SciComp and FEM in WS25

Exercise 8: individual projects

Deadline: Jan 28, 2026, 16:00

Coffee Mug. We consider the time dependent heat conductivity problem

$$\begin{aligned} c(x)\frac{\partial u(x,t)}{\partial t} - \nabla_x^T\left(\lambda(x)\nabla_x u(x,t)\right) &= f(x,t) \qquad (x,t) \in\ \Omega\times(0,T] \\ -\lambda(x)\frac{\partial u(x,t)}{\partial \vec{n}} &= \alpha(x)\left(u(x,t)-u_{out}(x,t)\right) \qquad (x,t)\in\ \partial\Omega\times(0,Z] \\ u(x,0) &= u_0(x) \qquad x\in\ \overline{\Omega}=\overline{\Omega}^{wall}\cup\overline{\Omega}^{fluid}\cup\overline{\Omega}^{air} \end{aligned} \tag{1}$$

with the initial condition $u_0(x,0)$ and $x=(x_1,x_2,x_3)$.

Figure 1: Domain of interest.

The domain of interest Ω is my coffee mug with the geometry:

geometry	quantity
lower (outer) diameter	50 mm
upper (outer) diameter	83 mm
height	105 mm
wall thickness	3 mm
bottom thickness	6 mm
height of coffee/water-fluid	66 mm

,

math	physics
c	heat storage capacity
λ	thermal conductivity
f	internal heat sources
α	heat transfer coefficient
u_{out}	outside temperature
u_0	initial temperature

.

It consist of 3 different regions: The ceramic mug Ω^{wall} itself, the inner part with fluid Ω^{fluid} and the air region between surface of fluid and the top edge of the cup Ω^{air}. The appropriate material coefficients c, λ, α differ with respect to the various subdomains. There are no inner heat sources f and only Robin boundary conditions[1] are assume.
We assume that all coefficients above do not change in time or depend on the solution u and we do not consider heat radiation. Otherwise we will end up with a non-linear problem.
That excludes also density driven convection in Ω^{air}.

Simulations. The following general effects should be investigated to certain extend.

(i) The initial temperature of the cup is the same as the surrounding air $u_0^{air,wall}=18°\text{C}$. The cup is filled with warm water such that $u_0^{fluid}=80°\text{C}$.
Q: How long does it take (T_{warm}) until the mug is heated to the optimal temperature[2] of 67°C? What changes with $u_0^{fluid}=80°\text{C}$?

(ii) We replace the water in the mug by coffee with temperature 85°C at time T_{warm}.
Q: How long does it take until the temperature of the coffee drops to 50°C?

(iii) You try to cool the coffee via blowing with velocity[3] $2\frac{m}{s}$ at the top edge. The temperature of your breath is 31°C.
Q: How much faster does the coffee cool down?

Presentation of results. In your git-repository, similar to evacuation of bees[4].

[1] `https://en.wikipedia.org/wiki/Robin_boundary_condition`
[2] `https://www.beanarella.at/de/welt-des-kaffees/kaffeetipps/die-perfekte-kaffeetemperatur`
[3] `https://www.atemmuskeltraining.com/de/patienten/normokapnische-hyperpnoe/koerperliche-leistung-atmung`
[4] `https://github.com/Mohadese561/Evacuation_of_Bees`

Project A: Rotationally symmetric conductivity

Let us consider the mug without the handle so that we achive the rotationally symmetric geometry Ω^{rot}.

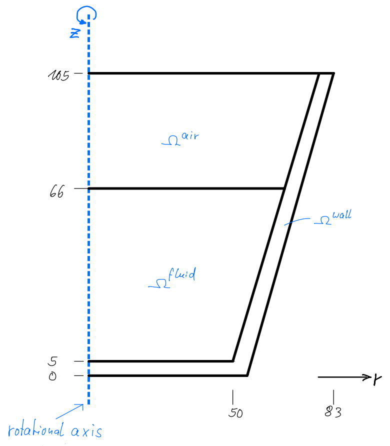

Figure 2: Ω^{rot}: rotationally symmetric

The transformation of coordinates $((x_1, x_2, x_3) \longrightarrow (\varphi, r, z))$ and the assumption that no quantity is a function of φ results in the rotationally symmetric PDE

