

Updates on the progresses from Mondaic on NN integration

August 26, 2026

H. Michel, F. Nguyen, S. Koley, et al.

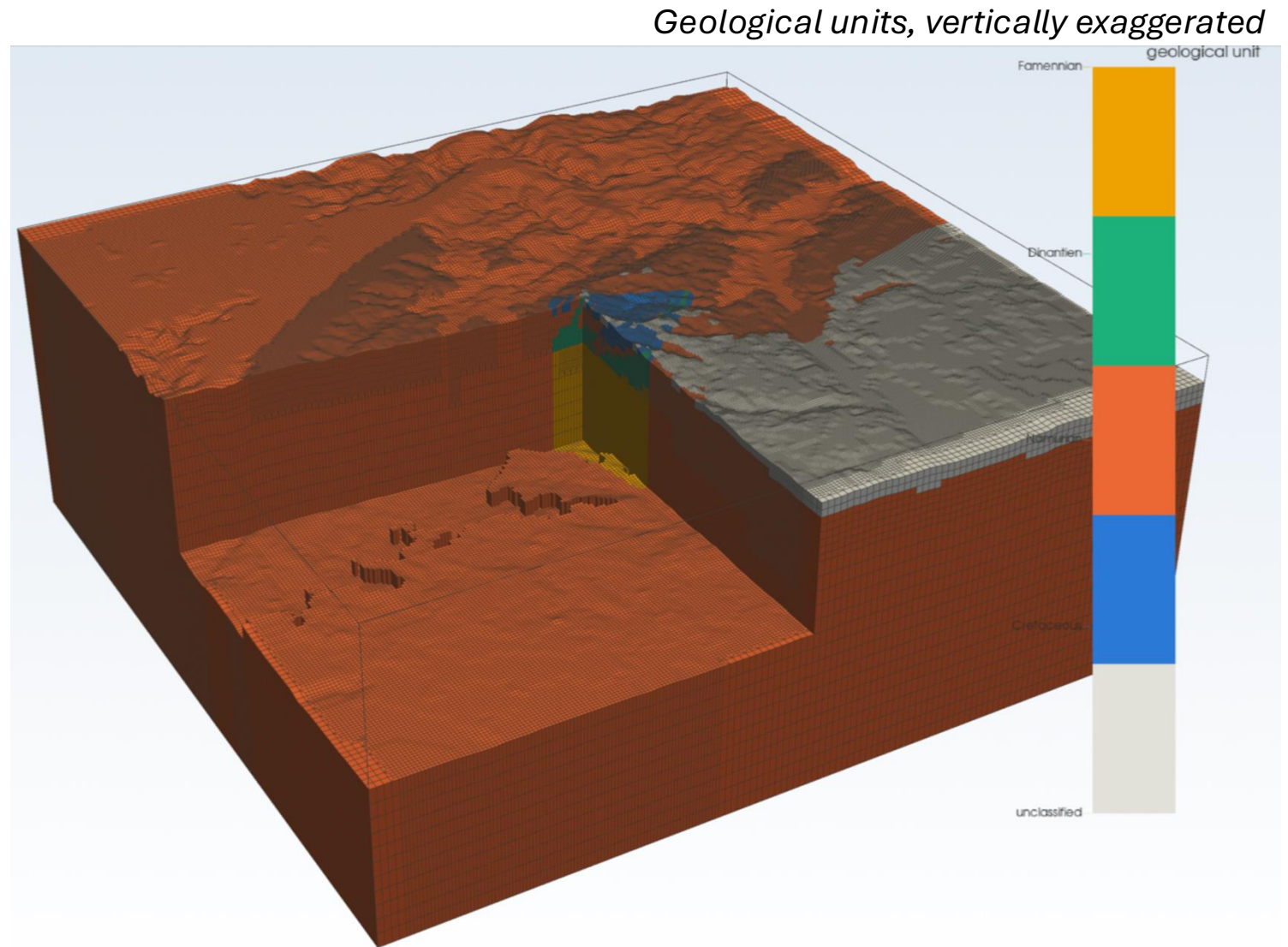
Slides modified from the weekly updates from Mondaic.

