pyelmer

Python interface for Elmer workflow

Elmer Webinar series

Espoo, Finland / online, March 11 – May 13, 2021

Arved Enders-Seidlitz

Kaspars Dadzis

Introduction

Arved Enders-Seidlitz, M. Sc.

Engineering Science

PhD Student
„Model development and validation for crystal growth process simulation“

Dr. Kaspars Dadzis

University of Latvia
Solar World / Uni. Freiberg

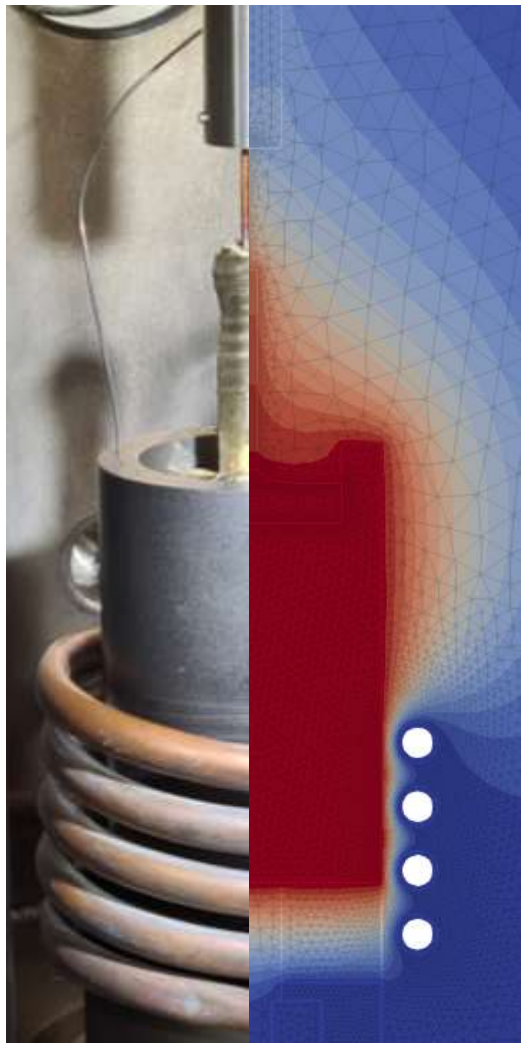
Group Leader „Model Experiments“
NEMOCRYS Project
funded by ERC Starting Grant

Leibniz Institute for Crystal Growth

Germany, Berlin-Adlershof

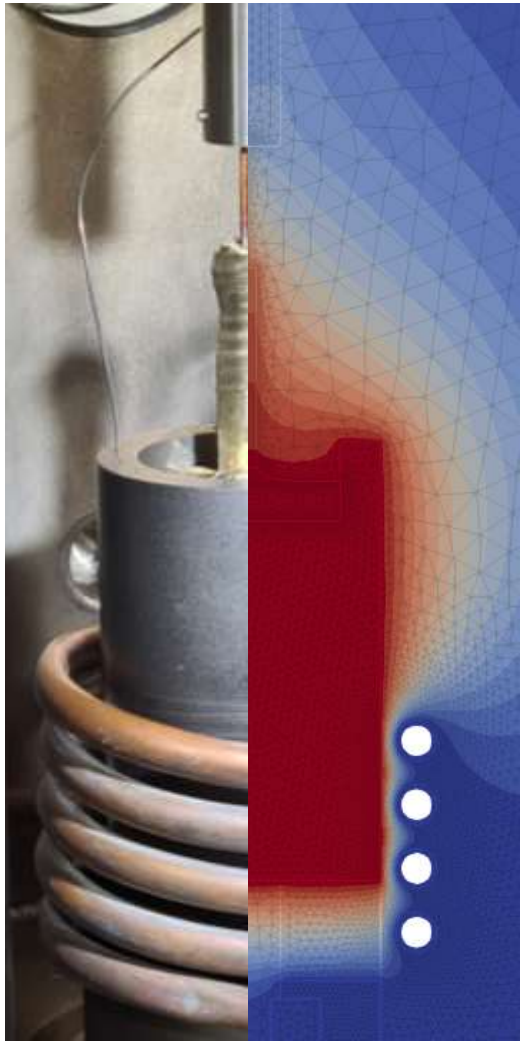
- Fundamental to pre-industrial research
- Development, characterization and production of crystalline materials

Content



- Motivation
- Getting started
- Examples & background
- Application in crystal growth
- Contribution & summary

Content



Motivation

Getting started

Examples & background

Application in crystal growth

Contribution & summary

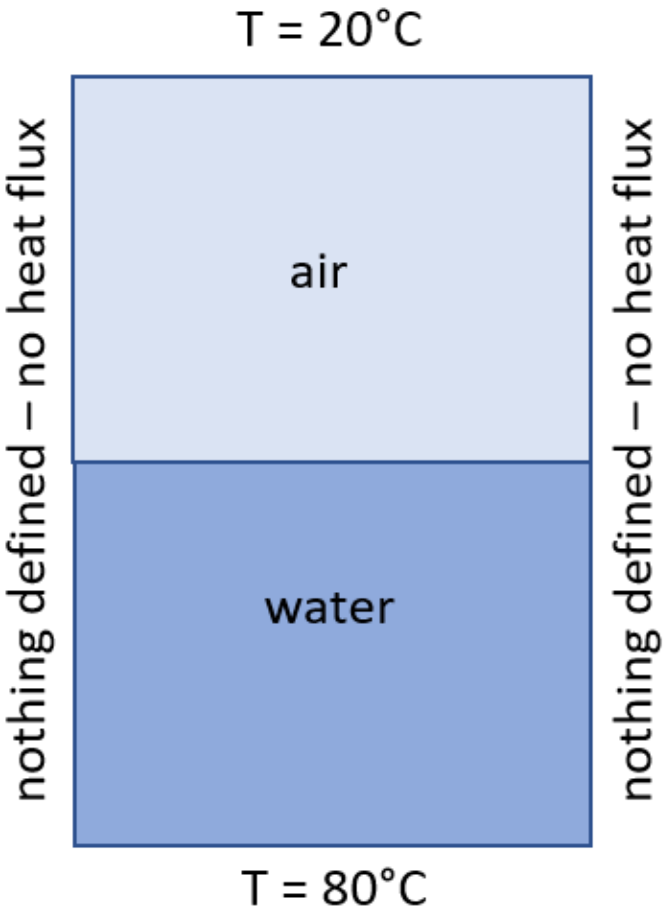
Motivation – the classical workflow

The classical workflow

1. Write *geo*-file, run Gmsh, export mesh
2. Write Elmer *sif*-file
3. Run ElmerGrid, ElmerSolver

Heating up water

- 2D steady state
- Water and air
- Heating at bottom
- Cold top surface
- No heat flux over boundaries
- Only heat conduction *(non-physical!)*



Motivation – the classical workflow

1. **Write *geo*-file, run Gmsh, export mesh**
2. Write Elmer *sif*-file
3. Run ElmerGrid, ElmerSolver

Workflow

- Create points, …
- … lines from points, …
- … loops from lines, …
- … surfaces from loops, …
- … and physical groups.

- Complex numbering required

```
Point(1) = {0, 0, 0};
Point(2) = {1, 0, 0};
Point(3) = {1, 1, 0};
Point(4) = {0, 1, 0};
Point(5) = {1, 2, 0};
Point(6) = {0, 2, 0};

Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};
Line Loop(101) =  {1, 2, 3, 4};
Plane Surface(101) = {101};

Line(5) = {5, 3};
Line(6) = {6, 5};
Line(7) = {4, 6};
Line Loop(102) = {7, 6, 5, 3};
Plane Surface(102) = {102};

Physical Surface ("Water") = {101};
Physical Surface ("Air") = {102};

Physical Line ("Bottom") = {1};
Physical Line ("Top") = {6};

Mesh(2);
Save "mesh.msh";
```

Motivation – the classical workflow

1. Write *geo*-file, run Gmsh, export mesh
2. **Write Elmer *sif*-file**
3. Run ElmerGrid, ElmerSolver

Workflow

- Copy setup from Elmer test cases
- Adapt solver parameter as far as you understand them
- Insert bodies, materials, boundaries
- Set relations by IDs

```
Header
  CHECK KEYWORDS "Warn"
  Mesh DB "." "."
End

Simulation
  Max Output Level = 4
  Coordinate System = Cartesian 2D
  Simulation Type = Steady state
  Steady State Max Iterations = 10
End

Equation 1
  Active Solvers(1) = 1
End

Solver 1
  Equation = HeatSolver
  Procedure = "HeatSolve" "HeatSolver"
  Variable = "Temperature"
  Variable Dofs = 1
  Calculate Loads = True
  Exec Solver = Always
  Nonlinear System Convergence Tolerance = 1e-06
  Nonlinear System Max Iterations = 1000
  Nonlinear System Relaxation Factor = 0.7
  Steady State Convergence Tolerance = 1e-06
  Stabilize = True
  Optimize Bandwidth = True
  Linear System Solver = Iterative
  Linear System Iterative Method = BiCGStab
  Linear System Max Iterations = 1000
  Linear System Preconditioning = ILU
  Linear System Precondition Recompute = 1
  Linear System Convergence Tolerance = 1e-08
  Linear System Abort Not Converged = True
  Linear System Residual Output = 1
End

Solver 2
  Exec Solver = After timestep
  Equation = ResultOutputSolver
  Procedure = "ResultOutputSolve" "ResultOutputSolver"
  VTU Format = True
  Save Geometry Ids = Logical True
End
```

