Advanced open source data formats for geometrically and physically coupled systems*

Andreas Naumann[1] Jens Saak[1,3] Stefan Sauerzapf[2] Julia Vettermann[1] Michael Beitelschmidt[2] Roland Herzog[1]

[1]Technische Universität Chemnitz, Faculty of Mathematics, 09107 Chemnitz, Germany (andreas.naumann, julia.vettermann, roland.herzog@mathematik.tu-chemnitz.de).
[2]TU Dresden, Faculty of Mechanical Science and Engineering, Institute of Solid Mechanics, Marschnerstraße 30, 01307 Dresden, Germany (stefan.sauerzapf, michael.beitelschmidt@tu-dresden.de).

[3]Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany (saak@mpi-magdeburg.mpg.de).

Abstract

Numerical calculations based on models are nowadays standard tools in all engineering disciplines. The tools, which faciliate the modeling, generally include all tasks in an engineering workflow. These tasks range from simple model descriptions to advanced visualization of results.

While the incorporation of all tasks in one tool fits neatly into a single-person scheme, it makes teamwork with shared tasks very hard. In particular, every member of the team has to use the exact same software, and every sub-task has to be available in the tool. This requirement, in turn, makes a joint development of advanced methods unnecessarily complicated. Especially the numerical analysis of problem tailored methods requires a detailed knowledge of all model ingredients.

The basis for joint workflows and teamwork are interfaces and common data formats. In this publication, we present a data format for geometrically and physically coupled systems. The data formats structure bases on the standardized format *JSON*, whereas the content is derived from a mathematical model. Finally, for presentational purposes, we present an instance of a simplified model.
Keywords: model language, software interfaces, partial differential equations

1 Introduction

Over the recent decades, numerical simulations have proven to be an indispensable tool for understanding scientific and industrial processes.

Part of the success story of numerical methods are the thorough numerical analysis and the increasing trust of the users. The users of numerical software are a very heterogeneous group. On the one end of the spectrum we find users who only concentrate on the results and runtime of the method. The quality of a solution is then assessed by comparison to experience, expectation and measurements. Deviations from the expectations are usually attributed to the model. On the opposite end we have those who have a deeper understanding of the method, and select numerical methods based on the properties of the model. This subgroup also considers numerical errors in addition to the model errors. In contrast to both, the numerical analysts consider the method itself as a research subject. They know the details and properties of the methods, and relate the efficiency of the method to the properties of the model. To them, the model error is not of interest, but serves as tolerance.

A development team for a new technical device, together with its digital twin, has to incorporate all the above user types. We aim to provide a workflow that suites all their needs and allows for maximum flexibility in the distribution of tasks among them, i. e. benefits teamwork in the best possible way. As an example we investigate the development of a thermo-elastic model of a machine.

The workflow was applied to a thermal model in Sauerzapf et al., 2020 and Vettermann et al., 2021 and a thermo-mechanical model in Naumann, Herzog, 2021 using the proposed data formats. Here, we concentrate on the interface and the key ideas. In particular, we consider the data format as an approach to extend Modelica® with functionality for coupled partial differential equations (PDEs).

The development of new machines requires, often prior to construction, a deep understanding of the resulting machine's behavior under load and other influences from the environment. Particularly the interaction between the machine components, and their influence on quantities of interests (QOI), require a thorough analysis. Nowadays, this analysis is generally carried out numerically with the help of a digital twin of the machine.

Computer aided design (CAD) representations of the machine form the basis for the numerical analysis. These CAD models represent the geometry of the machine. Depending on the specific question, the geometry can be cho-

*This work was supported by the DFG grants 174223256 – TRR 96 and 460135501 - NFDI 29/1 "MaRDI – Mathematische Forschungsdateninitiative".

sen more or less detailed.

The physical model extends the geometric model with the physical effects of interest. A physical model contains a variety of sub-models, where each sub-model describes a particular effect up to the required accuracy. For example, (thermal) loads inside a machine describe the heat input per volume, or the description of the interaction with the environment are sub-models. These examples belong to one physics, but different geometric parts. Nowadays multi physics models, which link several physics, are standard. Our use case, the thermo-elastic model of a machine tool, comprises the evolution of the heat distribution and the corresponding mechanical expansion. Therefore, a data format for the description of the physical system must be able to distinguish sub-models of the different physics.

We describe the overall physics using coupled systems of PDEs. The physical model, thus, consists of equations of different types. Among these types, there are systems of PDEs to describe the overall physics, like the evolution of the temperature field, algebraic equations for simple sources and ordinary differential equations (ODEs) to describe more complex sources. For the heat equation, these sources correspond to thermal loads inside and at the boundary. In the mechanical model, the sources correspond to forces acting inside a volume or at a boundary. In the thermo-elastic model, each physical type can exist.

To the authors knowledge, there are not many data formats to describe a system of geometrically and physically coupled PDEs. The authors are aware of

- extensions to support PDEs in Modelica®, see Li, Zhang, Zheng, 2008; Saldamli et al., 2005. So far, these extension to Modelica® only support very simple geometries, and, as a consequence, very simple (spatial) discretization methods.
- toolboxes in symbolic software Portela, Charafi, 2002. These toolboxes are tightly bound to the software, which we want to avoid.
- two similar approaches to describe finite element (FE) analysis using XML Michopoulos et al., 2001; Pinheiro, Moita, 2004. Their goal is to exchange a full set of FE analysis data, including results and solver data. Our aim is to separate the solver and tool specific configurations from the problem description. Sadly, the website `femml.sourceforge.net` had the last update in May 2002.
- a masters thesis Hvalstad-Nilsen, 2019, in which a YAML-based file format was developed. Similar to Michopoulos et al., 2001, the student describes a full FE analysis.

Due to this lack of tool-agnostic data formats, or libraries, for PDEs we developed a new format.

The numerical solution of the PDEs requires discretization in space and time. We follow the method-of-lines approach, which first performs a spatial-semidiscretization, leading to a system of ODEs.

For simplicity, we will present the structures and explain their relationships using the heat equation, and note the extensions for the mechanical behavior, where appropriate.

