UNIVERSITÄT AUGSBURG
Universitätsbibliothek

OPEN ACCESS
OPUS AUGSBURG

Fast approximation of fiber reinforced injection molding processes using eikonal equations and machine learning

Julian Greif, Philipp Lechner, Nils Meyer

Angaben zur Veröffentlichung / Publication details:

Greif, Julian, Philipp Lechner, and Nils Meyer. 2024. "Fast approximation of fiber reinforced injection molding processes using eikonal equations and machine learning." *Composites Part A: Applied Science and Manufacturing* 185: 108340.
https://doi.org/10.1016/j.compositesa.2024.108340.

Nutzungsbedingungen / Terms of use:

CC BY 4.0

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
CC-BY 4.0: Creative Commons: Namensnennung
Weitere Informationen finden Sie unter: / For more information see:
https://creativecommons.org/licenses/by/4.0/deed.de

Composites: Part A 185 (2024) 108340



Contents lists available at ScienceDirect

Composites Part A

journal homepage: www.elsevier.com/locate/compositesa





Fast approximation of fiber reinforced injection molding processes using eikonal equations and machine learning

Julian Greif [a], Philipp Lechner [a,b], Nils Meyer [a,b,*]

[a] University of Augsburg, Institute of Materials Resource Management, Augsburg, Germany
[b] University of Augsburg, Centre for Advanced Analytics and Predictive Sciences, Augsburg, Germany

ARTICLE INFO

Keywords:
Process simulation
Injection molding
Machine learning

ABSTRACT

Injection molding is a popular production process for short fiber reinforced components. The mechanical properties of such components depend on process-induced fiber orientations which are commonly predicted via numerical simulations. However, high computational costs prevent process simulations from being used in iterative procedures, such as topology optimization or finding optimal injection locations. We propose a fast approximation method that extracts nodal features and train a regression model to predict fill states, cooling times, volumetric shrinkage, and fiber orientations. The features are determined by solving eikonal equations with a fast iterative method and computing spatial moments to characterize node-adjacent material distributions. Subsequently, we use these features to train feed forward neural networks and gradient boosted regression trees with simulation data of a large dataset of geometries. This approach is significantly faster than conventional methods, providing 20x speed-up for single simulations and more than 200x speed-up in gate location optimization. It generalizes to arbitrary geometries and injection locations.

1. Introduction

Thermoplastic injection molding is used in a wide range of industries as an economic production process for complex three-dimensional components. During the process, a reciprocating screw injects a thermoplastic polymer melt into a mold cavity that represents the components shape. The polymer is commonly reinforced with glass or carbon fibers to improve the mechanical properties of the molded component. However, local fiber orientations are influenced by the polymer flow into the mold making the properties dependent on the molding process. For example, this results in typical core–shell structures with fibers oriented in flow direction in the shear-flow dominated shell region of a planar section and perpendicular oriented fibers in the extension-flow dominated core region [1,2]. As a qualitative understanding of reorientation is not sufficient to leverage full light-weighting potential or to predict warpage, there is a need for accurate quantitative predictions of the filling process.

Numerical simulations of the injection molding process range back to the 1980's [3] and are well established by now.

Several commercial tools are available to model the mold filling process and fiber orientations at the macroscopic scale, while research focuses i.a. on fiber migration phenomena [4–6], refining orientation models with high-fidelity microscale models [7–9], and flow-fiber coupling [10–12]. However, even relatively efficient commercial macroscale models require significant computational effort with simulations taking up to several hours. Fast estimation of the cooling time without a full simulation has been pursued in the past decades leading to a range of different models of different complexity [13,14]. These models usually build on analytical solutions of the heat diffusion equation in one or more dimensions to determine when the part has reached a predefined ejection temperature.

There are several reasons that necessitate fast surrogate models of injection molding simulations: First, optimization of process parameters typically requires iterative evaluations of the objective function (e.g. for different temperatures, pressures, injection locations, etc.), each of which requires a full simulation run. Second, meaningful structural optimization (e.g. shape, size, or topology) of the component itself requires a process simulation in each iteration as the changed geometry implies different process induced fiber orientations. Third, designers would greatly benefit from a tighter integration of fast prediction tools in their design software to iterate designs faster compared to sending them to a specialist performing a lengthy process simulation.

Several authors use fully-connected feed forward neural networks to predict a simulation result (e.g. warpage, stress on impact, clamp force) depending on processing conditions and apply optimization algorithms

* Corresponding author at: University of Augsburg, Institute of Materials Resource Management, Augsburg, Germany.
E-mail address: nils.meyer@uni-a.de (N. Meyer).

https://doi.org/10.1016/j.compositesa.2024.108340
Received 21 May 2024; Received in revised form 27 June 2024; Accepted 29 June 2024
Available online 3 July 2024
1359-835X/© 2024 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

(e.g. genetic algorithms, particle swarms) to find optimal process parameters [15–17]. Other groups used similar approaches to optimize manufacturing parameters for warpage and volumetric shrinkage [18–20]. These networks are trained with finite element simulations of a specific use case with a specific geometry and are therefore limited to this particular task without allowing for structural optimization of the underlying part. In addition, Kriging models and Gaussian processes have been used by several authors as surrogates for process optimization [21,22], but are also trained for specific use cases. Mixed effect Gaussian process models enable generalization to other geometries to some extend [23]. Ospald et al. [24] account for process-dependent fiber orientations in topology optimization using gradient information of the solution to the eikonal equation (i.e. they use the gradient of geodesic distances to the injection gate to define the fill front normal) as surrogate for the process simulation. While their approximation of fiber orientations generalizes to arbitrary domains, it requires an empirical parameter to adjust the degree of anisotropy. Another approach for process simulation surrogates, which are applicable to arbitrary domains, are direct regression models trained on meshes, point clouds or voxelized representations [25]. Uglov et al. [25] pre-process point cloud data from 158 Autodesk Moldflow simulations by computing shortest paths between all nodes and gates with Dijkstra's algorithm. Subsequently, they propose a gradient boosting model to predict fill times and a convolutional neural network with 2D projections in order to predict warpage. While clearly outperforming their naive estimator, the fill time prediction could be improved further to generalize beyond the geometry class of automotive dash panels used in their work.

In this work, we combine engineered nodal features and machine learning models to rapidly predict nodal fill times, cooling times, fiber orientations, and volumetric shrinkage in fiber reinforced injection molding processes. The method should generalize to arbitrary meshed geometries and injection gate locations, which is achieved with suitable normalization and denormalization transformations. We aim to accelerate the prediction by magnitudes, while keeping the relative root mean squared error of the predicted fields below 10%.

2. Training data generation

Training and validation of fast approximation models requires a ground truth. In this case, we generate the ground truth by running state-of-the-art injection molding simulations on 629 randomly sampled geometries from the ABC dataset [26], which is a large collection of Computer Aided Design (CAD) geometries for geometric deep learning. We mesh the sample surfaces with triangles using the open source software *gmsh* and pick a single random node on the surface as an injection location. Subsequently, we perform an automated *Autodesk Moldflow 2023* analysis for each geometry to simulate the injection molding process.

The injection molding simulation solves the balance of mass

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{1}$$

the balance of linear momentum (neglecting inertia and body forces)

$$\frac{\partial (\rho \mathbf{u})}{\partial t} = -\nabla p + \eta \Delta \mathbf{u} \tag{2}$$

and the balance of inner energy

$$\rho c_\mathrm{p} \dot{\theta} = \nabla \cdot (k \nabla \theta) + \eta \nabla \mathbf{u} : \nabla \mathbf{u} \tag{3}$$

for all locations $\mathbf{x} \in \Omega$ in a part domain Ω and all times t. Here, ρ denotes the polymer mass density, $\mathbf{u}$ denotes the flow velocity vector, p denotes the hydrodynamic pressure, and θ denotes the polymer temperature. The viscosity η is modeled with a Cross-WLF model and the pVT relation is modeled with a two-domain modified Tait model. The material for all simulations is a polypropylene with 30 wt% glass fiber reinforcement by *Celanese* with trade name *Factor PP GF 30* and all material parameters are taken from the *Autodesk Moldflow 2023* material database. The flow rate $\dot{V}$ at the injection gate $\partial\Omega_\mathrm{i}$ is determined automatically for each geometry by Autodesk Moldflow with a quick strip analysis in order to yield a low injection pressure. The injection temperature of the melt is set to 270 °C. We employ no-slip boundary conditions for the velocity at the mold walls Ω_w and the mold surface temperature is set to 45 °C. Volumetric shrinkage S_vol is defined as the local density increase from the packing phase to ambient conditions, the cooling time t_cool denotes the amount of time taken to cool the molten plastic down to a defined ejection temperature of 100 °C. Both quantities are derived from the solutions of Eqs. (1) to (3).

The fiber orientation state is described via a second order fiber orientation tensor [27], which is the second moment of the fiber orientation distribution function $\psi(\mathbf{p})$ defined as

$$\mathbf{A} = \int_S \mathbf{p} \otimes \mathbf{p}\ \psi(\mathbf{p})\ \mathrm{d}A, \tag{4}$$

where $\mathbf{p}$ is a fiber direction and $\int_S \cdot \mathrm{d}A$ describes integration over the surface of a unit sphere. The reorientation is computed with an evolution equation for the second order fiber orientation tensor via the ARD-RSC model [28] using velocity gradients from the solution of Eqs. (1) to (3) without flow-fiber coupling. After each computation, results are automatically exported as *XML* files and postprocessed to store the mesh as well as nodal values of fill time, cooling time, volumetric shrinkage and fiber orientation in a neutral *VTK* file. The term *fill time* refers to the fill time field, i.e. the time when the melt front reaches a point in the domain, and not to the single value describing the time to fill the entire domain. We made the dataset publicly available.[1]

3. Fast approximation method

Our proposed workflow (see Fig. 1) assumes inputs in the form of a tetragonally meshed domain Ω representing the component to be molded, injection locations $\partial\Omega_\mathrm{i}$ and a polymer melt flow rate $\dot{V}$ at the injection gates. It should predict the nodal values of fill time t_fill, cooling time t_cool, volumetric shrinkage S_vol, and fiber orientation states $\mathbf{A}$ as outputs.

The prediction is achieved by encoding feature tuples $\mathbf{X}$ for each node, which are then normalized to $\mathbf{X}^*$ and fed to a regression model. This regression model is trained on the injection molding simulation dataset from the previous section and will predict normalized nodal result tuples $\tilde{\mathbf{Y}}^*$ in inference mode. Finally, these results are denormalized (using process conditions and analytical solutions) to give a nodal output prediction $\tilde{\mathbf{Y}}$.

3.1. Node encoding

The meshed domain is a complex input parameter of varying size and complexity. In principle, it could be passed directly into a graph convolutional neural network, a point convolutional neural network, or a convolutional neural network after rasterization. However, these approaches likely require a lot of training data as they have to learn some basic physical observations. For example, the fill time of a node t_fill is affected by its distance to an injection gate and the orientation is influenced by the geometric shape in its neighborhood. Hence, the proposed workflow aims to encode such features for each node of the mesh in a computationally efficient way and train a regression model based on these encoded features.

3.1.1. Distance features

The distance metrics should evaluate geodesic distances through the domain Ω, i.e. a path that could be used by melt during filling. This metric may differ from a simple Euclidean distance, as illustrated in Fig. 2.

[1] https://doi.org/10.5281/zenodo.10027027.

