Hestia.jl: A Julia Library for Heat Conduction Modeling with Boundary Actuation

Stephan Scholz*, Lothar Berger

Control and Process Engineering, Ravensburg-Weingarten University of Applied Sciences, Germany
Web: https://forschung.rwu.de/forschungsgruppen/control-and-process-engineering
*stephan.scholz@rwu.de

SNE 33(1), 2023, 27-30, DOI: 10.11128/sne.33.sn.10634
Selected ASIM SST 2022 Postconf. Publication:2023-02-01;
Received Revised Improved:2023-02-27; Accepted:2023-03-10
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. Heat conduction modeling in three dimensions with boundary actuation plays an important role in thermal process engineering, for example in case of heating plates, laser welding or 3D printing. Here, the actuators and the induced energy have to be described exactly for such processes to guarantee a high simulation quality. We introduce Hestia.jl, a software library to model three-dimensional heat conduction with multiple spatially distributed heat sources on the boundary.

Introduction

The heat equation is a standard example in numerical analysis and control theory. Several software tools and libraries like OPENFOAM [1], FENICS [2], TRIXI.JL [3] and VORONOIFVM.JL [4] exist to solve heat equation models in one, two or three dimensions. These software tools are general purpose solvers for (specific types of) partial differential equations, which means they are also applicable for other models like the Poisson or Burgers' equation. However, they often require solid knowledge in the theory of finite volume or finite element methods.

We present HESTIA.JL [5], a Julia library [6] to simulate heat conduction in one, two and three dimensions with boundary actuation. A discretized heat conduction model is created in few steps using HESTIA.JL, without deep knowledge of numerical analysis. This heat conduction model is solved in time using high-order numerical integrators provided by DIFFERENTIALEQUATIONS.JL [7].

1 Problem formulation

Heat conduction is often modeled to occur inside geometrical objects like one-dimensional rods $\Omega = (0,L)$, two-dimensional plates $\Omega = (0,L) \times (0,W)$ and three-dimensional cuboids $\Omega = (0,L) \times (0,W) \times (0,H)$. These objects are implemented in HESTIA.JL as data types HeatRod, HeatPlate and HeatCuboid to store the original dimensions and the spatial approximation including sampling and number of grid points.

We distinguish linear and quasi-linear heat conduction depending on the definition of its physical properties: thermal conductivity λ, specific heat capacity c and mass density ρ. Constant properties (in case of linear heat conduction) are stored as a StaticIsoProperty whereas temperature-dependent properties (in case of quasi-linear heat conduction) are stored as a DynamicIsoProperty. In particular, temperature-dependent properties are assumed to be modeled as power series, e.g. $\lambda(\Theta) = \sum_{n=1}^{N} a_n \, \Theta^{n-1}$, and the coefficients are saved in an array, e.g $[a_1, \cdots, a_N]$. So, we assume the quasi-linear heat conduction model

$$\rho(\vartheta)\, c(\vartheta)\, \frac{\partial \vartheta(t,x)}{\partial t} \;=\; \operatorname{div}\left[\lambda(\vartheta)\, \nabla \vartheta(t,x)\right]$$

with $(t,x) \in (0,T) \times \Omega$ and final time $T > 0$ to be a generalization of the linear heat equation.

On boundary $\partial\Omega$ the variation of temperature ϑ is affected by the boundary conditions, namely linear heat transfer

$$-h(x)\, \left([\vartheta(\cdot,x) - \Theta_{amb}(x)]\right)$$

and nonlinear heat radiation

$$-\varepsilon(x)\, \sigma \left[\vartheta(\cdot,x)^4 - \Theta_{amb}(x)^4\right]$$

to the environment with ambient temperature Θ_{amb}, heat transfer coefficient h, emissivity ε and Stefan-

Boltzmann constant σ, see also [8]. The parameters h, ε and Θ_{amb} are stored in data type Emission and can be defined for each boundary side separately, see Table 1.

West	$\{0\} \times [0,W] \times [0,H]$
East	$\{L\} \times [0,W] \times [0,H]$
South	$[0,L] \times \{0\} \times [0,H]$
North	$[0,L] \times \{W\} \times [0,H]$
Underside	$[0,L] \times [0,W] \times \{0\}$
Topside	$[0,L] \times [0,W] \times \{H\}$

Table 1: Names and positions of boundary sides.

2 Actuator configuration

Actuators are only assumed on boundary sides - not inside the geometrical object. The actuated boundary sides (west, east, etc.) are partitioned using a checkerboard pattern and for each partition β_n an actuator and its spatial configuration b_p can be defined as introduced in [9]. This spatial configuration describes the possible spatially distributed intensity of actuation ranging from zero (no actuation) to one (full actuation). We define a radial symmetric configuration as

$$b_p(x) = \begin{cases} m_p \cdot \exp\left(-\|M_p(x-x_{c,n})\|^{2\nu_p}\right) & \text{for } x \in \beta_n, \\ 0 \quad \text{for } x \in B_A \setminus \beta_n \end{cases}$$

with scaling $m \in [0,1]$, curvature matrix $M \in \mathbb{R}^{3\times3}$, power $\nu \in \mathbb{N}_{>0}$ and central point $x_{c,n} \in \beta_n$ of the n-th partition. These coefficients are stored as RadialConfiguration. The choice of a spatial configuration shall approximate real-world scenarios where heating elements may not be able to induce the same amount of energy at each position of its surface. An example partition and its related actuator configuration are illustrated in Figure 1.