The thermal loads induce the majority of the thermal behaviour of the machine. Therefore, they must be considered properly in the thermal PDE model. At the same time, these loads determine the overall time scales. Prominent examples are rapidly varying loads and the coupling heat fluxes for relatively moving machine components. In particular the treatment of the temperature dependency of the coupling fluxes are of special interest when discretizing the PDE.

We propose the separation into a problem description and algorithm description. That way, only the algorithm description requires tool specific entries whereas the problem part concentrates on the common entries. Thus, we propose to use external software packages for the finite element (FE) discretization. Proprietary software solutions often provide a seamless workflow starting from the CAD model and proceeding all the way to the solution of the discretized ODEs. Consequently, the mathematical details and methods are completely hidden in the software.

This abstraction has the advantage that the construction of large and complicated models becomes feasible. At the same time, the specific numerical methods are harder, and more frequently not at all, extendable or even known. In contrast, the open-source packages are usually more specialized in the single tasks. For example, the mesh generation and the assembly (or, nowadays, the application) of the FE operators are usually implemented in different libraries. Thus, the implementation of a model requires a profound understanding of the underlying numerical methods. From the scientific point of view, access to the numerical methods and their implementation is key for further improvements and for understanding the efficiency of the simulation. At the same time, the open-source software in many cases has free licenses, thus the costs for the development can be reduced, when the license model allows proprietary downstream use, and modelers have received the appropriate training. See Sauerzapf et al., 2020 for an example where we show how to use open-source tools for discretization and model order reduction from within the proprietary FE package ANSYS®.

To facilitate the cooperation between scientific disciplines we propose an interface between

(i) industry standard modeling tools, like ANSYS® or COMSOL®,

(ii) open-source FE software packages, like DUNE or FEniCS,

(iii) ODE software packages.

In addition to the aforementioned tasks, the same interface connects to model order reduction (MOR), sensor place-

DOI
10.3384/ecp19381

ment Naumann, Herzog, 2021 or even parameter estimation. While the incorporation of the MOR toolbox relies mainly on the structure of the mathematical model, higher-level tasks such as sensor placement and parameter estimation require further operations. We concentrate on the extension of the FE discretization in space with the geometric coupling approaches. The interface to MOR and ODE packages will be discussed together.

For these purposes, the pointwise evaluation, as provided by a functional mockup unit (FMU), is not sufficient. In particular the MOR methods require the separation into inputs and corresponding coefficients. Thus, the FMU export and import between ANSYS® and Modelica® does not meet all our requirements.

We propose a tool-agnostic data format, which provides all required information about the model in an object-oriented layout. As a result, our proposed interface contains only the relationships between physical entities, but does not denote variables in the sense of a particular programming language, or environment.

From an abstract point of view, our format has similar goals as the SSP-standard Association, n.d. Both aim to separate common properties and relationships from tool specific ones. While the SSP-standard aims to bundle (technical) systems, we consider a mathematical model. In addition we made different technical decisions, like the basic file format or notation of URIs.

The structures of the (mathematical) PDE model and the ODE model are the basis for the interface. We briefly explain these models in Section 2. The two interfaces, one between the industry modeling tools and the open-source software, and one between the generated ODE and the simulation software packages, will be described in Section 3. Finally, we demonstrate the interface format with a simple model in Section 4 and give an outlook in Section 5.

2 Mathematical background

This section provides the mathematical foundations of the PDE and the ODE model at hand. Because the second stems from the discretization of the first, they have some parts in common.

The PDE model will use the heat equation for purpose of presentation. As it is a scalar equation for a scalar field, we can concentrate on the general structure. Other physical PDE, like Eulers equation, Navier-Stokes equations, equations of linear elasticity follow the same scheme. They comprise differential operators in space and time, which operate on a solution field. The tensor shape of the solution field is determined by the physical quantity. The boundary conditions close the system. Boundary conditions for vector valued problems can also restrict the vector to a sub-space and in general depend on the local basis of the vector field. For example, in linear elasticity, one fixes one coordinate direction, but leaves the others free to prevent penetration of a wall, but allow frictionless sliding along it.

In analogy, we present the ODE in Section 2.2 as first order ODE. The order is determined by the order (of the time derivative) of the PDE. Although one can transform every higher order ODE into a system of first-order ODEs, in a data format it is favorable to keep the second-order structure.

2.1 The PDE model

Real world applications often feature complicated geometries, where the FE discretization is the standard procedure. For the mathematical theory of the discretization we refer the reader to Grossmann, Roos, 1994; Zienkiewicz, Zhu, 1987.

We consider the conservation of heat in a system of solids with heat exchange between them. In the ith solid with domain $\Omega^{(i)}$ we describe the evolution of the temperature $T^{(i)}$ using the heat equation, i. e.

$$\partial_t \left(\rho^{(i)} C_p^{(i)} T^{(i)}\right) - \nabla \cdot \left(\lambda^{(i)} \nabla T^{(i)}\right) = Q^{(i)} \qquad \text{in } \Omega^{(i)} \tag{2.1a}$$

$$\partial_n \lambda^{(i)} T^{(i)} = g_{N,j}^{(i)} \qquad \text{on } \Gamma_{N,j}^{(i)} \tag{2.1b}$$

$$T^{(i)} = g_{D,j}^{(i)} \qquad \text{on } \Gamma_{D,j}^{(i)}. \tag{2.1c}$$

Inside the domain $\Omega^{(i)}$, the PDE (2.1a) holds, which depends on the material parameters density $\rho^{(i)}$, capacity at constant pressure $C_p^{(i)}$ and heat conduction $\lambda^{(i)}$, as well as on the volumetric thermal sources $Q^{(i)}$. Since the machine parts are often assembly groups consisting of several bodies, in addition, we assume to have a partition of the domain $\Omega^{(i)}$ into subdomains $\Omega_j^{(i)}$. Each domain $\Omega^{(i)}$, then, is a part and the subdomains $\Omega_j^{(i)}$ are the single bodies inside that part.