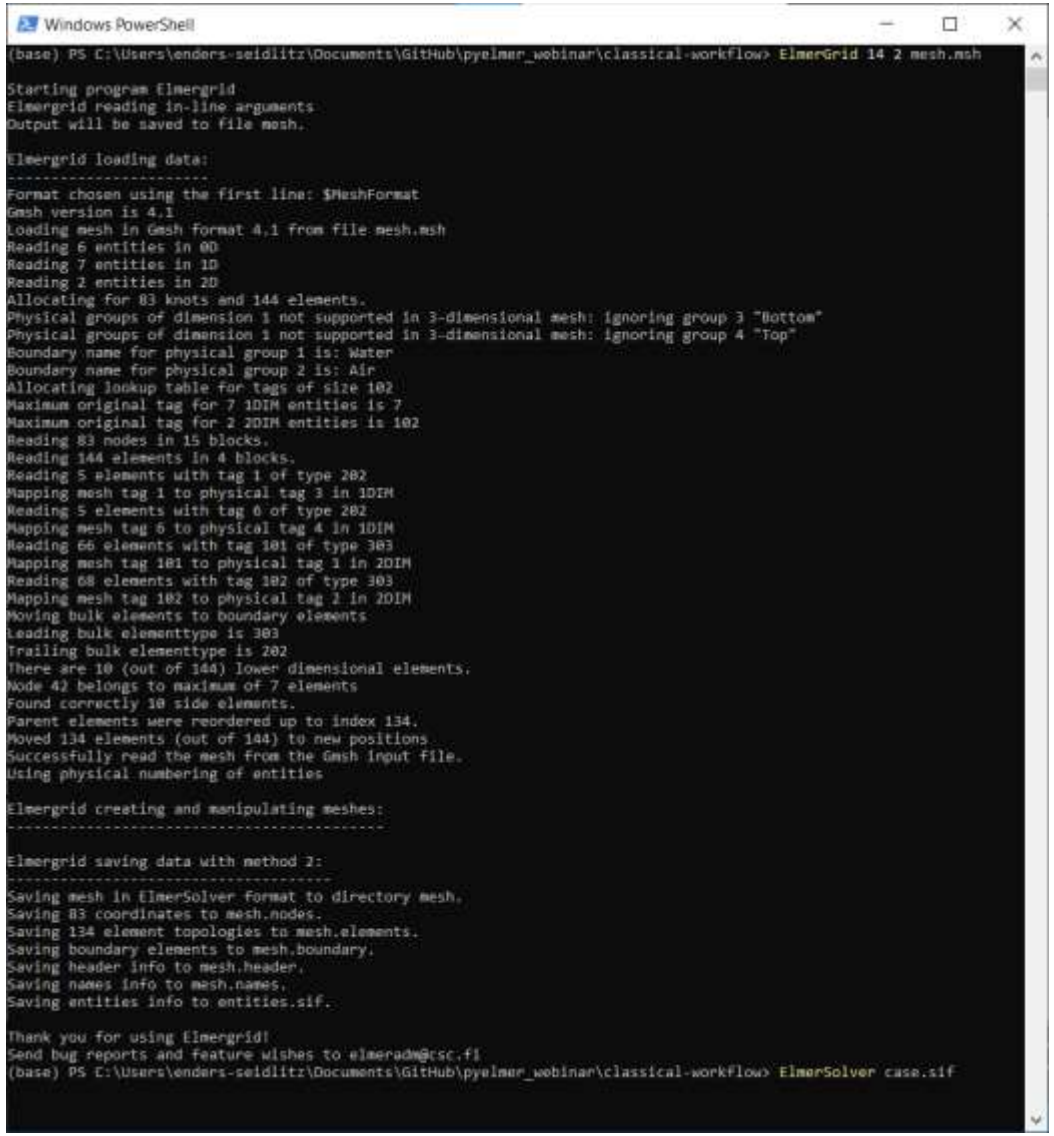
```
Material 1
  Density = 1.1885
  Heat Capacity = 1006.4
  Heat Conductivity = 0.025873
End

! water
Material 2
  Density = 1000.0
  Heat Capacity = 4182.0
  Heat Conductivity = 0.6
End

Body 1
  Target Bodies(1) = 1
  Equation = 1  ! main
  Initial Condition = 1  ! T0
  Material = 2  ! water
End

Body 2
  Target Bodies(1) = 2
  Equation = 1  ! main
  Initial Condition = 1  ! T0
  Material = 1  ! air
End

Boundary Condition 1
  Target Boundaries(1) = 3
  Temperature = 353.15
End

Boundary Condition 2
  Target Boundaries(1) = 4
  Temperature = 293.15
End

Initial Condition 1
  Temperature = 273.15
End
```

Motivation – the classical workflow

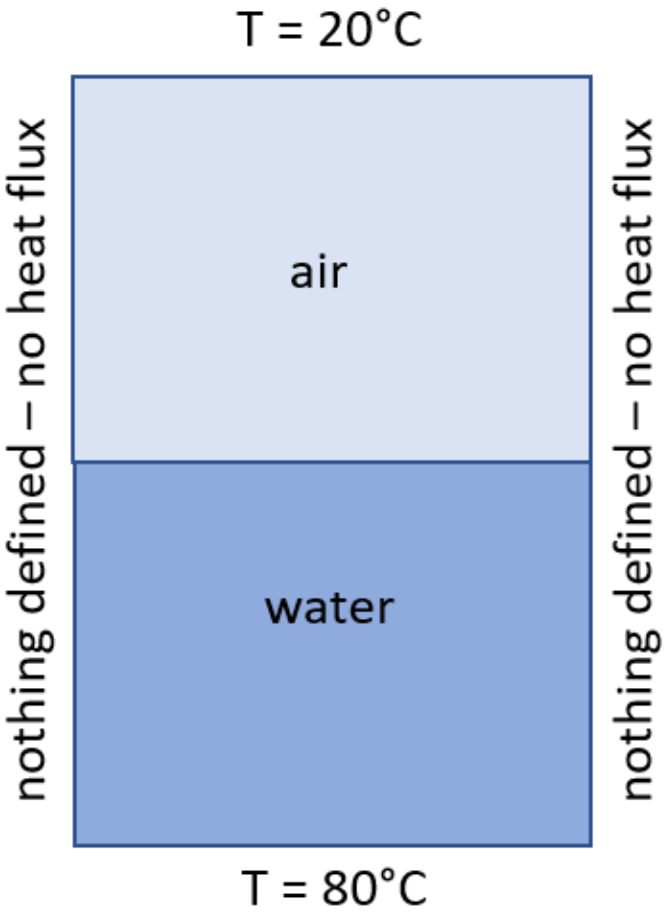
1. Write *geo*-file, run Gmsh, export mesh
2. Write Elmer *sif*-file
3. **Run ElmerGrid, ElmerSolver**

Workflow

- Open shell
- Execute ElmerGrid
- Execute ElmerSolver
- (Read solver output)



Motivation – the classical workflow



Conclusion

- Hard to understand
- Easy to make mistakes
- Difficult to debug

- Too much manual work
- Too many IDs

Motivation – the pyelmer workflow

Create clean code…

Keep it simple, stupid (KISS)

- Everything in one place
  - Geometry, mesh
  - Simulation setup, control
- Easy to read, write and understand

Do not repeat yourself (DRY)

- Default parameters stored separately
  - Solvers, Materials
- Re-usable by modular design

…for beginners and experts

Programming language: Python

- Most popular
- Infrastructure and feature set
- **…**

Open parameter space

- Default solvers for beginners
- Customizable for experts

Motivation – the pyelmer workflow

```python
import os, gmsh, pyelmer

sim_dir = "./simdata"
if not os.path.exists(sim_dir):
    os.mkdir(sim_dir)

gmsh.initialize()
gmsh.option.setNumber("General.Terminal", 1)
gmsh.model.add("heat-transfer-2d")
factory = gmsh.model.occ

water = factory.addRectangle(0, 0, 0, 1, 1)
air = factory.addRectangle(0, 1, 0, 1, 1)
factory.synchronize()
factory.fragment([(2, water)], [(2, air)])
factory.synchronize()
ph_water = pyelmer.gmsh.add_physical_group(2, [water], "water")
ph_air = pyelmer.gmsh.add_physical_group(2, [air], "air")

line = pyelmer.gmsh.get_boundaries_in_box(0, 0, 0, 1, 0, 0, 2, water)
ph_bottom = pyelmer.gmsh.add_physical_group(1, [line], "bottom")
line = pyelmer.gmsh.get_boundaries_in_box(0, 2, 0, 1, 2, 0, 2, air)
ph_top = pyelmer.gmsh.add_physical_group(1, [line], "top")

gmsh.model.mesh.setSize(gmsh.model.getEntities(0), 0.1)
gmsh.model.mesh.generate(2)
gmsh.write(sim_dir + "/case.msh")

sim = pyelmer.elmer.load_simulation("2D_steady")
air = pyelmer.elmer.load_material("air", sim)
water = pyelmer.elmer.load_material("water", sim)
solver_heat = pyelmer.elmer.load_solver("HeatSolver", sim)
solver_output = pyelmer.elmer.load_solver("ResultOutputSolver", sim)
eqn = pyelmer.elmer.Equation(sim, "main", [solver_heat])
T0 = pyelmer.elmer.InitialCondition(sim, "T0", {"Temperature": 273.15})

bdy_water = pyelmer.elmer.Body(sim, "water", [ph_water])
bdy_water.material = water
bdy_water.initial_condition = T0
bdy_water.equation = eqn

bdy_air = pyelmer.elmer.Body(sim, "air", [ph_air])
bdy_air.material = air
bdy_air.initial_condition = T0
bdy_air.equation = eqn

bndry_bottom = pyelmer.elmer.Boundary(sim, "bottom", [ph_bottom])
bndry_bottom.fixed_temperature = 353.15  # 80 °C
bndry_top = pyelmer.elmer.Boundary(sim, "top", [ph_top])
bndry_top.fixed_temperature = 293.15  # 20 °C

sim.write_startinfo(sim_dir)
sim.write_sif(sim_dir)

pyelmer.execute.run_elmer_grid(sim_dir, "case.msh2")
pyelmer.execute.run_elmer_solver(sim_dir)

err, warn, stats = pyelmer.post.scan_logfile(sim_dir)
print(„\nErrors:", err, „\nWarnings:", warn, „\nStatistics:", stats)
```

