin transfer mold filling. Int J Numer Methods Eng 39
17. Simacek P, Advani SG (2006) Role of acceleration forces in numerical simulation of mold filling processes in fibrous porous media. Composites Part A 37(11)
18. Simacek P, Advani SG (2018) Resin flow modeling in compliant porous media: an efficient approach for liquid composite molding. Int J Mater Form 11(4):503–515
19. Bruschke MV, Advani SG (1994) A numerical approach to model non-isothermal viscous flow through fibrous media with free surfaces. Int J Numer Methods Fluids 19:575–603. https://doi.org/10.1002/fld.1650190704
20. Tucker III CL, Dessenberger RB (1994) Governing equations for flow and heat transfer in stationary fiber beds, In: SG Advani (eds) Flow and rheology in polymer composites manufacturing, Elsevier
21. Maldonado J, Louis B, Klunker F, Ermanni P (2012) Reactive flow of thermosetting resins: implications to lcm processing, In: Eleventh international conference on flow processes in composite materials (FPCM-11), Auckland, New Zealand
22. Aydil T, Taniab H, Erdal M (2014) Resin transfer molding of particle-filled, continuous-fiber rein forced composites, In:

Proceedings of the american society for composites - 29th technical conference, ASC
23. Gangloff JJ, Hwang WR, Advani SG (2014) Characterization of bubble mobility in channel flow with fibrous porous media walls. Int J Multiph Flow 60:76–86
24. Simacek P, Advani SG (2003) A numerical model to predict fiber tow saturation during liquid composite molding. Comp Sci Technol 63(12):1725–1736. https://doi.org/10.1016/S0266-3538(03)00155-6
25. Modi D, Simacek P, Advani SG (2003) Influence of injection gate definition on the flow-front approximation in numerical simulations of mold-filling processes. Int J Numer Methods Fluids 42(11)
26. Niknafs Kermani N, Simacek P, Advani S (2020) bond-line porosity model that integrates fillet shape and prepreg facesheet consolidation during equilibrated co-cure of sandwich composite structures. Composites Part A 139:106071. https://doi.org/10.1016/j.compositesa.2020.106071
27. Bickerton S, Stadtfeld HC, Steiner KV, Advani SG (2001) Design and application of actively controlled injection schemes for resin-transfer molding. Compos Sci Technol 61(11):1625–1637
28. Sozer EM, Bickerton S, Advani SG (2000) On-line strategic control of liquid composite mould filling process. Composites Part A 31(12):1383–1394