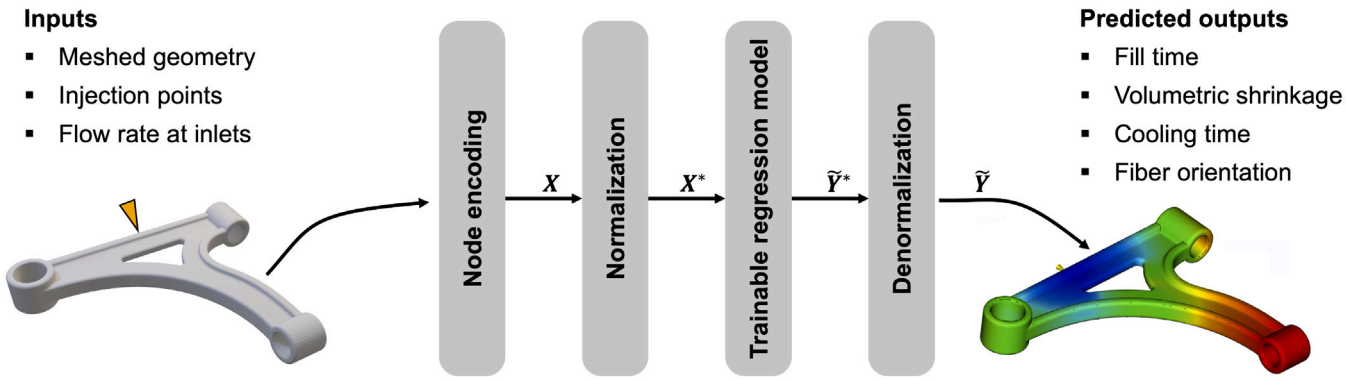

Fig. 1. Workflow for the proposed fast approximation of the injection molding process.

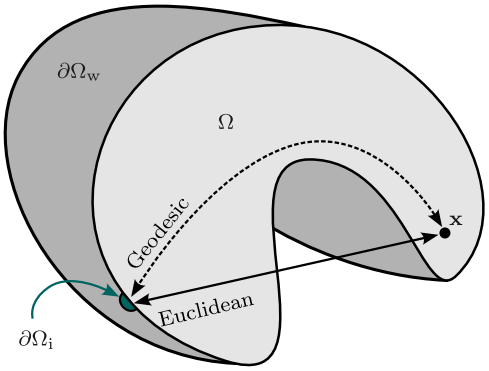

Fig. 2. Cut view of a three-dimensional domain Ω with its wall boundary Ω_w and concentrated injection boundary $\partial\Omega_\mathrm{i}$. The dashed line represents the shortest path from location $\mathbf{x}$ to the injection gate $\partial\Omega_\mathrm{i}$ within the domain and is different to the shortest Euclidean path.

We can compute such a distance metric as solution to the boundary value problem

$$\begin{aligned} \|\nabla\phi(\mathbf{x})\| &= \frac{1}{f(\mathbf{x})} \quad \mathbf{x}\in\Omega \\ \phi(\mathbf{x}) &= \phi^0 \quad \mathbf{x}\in\partial\Omega \end{aligned} \tag{5}$$

which computes the minimum travel time $\phi(\mathbf{x})$ from a point $\mathbf{x}$ in a domain Ω given the speed of travel $f(\mathbf{x})$ with an exit penalty $\phi^0(\mathbf{x})$. This partial differential equation is known as the *eikonal equation* and may be solved with the finite element method after reformulation [29] or iterative methods [30–32]. We utilize the package *fim-python* [33], which implements the tetragonal fast iterative method (tetFIM) by Fu et al. [32], to solve the eikonal boundary value problem efficiently on linear tetragonal domains with parallel computing architectures.

We solve three different configurations of Eq. (5), as summarized in Table 1. The wall distance $D_\mathrm{w}(\mathbf{x})$ and injection gate distance $D_\mathrm{i}(\mathbf{x})$ represent the shortest geodesic distance from a point $\mathbf{x}$ to the corresponding boundary. However, this injection gate distance does not account for faster travel speed of the polymer melt in wide channels and slower speed in narrow channels with higher shear rates and consequently higher flow resistance. Therefore, we use the wall distance $D_\mathrm{w}(\mathbf{x})$ as a proportional approximation of the travel speed to also compute a flow distance $D_\mathrm{f}(\mathbf{x})$, which should approximate the fill front progression through a domain. The exit penalty ϕ^0 could be used to model delayed injection gate opening sequences for multiple gates, but is set to 0 as we treat single gate only in this work.

Fig. 3 visualizes the differences of D_w, D_i and D_f using a simplified 2D example.

Table 1
Distance metrics computed with the eikonal Eq. (5).

Distance metric	ϕ	ϕ^0	f	$\partial\Omega$
Wall distance	D_w	0	1	$\partial\Omega_\mathrm{w}$
Injection distance	D_i	0	1	$\partial\Omega_\mathrm{i}$
Flow distance	D_f	0	D_w	$\partial\Omega_\mathrm{i}$

3.1.2. Spatial moment

We compute a second moment tensor of the spatial material distribution in the neighborhood of a node as

$$\mathbf{S}(\mathbf{x}_i) = \int_{\mathcal{N}_i} \frac{1}{V_i} I(\mathbf{x})\mathbf{d}_i(\mathbf{x})\otimes\mathbf{d}_i(\mathbf{x})\mathrm{d}V \approx \sum_{j\in\mathcal{N}_i} \frac{v_j}{V_i}\mathbf{d}_{ij}\otimes\mathbf{d}_{ij} \tag{6}$$

where $\mathbf{d}_i(\mathbf{x})$ denotes a unit vector pointing from $\mathbf{x}_i$ to $\mathbf{x}$, $I(\mathbf{x})$ is an indicator function that is 1 inside the part and 0 otherwise, and $V_i = \int_{\mathcal{N}_i} I(\mathbf{x})\mathrm{d}V$ denotes the total part volume in the neighborhood around $\mathbf{x}_i$. In the discrete approximation, $\mathbf{d}_{ij}$ denotes a unit vector pointing from node i located at $\mathbf{x}_i$ toward the center of cell j located at $\bar{\mathbf{x}}_j$ and v_j denotes the volume of that cell. The neighborhood of each node i is defined as

$$\mathcal{N}_i = \{j \mid \|\bar{\mathbf{x}}_j - \mathbf{x}_i\| < R \ \wedge \ |D_\mathrm{i}(\bar{\mathbf{x}}_j) - D_\mathrm{i}(\mathbf{x}_i)| < R\} \tag{7}$$

where R is a parameter that determines the size of the neighborhood. In this work, a ball tree algorithm provided by the *sciki-learn* library [34] is used for the fast evaluation of that neighborhood. The neighborhood size is set to $R = 3h_\mathrm{max}$, with h_max being the longest element edges present in the mesh. Subsequently, centers are excluded which differ in D_i with more than R, since otherwise points which are close in cartesian distance but not in geodesic distance (i.e. in spiral structures) reduce the accuracy. Essentially, the spatial moment describes the local directionality of the material distribution on a scale prescribed by R: At the center of large part, where the mold walls are further away than R, the spatial moment has an isotropic character. In a plate with a thickness less than $2R$, it will encode the transversal nature by having one smaller eigenvalue in the thickness direction and larger eigenvalues in the directions that are available for flow. In a rod with a diameter less than $2R$, it will have only one larger eigenvalue with a corresponding eigenvector pointing in the direction of the rod.

3.2. Feature normalization and preprocessing

The approximation method should handle arbitrary input geometries and injection locations. As a consequence, computed nodal features may differ by order of magnitudes between different geometries. As this has a negative impact on the ability of a regression model to generalize, normalization is applied to the nodal features before

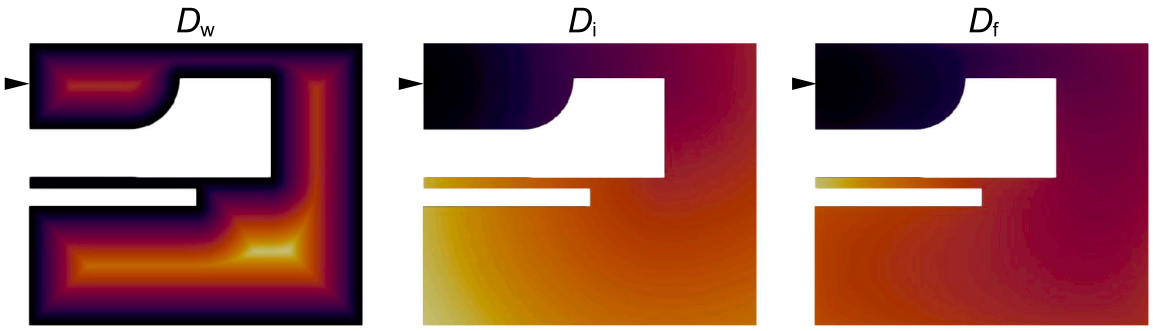

Fig. 3. Qualitative comparison of the three used distance metrics on a 2D example. While the injection distance D_i is the closest geodesic path from inlet gate to a specific node, for the flow distance D_f the wall distance D_w is used as an approximation for the travel speed of the polymer melt. Function values increase with color brightness..

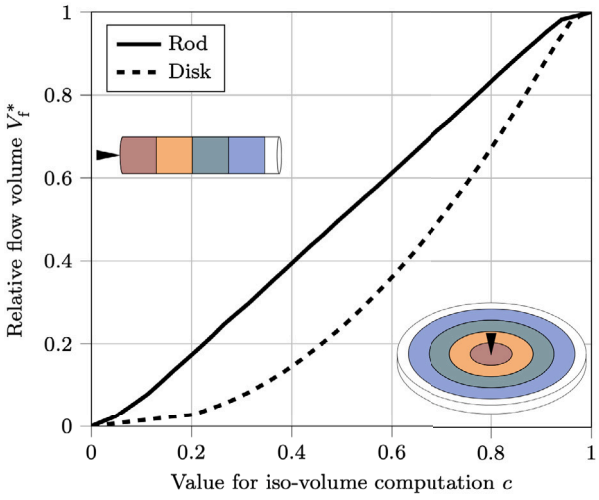

Fig. 4. Relative flow volume of a rod (solid) and a center-gated disk (dashed). The insets show isovolumes $\mathcal{I}_{D_f}$ for different c and illustrate how isovolumes increase non-linear with flow distance D_f in a disk and linear in a rod.

constructing a feature tuple for training. The spatial moment is already normalized by definition and normalized distances are computed via

$$D_w^* = \frac{D_w}{\max_\Omega D_w}, \quad D_i^* = \frac{D_i}{\max_\Omega D_i}, \quad \text{and} \quad D_f^* = \frac{D_f}{\max_\Omega D_f}, \tag{8}$$

where $\max_\Omega(\cdot)$ computes the maximum value within a domain Ω. However, the normalized flow distance D_f^* is not an ideal feature to encode melt flow propagation through the domain, because isovolumes

$$\mathcal{I}_{D_f^*}(c) = \{\mathbf{x} \in \Omega \mid D_f^*(\mathbf{x}) < c\} \tag{9}$$

for different values of $c \in [0, 1]$ are not necessarily linearly related to filled volumes during the filling process.

Consequently, we define a relative flow volume

$$V_f^*(c) = \frac{\int_{\mathcal{I}_{D_f}} \mathrm{d}V}{\int_\Omega \mathrm{d}V}, \tag{10}$$

which we expect to be a more meaningful feature for the fill front progress within a component in comparison to distance only. This can easily be understood by comparing a straight rod to a center-gated disk, as illustrated in Fig. 4: If the rod is subjected to a constant flow rate $\dot{V}$, the filled volume increases linearly with distance and the relative flow volume. In a center gated disk, the filled volume increases quadratically with distance for a constant flow rate at the gate. The relative flow volume accounts for this behavior and is thus a nodal feature that supports better generalization.