$$\begin{aligned} r c(\cdot)\frac{\partial u(\cdot,t)}{\partial t} - \nabla^T_{(r,z)}\left(r\lambda(\cdot)\nabla_{(r,z)}u(\cdot,t)\right) &= r f(\cdot,t) \qquad (\cdot,t) \in\ \Omega^{rot}\times(0,T] \\ -r\lambda(\cdot)\frac{\partial u(\cdot,t)}{\partial\vec{n}} &= r\alpha(\cdot)\left(u(\cdot,t)-u_{out}(\cdot,t)\right) \qquad (\cdot,t)\in\ \Gamma^{robin}\times(0,T] \\ \lambda(\cdot)\frac{\partial u(\cdot,t)}{\partial\vec{n}} &= 0 \qquad (0,z,t)\in\ \Gamma^{symm}\times(0,T] \\ u(\cdot,0) &= u_0(\cdot) \qquad (\cdot)\in\ \overline{\Omega}=\overline{\Omega}^{wall}\cup\overline{\Omega}^{fluid}\cup\overline{\Omega}^{air} \end{aligned} \tag{2}$$

with the initial condition $u_0(\cdot,0)$, the symmetry boundary $\Gamma^{symm} := \{(r,z) : r == 0\}$, the Robin boundary $\Gamma^{robin} := \partial\Omega^{rot}\setminus\Gamma^{symm}$.

1. Generate the 2D mesh with the 3 material domains Ω^{wall}, Ω^{fluid}, Ω^{air}.
   Derive the geometry description by changing *generate_mesh/chip_2materials.m* and using matlab. shm/generate_mesh
   The script above generates two files *chip_2materials.txt* (coordinates of vertices and the finite element connectivity) and *chip_2materials_sd.txt* (material number per element).
2. Follow the code structure in *mgrid_2/main.cpp* for the Jacobi branch (`#undef MG`) and check whether the stationary Dirichlet problem can be solved. shm/mgrid_2
3. Implement for class `FEM_Matrix` a new Method `CalculateLaplace_mult` derived from `CalculateLaplace` that takes into account constant but different conductivities for your domains.

4. Change the boundary conditions from Dirichlet b.c. to Robin b.c..
   Derive a method `ApplyRobinBC_mult` from `ApplyDirichletBC` in class `FEM_Matrix`.

5. Implement for class `FEM_Matrix` a new Method `CalculateLaplace_mult_rot` and a method `ApplyRobinBC_mult_rot` that realize the finite element matrix computation for the Laplace part (without $\frac{\partial u}{\partial t}$) in equ. (2).

6. Implement a method `AddMass_mult_rot` that realizes internally the calculation of the mass matrix entries $M_{i,j} := \int\int rc(r,z)\phi_i\phi_j drdz$ similar to the element calculation for the Laplace part of the matrix $K_{i,j}$ and adds it it to the existing matrix entries.

7. Write a funcion (or method) `Init_Solution_mult` that initializes the solution in the whole domain depending on the subdomain. That function might be useful in subtask (i) as well as in (ii).

8. Use an explicit scheme to discretize in time so that we end up with the discrete formulation of (2)
$$\left(\tfrac{1}{\tau}M + K\right)\underline{u}^{k+1} = \underline{f}^{k+1} + \tfrac{1}{\tau}M \cdot \underline{u}^k \tag{3}$$
   with τ denoting the chosen time step.

9. Solve (3) for subtask (i).

Project B: 2D coupling heat transfer $\leftrightarrow$ CFD

We consider a 2D cut of the mug as in Fig. 2 without rotation and without symmetry axis. The PDE is similar to (1) in x- and z-coordinates in Ω^{2d}.

1. Generate the 2D mesh with the 3 material domains Ω^{wall}, Ω^{fluid}, Ω^{air}.
   Derive the geometry description by changing *generate_mesh/chip_2materials.m* and using matlab.

2. Follow the code structure in *mgrid_2/main.cpp* for the Jacobi branch (`#undef MG`) and check whether the stationary Dirichlet problem can be solved.

3. Implement for class `FEM_Matrix` a new Method `CalculateLaplace_mult` derived from `CalculateLaplace` that takes into account constant but different conductivities for your domains.

4. Change the boundary conditions from Dirichlet b.c. to Robin b.c..
   Derive a method `ApplyRobinBC_mult` from `ApplyDirichletBC` in class `FEM_Matrix`.

5. Implement a method `AddMass_mult` that realizes internally the calculation of the mass matrix entries $M_{i,j} := \int\int c(x,z)\phi_i\phi_j dxdz$ similar to the element calculation for the Laplace part of the matrix $K_{i,j}$ and adds it it to the existing matrix entries.

6. **Couple** the top edge of the mug from Ω^{2d} with a CFD problem (air stream) in a rectangular domain Ω^{CFD}. Modelling air as fluid is not fully correct but good enough for our purpose of coupling two PDE solvers.