3 Demonstration example

Next, we explain how to build a simulation of a cooling-down and heating-up process for a three-dimensional cuboid. A full listing can be found on GitHub [10].

Setting up the model

We specify the physical properties of a quasi-linear heat conduction model as $\rho := 7800$,

$$c(\Theta) := 330 + 0.4\,\Theta \quad \text{and}$$
$$\lambda(\Theta) := 10 + 0.2\,\Theta - 10^{-4}\,\Theta^2.$$

These coefficients are stored in arrays as noted in Section 1 and a DynamicIsoProperty is created as in Listing 1.

```
property =
  createDynamicIsoProperty(
    [10.0, 0.2, -1e-4], [7800], [330, 0.4])
```

Listing 1: Create property type for quasi-linear dynamics.

We design a cuboid of length $L = 0.3$, width $W = 0.2$ and height $H = 0.1$, which is discretized by 40 cells in x_1-, 24 cells in x_2- and 10 cells in x_3-direction. The model is built as in Listing 2

```
cuboid =
  HeatCuboid(0.3, 0.2, 0.1, 40, 24, 10, property)
```

Listing 2: Create cuboid model.

The ambient temperature of the boundary conditions is set to 300 Kelvin. On boundaries west and east we assume the heat transfer coefficient $h = 10$ and emissivity $\varepsilon = 0.6$, see Listing 3. On boundaries north and east we consider only heat transfer with $h = 10$ and no heat radiation, all other boundary sides are considered with zero-Neumann boundary conditions ($h = 0$ and $\varepsilon = 0$) and do not have to be implemented explicitly.

```
boundary = initBoundary(cuboid)
emission = createEmission(10, 0.6, 300)
setEmission!(boundary, emission, :west)
```

Listing 3: Specify boundary conditions.

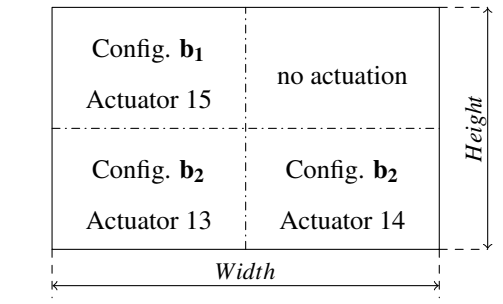
Cooling-down process

Now, the heat equation is approximated in space and forms an ordinary differential equation (ODE) which is solved with DIFFERENTIALEQUATIONS.JL. The specification (geometry, property, boundary conditions) and the temperatures are handed over to the diffusion! function to compute the right-hand side of the ODE, see Listing 4. The cool_down! function forms a standard interface for numerical ODE integration methods provided by DIFFERENTIALEQUATIONS.JL.

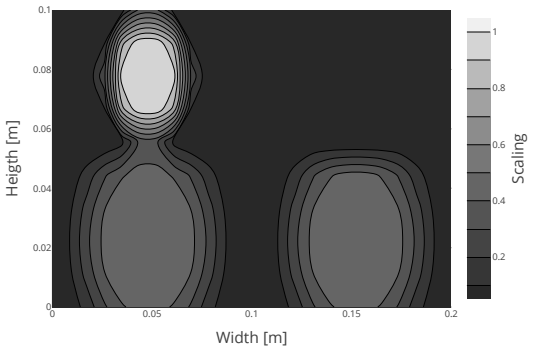
```
cool_down!(dv, v, p, t) =
  diffusion!(dv, v, cuboid, property, boundary)
```

Listing 4: Define interface for ODE solver for cooling-down process.

The cooling-down process is not depicted here because we focus on the heating-up process as described next.



(a) Boundary partition



(b) Actuator configuration

Figure 1: Partition (a) and configuration (b) on east boundary at $x_1 = L$ with configurations b_1: $(m_1, M_1, \nu_1) = (1.0, 50\ \mathrm{I}_{3\times3}, 3)$ and b_2: $(m_2, M_2, \nu_2) = (0.5, 30\ \mathrm{I}_{3\times3}, 2)$.

Heating-up process

The heating-up process extends the previous steps by specifying the position and spatial configuration of actuators on the boundary sides as discussed in Section 2. In this example, we assume actuation on boundaries UNDERSIDE and EAST. The underside is subdivided in 4×3 partitions and for each of it an individual actuator with configuration b_1 where $m_1 = 1$, $M_1 = 50\ \mathrm{I}_{3\times3}$, $\nu_1 = 3$ is specified as in Listing 5.

```
config = setConfiguration(1.0, 3, 50)
```

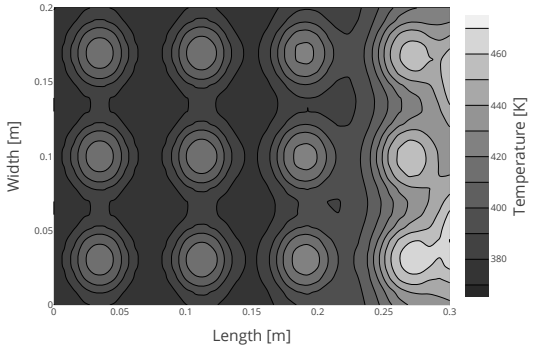


Figure 2: Temperature on the underside at $x_3 = 0$.

```
setIOSetup!( actuation, cuboid, (4,3),
                config , :underside )
```

Listing 5: Specify actuation on underside.