Further, we use gradients of the computed properties as features and introduce the notation

$$\nabla^*(\bullet) = \frac{\nabla \bullet}{\|\nabla \bullet\|} \tag{11}$$

for normalized gradients, i.e. the direction of a gradient.

3.3. Machine learning

The normalized features are computed for each geometry in the dataset reusing the Autodesk Moldflow mesh. They are summarized in a tuple

$$\mathbf{X}^* = \left(V_f^*, \nabla^* V_f^*, D_i^*, \nabla^* D_i^*, D_w, D_w^*, \nabla^* D_w^*, \mathbf{S}\right)^\top \tag{12}$$

for each node. Ideally, this fingerprint allows to infer some approximate information about the normalized injection molding result at the corresponding node, namely the tuple

$$\mathbf{Y}^* = \left(t_{\text{fill}}^*, t_{\text{cool}}^*, S_{\text{vol}}^*, \mathbf{A}\right)^\top. \tag{13}$$

In total, the dataset consists of 19 million pairs $\mathbf{X}^*$ and $\mathbf{Y}^*$ grouped in 629 geometries. The regression task is subsequently attempted with three different models: a naive benchmark model, a feed forward neural network, and a gradient boosting regression model.

3.3.1. Naive reference model

The naive model states that the predicted relative fill time t_{fill}^* is identical to the relative flow volume V_f^*, as this feature is engineered to be a meaningful predictor for the fill front progress. Cooling time is predicted using an approximated solution of a 1D heat diffusion equation (see Eq. (23)), normalized using the resulting t_{cool} at the point of maximum wall thickness D_w. Volumetric shrinkage is approximated as the increase in density of the material from injection conditions to ambient conditions. This value is also used for denormalization (see Section 3.4), therefore S_{vol}^* is 1 for all points in the naive model by definition. In addition, the naive prediction model adopts Ospald's [24] estimation of the orientation tensor

$$\mathbf{A} = \frac{\beta}{3}\mathbf{I} + (1-\beta)\nabla^* D_i \otimes \nabla^* D_i \tag{14}$$

with an adjustable parameter $\beta \in [0, 1]$ for the amount of anisotropy.

3.3.2. Feed forward neural network

The relation between input features $\mathbf{X}^*$ and outputs $\mathbf{Y}^*$ may be modeled with a simple feed forward neural network (FNN). In an FNN, inputs are processed by consecutive layers of artificial neurons that perform the operation

$$\mathbf{x}_{i+1} = f_i\left(\mathbf{x}_i \mathbf{W}_i^\top + \mathbf{b}_i\right) \tag{15}$$

with weight matrices $\mathbf{W}_i$, biases $\mathbf{b}_i$, and an activation function f_i. The weights and biases of each neuron are adjusted to minimize the error on a training set of pairs $(\mathbf{X}^*, \mathbf{Y}^*)$ employing the backpropagation algorithm. This is done by minimizing the mean squared error on the training set employing the Adam optimizer with learning rate 0.001 in batches of 1024 node pairs with *PyTorch*. Different sets of hyperparameters (hidden neurons, hidden layers and activation functions) are tested to yield a feasible balance between training effort and accuracy.

3.3.3. Gradient boosting

Gradient boosted regression trees (GBRT) are an alternative model for the regression task that maps from $\mathbf{X}^*$ to $\mathbf{Y}^*$. The basic idea of this machine learning model is a consecutive application of weak decision trees, in which each newly added tree improves the accuracy. Formally, this prediction model computes

$$\mathbf{Y}^* = \sum_i^k \nu_i F_i(\mathbf{X}) \tag{16}$$

with k decision trees $F_i \in \mathcal{F}$. The model is trained in an additive training process to minimize the mean squared residual of each added decision tree employing the *XGBoost* library. The factors ν_i depend on the learning rate and control the contribution of each tree to the final prediction. Together with the limited tree depth of individual trees, this prevents overfitting by limiting the complexity and contribution of individual trees.

3.4. Denormalization of output variables

All approximation models return normalized values for all output variables by design. In order to receive meaningful results, a denormalization or rescaling is needed for fill time, cooling time and volumetric shrinkage. The fiber-orientation tensor is normalized by definition and does not need to be changed.

We denormalize the fill time t_{fill} of each geometry according to

$$t_{\text{fill}} = \frac{V}{\dot{V}} t^*_{\text{fill}} \tag{17}$$

using the total volume V and the applied flow rate $\dot{V}$. We denormalize the volumetric shrinkage using the maximum theoretical shrinkage for the material as

$$S_{\text{vol}} = \tilde{S}_{\text{vol}} S^*_{\text{vol}} \tag{18}$$

with $\tilde{S}_{\text{vol}}$ being the density increase from molten state at high pressure to ambient conditions taken from pvT data. Cooling times are denormalized using an approximation at the point of maximum wall distance $\max_\Omega D_{\text{w}}$.

$$t_{\text{cool}} = \tilde{t}_{\text{cool}} t^*_{\text{cool}} \tag{19}$$

Consequently, we need a rapid approximation of $\tilde{t}_{\text{cool}}$ to rescale the cooling times to meaningful values. We use a spectral decomposition of the spatial moment $\mathbf{S}$ at the thickest point, i.e. at $\max_\Omega D_{\text{w}}$, to estimate the geometric shape as a position on a triangle representing feasible neighborhood shapes in the eigenbasis (see Fig. 5). Then, we can interpolate analytic solutions of the heat diffusion equation in 1D, 2D and 3D to approximate

$$\tilde{t}_{\text{cool}} = \frac{a w_a(r) t_{\text{cool,2D}} + b w_b(r) t_{\text{cool,1D}} + c w_c(r) t_{\text{cool,3D}}}{a w_a(r) + b w_b(r) + c w_c(r)}. \tag{20}$$

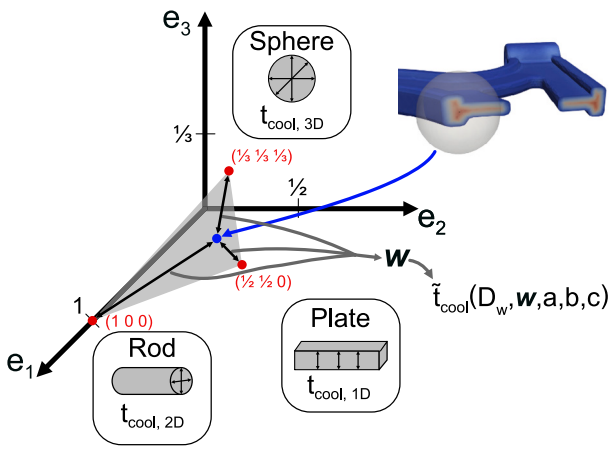

Fig. 5. Illustration of calculation of $\tilde{t}_{\text{cool}}$.

Here, $t_{\text{cool,nD}}$ are the analytic estimations as explained below, $w_x(r)$ are normalized distances in barycentric coordinates to the 1D, 2D, 3D reference cases and a, b, c are parameters fitted on the data set. The radius r for evaluating the spatial moment is defined as

$$r = r_f \cdot \max_\Omega D_{\text{w}}. \tag{21}$$

with r_f also being fitted to the dataset. The fitting process is detailed in Appendix B.

In a perfect sphere, the heat from the center point will diffuse perfectly symmetrical in all directions. The temperature at this point with respect to time can be estimated from the diffusion equation in 3D spherical coordinates. Solving the partial differential equation and solving for t needed to reach the ejection temperature T_{end} with a constant mold temperature T_{mold} and a starting temperature T_{start} leads to

$$t_{\text{cool, 3D}} \approx -\frac{D_{\text{w}}^2}{\alpha \pi^2} \ln\left(\frac{1}{2}\frac{T_{\text{end}} - T_{\text{mold}}}{T_{\text{start}} - T_{\text{mold}}}\right) \tag{22}$$

with α being the thermal diffusivity which is calculated from the density and temperature-averaged values of the thermal conductivity and the specific heat capacity of the used plastic. In contrast, a point in the middle of an infinite plane sheet is surrounded by points of the same temperature in two directions and can therefore be approximated using the heat diffusion equation in 1D, leading to

$$t_{\text{cool, 1D}} \approx -\frac{4 D_{\text{w}}^2}{\alpha \pi^2} \ln\left(\frac{\pi}{4}\frac{T_{\text{end}} - T_{\text{mold}}}{T_{\text{start}} - T_{\text{mold}}}\right). \tag{23}$$

A point in the middle of a long cylinder can distribute heat in a plane and can be approximated using the solution of the 2D heat equation

$$t_{\text{cool, 2D}} \approx -\frac{D_{\text{w}}^2}{5.783 \alpha} \ln\left(\frac{1}{1.602}\frac{T_{\text{end}} - T_{\text{mold}}}{T_{\text{start}} - T_{\text{mold}}}\right). \tag{24}$$

More information regarding the derivation of Eqs. (22)–(24) can be found in Appendix A.

4. Results and discussion

First, we evaluate the prediction accuracy on test sets sampled from the generated ABC injection molding simulations. Subsequently, we illustrate inference quality, generalization potential, and computational performance gains on a generic application example.

Different hyperparameters were tested for the machine learning models and the performance metrics are detailed in Appendix C. For the following results, we use an FNN with 100 neurons per hidden layer, ReLu activation and four layers as well as a GBRT model with 100 trees

of maximum depth of $k = 8$ and a learning rate of 0.1. These hyperparameters represent a balance between accuracy and computational performance.

4.1. Evaluation of prediction errors on ABC dataset

We created five different random samplings, each splitting the generated dataset of 629 geometries into a test set with 315 geometries and a training set containing the remaining geometries. We trained the models (Naive, FNN, GBRT) on each training set and evaluate the prediction for each node of each geometry in the corresponding test set. Then, we compute the relative root mean squared error (RRMSE) between predictions $\tilde{\mathbf{Y}}$ and simulation results $\mathbf{Y}$ for each geometry of each test set. Error curves for each predicted variable are averaged over all five test sets, only plotting the mean curves due to low variance. For performance of the models on different size ratios of training set and test set, see Appendix D.

4.1.1. Fill time

Fig. 6 shows the sorted distribution of fill time errors averaged over all test sets. It becomes apparent that the machine learning models perform only slightly better than the naive approximation of fill time. This is expected as the feature V_f^* was specifically engineered to describe the fill front progression and thus forms an excellent predictor for the fill time. Additional features, which may be used by the learned models, do not improve the prediction significantly. Increasing the number of learnable parameters and tuning the hyperparameters in both machine learning models does not notably outperform the naive estimator (see Appendix C).

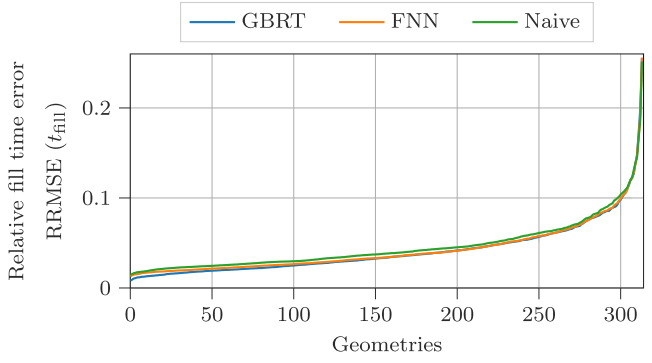

Fig. 6. Sorted distribution of fill time errors on test sets averaged over five different sets.

