Thermal Twin (TES Digital Twin): Transfer of Knowledge Report

Project Summary for Handover - Alexandros Kinanis

December 17, 2025

Executive Summary

Thermal Twin is a configurable, scenario-driven thermal simulation framework designed to predict transient temperature evolution in composite heated assemblies (geopolymer blocks, heaters, bedding/contact media, and insulation layers). The pipeline couples: (1) parametric CAD construction and meshing using `gmsh`, (2) transient heat diffusion solving using FiPy (finite volume method on unstructured tetrahedra), and (3) telemetry/probe extraction and plotting for direct comparison to experimental measurements.

The system is intentionally modular. A user defines a physical test article using a **geometry JSON** (volumes and regions) and a **scenario YAML** (heater schedule, boundary conditions, probes, mesh sizes). Temperature-dependent material properties are provided in `materials.json`. This allows rapid iteration across experimental configurations (single block, multi-block stacks, different insulation levels) without changing solver code.

1 Repository Overview

1.1 High-level structure

Key directories:

- `src/thermal_twin/`: installable package.
- `configs/geometry/`: geometry JSON definitions (blocks, shells, heaters, gaps, pilot scheme).
- `configs/scenarios/`: scenario YAML definitions for meshing/simulation.
- `materials.json`: temperature-dependent material catalog.
- `outputs/`: meshes and run telemetry (ignored by git).
- `scripts/`: utilities for mesh visualization.

Core modules:

- **Geometry**: `src/thermal_twin/geometry/elements.py` defines element types and their gmsh OCC builders (block, cylinder, shell, plus a helical pipe sweep for the pilot).
- **Meshing**: `src/thermal_twin/meshing/builder.py` builds the gmsh model, fragments volumes, registers physical groups, applies mesh sizing, and exports `.msh`/`.vtu`.

- **Solver**: `src/thermal_twin/solver/core.py` loads the mesh into FiPy, assembles material fields, applies heater and boundary conditions, advances time, and streams telemetry.
- **Scenarios/CLI**: `src/thermal_twin/scenarios/schema.py`, `src/thermal_twin/scenarios/runner.py`, `src/thermal_twin/cli.py` provide the configuration schema and runnable commands.
- **Analysis**: `src/thermal_twin/analysis/plot_telemetry.py` plots NDJSON telemetry into standard time-series charts.

1.2 Design rationale

The project uses configuration files rather than hard-coded geometries to support:

1. fast iteration (geometry changes do not require solver edits),
2. systematic calibration (loss coefficients and contact/bedding properties can be tuned),
3. reproducibility (a scenario file fully captures assumptions and run settings).

2 Physics Model

2.1 Governing equation (transient heat diffusion)

In each solid region, temperature evolves according to:

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + q, \tag{1}$$

where T is temperature, ρ density, c_p specific heat, k thermal conductivity, and q volumetric heat generation.

In current configurations, the heater is imposed as a **Dirichlet temperature constraint** (so $q = 0$), which is appropriate when heater surface temperature is known/controlled or will be measured directly in future hardware.

2.2 Heater model (Dirichlet ramp schedule)

For heater cells/volumes:

$$T = T_h(t), \tag{2}$$

with a linear ramp:

$$T_h(t) = \begin{cases} T_\infty + (T_{\max} - T_\infty)\, t/t_{\text{ramp}}, & t < t_{\text{ramp}}, \\ T_{\max}, & t \geq t_{\text{ramp}}. \end{cases} \tag{3}$$

Here T_∞ is ambient temperature, $T_{\max}$ the target heater setpoint, and t_{ramp} the ramp time.

Heater efficiency (concept). In practical systems, not all electrical power reaches the block (lead losses, fixture losses, imperfect coupling, radiative losses, etc.). A simple and robust way to capture this when only heater temperature is prescribed is to introduce an efficiency factor $\eta_h \in (0, 1]$ such that:

$$T_{\text{max,eff}} = \eta_h\, T_{\max}. \tag{4}$$

This is a calibration knob; it should be used cautiously and ideally replaced by measured heater temperature telemetry (future plan).