When the assembly groups move relative to each other, the domain $\Omega^{(i)}$ is time-dependent. Thus, the domain $\Omega^{(i)}$ at time t can be defined as

$$\Omega^{(i)}(t) \coloneqq \left\{ x^{(i)} \,\middle|\, x^{(i)}(t) = g_M^{(i)}(t, \hat{x}^{(i)}) \; \forall \hat{x}^{(i)} \in \tilde{\Omega}^{(i)} \right\}, \tag{2.2}$$

where $\tilde{\Omega}^{(i)}$ is a fixed reference domain and $g_M^{(i)}$ represents the movement. For simplicity, we restrict the movements to expressions of the form $g_M^{(i)}(t, \hat{x}^{(i)}) = O^{(i)}(t)\hat{x}^{(i)} + b^{(i)}(t)$, where $O^{(i)}$ are orthogonal matrices. In case of time-dependent non-orthogonal coordinate transformations, the space-discrete system gets additional time-dependent terms.

Each boundary $\Gamma^{(i)}$ is also separated into sub-boundaries $\Gamma_{N,j}^{(i)}$ and $\Gamma_{D,j}^{(i)}$. At each sub-boundary we apply either condition (2.1b), or (2.1c), whichever corresponds to the sub-boundary type. Please note that the subset of the boundaries is independent of the selection of the bodies, although they might intersect. The subscripts N and D represent Neumann and Dirichlet conditions, respectively.

In general, the material parameters and right-hand side of the boundary conditions can be arbitrary expressions. In particular, all expressions are allowed to depend on

time, space and temperature, or a mixture thereof. The dependencies of the material parameters, the sources and the boundary conditions on the temperature determine the classification of the PDE.

Despite the simplicity, temperature-independent material parameters and at most linearly temperature-dependent sources and boundary conditions, often describe the reality to sufficient accuracy. This special case renders the PDE model linear, which carries over to the ODE, after spatial semi-discretization. The solution of PDEs with constant coefficients is well understood and a wide range of numerical methods exists. The discretization requires the dependency information for all expressions.

We equip the system of heat equations with QOIs

$$y = \mathscr{C}(T), \tag{2.3}$$

where the operator $\mathscr{C}$ maps all temperature fields to one vector. The following output operators are of special interest:

(i) A linear, block-structured operator $\mathscr{C}$, such that the QOI y_i depends on the temperature field $T^{(i)}$ in the i-th body. This case includes the temperature in a point, the average temperature in the whole domain, in parts of the domain, or averages on surfaces. This structure was used in Vettermann et al., 2021 to describe thermally coupled, relatively moving parts.

(ii) A linear, block structured operator $\mathscr{C}$, such that a QOI depends on the temperature fields in two parts. These QOIs can describe the heat flux between neighboring parts, in form of a scaled multiple of the temperature difference.

(iii) An unstructured operator, which correspond to displacements in certain important points. These displacements describe the translation and rotation of a tool. The operator is the application of the inverse of the discrete linear elasticity on a temperature field. Due to the mechanical coupling, the operator acts on all temperature fields.

Thus, the complete PDE model describes a scalar PDE in a moving domain with mixed boundary conditions and user defined output operators. Please note, that the coupling of different physics can be hidden in the output operators.

2.2 The ODE model

The discretization in space transforms the system of PDEs into a system of ODEs. This ODE is block structured, where every block row represents the contribution of the PDE for one solid. In the following, we will omit the superscripts and describe the construction of a single block.

The basis for the FE discretization in space is the weak formulation of Eq. (2.1) on the reference domain $\widetilde{\Omega}$, i. e.

$$\begin{aligned}\int_{\widetilde{\Omega}} \partial_t(\rho C_p T)\psi dx + \int_{\widetilde{\Omega}} (\lambda \cdot \nabla T)\cdot\nabla\psi dx = \\ \int_{\widetilde{\Omega}} Q\psi dx + \int_{\widetilde{\Gamma}_N} g_N \psi dS,\end{aligned} \tag{2.4}$$

with suitable test functions ψ. Finally, we represent the solution $T(t,x)$ in the form $T(t,x) = \sum_l \mathbf{T}_l(t)\varphi_l(x)$. Thus, the solution is decomposed into a time-dependent vector and a set of space-dependent, but time-independent, functions φ_l. The functions φ_l form a basis for the function space, which includes the Dirichlet conditions. The test functions ψ vanish on the Dirichlet boundary Γ_D.

Table 2.1. Expressions corresponding to the differential operators of Eq. (2.1a).

dependency	$\partial_t(\rho C_p T)$	$-\nabla\cdot(\lambda\cdot\nabla T)$
constant	$\rho C_p M_V \dot{\mathbf{T}}$	$\lambda L_V \mathbf{T}$
time	$\partial_t(\rho C_p) M_V \mathbf{T} + (\rho C_p)(t) M_V \dot{\mathbf{T}}$	$\lambda(t) L_V \mathbf{T}$

The dependencies of the material parameters, the sources, and the boundary conditions on the temperature, space or time can lead to more terms, or a higher complexity. Note that a space parameter also has a representation in the same form as the solution. Therefore, we can neglect the space dependency in the following and concentrate on the time- and solution-dependence.

The Tables 2.1 and 2.2 represent the FE discretization for the volume and surface expressions in Eq. (2.1). Please note the linearity or independence of the solution T. Therefore, the PDE is linear too, and the discretized system will remain linear in the states.

The expressions M_V, L_V, b_V, $b_{S,j}$ represent the space-dependent part of the FE discretization. We use the subscripts V, S and C to refer to the integrals over a volume, surface and surface intersections, respectively. The symbols M, A and b refer to the FE mass matrix, the discrete Laplacian matrix and the integral vectors of the test functions, i.e.

$$\begin{aligned}[M_V]_{kl} &= \int_{\widetilde{\Omega}} \varphi_l \psi_k & [b_V]_k &= \int_{\widetilde{\Omega}} \psi_k \\ [M_{S,j}]_{kl} &= \int_{\widetilde{\Gamma}_j} \varphi_l \psi_k & [b_S]_k &= \int_{\widetilde{\Gamma}} \psi_k \\ [L_V]_{kl} &= \int_{\widetilde{\Omega}} \nabla\varphi_l \cdot \nabla\psi_k & [M^{(i,j)}_{C,m}]_{kl} &= \int_{\Gamma^{(i)}_{N,m}\cap\Gamma^{(j)}_{N,m}} \varphi_l^{(i)} \psi_k^{(j)}.\end{aligned}$$

All matrices depend only the discretization, but not on material parameters. Thus for example the mass matrix is neither the usual capacity matrix, nor the mass matrix from mechanics.