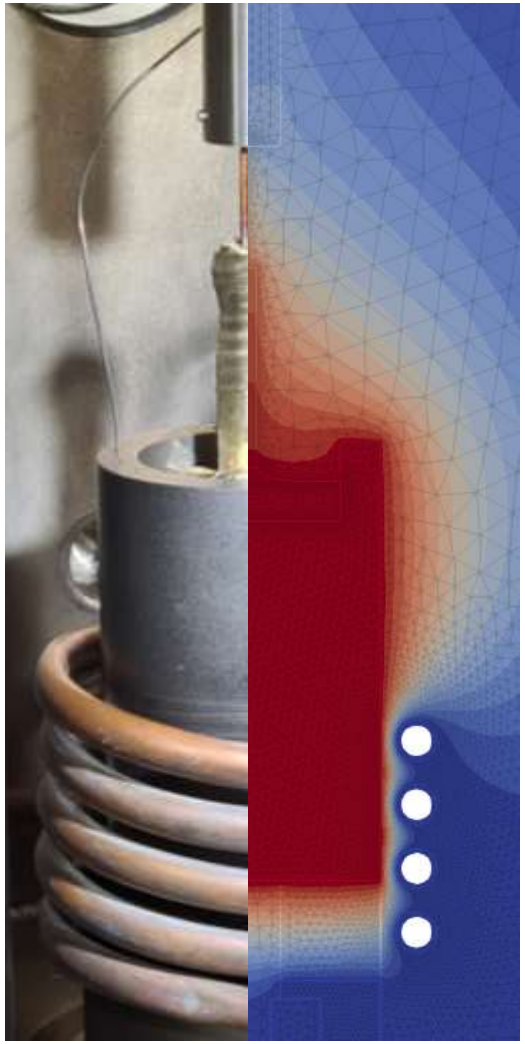
pyelmer

- Gmsh python API
- Object-oriented relation management
- Execute & evaluate

- 50 lines of code for “heating up water” case

  (geo-file: 23 lines, sif-file: 74 lines)

Content



Motivation

Getting started

Examples & background

Application in crystal growth

Contribution & summary

Getting started with pyelmer

Where to find it

- https://github.com/nemocrys/pyelmer
- https://pypi.org/project/pyelmer/

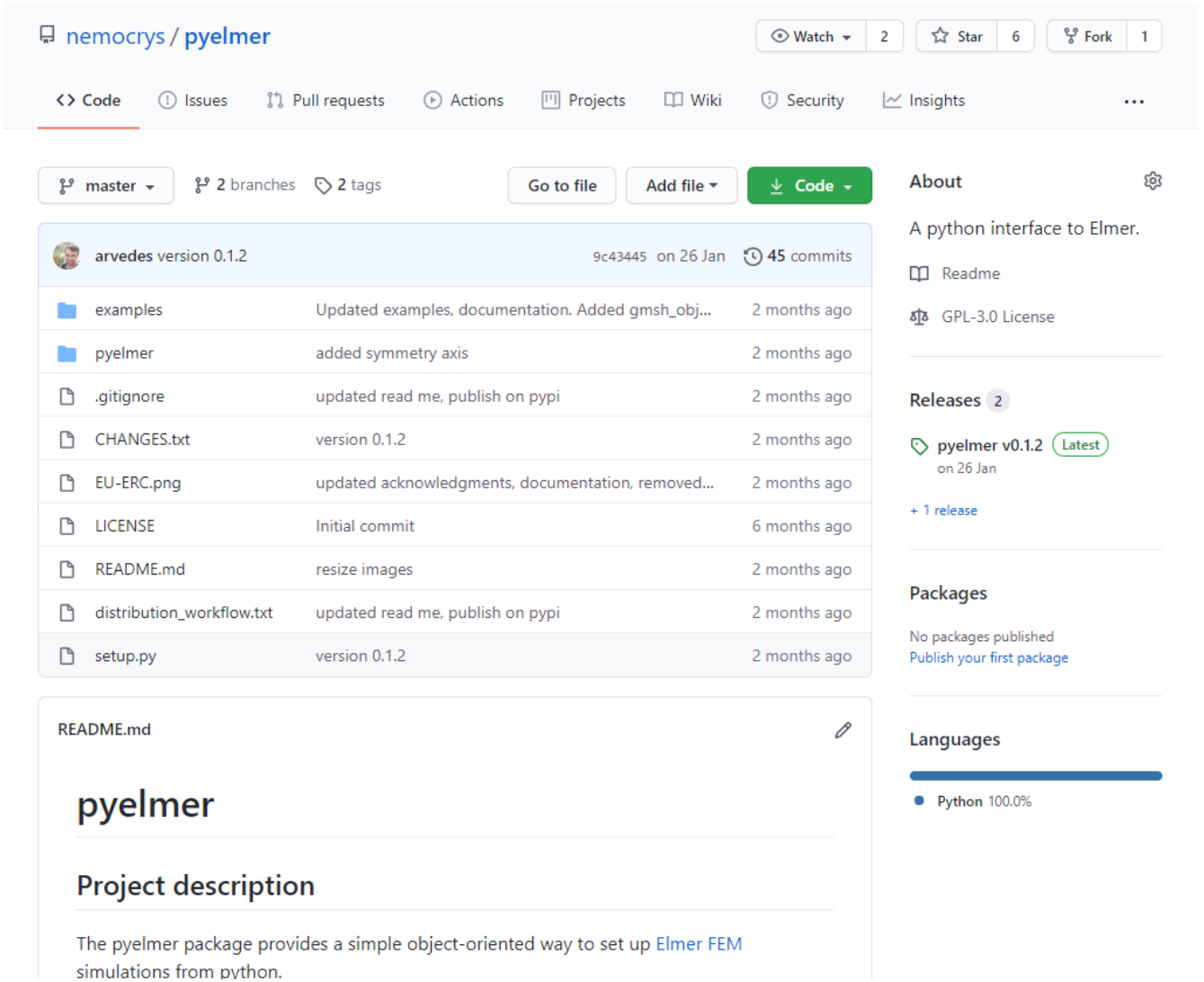
How to install it

- `pip install pyemer`
- ElmerSolver, ElmerGrid executable on path

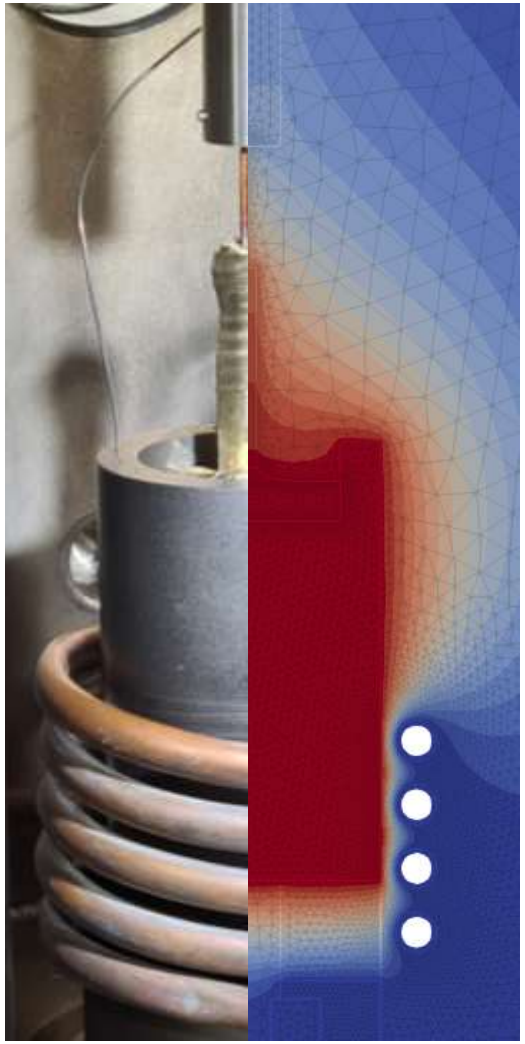
What to do with it

- Examples in readme on Github
- Or in examples folder
- Docstrings in source code