2.3 Boundary heat losses (convection + radiation as Robin BC)

Exposed faces exchange heat with ambient via convection and radiation. In the solver, this is modeled as a Robin boundary:

$$-k_{\text{face}}\,\mathbf{n}\cdot\nabla T = h\,(T-T_\infty) + h_{\text{rad}}\,(T-T_\infty), \tag{5}$$

where h is convective heat transfer coefficient, and h_{rad} is a linearized radiation coefficient:

$$h_{\text{rad}} = 4\sigma\varepsilon T_\infty^3, \tag{6}$$

with emissivity ε and Stefan–Boltzmann constant σ. The effective loss coefficient is:

$$h_{\text{eff}} = h + h_{\text{rad}}. \tag{7}$$

In implementation, each boundary face contributes an implicit sink term to adjacent cells, scaled by face area and cell volume.

2.4 Imperfect contact between blocks (gap modeling)

When assemblies are not perfectly monolithic, thermal resistance at joints can dominate transient response. Thermal Twin supports a practical, mesh-based representation:

- Insert **thin low-k volumes** between blocks (e.g., `air_gap`) to emulate imperfect contact.
- This avoids specialized contact conductance models while remaining configurable and calibratable.

This approach is effective when contact is partial (touching at points but with voids), and when the detailed micro-contact physics are unknown.

3 Numerical Method and Implementation

3.1 Discretization and solver

The PDE is discretized using FiPy finite volumes on unstructured tetrahedra. The time integration uses an implicit transient term with a diffusion term:

$$\texttt{TransientTerm}(\rho c_p) = \texttt{DiffusionTerm}(k) + \text{boundary sinks}. \tag{8}$$

Implicit time stepping improves stability for stiff diffusion systems and strongly varying material properties.

3.2 Mesh generation

Geometries are built in gmsh using OpenCASCADE primitives and boolean fragmentation to ensure non-overlapping volumes. Each region is assigned a physical group for:

- volumetric material assignment (3D physical groups),
- boundary-face extraction for Robin BC application (2D physical faces).

Mesh sizing is controlled by scenario fields (global element size, heater refinement). Mesh quality is evaluated using gmsh element quality metrics.

3.3 Stability safeguards

Unstructured meshes can contain degenerate distances that cause divisions by zero in FiPy. The solver clamps near-zero cell distances early (distance floor) to prevent numerical blow-up.

4 Configuration System (How Users Define New Systems)

4.1 Materials: `materials.json`

Materials are defined with temperature-dependent arrays:

- `T`: temperature samples (°C),
- `rho`, `cp`, and either `k` or `alpha` (if only diffusivity is known).

The solver interpolates properties at runtime using linear interpolation.

4.2 Geometry JSON

A geometry file defines named elements with types such as:

- `block`: axis-aligned box,
- `cylinder`: heater rods / bedding volumes,
- `shell`: insulation layers wrapping targets,
- `helix_pipe`: swept helical pipe (pilot scheme).

Each element may assign a `material` (maps to `materials.json`).

4.3 Scenario YAML

A scenario binds geometry to physics and runtime:

- mesh controls (global size, refinement, optional min quality),
- heater schedule (setpoint, ramp time, optional turn-off),
- boundary conditions (region name, convection/radiation parameters, optional face filtering),
- measurement probes (positions),
- solver runtime settings (dt, total time).

This design allows end users to define new assemblies without code changes.

5 How to Run the Codebase

5.1 Environment setup (Windows)

From repository root:

```
python -m venv .venv
.\.venv\Scripts\Activate.ps1
pip install -e .
```

5.2 Mesh build and preview

Generate mesh from a scenario (exports `.msh` and a preview `.vtu`):

```
.\.venv\Scripts\python -m thermal_twin.cli mesh build <scenario.yml> --preview outputs/
    meshes/<name>.vtu
```

5.3 Run a simulation with telemetry and live plot

```
.\.venv\Scripts\python -m thermal_twin.cli simulate <scenario.yml> \
  --live-plot --keep-live-plot-open \
  --telemetry-log outputs/runs/<run>.ndjson \
  --output outputs/runs/<run>.json
```

5.4 Plot telemetry after the run

```
.\.venv\Scripts\python -m thermal_twin.analysis.plot_telemetry outputs/runs/<run>.ndjson
    --save report/plot.png
```

6 Results (Current Baselines)

This section summarizes the current validation baselines used for calibration and handover.