The volume integrals on the left-hand side can be further decomposed into bodies to account for piecewise material parameters. Thus, for piecewise material, the expressions $\rho C_p M_V$ and λL_V in Table 2.1 can be decomposed into sums over all bodies.

The Dirichlet boundary conditions (2.1c) describe the value in all nodes on the surface Γ_D. Thus, the test functions ψ vanish on Γ_D. In turn the entries $[b_V]_k$ and $[b_S]_k$

DOI
10.3384/ecp19381

Table 2.2. Expressions corresponding to the sources and the boundary conditions Eqs. (2.1a) and (2.1b).

	Q	$g_{N,j}$	$g_{D,j}$
constant	Qb_V	$g_{N,j}b_{S,j}$	$g_{D,j}b_{D,j}$
time	$Q(t)b_V$	$g_{N,j}(t)b_{S,j}$	$g_{D,j}(t)b_{D,j}$
$\alpha(t)T+\beta(t)$	$\alpha(t)M_D\mathbf{T}+\beta(t)b_V$	$\alpha(t)M_{S,j}\mathbf{T}+\beta(t)b_{S,j}$	—
$\alpha(t)\left(T^{(j)}-T^{(i)}\right)$	—	$\alpha(t)\left(M_{C,m}^{(i,j)}\mathbf{T}^{(j)}-M_{C,m}^{(i,i)}\mathbf{T}^{(i)}\right)$	—
nonlinear in T	$Q(y)b_V$	$g_{N,j}(y)b_{S,j}$	—

and kth rows of M_S and M_V vanish for $x_k \in \Gamma_D$. Instead we insert the condition $T_l = g_{D,j}(x_l)$ into L_V and a corresponding vector b_D, where the subscript D refers to the Dirichlet conditions. We replace the row k of L_V by the unit row and set b_D to the kth column of L_V afterwards.We refer the reader to Logg, Mardal, Wells, 2012, Section 6.3 for a local element approach. Please note that the Dirichlet conditions cause algebraic equations and therefore lead to a differential algebraic equation (DAE) system instead an ODE system. For the sake of presentation, we consider only systems of ODEs.

The Table 2.2 lists the expressions, which arise from the discretization of the sources and the boundary conditions. The rows are sorted with increasing dependency, where the time-dependence has lower complexity than temperature-dependence and a linear expression has also a lower complexity than a nonlinear one.

The fourth row in Table 2.2 represents the coupling between the thermal fields through a heat flux. The heat flux into part i is assumed to be linear in the temperature. Thus, the surface integrals for $M_{C,m}$ are restricted to some common surface between the components. Due to the relative movement, this surface is in general time-dependent, thus the matrices $M_{C_m}^{(i,j)}$ and $M_{C_m}^{(i,i)}$ are in general time-dependent too, even if the heat transfer coefficient α is constant. In an efficient implementation, this part is interpreted as an operator instead of a matrix.

The fifth row in Table 2.2 provides an additional approach to approximate the heat flux between different parts and accounts for a simplified approximation of nonlinear sources and boundary conditions. The convergence of an FE discretization requires an approximation of the integrals in Eq. (2.4) up to the required order, with a suitable quadrature formula. Thus, the nonlinear boundary condition g_N has to be evaluated in the quadrature points at the element level. Using piecewise constant temperatures, where the pieces are subdivisions of the domain (or surface), leads to a cheaper approximation at the costs of the order of the approximation.

The piecewise constant temperature on each piece is the average temperature, which can also serve as a QOI of the model. The same idea can be used to approximate the heat exchange. The coupling matrices $M_{C_m}^{(i,j)}$ depend on the meshes of both parts and the exchange between the meshes can be quite demanding. Thus, approximation uses the subdivision of the surface and replaces the pointwise temperatures $T^{(i)}$ and $T^{(j)}$ by their averages over the pieces on their meshes. This makes the meshes independent and the computations far cheaper. At the same time we can also represent the heat exchange between moving geometries without time dependent coefficients. The disadvantages of the piecewise constant approximation is the lower accuracy and the dependency of inputs in one block on outputs of another block.

Block ODE The full block system consists of all blocks from the previous paragraph combined with a set of external systems without any particular structure. Thus, we collect the terms by coefficients of $\dot{x}$, x, dependence on time t and for nonlinear T and outputs y in this order to obtain the first block in the ODE

$$M\dot{x} = (A_c + A_t(t))x + B_c u_c + B_t u_t(t) + B_y u_y(t, Y) \tag{2.5a}$$
$$y = Cx \tag{2.5b}$$
$$\dot{x}_{\text{ext}} = f_{\text{ext}}(t, x_{\text{ext}}, Y) \tag{2.5c}$$
$$y_{\text{ext}} = h_{\text{ext}}(x_{\text{ext}}). \tag{2.5d}$$

The second block consists of all right-hand sides, which are provided by external functions. A particular example for the external functions are the FMUs in model exchange mode. By contrast, an FMU in Co-Simulation mode contributes to the inputs u_y, where the subscript y denotes the dependency on outputs. Thus, the ODE consists of all terms of the form in Tables 2.1 and 2.2, but they are decomposed into the coefficients and the inputs. The matrix M is composed of all coefficients of the time derivatives, and is block diagonal. The matrices A_c and A_t are the block matrices, composed of the coefficients of $\mathbf{T}$, which includes the (negative) discrete Laplacians and the mass matrices M_S and M_C from the flux boundary conditions g_N. Please note the different origins of the time-dependence of the coefficients. These can originate from time-dependent heat transfer coefficients (HTCs), or from relatively moving parts. While the time-dependent HTCs lead to a time-dependent scalar coefficient with a constant surface mass matrix, the coupled surfaces lead to time-dependent operators, involving the integrals on the common surface.