License: GPL v3



Content



Motivation

Getting started

Examples & background

Application in crystal growth

Contribution & summary

Example 1 – heating up water

https://youtu.be/KZ1N2FM6J7E

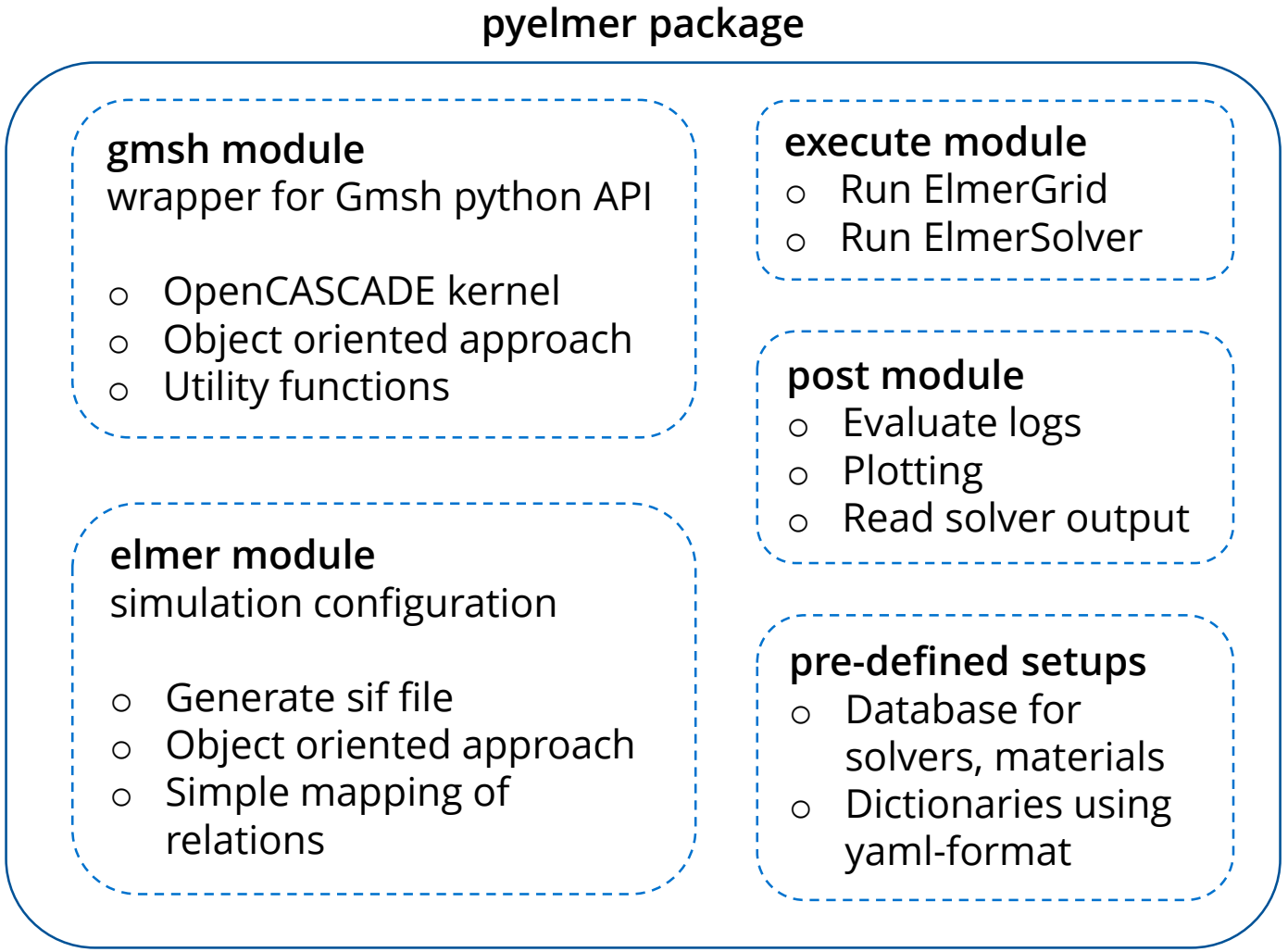
Background – design of pyelmer

Focused on workflow

- Mesh generation in Python using Gmsh
- Setup with in Python
- Simulation with ElmerSolver
- Postprocessing with paraview, pyelmer, custom python scripts

Design considerations

- Follow Elmer defined structure (sif file)
- Provide open keyword space
- Fully automatize the annoying stuff



Example 2 – crystal growth

https://youtu.be/0lnX-ytKTH8

Background – gmsh module

pyelmer package

gmsh module
wrapper for Gmsh python API

- OpenCASCADE kernel
- Object oriented approach
- Utility functions

execute module
- Run ElmerGrid
- Run ElmerSolver

post module
- Evaluate logs
- Plotting
- Read solver output

elmer module
simulation configuration

- Generate sif file
- Object oriented approach
- Simple mapping of relations

pre-defined setups
- Database for solvers, materials
- Dictionaries using yaml-format

Mesh generation with gmsh

- gmsh python API
  - Native way to use it
  - Complex and cumbersome
- pyelmer.gmsh
  - Object-oriented top-to-bottom approach (with OpenCASADE kernel)
  - Utility functions for gmsh python API
- pygmsh
  - Independent gmsh interface
  - Improved workflow

Background – gmsh module

Classes for geometry modeling and mesh generation

`class Model`

- Collect geometries (Shape objects)
- Generate and export mesh
- Global Gmsh operations

`class Shape`

- Wrapper for any kind of shape:

  Bodies, surfaces, lines, …
- Defined by Gmsh geometry ID
- Simplifies Gmsh operations: Extract boundaries

Classes for mesh control

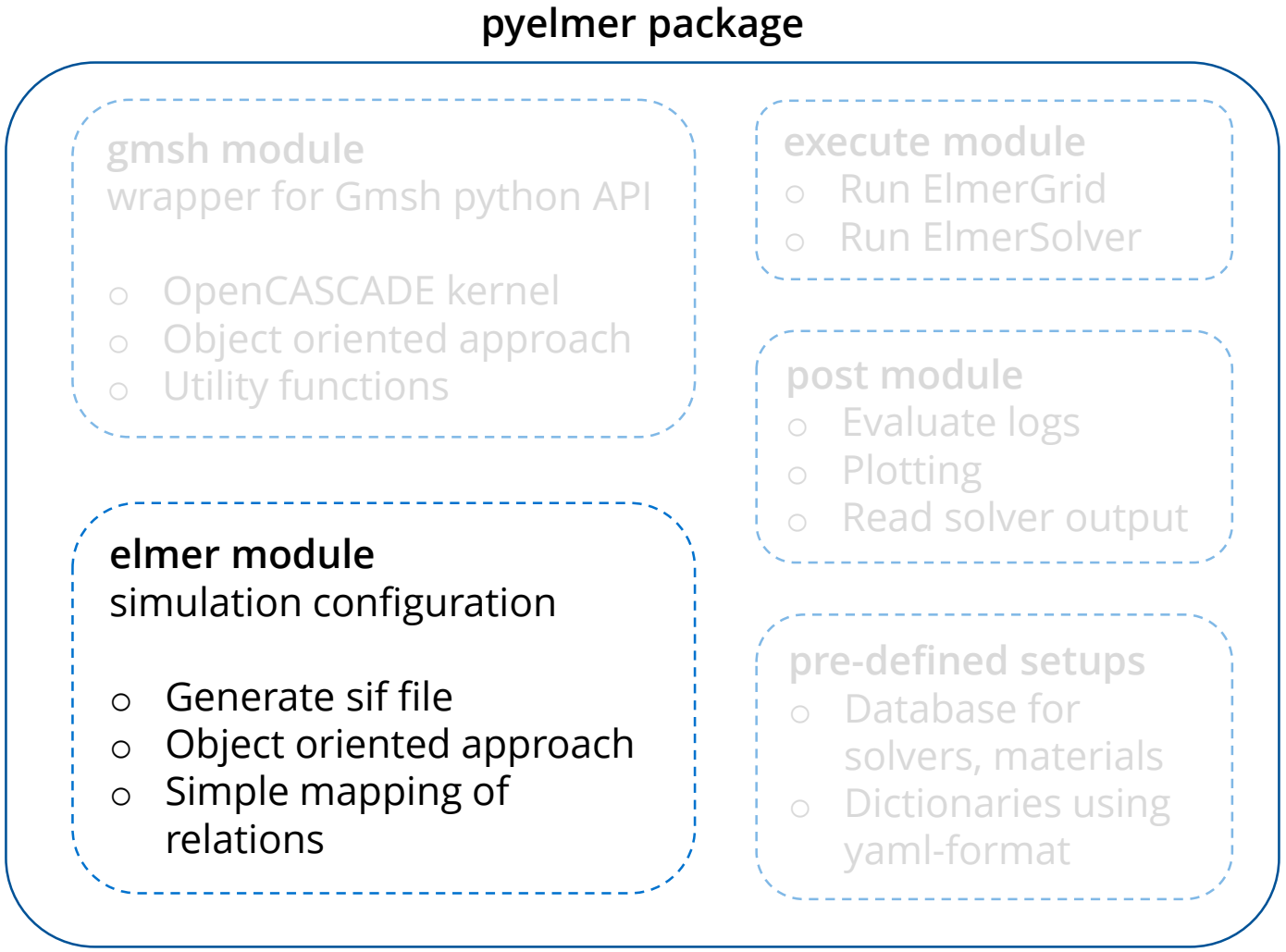
- `class MeshControlConstant`
- `class MeshControlLinear`
- `class MeshControlExponential`

Working principle

- Wrapper for Gmsh API
- In case of errors: plot the model

```
model.synchronize()
model.show()
```