While the majority of RRMSE values (300 out of 314) are well below 0.1, a few geometries feature quite significant deviations between approximation and injection molding simulation. The four geometries with the greatest prediction error are visualized in Fig. 10. Significant approximation errors mainly occur if there are considerable jumps in wall thicknesses. In this case, the assumed travel speed in the flow distance computation is an oversimplification of the actual physical balance laws. However, in most cases, all models allow a quite accurate prediction of the filling process without solving the full transient PDE system stated in Eqs. (1)–(3). For fill time prediction, comparable errors have been shown in literature before, with Uglov et al. [25] reporting absolute RMSE values of 0.357 s with a total fill time of about 9 s, being an RRMSE value of 0.04. This matches the median prediction accuracy of the model presented here, however their prediction is limited to a very specific task (automotive dashboard).

4.1.2. Orientation

Fig. 7 shows the sorted distribution of orientation errors averaged over all test sets. The naive model performs best for $\alpha = 0.85$. The machine learning models outperform the naive benchmark noticeably.

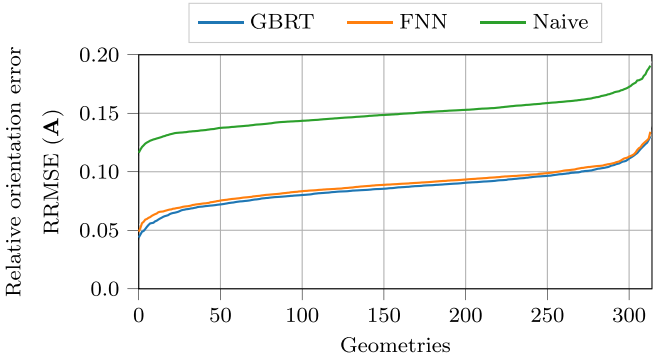

Fig. 7. Sorted distribution of orientation error on test sets averaged over five different sets.

In comparison to fill time prediction, the distribution is less skewed and error levels overall are higher.

The worst predictions occur when the fill front is moving orthogonal to the spatial moment of the part, since the fiber orientation prediction is strongly correlated with this feature. This is especially noticeable in Fig. 11(c), (f). Due to the inlet position, the fill front progresses in y-direction through the part, whereas the spatial moment assumes a structural directionality in z-direction. Other inaccuracies can happen, if the filtering mechanism during the evaluation of the neighborhood fails due to similar values of D_i, e.g. in a spring which is injected in the middle. In that case, close to the injection location, the spatial moment computation takes into account nodes that are not relevant for the local direction, and consequently yields wrong predictions. This could be mitigated by a geodesic measure to define the neighborhood $\mathcal{N}$, which is computationally expensive. Alternatively, a more sophisticated adaptive method for calculating the radius R for an individual geometry would be needed.

4.1.3. Cooling time

Fig. 8 shows the sorted distribution of cooling time errors for the three models. For estimating the performance of the machine learning models decoupled from additional uncertainties from the calculation of $\tilde{t}_{cool}$, Fig. 8 also shows the error distribution when using the denormalization value as extracted from the original numerical solution $\tilde{t}_{cool,\ exact}$.

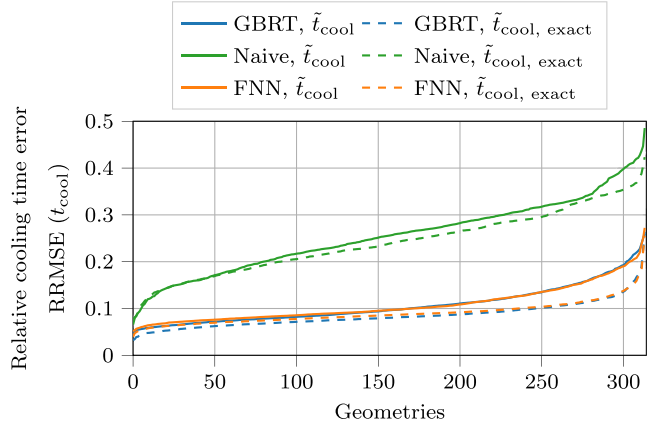

Fig. 8. Sorted distribution of cooling time errors on test sets averaged over five different sets.

The plots are averaged over all test sets. Naturally, those using the additional approximation show higher errors than their counterparts, with the GBRT model performing slightly better than the FNN. However, both machine learning models show median errors of 0.1. The

Naive reference model using the 1D approximation performs considerably worse in contrast.

Fig. 12 shows representative geometries with the highest errors for both denormalization methods. The calculation for $\tilde{t}_{\text{cool}}$ is most inaccurate for parts where the calculation of the spatial moment has similar problems as explained before, i.e. points being taken into account which are close in cartesian but far in geodesic distance. For example, a long, thick-walled pipe is approximated as a cylinder (Fig. 12 b,e). Large errors when denormalizing with $\tilde{t}_{\text{cool, exact}}$ mostly occur when the point of maximum wall distance is located close to the injection gate. Longer cooling times in this regime lead to an overestimation in the whole part. Only one part (see Fig. 13) shows high errors for both variants of denormalization. This can be explained by it being the smallest part in the dataset with a size of $1 \times 1 \times 0.5$ mm. We suspect that for such a small volume, the hot region around the injection gate is as large as the part itself. This is unique to this single geometry instance and thus not properly learned during training.

4.1.4. Volumetric shrinkage

Fig. 9 shows the sorted distribution of errors for the volumetric shrinkage of the three models averaged over all test sets. Being a constant value, the naive model exhibits very high errors, whereas both machine learning models show errors mostly below 0.18.

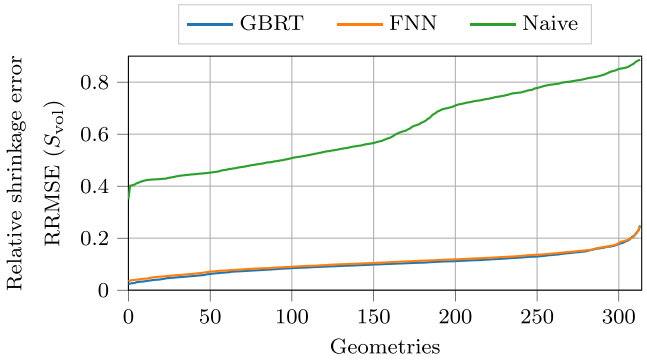

Fig. 9. Sorted distribution of volumetric shrinkage errors on test sets averaged over five different sets.

Fig. 14 shows the three parts with highest errors, which do not share explicit characteristics. This is reflected in the error curves, which do not show a strong increase for the last structures like for other variables. While rectangular slabs are approximated best, no clear trend in the worse approximated structures is found.

4.2. Generic application example

The approximation workflow is applied to a generic geometry, which is not contained in the ABC dataset, to demonstrate the usage phase after training and to analyze the computational performance in inference mode.

4.2.1. Prediction accuracy

A quantitative evaluation of RRMSEs for the entire geometry is given in Table 2. Fig. 15 compares the predicted fields of the FNN and GBRT model with a full numerical simulation.

For the mold filling process, it can be seen that both approximation methods are in accordance with the simulation, only showing a slight delay in the middle of the part and a small overestimation on one end of the part. Regarding the fiber orientation, the predictions correctly determine regions with predominant fiber orientation in the x-direction. However, the degree of orientation is underestimated when compared to the simulated result. The general distribution of cooling times is well approximated by both models, however the longer times in thicker regions of the part are underestimated. Areas of longer cooling are predicted thinner than in the simulation by both models and the approximation for $\tilde{t}_{\text{cool}}$ underestimates the actual denormalization value. Both models overestimate the maximum volumetric shrinkage by up to 1%, but capture the overall distribution correctly including the difference around the injection location. The FNN model additionally introduces some variation in the planar areas of the part.

Table 2
RRMSEs of the machine learning models on the generic control structure.

ML model	Fill time	Orientation	Cooling time	Shrinkage
GB	0.042	0.105	0.116	0.037
FNN	0.044	0.105	0.108	0.043

4.2.2. Computational performance

The generic application example is meshed with 117,357 nodes and 629,846 linear tetrahedral elements. The comparison was performed on an AMD Ryzen Threadripper PRO 5995WX workstation with 64 cores, 512 GB RAM and an NVIDIA RTX4090 GPU, respective computational times are given in Table 3. On this machine, meshing with Autodesk Moldflow 2023 took 62 s and the injection molding simulation required additional 1647 s. Training of the GBRT ($k = 100$) model and FNN ($n = 100$) takes 165 s and 688 s, respectively. Once meshed, computation of the distance measures on CPU takes approximately 140 s, split into 12 s for solver initialization, 44 s for D_{w}, 45 s for D_{i} and 39 s for D_{f}. This can be accelerated significantly with a GPU down to 20 s (12, 1.6, 3.3 and 2.8 s). Computing the spatial moment takes about 4 s, which is caused in equal parts by the neighborhood search utilizing a ball tree algorithm and computing the tensor products from Eq. (6). Finally, evaluation of the ML models based on the computed features adds negligible computational time to the approximation.

Table 3
Computational performance of the full numerical calculation against the proposed approximation model on generic application example with 117,357 nodes and 629,846 linear tetrahedral elements.

Software	Task	Full simulation	Approximation
Moldflow	Meshing	62 s	62 s
Moldflow	PDE solver	1647 s	–
tetFIM	Initialization	–	12 s
tetFIM	D_{i}	–	3.3 s
tetFIM	D_{w}	–	1.6 s
tetFIM	D_{f}	–	2.8 s
SciPy, NumPy	Spatial moment	–	4 s
		1709 s	**85.7 s**

For a single evaluation, these results indicate a significant acceleration to 5% of the full simulation time, which might be further enhanced with more efficient meshing or if a mesh is already available. However, the real benefit comes from repeated evaluation, e.g. if the injection location should be optimized. In this scenario, only D_{i} and D_{f} need to be recomputed, which would take 6 s while a full new simulation with a different injection gate would take 1650 s - a 275-fold acceleration.

4.2.3. Dependency on meshsize

The generic geometry is also used to investigate the influence of mesh density on the approximation workflow. Fig. 16 shows errors of different mesh densities for Moldflow simulations and the three fast approximation models (Naive, FNN, GBRT) without mesh refinement around the injection location. The errors are relative to the simulation with the finest mesh, i.e. an edgelength of 2 mm. While the numerical simulation shows a decrease in accuracy with a coarser mesh for all variables, the approximation errors are relatively constant. The full numerical solution is heavily dependent on the underlying mesh and requires sufficient refinement at regions with large velocity gradients, whereas for the approximation models only the calculation of geodesic distances is influenced by the mesh size. The approximation for $\tilde{t}_{\text{cool}}$ may also be influenced in the computation of the spatial moment, explaining the small variation in accuracy. For volumetric shrinkage,

the approximation methods perform even better than the numerical simulation with lower resolution, but also loose accuracy at a similar rate. This can be explained with the representation of the thicker area surrounding all edges of the part, which becomes worse for lower resolution meshes.

The ability to use coarser meshes makes the proposed approximation method even more suitable for optimization tasks. Computational effort for both meshing and calculation of distances scale linearly with the amount of nodes in the mesh, leading to even faster evaluation.