The matrices B originate from the vectors b and commonly correspond to the inputs u of the ODE. This includes all sources and boundary conditions, irrespective of the type. The only differences, denoted by the subscripts c, t and y, are the dependencies on time and outputs. Note that the output dependencies u_y introduce an indirect de-

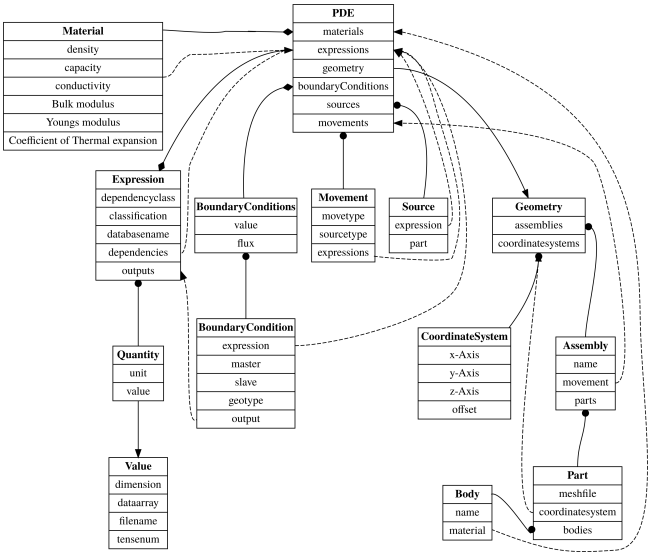

Figure 3.1. Relation between entities in the PDE model. Solid edges denote an aggregation, where the heads of the edges determine the aggregation types. A diamond head represents a dictionary, a circle head an array and an arrow corresponds to a single value of the type. Heads without marker correspond to single values of the referenced structur. Dashed lines represent indirect relations using keys of dictionaries or indices into an array.

pendence on the states and can render the ODE a nonlinear system, as well.

The QOIs $\mathscr{C}$ are represented by the operator C. We will consider only linear output operators, therefore C can be represented by a matrix.

3 Interfaces

The preceding section described the model equations for the PDE and the DAE. Each equation provides a structure and consists of several terms. Usually, the expressions in the terms correspond to physical properties. Thus, a hierarchical model description denotes the physical properties as objects, whereas the equations link them together. This section is devoted to the structure and content of the new model descriptions.

3.1 PDE model

The Eqs. (2.1) and (2.2) show all components of the PDE model:

- the reference domains $\widetilde{\Omega}^{(i)}$, which are approximated by mesh files,
- the movements, which are vector valued expressions, in contrast to the field valued sources and boundary conditions,
- the material parameters ρ, C_p and λ, which are scalar expressions,
- each boundary condition $g_{N,j}$ and $g_{D,j}$, which is a pair of a scalar expression and the reference to the corresponding sub-boundary.

The components of the PDE model are depicted in Fig. 3.1. The central node with the title “PDE” combines all other entries. The entry *materials* is separated from the remaining model, and only relates to the expressions.

The model joins all geometric properties and relations using the class *Geometry* and the entry *geometry* of the class PDE. As a consequence, the subgraph of the geometry contains the assemblies, which consist of parts and bodies. Each part is attached to a coordinate system, which is also part of the geometry subgraph.

The entry *boundaryConditions*, which is an instance of the class *BoundaryConditions*, contains value and flux conditions. Each condition references the corresponding expression and is attached to the *master*. The rank of these conditions must be the same, as the rank of the field. Thus, for the heat equation, the expression must be scalar, whereas for the equations of elasticity the expressions are vector valued.

We distinguish coupling from non-coupling boundary condition with the enum *geotype*. A value of *Face* represents loads, which are active on one surface, whereas *FaceFace* represents the coupling of two neighboring surfaces. The corresponding facing / opposite surface is determined by the entry *slave*. The boundary conditions and their associated expressions must match to their physics, respectively. Therefore, the description of mixed systems of PDEs modeling different physical quantities require an appropriate association. Please note the missing association with the physics. As our models are the equations of thermo-elasticity, we had no need to specify that explicitly.

All the assembly’s movements are part of the list *movements*, where every entry is an instance of the class *Movement*. Each assembly references the local movement, which is relative to the kinematic parent. Thus, the global movement of an assembly is given by the successive composition of the local movements of all predecessors.

The expressions are the common structure between the PDE model and the ODE model. Due to their central nature, we explain them in detail in Section 3.3.

3.2 ODE model

We depict the ODE model in Fig. 3.2. The model consists of a *blockIO*, the *expressions* and *associations*. The mathematical basis is a block structured ODE as given by Eq. (2.5). Each matrix entry of the *blockIO* represents a block matrix. While the output matrices C and the mass matrices E are block diagonal, the coefficient matrices A_c and A_t might be dense. Thus, the former are arrays of submatrices, whereas the latter are arrays of arrays of submatrices. The matrices E, A_c and A_t have coefficients, which are stored in the arrays m, α and β, respectively. In analogy the input matrices B_c, B_t and B_y correspond to the inputs u_c, u_t and u_y. The sub-scripts refer to the dependencies of the expressions, i. e. constant, time-dependent and output-dependent. The associations B_y link the outputs to the expression. Please note the missing entry of the size

DOI
10.3384/ecp19381

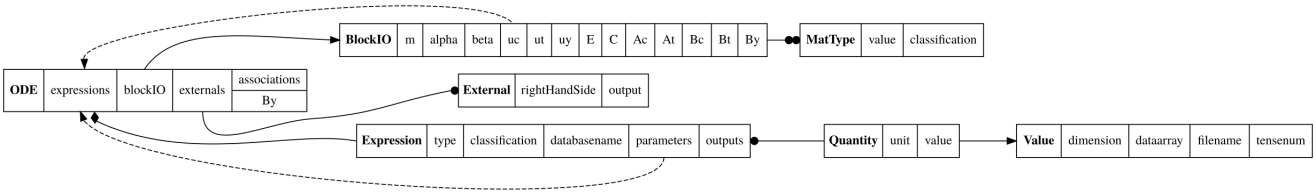

Figure 3.2. The relationship of the ODE model. The line and arrow styles are the same as in Fig. 3.1.

of the blocks. This must be inferred from the size of the sub-matrices.