Background – elmer module



`class Simulation`

- Global settings
- Sub-components
- Writes sif-file

Classes for sub-compontens

`Solver, Equation, Material, Body, BodyForce, InitialCondition, Boundary`

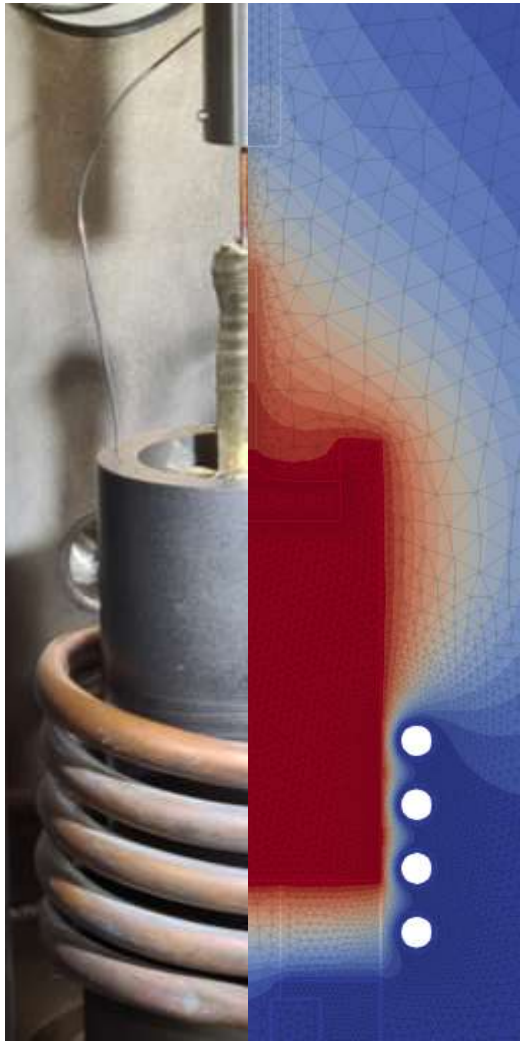
Functions to load setups

`Simulation, Solver, Material`

Background – elmer module

Let’s have a look at the code!

Content



Motivation

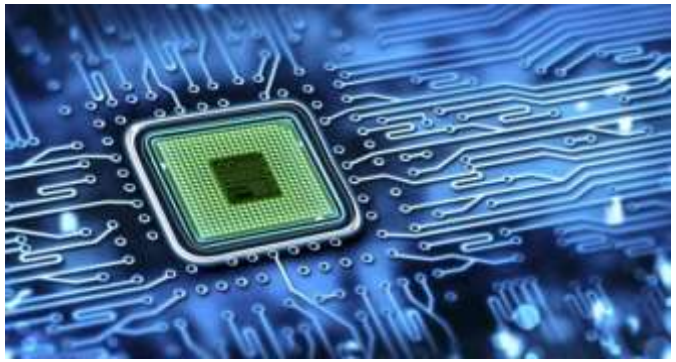
Getting started

Examples & background

Application in crystal growth

Contribution & summary

Application in crystal growth



http://www.knoda.org/back-history-discovery-very-first-silicon-chip-digital-computers/

https://cen.acs.org/energy/solar-power/Supercharging-silicon-solar-cell/97/web/2019/07

Computer technology, solar energy

https://www.sciencedirect.com/topics/chemistry/czochralski-process

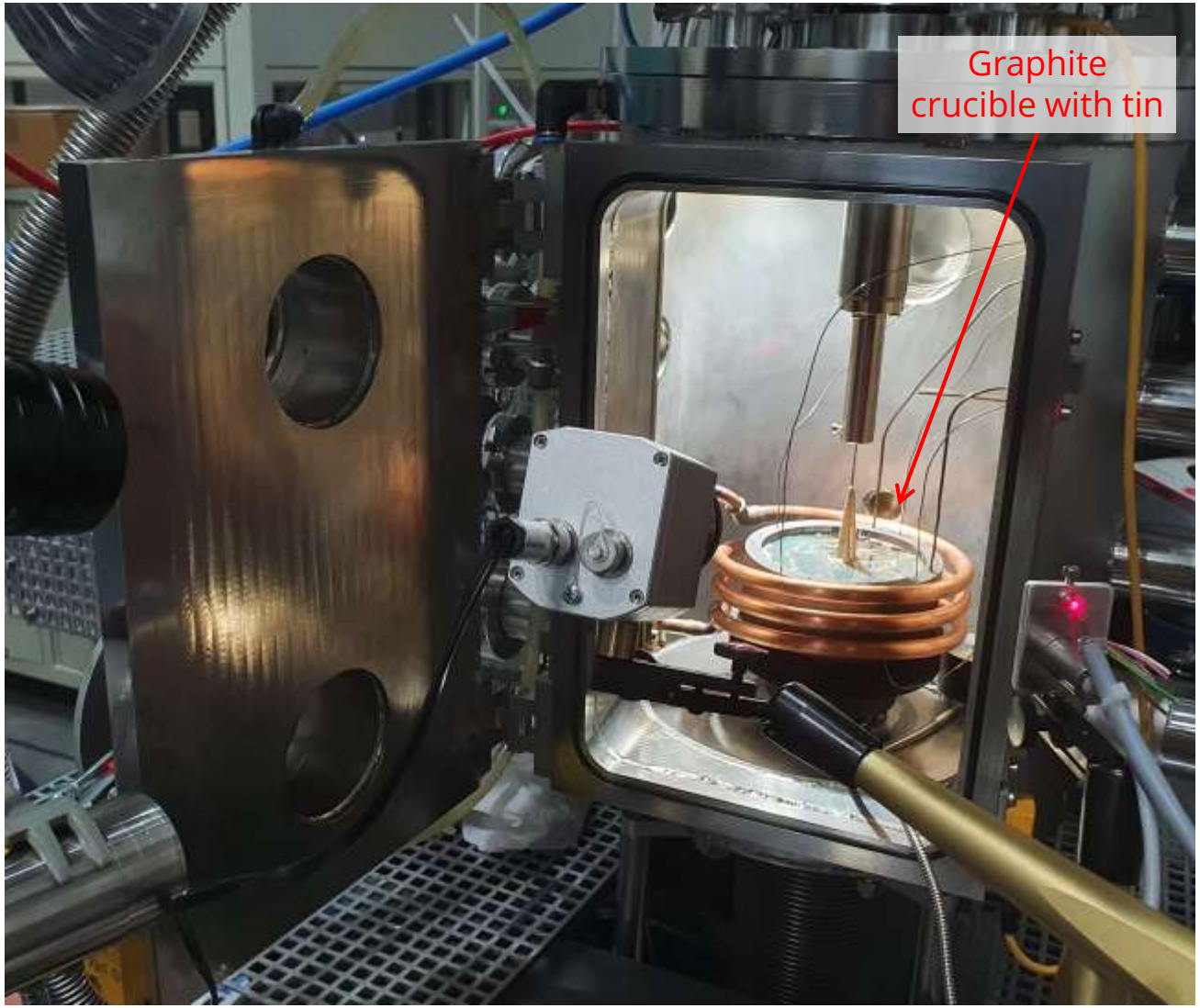
Silicon single crystal

https://www.pvatepla-cgs.com/anlagen/czochralski/

Silicon growth furnace

Application in crystal growth

NEMOCRYS: Next Generation Multiphysical Models for Crystal Growth Processes



Model materials

- Tin, $T_{melt} = 232°C$
- Bismuth, NaNO3, …

Conditions

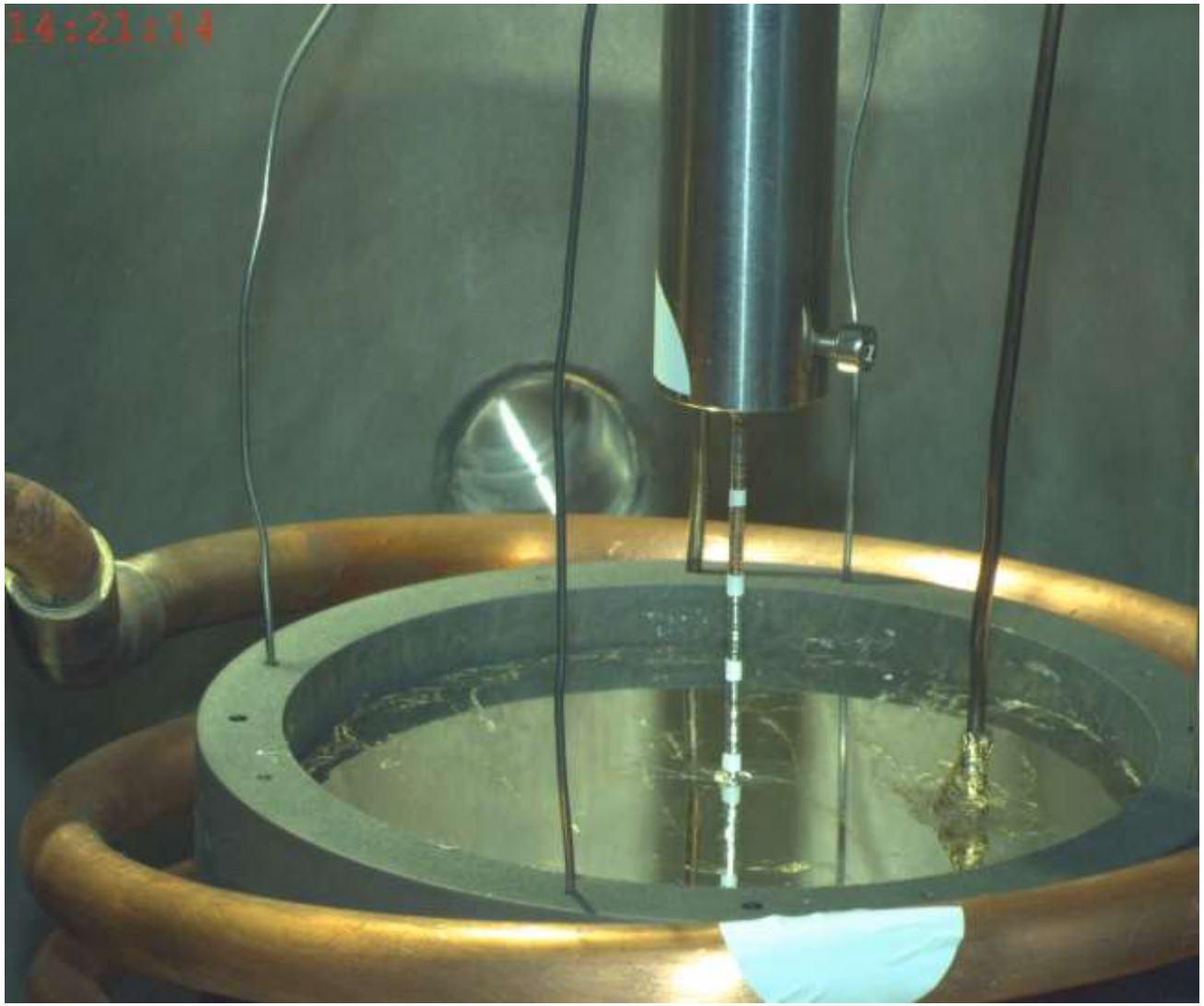
- Air atmosphere (cooling with fan)
- Vacuum