6.1 Single insulated block: 700 °C case

Figure 1 shows the single-block geometry. Figure 2 shows the simulated probe temperature response for a 700 °C target.

6.2 Four-block naked stack: 600 °C case

Figure 3 shows the four-block geometry with internal gaps. Figure 4 shows the simulated response for a 600 °C target without insulation.

6.3 Pilot scheme (next step)

Figure 5 shows the planned pilot configuration: a 3×3 stack of long block arrays, with a central block containing a helical stainless pipe and the entire assembly wrapped in rockwool insulation. This geometry exists to support the next experimental phase, where measured data will enable calibration of contact losses, insulation performance, and heater efficiency.

7 Applicability and Future Extensions

Thermal Twin is applicable to any assembly that can be approximated as piecewise-homogeneous solids with known or calibratable boundary losses. The strongest use case is experimental replication and parameter calibration:

- Calibrate h and ε to match observed heating/cooling.

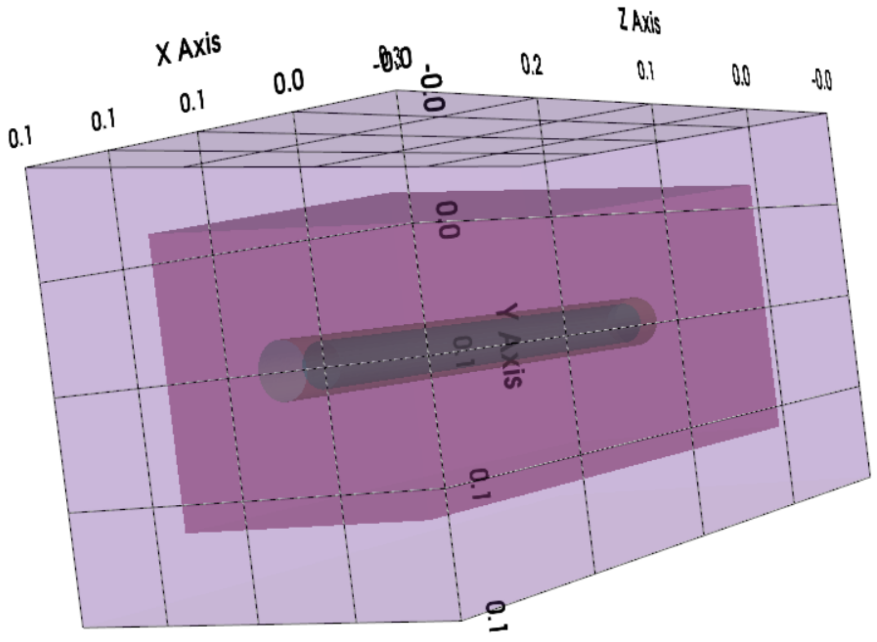

Figure 1: Single block geometry visualization (insulated assembly with heater rod and bedding).