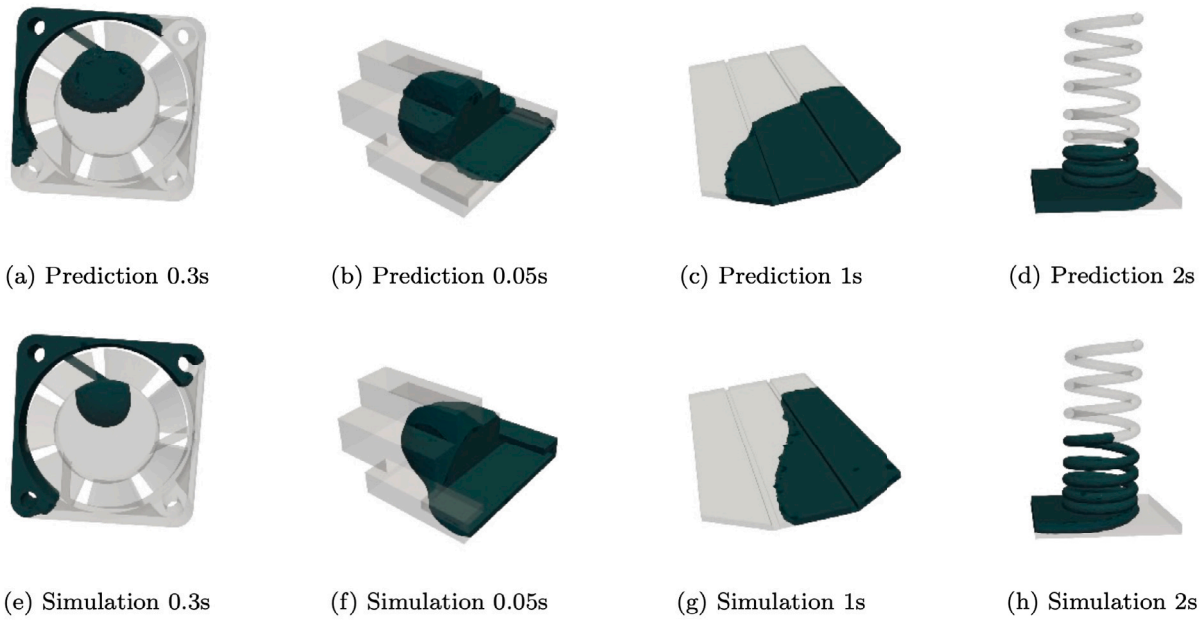

Fig. 10. Worst four geometries with highest fill time RRMSE. The visualizations show isovolumes for the indicated time threshold representing the fill front at this time.

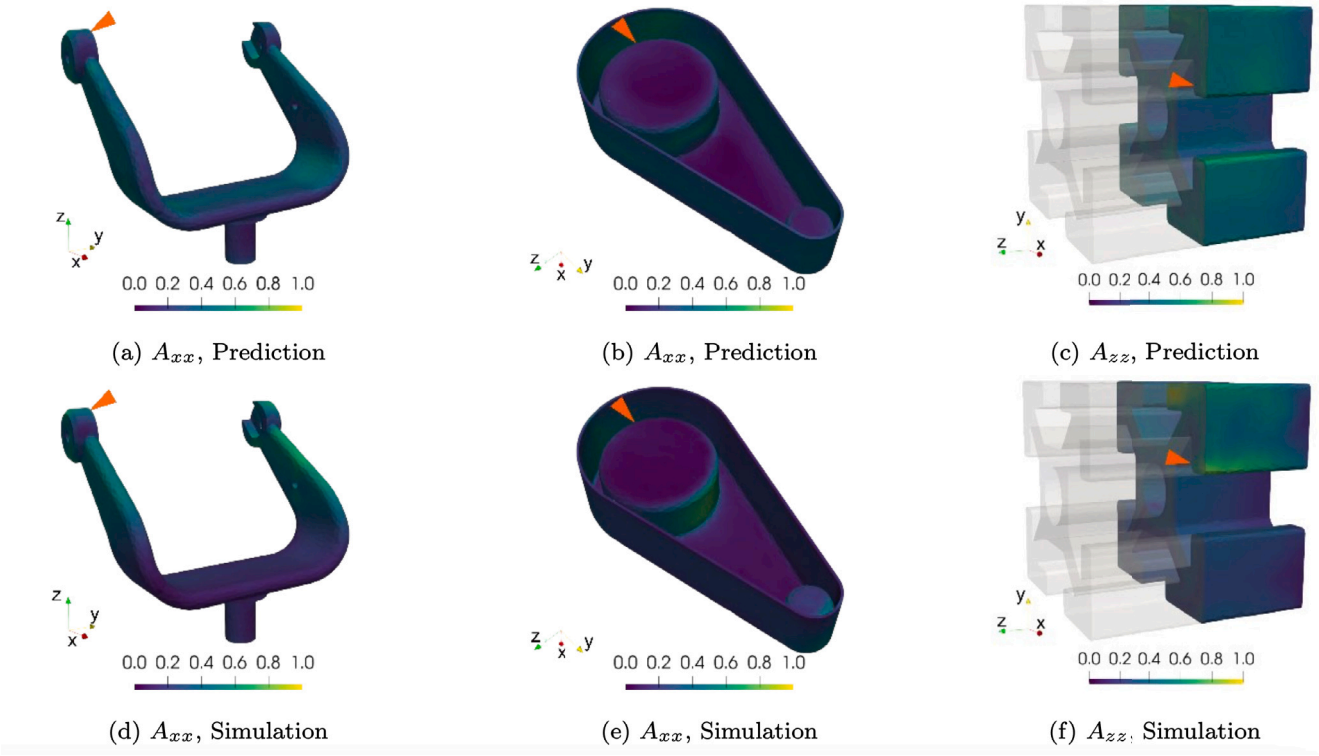

Fig. 11. Worst three geometries with highest orientation RRMSE. Folded structures lead to errors in the spatial moment computation, further errors occur if the flowfront is orthogonal to the calculated spatial moment.

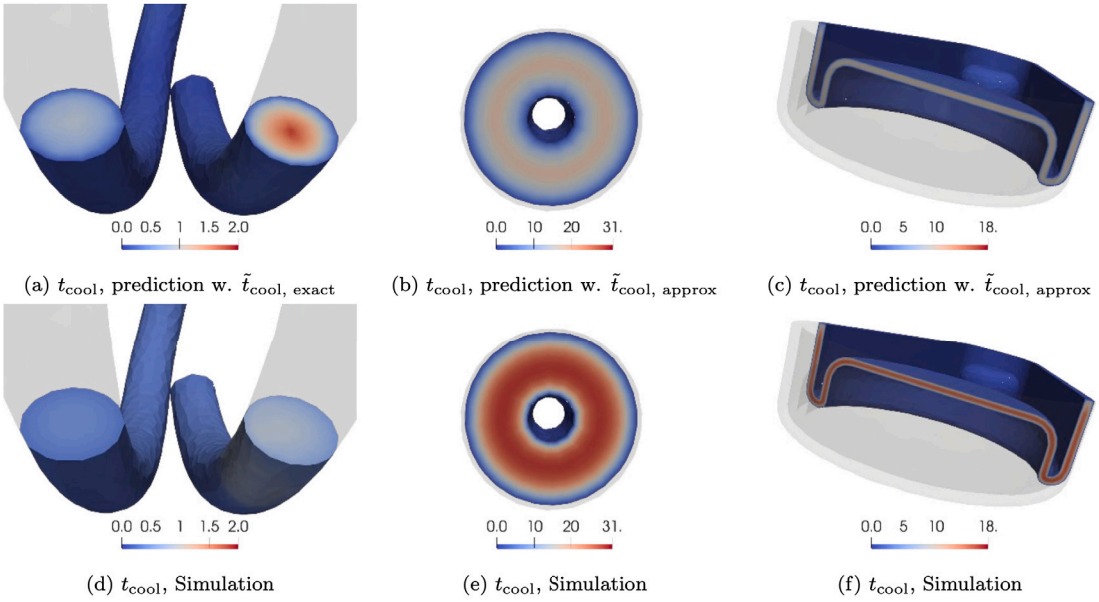

Fig. 12. Geometries with highest errors for the approximated cooling time. If the point of maximum wall thickness is located in proximity to the injection location, cooling times are overestimated when denormalized with $\tilde{t}_{\text{cool, exact}}$ (a, d). The approximation tends to underestimate the cooling time when the part resembles the 1D cooling scenario but is classified otherwise by the spatial moment (b,c,e,f).

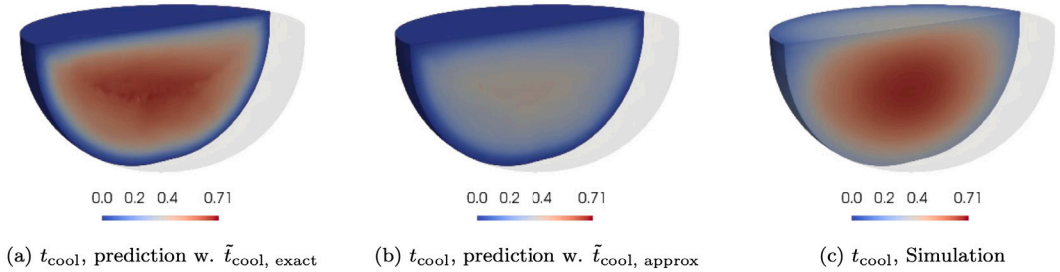

Fig. 13. Geometry with high cooling time errors for both rescaling methods.

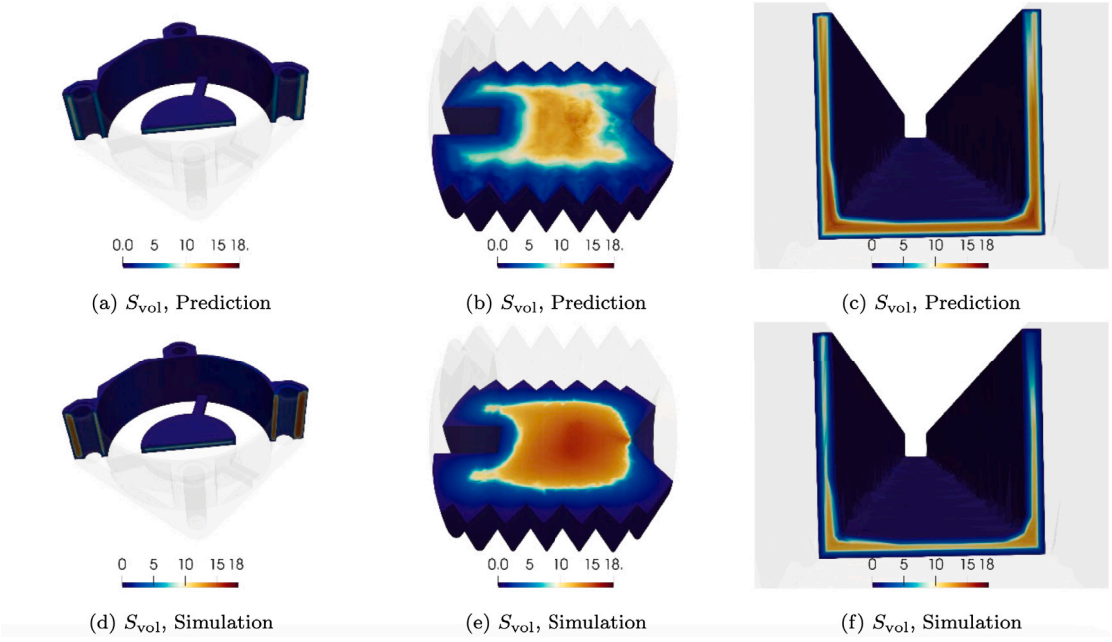

Fig. 14. Three geometries with highest RRMSE for volumetric shrinkage.