Measurements

- Temperatures
  - Thermocouples, Pt100
  - IR Camera
  - Pyrometer
- Electromagnetism
  - Heating power
  - Magnetic field
- Flows, thermal stresses

Application in crystal growth

NEMOCRYS: Next Generation Multiphysical Models for Crystal Growth Processes



Model materials

- Tin, $T_{melt} = 232°C$
- Bismuth, NaNO3, ...

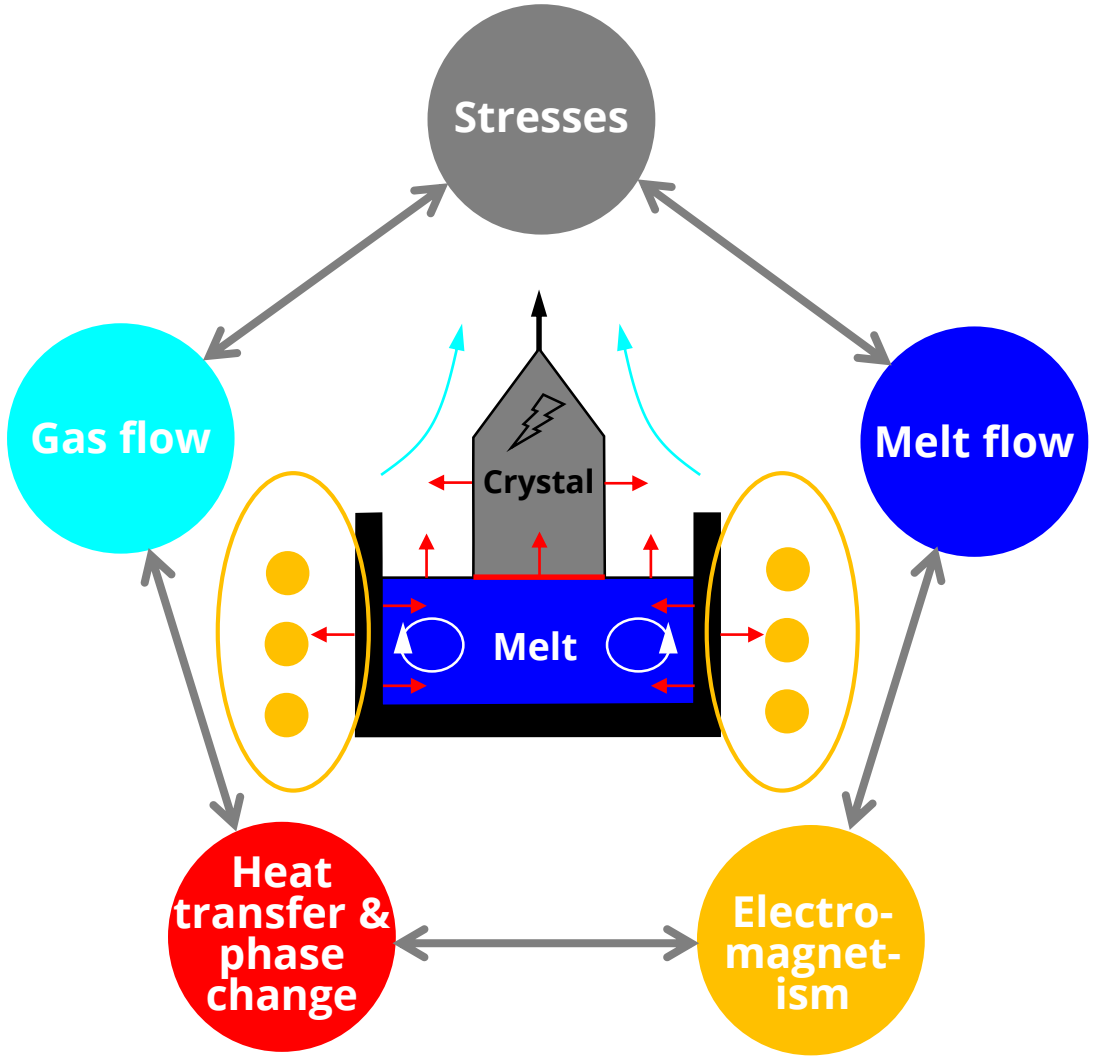
Conditions

- Air atmosphere (cooling with fan)
- Vacuum

Measurements

- Temperatures
  - Thermocouples, Pt100
  - IR Camera
  - Pyrometer
- Electromagnetism
  - Heating power
  - Magnetic field
- Flows, thermal stresses

Application in crystal growth



Numerical challenges

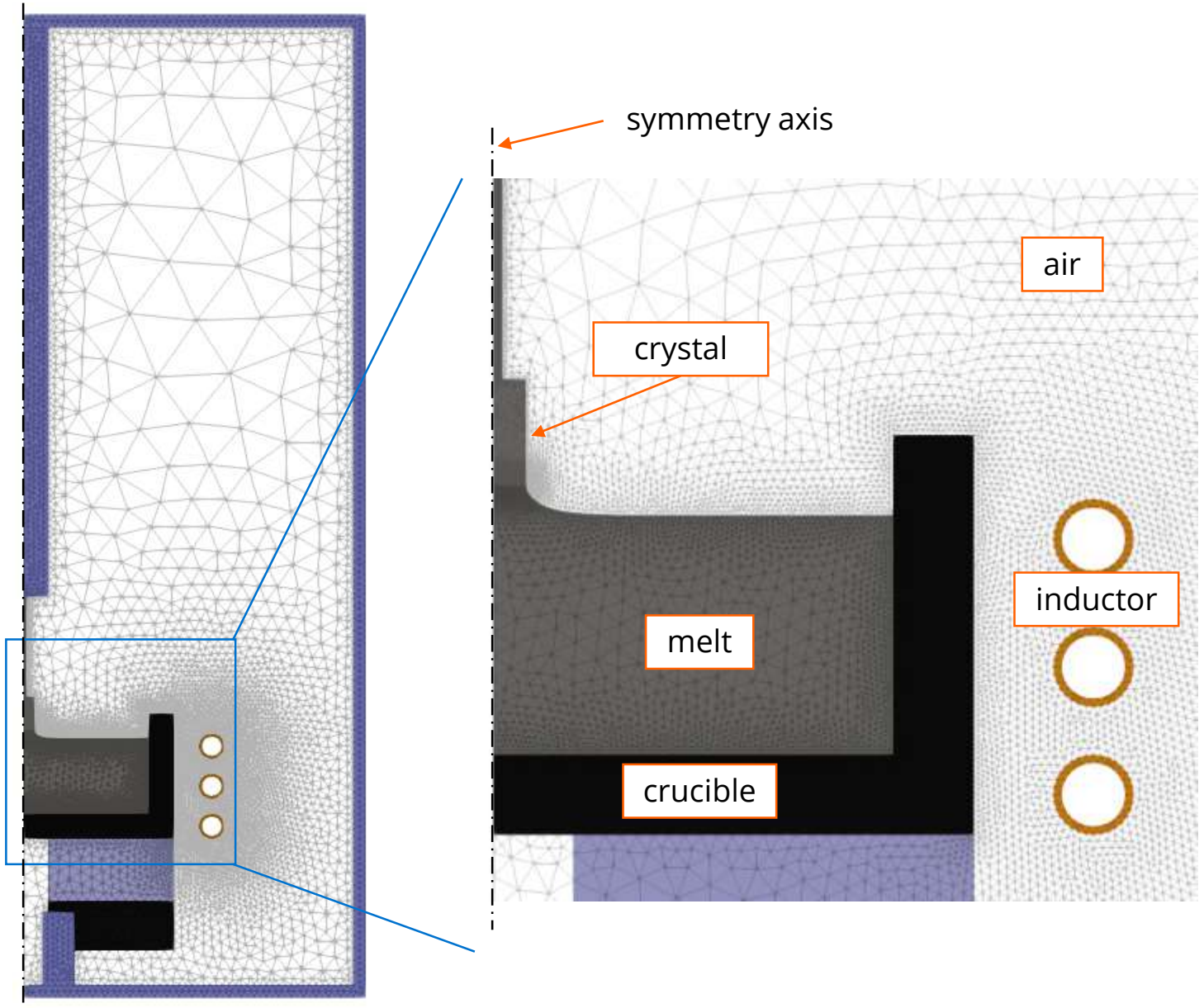
- Complex coupled physics
- Moving geometries
- Heat radiation
- …