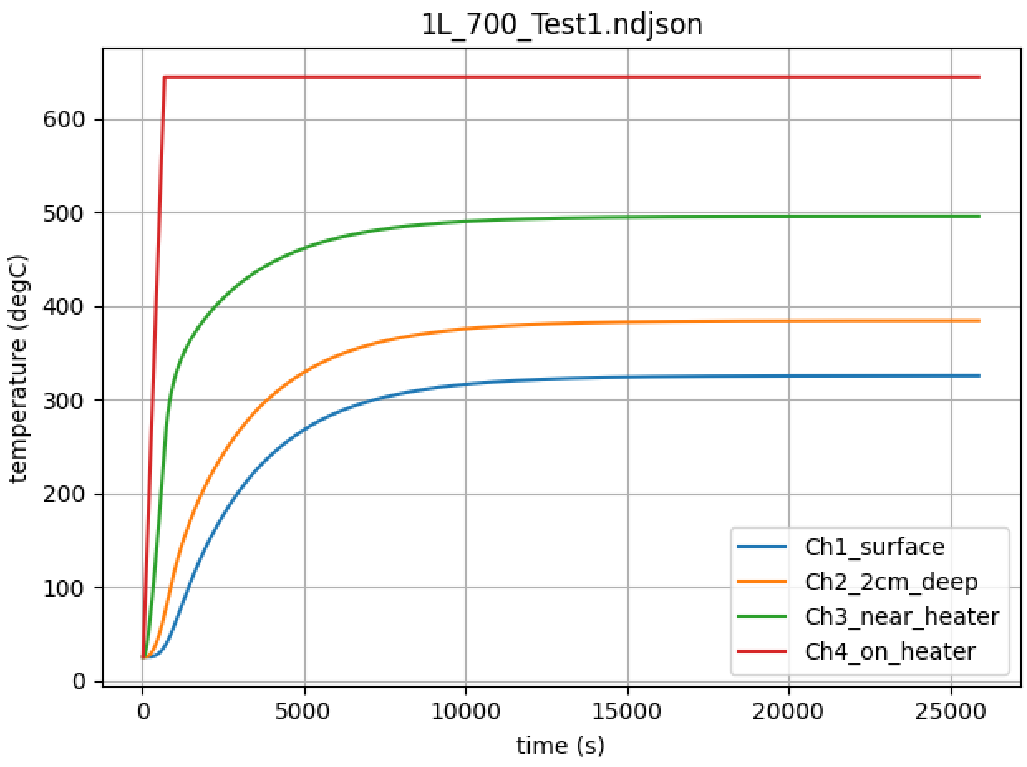

Figure 2: Single block (1L) simulated temperature response for the 700 °C target.

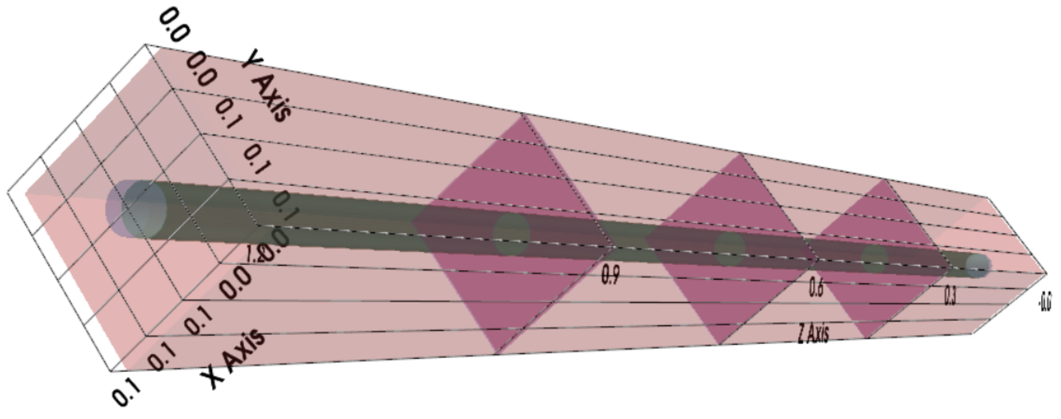

Figure 3: Four-block naked assembly geometry visualization. The internal planes indicate low-k contact layers used to emulate imperfect block-to-block coupling.