This format proves to be very flexible and easy to extend, in a backward compatible way. Thus, it allows for the incorporation of advanced mathematical methods for the considered models. We highlight some aspects which were included in the data format to meet the method's special requirements using the example of MOR for linear time-invariant as well as linear parameter-varying systems.

MOR is used to compute low-dimensional surrogate models, which allow for accelerated simulations, see e.g. Antoulas, 2005; Benner, Grivet-Talocia, et al., 2021. System-theory based MOR methods require a model in input-output form as given in Eq. (2.5). Especially, in the case where a physical parameter is supposed to be preserved in the reduced-order model, i. e. in the parametric MOR (PMOR), the splitting into time-dependent coefficients β and constant matrices A_t coincides with the parameter-affine format proposed, e. g., in Benner, Gugercin, Willcox, 2015,. Then, the corresponding MOR methods, based on interpolation in parameter direction, come into consideration. Therefore, a possible parameter p and corresponding coefficients $\beta(p)$ were added to the model description. Moreover, these methods require a suitable parameter interval $[a,b]$. On this interval, local reduced-order models (ROMs) are computed in some parameter sample points. For a given parameter value $p \in [a,b]$ these sample ROMS are then interpolated in one way or another. These PMOR methods can, thus, be directly applied to the models at hand.

Furthermore, the expected magnitude of the inputs u is important for reliable ROMs, as the MOR error scales with, both, the system approximation error, and u, see Vettermann et al., 2021 and the references therein for further information. Thus, this magnitude is computed using the provided expressions u_c, u_t and u_y and is then used to transfer the scaling to the columns of the input matrices B_c, B_t and B_y prior to the MOR step, such that it can be taken into account in the system approximation and u itself is normalized.

The addition of the geometric coupling approaches, as stated in Section 2, allows for tailored MOR strategies utilizing the special structure of these models, see Vettermann et al., 2021. By including all necessary information in the ODE model-description the MOR methods can be applied in a user-friendly semi-automatic process.

3.3 Expressions

The expressions are the common structures, which are shared between the PDE and ODE model.

We separate the expressions with respect to two classifications

(i) The dependency on other expressions or solution fields with the entry *dependencyclass*. These classes are

- **Constant** are constant values, which do not depend on anything else.
- **Equationset** represents compound expressions, which involve further expressions. The dependencies between expressions are denoted by the map *dependencies*. These also include characteristic maps, which are represented by file names. FMUs are treated as a particular (complicated) expression. There, interaction with the solution is determined by the classification. The interaction between different FMUs is described by the dependency. Thus the expressions here are comparable to the transformations in the SSP-standard Association, n.d., p. 23.
- **Sensor** represents a linear functional of a solution field on one part. The specific geometry entity is described in the dependency map in analogy to the dependency on other expressions. Examples for these are the temperature average on a surface and the temperature value at a point.

(ii) The function classification, given by *classification* separates the expressions into time, time and one field value and time and two field values, or a vector of outputs. Thus, this classification concretises an expression of dependency class *Equationset*.

The first classification separates the expressions into simple, compound field-independent expressions and field-dependent expressions. Please note that an expression with dependency **Sensor** can exist only in a PDE model and is the only expression which can reference a geometric object. Therefore, during the ODE generation, the generator must keep track of all Sensor-type expressions and replace their occurrences by a reference to the corresponding output.

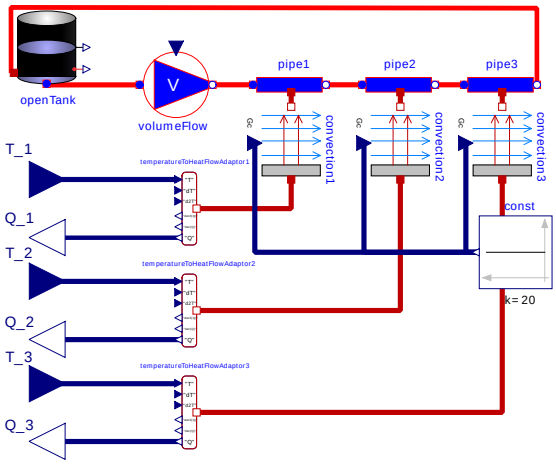

Figure 4.1. The cooling model with three flux outputs and three temperature inputs.

The second classification provides the interface for the function implementation, including the dependencies, to other expressions. Please note that this classification corresponds to the inputs u in the ODE system (2.5). At the same time, this classification implicitly serves as a notation for the dependency between the expressions and the solutions. Thus, when generating an ODE, an expression which depends on a solution value in the PDE model must be either decomposed according to Table 2.2 or related to an output.

Finally, an expression can have outcomes of different sizes or even different units. Every outcome is an entry in the list *outputs*, which is an instance of the class *Quantity*. A *Quantity* denotes the physical unit, and the description of the value. Each value consists of a vector *dimension*, which encodes the shape. Constant expressions also have a data-array, which contains the value, or a filename, if the values are stored elsewhere.

A numerical solution of an ODE requires the evaluation of the expressions. These implementations are encapsulated in an accompanying library, which represents a database of functors. This database maps a string to a functor with the signature according to the function classification (Item (ii)). The key of the corresponding functor is the entry *databasename* in the expression. It is up to the implementation of the database to provide further properties and operations, like linearity and derivatives with respect to constant arguments.

4 Two body model

To illustrate the usage of the interface formats described in Section 3, a simple model along with excerpts from json files. These files are exchanged between the interdisciplinary groups in the CRC/TR96. In detail, the json file Fig. 4.4 was generated using ANSYS® by one author, and used the code from a second author to generate Fig. 4.5. This ODE model will be fed to the MOR toolbox, written by the third author. Please note, that the ODE model could also be generated using ANSYS® directly.

The Fig. 4.2 shows the geometry of the example model. It comprises two assemblies, where each one consists of one part. Each part comprises two bodies, which are highlighted by the colors. Both assemblies exchange heat through a neighbouring surface with a linear temperature-dependent flux of the form $q = \alpha(T^{(1)} - T^{(2)})$.

The boundary conditions we applied are shown in Fig. 4.3. In this particular case, the cooling system shown in Fig. 4.1 is integrated as an FMU.