Goals in NEMOCRYS Project

- Validation: Using model experiments
- Open source implementation

→ Elmer provides the best feature set

Application in crystal growth



Induction heating (2D axisymmetric, harmonic)

$$\nabla \times \left(\frac{1}{\mu} \nabla \times A_\varphi \boldsymbol{e}_\varphi \right) + i\omega\sigma A_\varphi \boldsymbol{e}_\varphi = j_\varphi$$

Heat transfer (2D axisymmetric)

$$\rho c_p \left(\frac{\partial T}{\partial t} + (\boldsymbol{u} \cdot \nabla) T \right) - \nabla \cdot (\lambda \nabla T) = \rho h$$

Phase change (2D axisymmetric, steady-state)

$$q = L\,\rho\,\boldsymbol{v} \cdot \boldsymbol{n}\,, \quad s_y = \left(y_{j,1} - y_i\right) + (x_i - x_{j,1}) \frac{y_{j,2} - y_{i,1}}{x_{j,2} - x_{j,1}}$$

Radiation (at solid/air boundaries)

$$-\lambda_k \frac{\partial T_k}{\partial \boldsymbol{n}_k} = \sigma_\varepsilon \varepsilon_k \left(T_k^4 - \frac{1}{A_k \varepsilon_k} \sum_{i=1}^{N} G_{ik} \varepsilon_i T_i^4 A_i \right)$$

P. Råback et al.: Elmer Models Manual, CSC – IT Center for Science, 10.11.2020. https://www.nic.funet.fi/pub/sci/physics/elmer/doc/

Application in crystal growth

Procedure using Elmer / Gmsh

Steady state simulation

↓ T ↓

Mesh generation (with new L)

↓

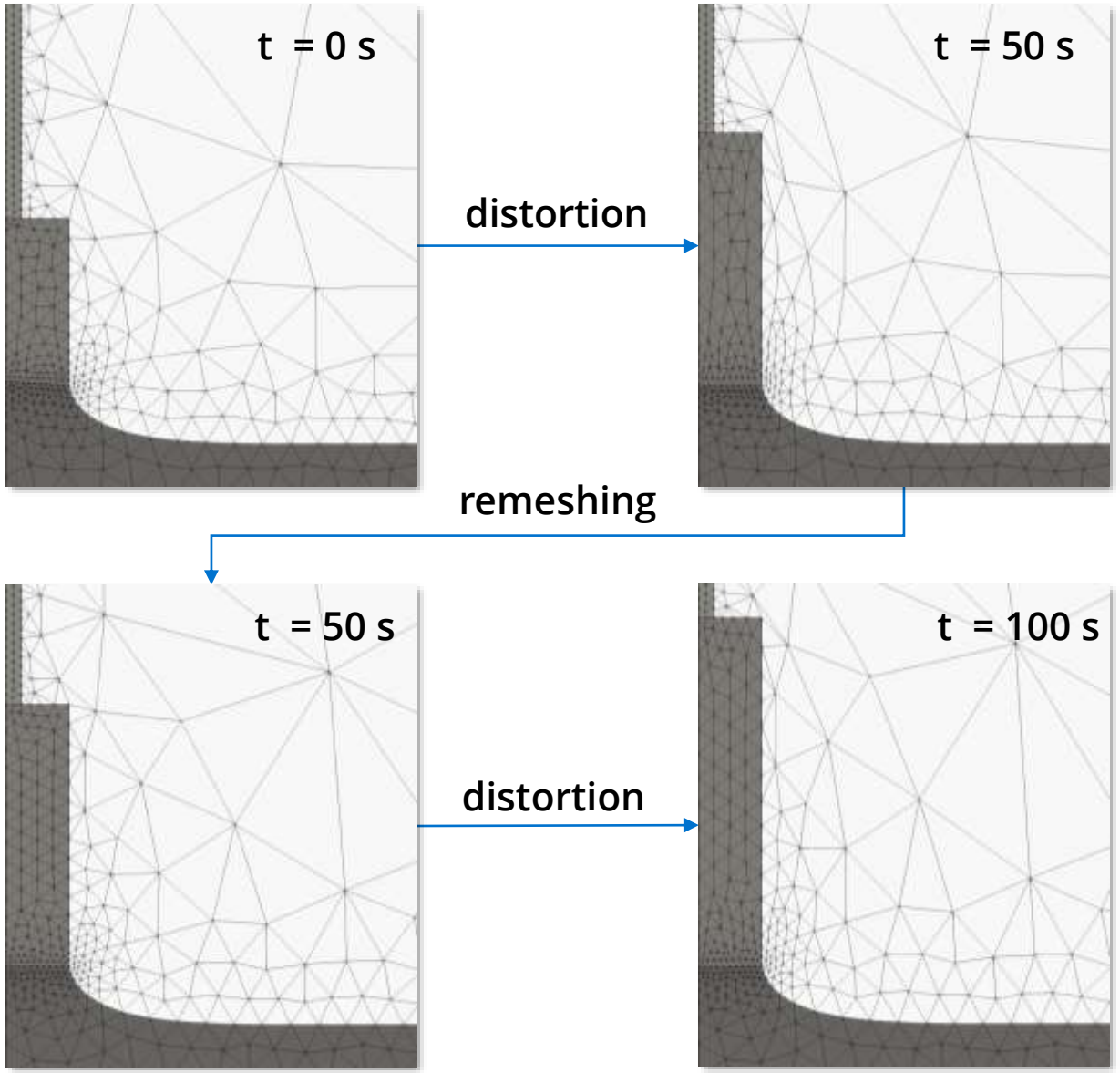
Mesh to mesh interpolation in Elmer

↓

Transient simulation
with mesh distortion

if $l_{crstal} < l_{max}$ | L, T ↓ *else*

Combination of sub-simulations to
one final result in Python



Application in crystal growth

Complex Elmer setup

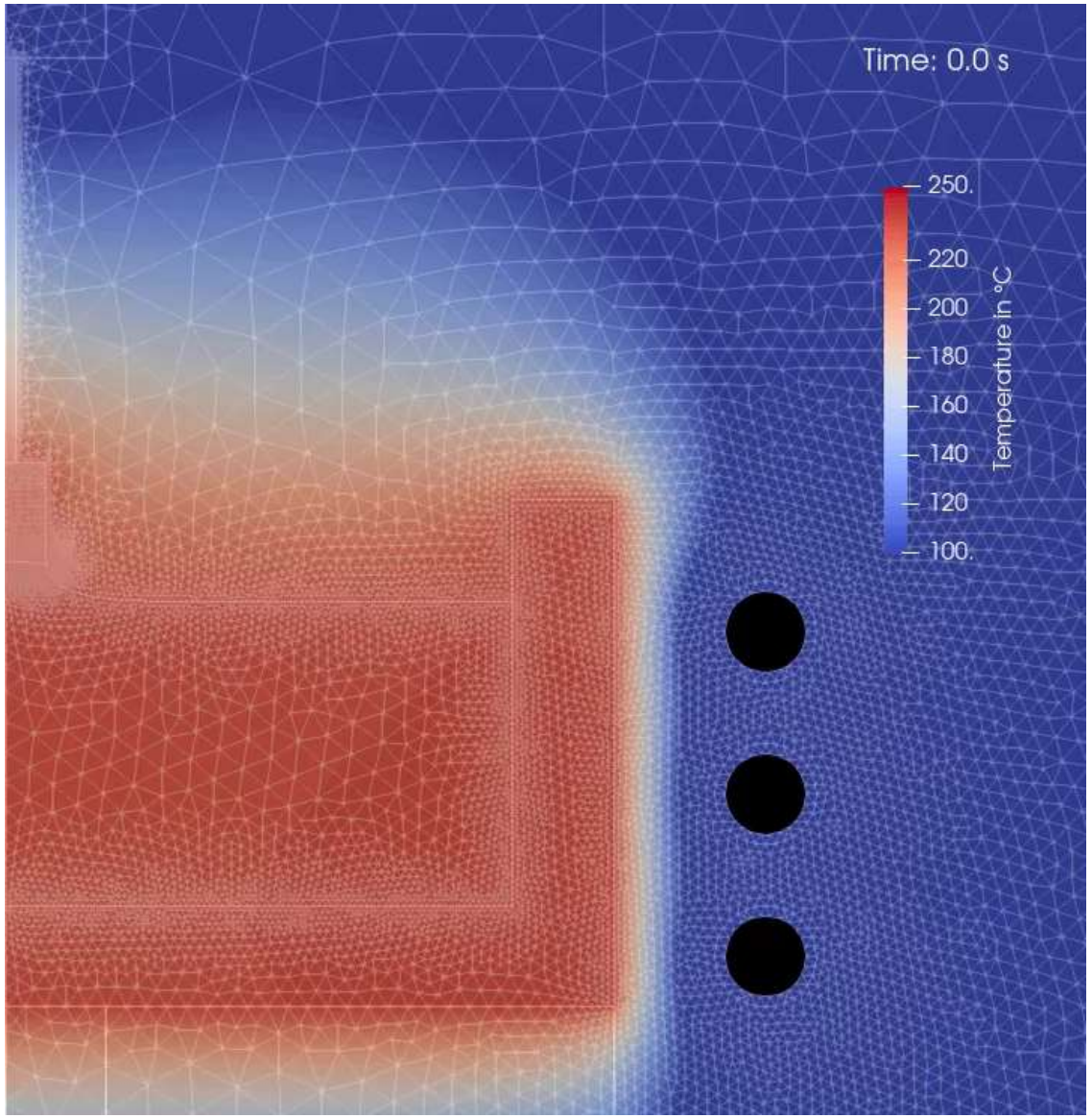
- 9 different solvers, e.g.
  - StatMagSolver / MagnetoDynamics2DHarmonic
  - HeatSolver
  - SteadyPhaseChange
- 12 bodies
- 25 boundary conditions