   - Use the given solver for the heat conductivity problem shm/mgrid_2
   - Use OpenFoam for the CFD part (intro[5], tutorial[6], download[7])
   - Use preCICE for coupling (intro[8], docu[9], quickstart[10], adapters[11], fluid-temperature[12]).
     Have a look at the example code in f90, especially files *precice_config.xml* and *main_precice.f90* . shm/mch

   Start with one time step for the stationary problem.

7. Solve the coupled time-dependent problem.

[5] `https://www.openfoam.com/`
[6] `https://www.openfoam.com/documentation/tutorial-guide`
[7] `https://www.openfoam.com/current-release`
[8] `https://precice.org/`
[9] `https://precice.org/docs`
[10] `https://precice.org/quickstart`
[11] `https://precice.org/adapters-overview`
[12] `https://precice.org/tutorials-flow-over-heated-plate`

Project C: mug simulation with ngsolve

Discretize and solve the full 3D problem using ngsolve[13]. Consider the mug without handle as domain Ω. ngsolve

1. Login to the jupyter hub[14] on mephisto and start the ngsolve kernel.
2. Have a look at the tutorials[15] and try some of them in the jupyter hub.
3. Find out hout to describe the geometry[16] and how produce a 3d mesh from it. netgen
   Start with the 2D geometry from Task A or B for the first experiments.
4. Solve the stationary heat conductivity problem (Laplace-like from (1)) in Ω.
5. Solve the instationary heat conductivity problem (1).
6. Try the coupling with CFD in the top surface of the mug.

[13] `https://ngsolve.org/`
[14] `https://imsc.uni-graz.at/jupyter`
[15] `https://docu.ngsolve.org/latest/i-tutorials/index.html`
[16] `https://docu.ngsolve.org/latest/i-tutorials/unit-4.1.1-geom2d/geom2d.html`

Riser-Cast-Sand. We consider a domain $\overline{\Omega} = \overline{\Omega^{RS}} \cup \overline{\Omega^{C}} \cup \overline{\Omega^{S}}$ with 3 sorts of physical effects, see Fig. 3-left. The FORTRAN-code by Diego Vasco[17] is able to handle rectangular domains (Sleeve-Riser Ω^{RS}, Cube with all physics Ω^{C}) as depicted in Fig. 3-right. Although a simple

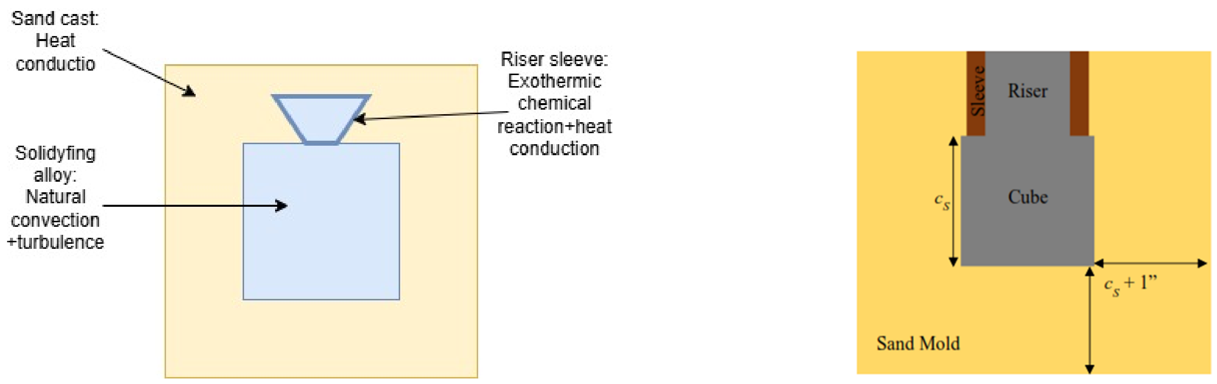
Sand cast: Heat conductio
Solidyfing alloy: Natural convection +turbulence
Riser sleeve: Exothermic chemical reaction+heat conduction
Sleeve
Riser
Cube
c_s
$c_s + 1"$
Sand Mold

Figure 3: left: original domain; right: simplified domain

heat conductivity equation, the sand domain Ω^S cannot be handled by Vasco's solvers and so a second solver for the heat conductivity is needed. Therefore, we have to consider a coupling of the two codes via software *preCICE*.

- We assume Robin b.c. for the outer boundary of Γ^S.

$$-\lambda(x)\frac{\partial u(x,t)}{\partial \overrightarrow{n}} = \alpha(x)\left(u(x,t) - u_{out}(x,t)\right) \qquad (x,t) \in \ (\partial\Omega^S \cap \partial\Omega) \times (0, Z]$$

- $\lambda = 0.27\frac{W}{mK}$ (or assume 80%-dry Sandy Soils[18] from saturated soils?)
- $\alpha = 10\frac{W}{m^2K}$
- $u_{out} = 290K$
- no internal source, i.e. $f = 0$.