Collaborators in the authors research project constructed a complex cooling system in Shabi, Weber, Weber, 2017, which is part of the thermal model of a complex machine tool. For the sake of presentation, we simplified this model to a tank, an ideal pump and three pipes. The heat exchange between fluid and body walls is realized using convection subsystems in the Modelica® model.

The connection to the machine is two-fold:. The FMU inputs are the (averaged) temperatures at the corresponding surfaces. Internally, the FMU computes the heat fluxes for every pipe. These are the outputs of the FMU, and serve as the heat fluxes at the aforementioned surfaces.

Figure 4.4 depicts the PDE model. This relates to Figs. 4.2 and 4.3 using the same colors. We added the FMU using a single expression and reference the corresponding outputs in the boundary condition. As a consequence, practically, the FMU is evaluated only once.

The ODE model in Fig. 4.5 shows the relation of the coefficient matrix to the piecewise materials. In addition, we also highlight the corresponding expressions for the boundary conditions. As can be seen from the coefficient *alpha*, we decompose the coupling heat flux and use only the coefficient. Also note the change of the FMU expression. Instead of multiple outputs, we use a single vector valued output and classify the expression to depend on time and outputs.

5 Summary and outlook

This contribution describes a PDE model and an ODE model and introduces a data format to exchange each of them. Each model type comprises all entities arising in the mathematical model except the equations. Their relation is therefore only part of the documentation.

The simple two body model showed the basic thermal properties and a thermal load given as a simplified cooling system, which is provided as an FMU. Thus, the whole complexity, and possible intellectual properties, remain hidden in the FMU. This very simple example shows that both descriptions are general enough to describe geometrically coupled thermal problems with very complex loads, developed by collaborators in the same research project. Despite the simplicity, the example model also highlights the successful simplification of the model exchange between mathematicians and engineers. The engineers got access to state-of-the-art open source implementations of FE discretizations and MOR techniques. In the other di-

DOI
10.3384/ecp19381

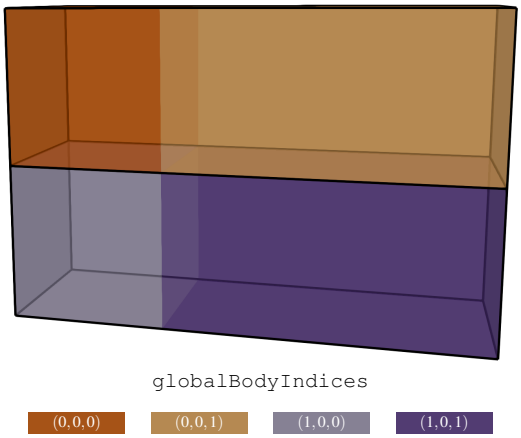

Figure 4.2. The geometry of the example model. Each color represents one body, whereas the legend indices denote the logical position as (assembly, part, body) in the description.

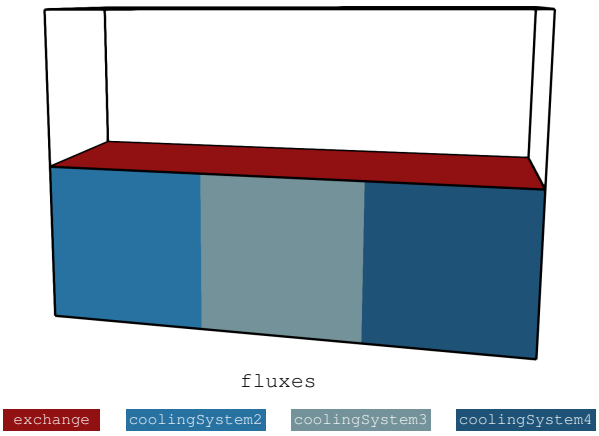

Figure 4.3. The boundary conditions in the example model. The colors red and light to dark blue represent the heat exchange and the cooling boundary conditions, respectively.

```
{
 "geometry": {
  "assemblies": [
   {
    "parts": [
     {
       "meshfile": "coupledflux_top.json",
       "name": "top",
       "bodies": [
        {"name": "left", "material": "steel" },
        {"name": "right", "material": "alu" }
       ],
       "coordinatesystem": 0
     }
    ],
    "name": "assembly"
   },
   {...}
   }
 ],
 "coordinatesystems": [...]
 },
},
"materials": {...},
                    :
```

```
                    :
"boundaryConditions": {
 "flux" :[
  {
   "master": [0, 0, 0],
   "expression": "alphaEx",
   "name": "alphaEx_0_1",
   "slave": [1, 0, 0],
   "geotype": "FaceFace"
  },
  {
   "master": [1, 0, 1],
   "expression": "coolingSystem",
   "name": "coolingSystem2",
   "output" : 0
   "geotype": "Face"
  },
  {
   "master": [1, 0, 2],
   "expression": "coolingSystem",
   "name": "coolingSystem4",
   "output" : 1
   "geotype": "Face"
  },
  {...}
 ]
},          :
```

```
                    :
"expressions": {
  "steel_capacity": {...},
  "alphaEx": {
   "dependencyclass": "Equationset",
   "classification": "timeTempTemp",
   "databasename": "chiAlphaToMT",
   "outputs": [{
      "value": {"tensenum": "Scalar"},
      "unit": {"unitstring": "kg s^-3"}
   }],
   "arguments": ["alphaIkb"]
  },
  "coolingSystem" : {
   "dependencyclass" : "Equationset",
   "classification" : "timeTemp",
   "databasename": "fmuWithTemp",
   "dependencies" : [...],
   "outputs": [{
      "value": {"tensenum": "Scalar"},
      "unit": {"unitstring": "kg s^-3"}
   }, {...}, {...}]
  },
    ...
 },
 "sources": []
 "movements": [] }
```