Boundary EAST is subdivided manually with 2×2 partitions, as portrayed in Figure 1, where two fields are defined by configuration b_2 with $m_2 = 0.5$, $M_2 = 30\ \mathrm{I}_{3\times3}$, $\nu_2 = 2$, one field is defined by b_1 as noted above and one field is not actuated. See also the complete listing [10].

A constant heat input $u_n(t) = 4 \cdot 10^5$ for actuator $n = 1, \ldots, 15$ (12 actuators on the underside and 3 on the east boundary) is set and the ODE interface is implemented as in Listing 6.

```
u_in = 4e5 * ones(15)
heating_up!(dv, v, param, t) =
  diffusion!(dv, v, cuboid, property,
                boundary, actuation, u_in)
```

Listing 6: Define interface for ODE solver for heating-up process.

The heating-up process is simulated for $T_f = 200$ seconds and the final temperature distribution on the underside (at $x_3 = 0$) and east boundary (at $x_1 = L$) are portrayed in Figure 2 and 3. They unveil the strong influence of the actuator configuration on the resulting temperature distribution which reaches up to 470 Kelvin. A three-dimensional temperature distribution is illustrated in Figure 4 for temperatures above 360 Kelvin.

4 Conclusion

We introduced the software library HESTIA.JL for modeling of three-dimensional heat conduction with boundary actuation. The recent version is able to approximate linear and quasi-linear (isotropic) heat conduction and

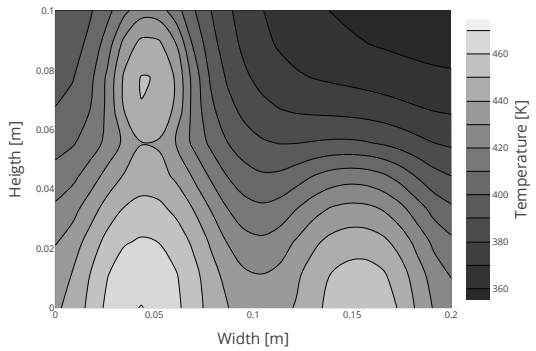


Figure 3: Temperature on the east boundary at $x_1 = L$.

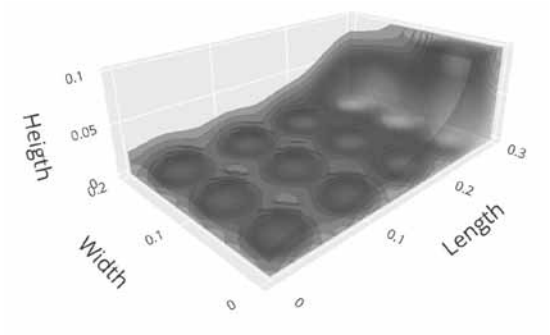


Figure 4: Temperature distribution in the cuboid for temperatures higher than 360 Kelvin.

to handle radial symmetric actuator configurations. Our further research will focus on the development of optimal control modules for HESTIA.JL, and on its good integration in Julia's Scientific Machine Learning ecosystem.

References

[1] Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*. 1998; 12(6): 620–631.

[2] Logg A, Wells GN. DOLFIN: Automated finite element computing. *ACM Transactions on Mathematical Software*. 2010; 37(2), 1-28.

[3] Schlottke-Lakemper M, Gassner GJ, Ranocha H, Winters AR, Chan J. Trixi.jl. Zenodo. 2022. Available: `https://zenodo.org/record/6372038`

[4] Fuhrmann J, contributors. VoronoiFVM.jl: Finite volume solver for coupled nonlinear partial differential equations. Zenodo. 2022. Available: `https://doi.org/10.5281/zenodo.6151074`

[5] Scholz S. Hestia.jl. Zenodo. 2023. Available: `https://doi.org/10.5281/zenodo.7685941`

[6] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: A fresh approach to numerical computing. *SIAM Review*. 2017; 59(1): 65-98.

[7] Rackauckas C, contributors. SciML/DifferentialEquations.jl. Zenodo. 2022. Available: `https://doi.org/10.5281/zenodo.5837925`

[8] Baehr HD, Stephan K. *Heat and mass transfer*. Springer Science & Business Media, 2011.

[9] Scholz S, Berger L. Modeling of a multiple source heating plate. *arXiv preprint arXiv:2011.14939*. 2020.

[10] Scholz S. HestiaDemonstration.jl. GitHub. 2022. Available: `https://github.com/stephans3/HestiaDemonstration.jl`