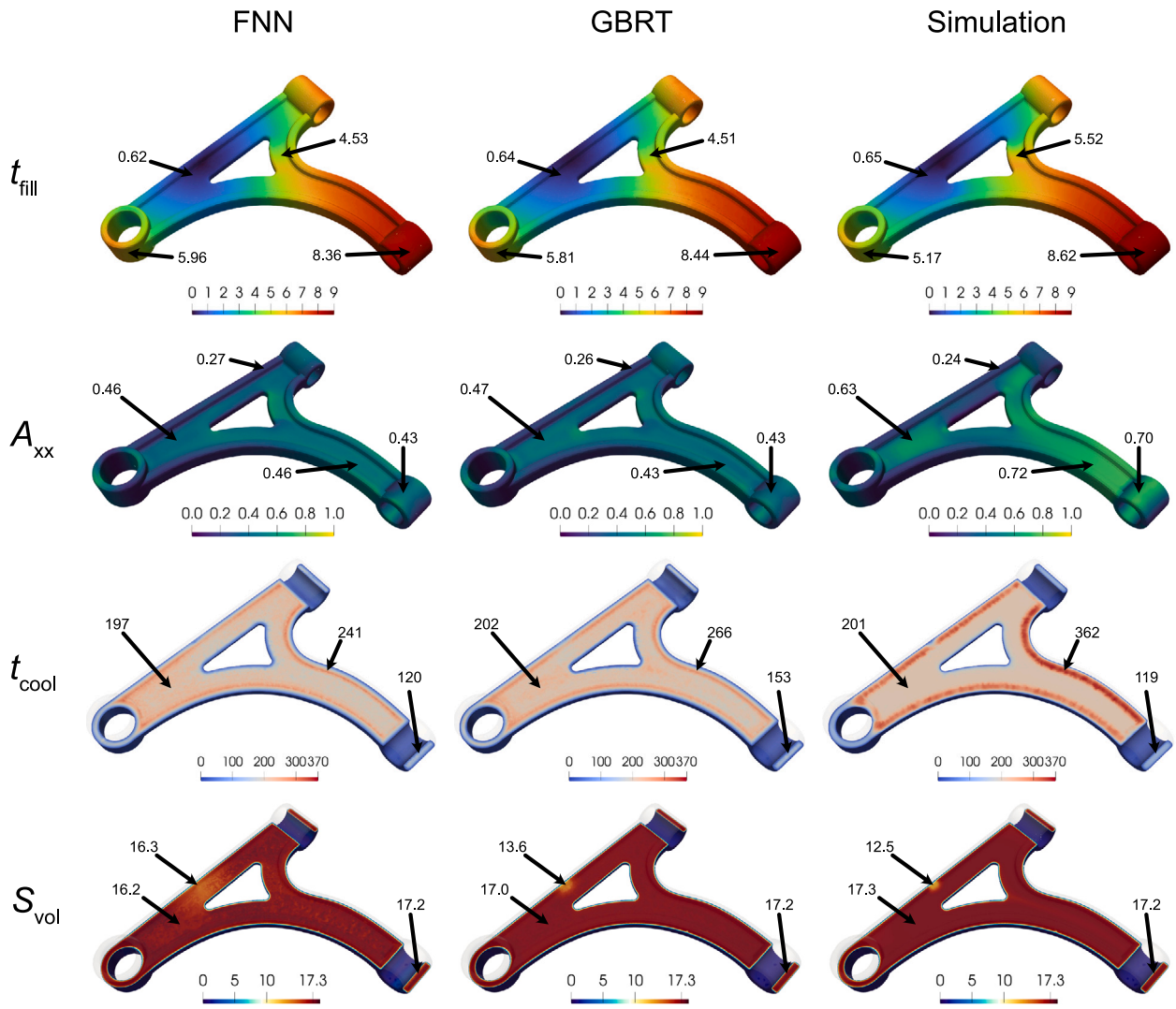

Fig. 15. Comparison of approximation methods and full numerical simulation on the generic reference geometry along with specific nodal values. Fill time and cooling time given in seconds, volumetric shrinkage given in percent.

5. Conclusion

In this work, we generate a dataset of 3D injection molding simulations, computing several variables of interest in 629 geometries sampled from the ABC geometry dataset. This dataset is publicly available to accelerate the development of data-driven process models for injection molding in the community.

We propose a fast approximation method, which uses nodal features based on the eikonal equation to encode distance metrics and a spatial moment tensor to encode the local geometry around each node. The distance based features are engineered such that they resemble the flow front progression through the domain accounting for different flow velocities and the flow rate at the injection gate. We train a feed forward neural network and a gradient boosted regression tree model to rapidly predict fill time, fiber orientation, cooling time, and volumetric shrinkage at each node in a normalized form. The predictions are denormalized, where we introduce an additional approximation of cooling time based on analytic solutions of the heat diffusion equation.

The models are tested on a test set of geometries extracted from the ABC dataset for a critical statistical evaluation of the relative root mean squared errors. The machine learning models greatly outperform naive benchmark models for all quantities but fill time, where the underlying distance feature is already a good predictor. For a generic application example, we are able to predict fill times and shrinkage with approximately 4% error as well as fiber orientations and cooling times with approximately 10% error, while being 20 times faster than a conventional injection molding simulation. The errors are computed with respect to state of the art injection molding simulations, which may feature errors in a similar range. However, opposed to full simulations, errors of the approximation models are barely influenced by the resolution of the underlying mesh. This opens the possibility to use coarser meshes opposed to what is needed for simulating the injection molding process. In a next step, the fast prediction models should be combined with a differentiable warpage simulation and trained with real world data to potentially learn additional effects which are not included in numerical models yet.

The computational speedup is further increased, if the mesh can be reused, i.e. in optimization tasks. We envision to replace expensive numerical simulations in such optimization tasks with the proposed fast approximation model such that conventional injection molding simulations are only needed to verify the final iteration.

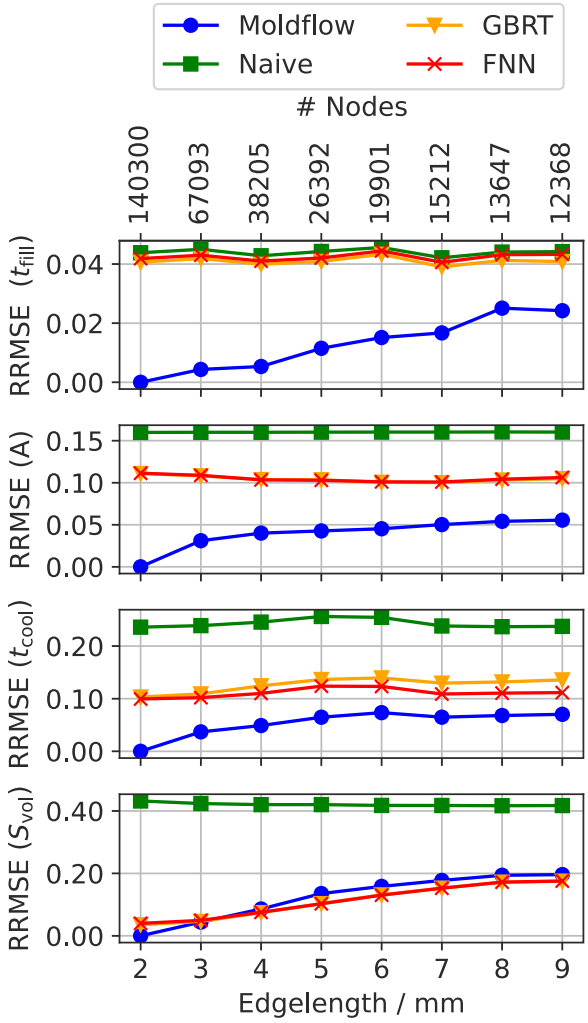

Fig. 16. RRMSEs of full numerical simulation and approximation models depending on the edgelength of the underlying mesh, investigated on the generic validation sample.

CRediT authorship contribution statement

Julian Greif: Writing – original draft, Software, Methodology, Investigation. **Philipp Lechner:** Writing – review & editing. **Nils Meyer:** Writing – original draft, Supervision, Software, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The research documented in this manuscript has been supported by the Hightech Agenda Bavaria. The support by the Bavarian State Government is gratefully acknowledged.

Appendix A. Derivation of cooling time approximations

This derivation is adopted from Mehrer [35]: We need to find the solution to the boundary-value problem

$$\frac{\partial u(x,t)}{\partial t} = \alpha \frac{\partial^2 u(x,t)}{\partial t^2} \tag{A.1}$$

with the following initial conditions

$$u(0 < x < 2D_w, t = 0) = T_{start} - T_{mold} \tag{A.2}$$

and the boundary conditions

$$u(x = 0, t) = 0 \tag{A.3}$$

$$u(x = 2D_w, t) = 0. \tag{A.4}$$

Separation of variables leads to two non-connected solutions which are recombined:

$$T(x,t) = X(x)T(t) \tag{A.5}$$

$$\frac{1}{\alpha T(t)}\frac{\mathrm{d}T(t)}{\mathrm{d}t} = \frac{1}{X(x)}\frac{\mathrm{d}X^2(x)}{\mathrm{d}x^2} = -\lambda^2 \tag{A.6}$$

$$T(t) = T_0 e^{-\alpha\lambda^2 t} \qquad X(x) = a\sin(\lambda x) + b\cos(\lambda x) \tag{A.7}$$

$$\begin{aligned} T(x,t) &= (a\sin(\lambda x) + b\cos(\lambda x)) \cdot T_0 e^{-\alpha\lambda^2 t} \\ &= (A\sin(\lambda x) + B\cos(\lambda x)) \cdot e^{-\alpha\lambda^2 t} \end{aligned} \tag{A.8}$$

All linear combinations are also possible solutions to the equation, i.e.

$$T(x,t) = \sum_{n=1}^{\infty} A_n \sin(\lambda x) + B_n \cos(\lambda x) \cdot e^{\left(-\alpha\lambda_n^2 t\right)}. \tag{A.9}$$

Eq. (A.3) leads to $B_n = 0$, Eq. (A.4) leads to:

$$\begin{aligned} T(2D_w, t) &= \sum_{n=1}^{\infty} A_n \sin\left(\lambda 2D_w\right) \cdot e^{\left(-\alpha\lambda_n^2 t\right)} = 0 \\ &\rightarrow \sum_{n=1}^{\infty} \sin\left(\lambda 2D_w\right) = 0 \\ &\rightarrow \lambda = \frac{n\pi}{2D_w} \end{aligned} \tag{A.10}$$

The starting condition (A.2) leads to

$$\sum_{n=1}^{\infty} A_n \sin\left(\frac{n\pi}{2D_w}\right) = T_{start} - T_{mold} \tag{A.11}$$

which can be solved by multiplying with $\sin(m\pi x/2D_w)$ (where $m \in \mathbb{N}$), integrating from 0 to $2D_w$ and using the orthogonality of trigonometric functions:

$$\begin{aligned} &\sum_{n=1}^{\infty} A_n \int_0^{2D_w} \sin\left(\frac{m\pi x}{2D_w}\right)\sin\left(\frac{n\pi x}{2D_w}\right)\mathrm{d}x \\ &= (T_{start} - T_{mold}) \int_0^{2D_w} \sin\left(\frac{m\pi x}{2D_w}\right)\mathrm{d}x \end{aligned} \tag{A.12}$$

Right side:

$$(T_{start} - T_{mold}) \int_0^{2D_w} \sin\left(\frac{m\pi x}{2D_w}\right)\mathrm{d}x = \begin{cases} \frac{4D_w(T_{start}-T_{mold})}{m\pi} & \text{for odd } m \\ 0 & \text{for even } m \end{cases} \tag{A.13}$$