[17]`https://sciprofiles.com/profile/842146`
[18]`https://en.wikipedia.org/wiki/List_of_thermal_conductivities`

Project D: 3D Coupling heat transfer ↔ CFD+chemical reaction

We are going to solve the problem on page 6 unsing code *CUBE_3D_Turbulent* and combine it with *OpenFoam* in 3D. 2026_Vasco/CUBE

1. Run the code *CUBE_3D_Turbulent*.

2. Determine the geometry of Ω^S.

3. Set up *OpenFoam* for the heat conductivity part in Ω^S with Dirichlet b.c. on Γ^S. (intro[19], tutorial[20], equations[21] download[22])

4. **Couple** the two solvers via *preCICE*.

   - Use the given solver for the heat conductivity problem shm/mgrid_2
   - Use preCICE for coupling (intro[23], docu[24], quickstart[25], adapters[26], fluid-temperature[27]).
   - Have a look at the example code in f90, especially files *precice_config.xml* and *main_precice.f90* . shm/mch

   Start with one time step for the stationary problem.

5. Solve the coupled time-dependent problem.

[19] https://www.openfoam.com/
[20] https://www.openfoam.com/documentation/tutorial-guide
[21] https://doc.openfoam.com/2312/tools/processing/numerics/schemes/
[22] https://www.openfoam.com/current-release
[23] https://precice.org/
[24] https://precice.org/docs
[25] https://precice.org/quickstart
[26] https://precice.org/adapters-overview
[27] https://precice.org/tutorials-flow-over-heated-plate

Project E: 2D-rotational Coupling heat transfer ↔ CFD+chemical reaction

We are going to solve the problem on page 6 using code *CUBE_RISER_SLEEVE_2D* and combine it with our adapted teaching code for the rotational symmetric heat equation (2) from page 2 in Ω^S. 2026_Vasco/CUBE

1. Run the code *CUBE_RISER_SLEEVE_2D*.
2. Determine the geometry of Ω^S.
3. Setup/Change the teaching solver:
   (a) Generate the 2D mesh with the material domain Ω^S.
   Derive the geometry description by changing *generate_mesh/chip_2materials.m* and using matlab. shm/generate_mesh
   The script above generates two files *chip_2materials.txt* (coordinates of vertices and the finite element connectivity) and *chip_2materials_sd.txt* (material number per element).
   (b) Follow the code structure in *mgrid_2/main.cpp* for the Jacobi branch (`#undef MG`) and check whether the stationary Dirichlet problem can be solved. shm/mgrid_2
   (c) Change the boundary conditions from Dirichlet b.c. to Robin b.c..
   Derive a method `ApplyRobinBC` from `ApplyDirichletBC` in class `FEM_Matrix`.
   (d) Implement for class `FEM_Matrix` a new Method `CalculateLaplace_rot` and a method `ApplyRobinBC_rot` that realize the finite element matrix computation for the Laplace part (without $\frac{\partial u}{\partial t}$) in equ. (2).
   (e) Implement a method `AddMass_rot` that realizes internally the calculation of the mass matrix entries $M_{i,j} := \int\int r c(r,z)\phi_i\phi_j drdz$ similar to the element calculation for the Laplace part of the matrix $K_{i,j}$ and adds it it to the existing matrix entries.
   (f) Use an explicit scheme to discretize in time so that we end up with the discrete formulation of (2)
   $$\left(\tfrac{1}{\tau}M + K\right)\underline{u}^{k+1} = \underline{f}^{k+1} + \tfrac{1}{\tau}M\cdot\underline{u}^k \tag{4}$$
   with τ denoting the chosen time step.
   (g) Solve (3).
4. **Couple** the two solvers via *preCICE*.
   - Use the changed solver for the heat conductivity problem shm/mgrid_2
   - Use preCICE for coupling (intro[28], docu[29], quickstart[30], adapters[31], fluid-temperature[32]).
   - Have a look at the example code in f90, especially files *precice_config.xml* and *main_precice.f90* . shm/mch

   Start with one time step for the stationary problem.
5. Solve the coupled time-dependent problem.

[28] `https://precice.org/`
[29] `https://precice.org/docs`
[30] `https://precice.org/quickstart`
[31] `https://precice.org/adapters-overview`
[32] `https://precice.org/tutorials-flow-over-heated-plate`