Overview

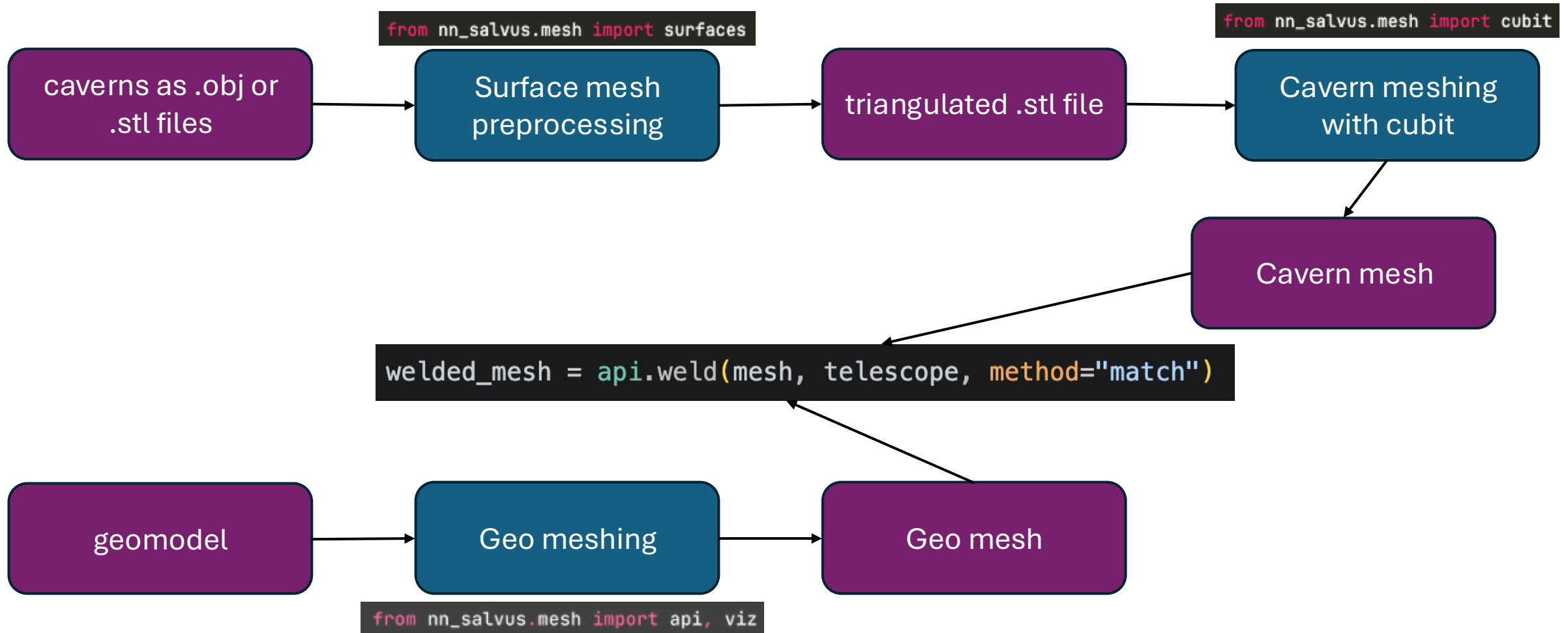
- Task 1: Meshing
 - Geological meshing → Solved
 - Caverns meshing → Mostly solved (some details lost in the process)
 - Merging → Working, still experimental
 - Optimization → Working
- Task 2: NN integration
 - 'nn_salvus': Python package for post-processing of Time-domain snapshots
 - Internal on-the-fly to Salvus
- Task 3: Frequency domain output on GPU
 - On Pause
- Task 4: Full (mostly) automated workflow
 - From geological model, caverns layout and sources distributions to NN
 - Ongoing work