Left side:

$$\int_0^{2D_w} \cos\left(\frac{(m-n)\pi x}{2D_w}\right) - \cos\left(\frac{(m+n)\pi x}{2D_w}\right)\mathrm{d}x = \begin{cases} D_w & \text{for odd } m \\ 0 & \text{for even } m \end{cases} \tag{A.14}$$

Therefore, it can be concluded:

$$D_w A_m = \begin{cases} \frac{4D_w(T_{start}-T_{mold})}{m\pi} & \text{for odd } m \\ 0 & \text{for even } m \end{cases} \tag{A.15}$$

With $m = 2j + 1$ and $A_m = \frac{4(T_{\text{start}} - T_{\text{mold}})}{m\pi}$, the solution to (A.1) is:

$$T(x,t) = \frac{4(T_{\text{start}} - T_{\text{mold}})}{\pi} \sum_{j=0}^{\infty} \frac{1}{2j+1} \sin\left(\frac{(2j+1)\pi x}{2D_{\text{w}}}\right) \cdot \exp\left(-\frac{\alpha(2j+1)^2\pi^2 t}{4D_{\text{w}}^2}\right) \tag{A.16}$$

For $t \gg 1$, only using $j = 0$ is a sufficient approximation:

$$T(x, t \gg 1) \approx \frac{4(T_{\text{start}} - T_{\text{mold}})}{\pi} \sin\left(\frac{\pi x}{2D_{\text{w}}}\right) \cdot \exp\left(-\frac{\alpha\pi^2 t}{4D_{\text{w}}^2}\right) \tag{A.17}$$

For approximating the time, the center point takes to reach the ejection temperature, we set $T = T_{\text{end}} - T_{\text{mold}}$ and $x = D_{\text{w}}$ and solve for t to receive

$$t_{\text{cool, 1D}} \approx -\frac{4D_{\text{w}}^2}{\alpha\pi^2} \ln\left(\frac{\pi(T_{\text{end}} - T_{\text{mold}})}{4(T_{\text{start}} - T_{\text{mold}})}\right). \tag{A.18}$$

The heat equation in 3D and 2D is solved similarly using the separation of variables in the appropriate coordinate system (spherical coordinates in 3D, cylindrical coordinates for 2D), yielding

$$T(x = D_{\text{w}}, t)_{\text{3D}} \approx 2(T_{\text{start}} - T_{\text{mold}}) \exp\left(\frac{-\alpha t\pi^2}{D_{\text{w}}^2}\right) \tag{A.19}$$

$$t_{\text{cool, 3D}} \approx -\frac{D_{\text{w}}^2}{\alpha\pi^2} \ln\left(\frac{1}{2}\frac{T_{\text{end}} - T_{\text{mold}}}{T_{\text{start}} - T_{\text{mold}}}\right) \tag{A.20}$$

and

$$\begin{aligned} T(x = D_{\text{w}}, t)_{\text{2D}} &\approx \frac{2(T_{\text{start}} - T_{\text{mold}})}{\alpha_{B,0} J_1(\alpha_{B,0})} \exp\left(\frac{-\alpha t\alpha_{B,0}^2}{D_{\text{w}}^2}\right) \\ &= 1.602(T_{\text{start}} - T_{\text{mold}}) \exp\left(\frac{-5.783\alpha t}{D_{\text{w}}^2}\right) \end{aligned} \tag{A.21}$$

$$t_{\text{cool, 2D}} \approx -\frac{D_{\text{w}}^2}{5.783\alpha} \ln\left(\frac{1}{1.602}\frac{T_{\text{end}} - T_{\text{mold}}}{T_{\text{start}} - T_{\text{mold}}}\right). \tag{A.22}$$

Here, $J_1(x)$ is the Bessel's function of first order and $\alpha_{B,0}$ is the first positive root of the Bessel's function zeroth order $J_0(x)$.

Appendix B. Fitting of cooling time approximation model

This section explains how the parameters a, b, c and r_f where determined for the approximation model of cooling times at points of maximum thickness. The minimization problem is given as

$$\min_{a,b,c,r_f} \sum_{i}^{n} \left| \frac{t_{\text{cool,pr.,}i} - t_{\text{cool,ex.,}i}}{t_{\text{cool,ex.,}i}} \right| \tag{B.1}$$

with

$$t_{\text{cool,pr.,}i} = \frac{a w_{a,i}(r_f) t_{\text{cool, 2D}} + b w_{b,i}(r_f) t_{\text{cool, 1D}} + c w_{c,i}(r_f) t_{\text{cool, 3D}}}{a w_{a,i}(r_f) + b w_{b,i}(r_f) + c w_{c,i}(r_f)} \tag{B.2}$$

and n being the amount of geometries in the dataset. Without using the relative error, thicker parts with higher cooling times would be weighted more than thin parts, which is undesirable.

For optimizing, the *L-BFGS-B* algorithm [36,37] as implemented in the Python module *SciPy* was used with an analytical Jacobian for gradient evaluation. The used loss function $\mathcal{L}$ is given as

$$\mathcal{L} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{t_{\text{cool,pr.,}i} - t_{\text{cool,ex.,}i}}{t_{\text{cool,ex.,}i}}\right)^2} \tag{B.3}$$

with the corresponding Jacobian defined as:

$$\begin{pmatrix} \frac{\partial\mathcal{L}}{\partial a} \\ \frac{\partial\mathcal{L}}{\partial b} \\ \frac{\partial\mathcal{L}}{\partial c} \\ \frac{\partial\mathcal{L}}{\partial r_f} \end{pmatrix} = \frac{1}{\sqrt{n\sum_j^n \left(\frac{t_{\text{cool,pr.,}j} - t_{\text{cool,ex.,}j}}{t_{\text{cool,ex.,}j}}\right)^2}} \cdot \begin{pmatrix} \sum_i^n \left(\frac{t_{\text{cool,pr.,}i} - t_{\text{cool,ex.,}i}}{t_{\text{cool,ex.,}i}^2}\right) w_{a,i}(r_f) t_{\text{cool,2D,}i} \\ \sum_i^n \left(\frac{t_{\text{cool,pr.,}i} - t_{\text{cool,ex.,}i}}{t_{\text{cool,ex.,}i}^2}\right) w_{b,i}(r_f) t_{\text{cool,1D,}i} \\ \sum_i^n \left(\frac{t_{\text{cool,pr.,}i} - t_{\text{cool,ex.,}i}}{t_{\text{cool,ex.,}i}^2}\right) w_{c,i}(r_f) t_{\text{cool,3D,}i} \\ \mathrm{d}w_i(r_f) \end{pmatrix}. \tag{B.4}$$

Here, $\mathrm{d}w_i(r_f)$ denotes the central difference given as

$$\mathrm{d}w_i(r_f) = \frac{t_{\text{cool,pr.,}i}(r_f + 0.1) - t_{\text{cool,pr.,}i}(r_f - 0.1)}{2 \cdot 0.1} \tag{B.5}$$

which is used because of the discontinuity of the function with respect to this variable. The variable r_f defines the radius used for tensor evaluation, where very small changes do not lead to more nodes being included in the neighborhood evaluation. The fitting was done on all geometries in the dataset for including the maximum variety of different structures and the most versatile approximation. A maximum of 25 steps were conducted starting from 15 different randomized points (a, b, c, r_f). No boundaries were imposed except for r_f, which was constrained between 1.1 and 10. The results from the best optimization are shown together with the corresponding loss value $\mathcal{L}$ in Table B.4.

Table B.4
Parameters for cooling time approximation model as well as loss-function $\mathcal{L}$ for optimization before and after optimization.

	a	b	c	r_f	$\mathcal{L}$
Starting point	4.283	8.653	7.275	8.838	412.387
Optimized	17.825	8.871	0.146	4.227	0.034

Appendix C. Machine learning errors

Tables C.5 and C.6 list the minimum, maximum and median errors for different machine learning configurations, averaged over the same five test- and training set combinations. For the FNN models no extreme differences are apparent, with some models performing slightly better than others for single output variables. For the GBRT models, it can be seen that a maximum of ten decision trees is not able to provide accurate prediction, while other configurations show similar errors.

Appendix D. Error dependence on training set size

In order to find the influence of training set size on the resulting machine learning models, the dataset was split in different ratios of training set and test set from 10%/90% to 80%/20%. Five different splits were considered per ratio with one FNN and one GBRT model being trained on each split. Tables D.7 and D.8 list minimum, median and maximum errors averaged over all five splits per ratio. It can be seen that, especially for median errors, a 50–50 split shows good results, with smaller training sets showing higher errors and larger training sets only leading to negligibly better predictions. Maximum errors alone should not be considered for performance evaluation because they are too dependent on the random decision of having difficult geometries (see Figs. 10, 11, 12, 14) in the test set.

Table C.5
Machine learning errors for different FNN configurations, averaged over five test- and training set samplings. Configurations are defined as (Nodes per layer, Amount of layers, Activation function).

RRMSE	100,3,ReLu	100,4,ReLu	100,4,tanh	100,5,ReLu	200,4,ReLu	50,4,ReLu
min(t_{fill})	0.0129	0.0130	0.0143	0.0139	0.0149	0.0124
max(t_{fill})	0.2547	0.2565	0.2564	0.2558	0.2556	0.2562
median(t_{fill})	0.0340	0.0339	0.0352	0.0347	0.0354	0.0338
min(**A**)	0.0517	0.0503	0.0524	0.0493	0.0504	0.0562
max(**A**)	0.1341	0.1354	0.1339	0.1333	0.1339	0.1359
median(**A**)	0.0907	0.0895	0.0919	0.0895	0.0878	0.0933
min(t_{cool})	0.0529	0.0516	0.0576	0.0500	0.0495	0.0552
max(t_{cool})	0.2729	0.2723	0.2721	0.2818	0.2763	0.2774
median(t_{cool})	0.0975	0.0968	0.1016	0.0962	0.0952	0.1013
min(S_{vol})	0.0388	0.0341	0.0362	0.0314	0.0316	0.0383
max(S_{vol})	0.2494	0.2466	0.2531	0.2432	0.2434	0.2517
median(S_{vol})	0.1097	0.1072	0.1127	0.1058	0.1041	0.1100

Table C.6
Machine learning errors for different GBRT configurations, averaged over five test- and training set samplings. Configurations are defined as (Amount of decision trees, Max. depth of each tree). The learning rate is equal to 0.10 for all configurations.

RRMSE	10, 8	100, 10	100, 6	100, 8	200, 8	50, 8
min(t_{fill})	0.1091	0.0090	0.0078	0.0082	0.0093	0.0082
max(t_{fill})	0.2476	0.2557	0.2549	0.2550	0.2558	0.2536
median(t_{fill})	0.1462	0.0347	0.0331	0.0338	0.0344	0.0334
min(**A**)	0.0996	0.0425	0.0457	0.0436	0.0429	0.0453
max(**A**)	0.1629	0.1296	0.1313	0.1302	0.1294	0.1314
median(**A**)	0.1293	0.0848	0.0890	0.0863	0.0847	0.0891
min(t_{cool})	0.0676	0.0442	0.0504	0.0483	0.0447	0.0516
max(t_{cool})	0.3618	0.2604	0.2610	0.2600	0.2611	0.2629
median(t_{cool})	0.1457	0.0947	0.0986	0.0962	0.0941	0.0983
min(S_{vol})	0.0925	0.0220	0.0260	0.0229	0.0225	0.0235
max(S_{vol})	0.3115	0.2507	0.2444	0.2493	0.2498	0.2486
median(S_{vol})	0.1958	0.0976	0.1042	0.1003	0.0981	0.1018

Table D.7
Minimum, median and maximum RMSEs of FNN models depending on training set size. Averaged over five models per size.