Combination of different simulation

- Initial steady state simulation
- Series of transient simulations with mesh update

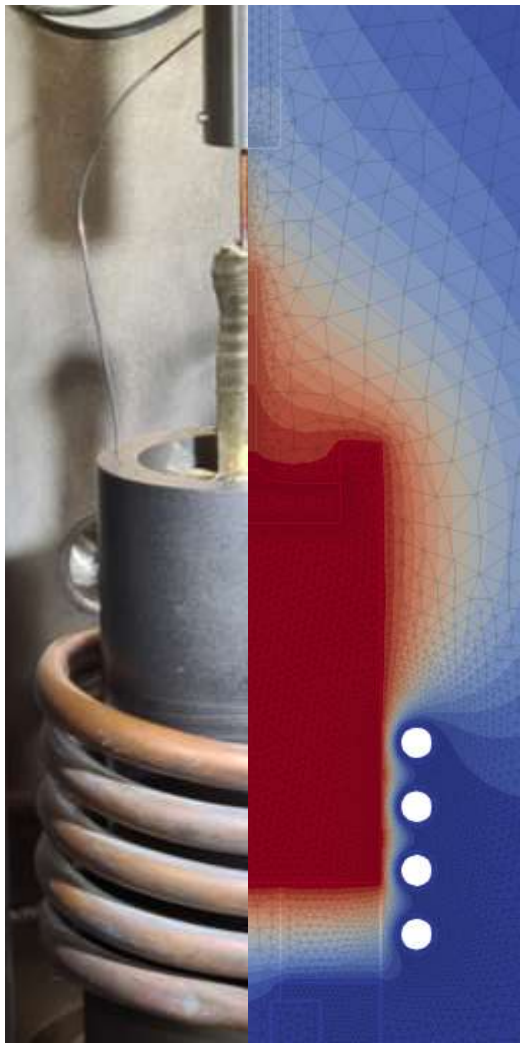
Parameter studies

- Material parameters: 35
- Shape parameters: 30
- Numerical parameters, e.g. mesh, convergence criteria



https://youtu.be/bM0DapM4Ngk

Content



Motivation

Getting started

Examples & background

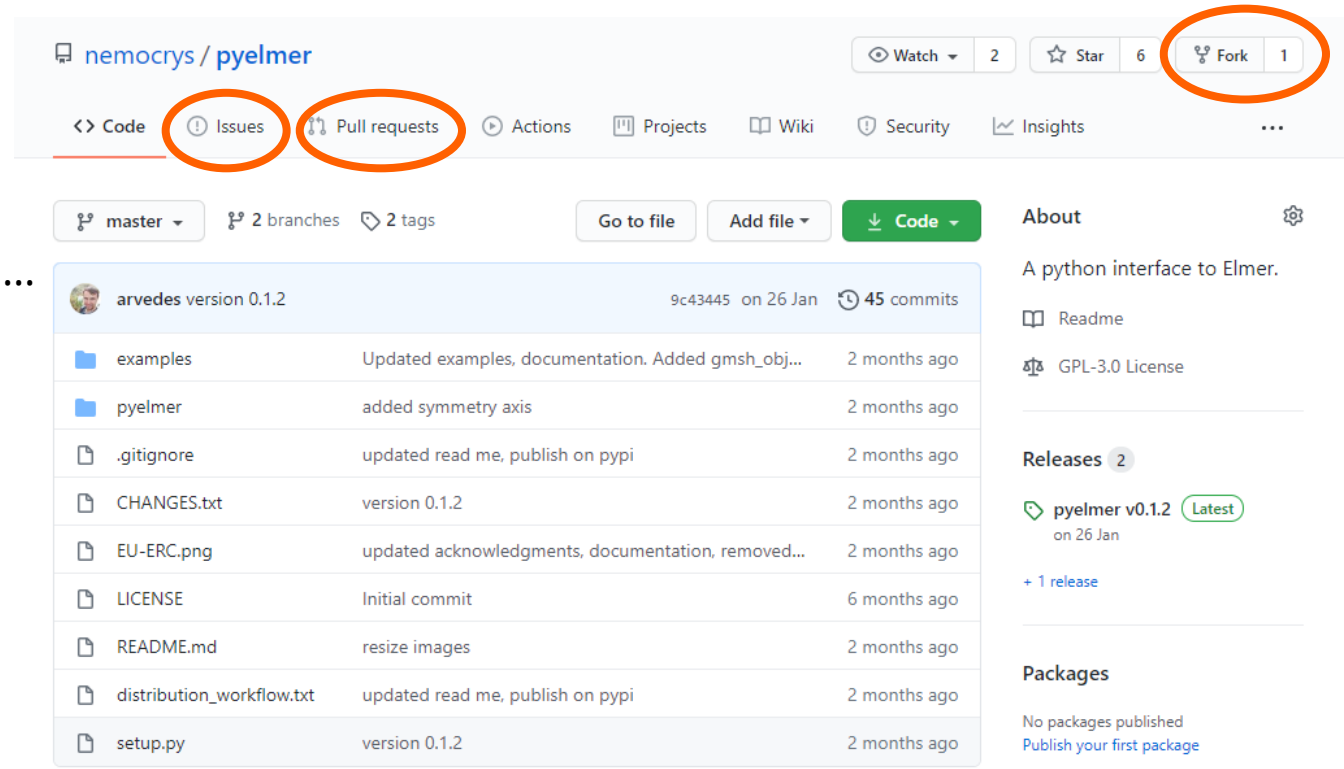
Application in crystal growth

Contribution & summary

Contribution

Open points in the pyelmer development

- Testing
- Documentation – conforming with Elmer docs
- New parameters for bodies, boundary conditions, …
- Add solvers, materials in yaml database
- Consistency checks (for sif file)
- MPI support
- Post processing
  - Evaluation of logs
  - Plot residuals
  - Read output to useful python data format (pandas?)
- # TODO in code
- Additional ideas welcome!



Use github: https://github.com/nemocrys/pyelmer

- Open issue for bugs / feature requests
- Create fork fork / pull request for contribution

Summary

Python workflow for Elmer

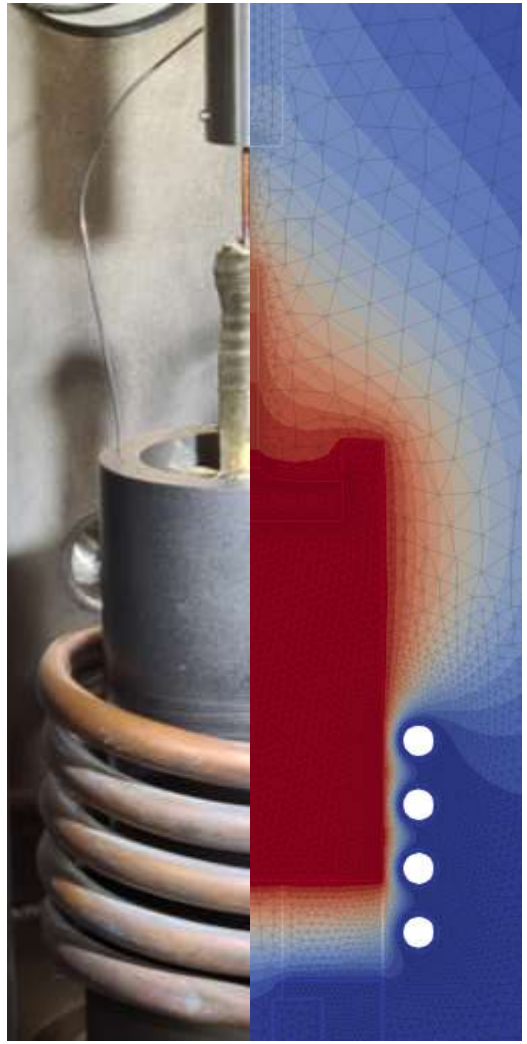
- Geometry & meshing with Gmsh
- Object oriented setup of Elmer

Different examples

- Easy to use for beginners
- High flexibility for all kinds of simulation
- Applied in complex crystal growth simulation

Support us

- Cite pyelmer: https://doi.org/10.5281/zenodo.4431440
- Give us a star on Github
- Provide feedback
- NEMOCRYS Project on ResearchGate



Acknowledgements

Thank you for your attention!

We'd like to acknowledge the developers of Elmer and Gmsh for creating such great open-source software.
Special thanks go to Peter Råback for his support regarding the usage of Elmer.

We'd like to thank Anil Kunwar for his contribution to pyelmer.

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 851768).