Figure 4.4. Excerpt from json-file used to exchange models in the PDE-Interface format. The colors mark the corresponding entities in Fig. 4.2 and Fig. 4.3.

```
{
 "blockIO": {
  "m": [
   ["BG_0_mass_factor_1", "BG_0_mass_factor_2"],
   ["BG_1_mass_factor_1", "BG_1_mass_factor_2"]
  ],
  "alpha": [
   [["BG_0_lapl_factor_1", "BG_0_lapl_factor_2", "alphaIkb"], ["alphaIkb"]]
   [["alphaIkb"], ["BG_1_lapl_factor_1", "BG_1_lapl_factor_2", "alphaIkb"]]
  ],
  "uy": [
   [],
   ["coolingSystem"]
  ],
  "Ac": [...],
  :
  },
  "externals": [],
  "associations": {
   "By": [
    [],
    [[[1, 0], [1, 1], [1, 2]]]
   ],
   :
  },
                :
                                :
 "expressions": {
  "alphaIkb": {...},
  "alphaEx": {
   "dependencyclass": "Equationset",
   "classification": "timeTempTemp",
   "databasename": "chiAlphaToMT",
   "outputs": [{
      "value": {"tensenum": "Scalar"},
      "unit": {"unitstring": "kg s^-3"}
   }],
   "arguments": ["alphaIkb"]
  },
  "coolingSystem" : {
   "dependencyclass" : "Equationset",
   "classification" : "timeOutputs",
   "databasename": "fmuHandler",
   "outputs": [{
      "value": {"tensenum": "Vector", "dimension": [3]},
      "unit": {"unitstring": "m^2 kg s^-3"}
   }]
  },
  :
 }
}
```


Figure 4.5. Excerpt from json-file used to exchange models in the ODE-Interface format. The colors mark the corresponding entities in Fig. 4.2 and Fig. 4.3

rection, the mathematicians optimized sensor placements for the same models.

The main advantage is the simplicity of the underlying data format. This simplicity renders the (ODE) description easily extendable with further entries for more sophisticated methods, like parametric MOR approaches.

Nevertheless models with systems of PDEs describing various physics require extensions of the PDE model.

We concentrated on the equations for thermo-elasticity, where the elasticity is assumed to be linear and stationary. In that particular case, boundary conditions apply either to the heat equation or the equations of linear elasticity. Thus, they can be distinguished by the dimension and unit of the associated expression. In a complicated model, an additional property would be less error prone and might be used for consistency checks.

In particular physically coupled models, like thermo-elastic models, require the addition of another kind of physics. These require the extension of the boundary conditions to assign them to the particular physics. The thermo-elastic models with stationary equations of elasticity can already be described using the output operators in the ODE model. At the same time the space-discrete dynamic equations of linear elasticity are second-order systems. Thus, the ODE model requires additional coefficients to represent these.

References

Antoulas, A. C. (2005). *Approximation of Large-Scale Dynamical Systems*. Vol. 6. Advances in Design and Control. With a foreword by Jan C. Willems. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM). DOI: 10.1137/1.9780898718713.

Association, M. (n.d.). *System Structure and Parameterization*. Version 1.0. URL: https://ssp-standard.org/publications/SSP10/SystemStructureAndParameterization10.pdf.

Benner, P.; S. Grivet-Talocia; A. Quarteroni; G. Rozza; W. Schilders; L. M. Silveira, eds. (Oct. 2021). *Model Order Reduction. Volume 1: System- and Data-Driven Methods and Algorithms*. De Gruyter. DOI: 10.1515/9783110498967.

Benner, P.; S. Gugercin; K. Willcox (Jan. 2015). "A Survey of Projection-Based Model Reduction Methods for Parametric Dynamical Systems". *SIAM Review* 57.4, pp. 483–531. DOI: 10.1137/130932715.

Grossmann, C.; H.-G. Roos (1994). *Numerik partieller Differentialgleichungen*. Stuttgart: Teubner. DOI: 10.1007/978-3-322-96752-7.

Hvalstad-Nilsen, O. (2019). "YAML Based Input File Format For Finite Element Analysis". MA thesis. NTNU Norwegian University of Science and Technology.

Li, Z.; H. Zhang; L. Zheng (2008). "Description of PDE models in Modelica". *2008 International Symposium on Computer Science and Computational Technology*. Vol. 1. IEEE, pp. 809–813.

Logg, A.; K.-A. Mardal; G. N. Wells, eds. (2012). *Automated Solution of Differential Equations by the Finite Element Method*. Springer. DOI: 10.1007/978-3-642-23099-8.

Michopoulos, J.; P. Mast; T. Chwastyk; L. Gause; R. Badaliance (2001). "FemML for data exchange between FEA codes". *ANSYS Users' Group Conference*.

Naumann, A.; R. Herzog (Jan. 2021). "Optimal sensor placement for thermo-elastic coupled machine

models". *PAMM* 20.1. DOI: 10 . 1002/pamm . 202000255.

Pinheiro, M. C.; G. F. Moita (2004). "Usando XML para criação de um padrão para o intercâmbio de dados entre programas de elementos finitos". *Educação & Tecnologia* 9.1.

Portela, A.; A. Charafi (2002). *Finite Elements Using Maple*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-55936-5.

Saldamli, L.; B. Bachmann; H. Wiesmann; P. Fritzson (2005). "A framework for describing and solving pde models in modelica". *Paper presented at the 4th International Modelica Conference*.

Sauerzapf, S.; J. Vettermann; A. Naumann; J. Saak; M. Beitelschmidt; P. Benner (2020). "Simulation of the thermal behavior of machine tools for efficient machine development and online correction of the tool center point (TCP)-displacement". euspen Special Interest Group Meeting: Thermal Issues. Aachen, Germany, pp. 135–138.

Shabi, L.; J. Weber; J. Weber (2017). "Model-based Analysis of Decentralized Fluidic Systems in Machine Tools". *Proceedings of 15: th Scandinavian International Conference on Fluid Power, June 7-9, 2017, Linköping, Sweden*. 144. Linköping University Electronic Press, pp. 116–123.

Vettermann, J.; S. Sauerzapf; A. Naumann; J. Saak; P. Benner; M. Beitelschmidt; R. Herzog (Apr. 20, 2021). "Model order reduction methods for coupled machine tool models". Special Issue ICTIMT 2021.

Zienkiewicz, O. C.; J. Z. Zhu (1987). "A simple error estimator and adaptive procedure for practical engineering analysis". *International Journal for Numerical Methods in Engineering* 24.2, pp. 337–357. DOI: 10.1002/nme.1620240206.