RRMSE	10-90	30-70	50-50	70-30	80-20
min(t_{fill})	0.0190	0.0143	0.0130	0.0132	0.0132
median(t_{fill})	0.0405	0.0364	0.0339	0.0350	0.0329
max(t_{fill})	0.2843	0.2840	0.2565	0.2682	0.1738
min(**A**)	0.0548	0.0512	0.0503	0.0488	0.0537
median(**A**)	0.0973	0.0914	0.0895	0.0881	0.0873
max(**A**)	0.1614	0.1338	0.1354	0.1261	0.1245
min(t_{cool})	0.0569	0.0521	0.0516	0.0491	0.0542
median(t_{cool})	0.1067	0.0989	0.0968	0.0937	0.0966
max(t_{cool})	0.3016	0.2985	0.2723	0.2986	0.2586
min(S_{vol})	0.0369	0.0337	0.0341	0.0315	0.0335
median(S_{vol})	0.1163	0.1108	0.1072	0.1032	0.1029
max(S_{vol})	0.3159	0.2577	0.2466	0.2356	0.2311

Table D.8
Minimum, median and maximum RMSEs of GB models depending on training set size. Averaged over five models per size.

RRMSE	10-90	30-70	50-50	70-30	80-20
min(t_{fill})	0.0099	0.0085	0.0082	0.0081	0.0082
median(t_{fill})	0.0387	0.0348	0.0338	0.0341	0.0320
max(t_{fill})	0.2894	0.2849	0.2550	0.2701	0.1735
min(**A**)	0.0449	0.0427	0.0436	0.0428	0.0474
median(**A**)	0.0912	0.0874	0.0863	0.0856	0.0853
max(**A**)	0.1484	0.1323	0.1302	0.1270	0.1223
min(t_{cool})	0.0426	0.0468	0.0483	0.0472	0.0501
median(t_{cool})	0.1018	0.0980	0.0962	0.0946	0.0967
max(t_{cool})	0.3122	0.3015	0.2600	0.2804	0.2513
min(S_{vol})	0.0248	0.0229	0.0229	0.0231	0.0252
median(S_{vol})	0.1068	0.1034	0.1003	0.0990	0.0966
max(S_{vol})	0.3090	0.2536	0.2493	0.2260	0.2339

References

[1] Givler RC, Crochet MJ, Pipes RB. Numerical prediction of fiber orientation in dilute suspensions. J Compos Mater 1983;17(4):330–43. http://dx.doi.org/10.1177/002199838301700404.

[2] Bernasconi A, Cosmi F, Dreossi D. Local anisotropy analysis of injection moulded fibre reinforced polymer composites. Compos Sci Technol 2008;68:2574–81. http://dx.doi.org/10.1016/j.compscitech.2008.05.022.

[3] Hieber CA, Shen SF. A finite-element/finite-difference simulation of the injection-molding filling process. J Non-Newton Fluid Mech 1980;7:1–32. http://dx.doi.org/10.1016/0377-0257(80)85012-9.

[4] Tseng H-C, Chang R-Y, Hsu C-H. Predictions of fiber concentration in injection molding simulation of fiber-reinforced composites. J Thermoplast Compos Mater 2018;31:1529–44. http://dx.doi.org/10.1177/0892705717738302.

[5] Perumal V, Gupta RK, Bhattacharya SN, Costa FS. Fiber migration in shear flow: Model predictions and experimental validation. Polym Compos 2019;40:3573–81. http://dx.doi.org/10.1002/pc.25219.

[6] Goris S, Osswald T. Process-induced fiber matrix separation in long fiber-reinforced thermoplastics. Composites A 2018;105:321–33. http://dx.doi.org/10.1016/j.compositesa.2017.11.024.

[7] Sasayama T, Sato N, Katagiri Y, Murayama Y. Particle-level simulation for the prediction of short fiber orientation in injection molding. Composites A 2020;139:106115. http://dx.doi.org/10.1016/j.compositesa.2020.106115.

[8] Kugler SK, Kech A, Cruz C, Osswald T. Fiber orientation predictions—A review of existing models. J Compos Sci 2020;4:69. http://dx.doi.org/10.3390/jcs4020069.

[9] Meyer N, Saburow O, Hohberg M, Hrymak AN, Henning F, Kärger L. Parameter identification of fiber orientation models based on direct fiber simulation with smoothed particle hydrodynamics. J Compos Sci 2020;4(2):77–96. http://dx.doi.org/10.3390/jcs4020077.

[10] Li T, Luyé J-F. Flow-fiber coupled viscosity in injection molding simulations of short fiber reinforced thermoplastics. Int Polym Process 2019;34:158–71. http://dx.doi.org/10.3139/217.3706.

[11] Wittemann F, Kärger L, Henning F. Influence of fiber breakage on flow behavior in fiber length- and orientation-dependent injection molding simulations. J Non-Newton Fluid Mech 2022;310:104950. http://dx.doi.org/10.1016/j.jnnfm.2022.104950.

[12] Karl T, Zartmann J, Dalpke S, Gatti D, Frohnapfel B, Böhlke T. Influence of flow–fiber coupling during mold-filling on the stress field in short-fiber reinforced composites. Comput Mech 2023;71:991–1013. http://dx.doi.org/10.1007/s00466-023-02277-z.

[13] Liang JZ, Ness JN. The calculation of cooling time in injection moulding. J Mater Process Technol 1996;57(1):62–4. http://dx.doi.org/10.1016/0924-0136(95)02044-6.

[14] Zarkadas DM, Xanthos M. Prediction of cooling time in injection molding by means of a simplified semianalytical equation. Adv Polym Technol 2003;22(3):188–208. http://dx.doi.org/10.1002/adv.10048.

[15] Kurtaran H, Ozcelik B, Erzurumlu T. Warpage optimization of a bus ceiling lamp base using neural network model and genetic algorithm. J Mater Process Technol 2005;169:314–9. http://dx.doi.org/10.1016/j.jmatprotec.2005.03.013.

[16] Yin F, Mao H, Hua L. A hybrid of back propagation neural network and genetic algorithm for optimization of injection molding process parameters. Mater Des 2011;32:3457–64. http://dx.doi.org/10.1016/j.matdes.2011.01.058.

[17] Xu Y, Zhang Q, Zhang W, Zhang P. Optimization of injection molding process parameters to improve the mechanical performance of polymer product against impact. Int J Adv Manuf Technol 2015;76:2199–208. http://dx.doi.org/10.1007/s00170-014-6434-y.

[18] Song Z, Liu S, Wang X, Hu Z. Optimization and prediction of volume shrinkage and warpage of injection-molded thin-walled parts based on neural network. Int J Adv Manuf Technol 2020;109(3):755–69. http://dx.doi.org/10.1007/s00170-020-05558-6.

[19] Shiroud Heidari B, Hedayati Moghaddam A, Davachi SM, Khamani S, Alihosseini A. Optimization of process parameters in plastic injection molding for minimizing the volumetric shrinkage and warpage using radial basis function (RBF) coupled with the k-fold cross validation technique. J Polym Eng 2019;39(5):481–92. http://dx.doi.org/10.1515/polyeng-2018-0359.

[20] Rosli MU, Ahmad Termizi SNA, Khor CY, Nawi MAM, Akmal Omar A, Ikman Ishak M. Simulation based optimization of thin wall injection molding parameter using response surface methodology. IOP Conf Ser Mater Sci Eng 2020;864(1):012193. http://dx.doi.org/10.1088/1757-899X/864/1/012193.

[21] Liao XP, Ruan T, Xia W, Ma JY, Li LL. Multi-objective optimization by Gaussian genetic algorithm and its application in injection modeling. In: New materials, applications and processes. Adv Mater Res 2012;Vol. 399:1672–6. http://dx.doi.org/10.4028/www.scientific.net/AMR.399-401.1672.

[22] Zhao J, Cheng G. An innovative surrogate-based searching method for reducing warpage and cycle time in injection molding. Adv Polym Technol 2016;35(3):288–97. http://dx.doi.org/10.1002/adv.21554.

[23] Luo L, Yao Y, Gao F, Zhao C. Mixed-effects Gaussian process modeling approach with application in injection molding processes. J Process Control 2018;62:37–43. http://dx.doi.org/10.1016/j.jprocont.2017.12.003.

[24] Ospald F, Herzog R. SIMP based topology optimization for injection molding of SFRPs. In: Schumacher A, Vietor T, Fiebig S, Bletzinger K-U, Maute K, editors. Advances in structural and multidisciplinary optimization. Cham: Springer International Publishing; 2018, p. 850–61. http://dx.doi.org/10.1007/978-3-319-67988-4_65.

[25] Uglov A, Nikolaev S, Belov S, Padalitsa D, Greenkina T, Biagio MS, Cacciatori FM. Surrogate modeling for injection molding processes using deep learning. Struct Multidiscip Optim 2022;65:305. http://dx.doi.org/10.1007/s00158-022-03380-0.

[26] Koch S, Matveev A, Jiang Z, Williams F, Artemov A, Burnaev E, Alexa M, Zorin D, Panozzo D. ABC: A big CAD model dataset for geometric deep learning. In: The IEEE conference on computer vision and pattern recognition. CVPR, 2019, p. 1–15. http://dx.doi.org/10.48550/arXiv.1812.06216.

[27] Advani SG, Tucker CL. The use of tensors to describe and predict fiber orientation in short fiber composites. J Rheol 1987;31(8):751–84. http://dx.doi.org/10.1122/1.549945.

[28] Phelps JH, Tucker CL. An anisotropic rotary diffusion model for fiber orientation in short- and long-fiber thermoplastics. J Non-Newton Fluid Mech 2009;156(3):165–76. http://dx.doi.org/10.1016/j.jnnfm.2008.08.002.

[29] Fares E, Schröder W. A differential equation for approximate wall distance. Internat J Numer Methods Fluids 2002;39(8):743–62. http://dx.doi.org/10.1002/fld.348.

[30] Kimmel R, Sethian JA. Computing geodesic paths on manifolds. Proc Natl Acad Sci USA 1998;95(15):8431–5. http://dx.doi.org/10.1073/pnas.95.15.8431.

[31] Zhao H. A fast sweeping method for eikonal equations. Math Comp 2004;74(250):603–27. http://dx.doi.org/10.1090/s0025-5718-04-01678-3.

[32] Fu Z, Kirby RM, Whitaker RT. A fast iterative method for solving the Eikonal equation on tetrahedral domains. SIAM J Sci Comput 2013;35(5):473–94. http://dx.doi.org/10.1137/120881956.

[33] Grandits T. A fast iterative method python package. J Open Source Softw 2021;6(66):3641. http://dx.doi.org/10.21105/joss.03641.

[34] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: Machine learning in python. J Mach Learn Res 2011;12:2825–30. http://dx.doi.org/10.48550/arXiv.1201.0490.

[35] Mehrer H. Solutions of the diffusion equation. In: Diffusion in solids: fundamentals methods, materials, diffusion-controlled processes. Berlin, Heidelberg: Springer; 2007, p. 37–53. http://dx.doi.org/10.1007/978-3-540-71488-0_3.

[36] Zhu C, Byrd RH, Lu P, Nocedal J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans Math Software 1997;23(4):550–60. http://dx.doi.org/10.1145/279232.279236.

[37] Byrd RH, Lu P, Nocedal J, Zhu C. A limited memory algorithm for bound constrained optimization. SIAM J Sci Comput 1995;16(5):1190–208. http://dx.doi.org/10.1137/0916069.