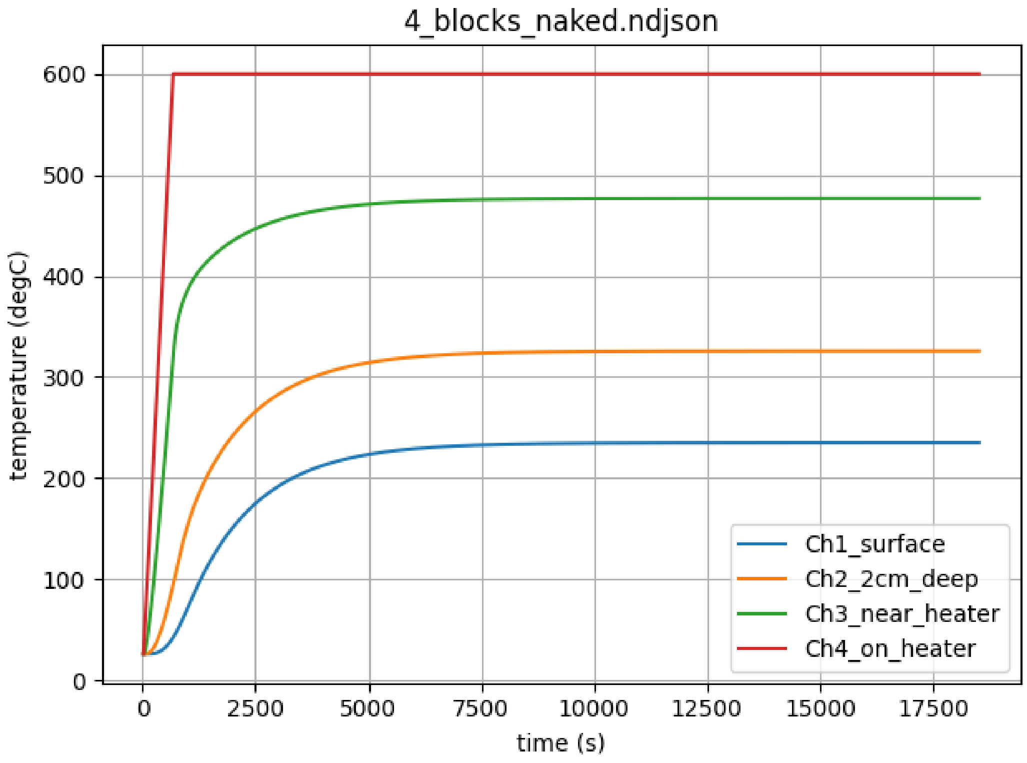

Figure 4: Four-block naked simulated probe response for a 600 °C heater target. Heater efficiency was not aggressively tuned due to unknown loss physics; calibration is expected after pilot measurements.

- Calibrate bedding k and contact gap parameters to match ramp dynamics.

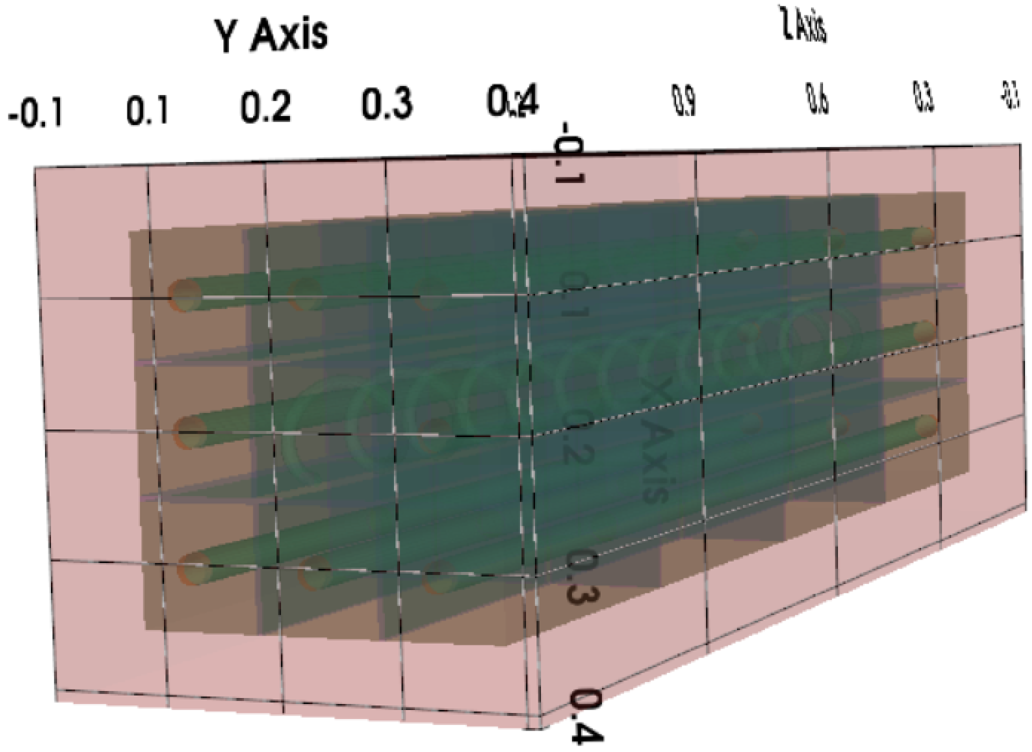

Figure 5: Pilot scheme geometry diagram (3×3 stack, separator geopolymer layers, central helical pipe, rockwool shell).

- Replace prescribed heater schedule with measured heater telemetry in future hardware.

Planned upgrades include nonlinear radiation (full $\sigma\varepsilon(T^4 - T_\infty^4)$), explicit thermal contact resistance models, and optimization-based fitting to experimental datasets.