Task 1: Meshing

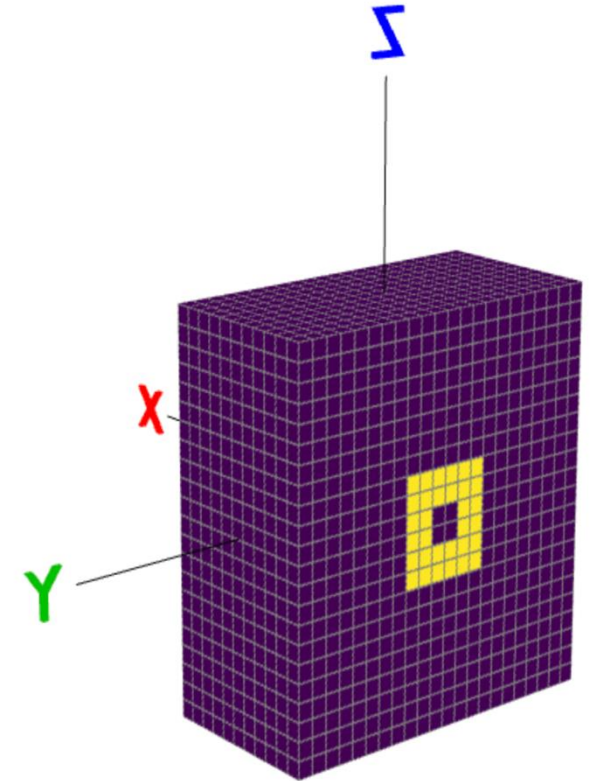
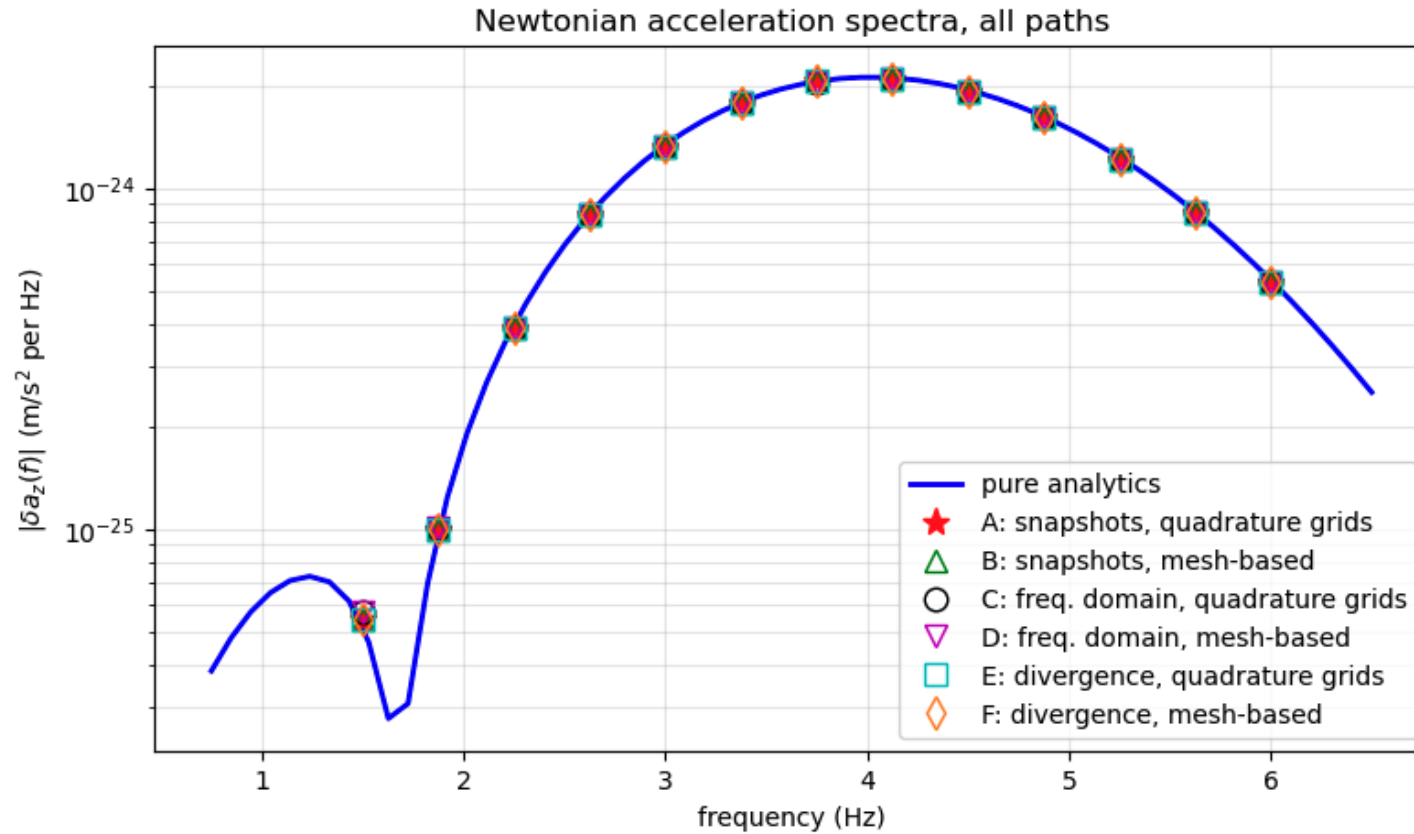
- Meshing beased on STL from the geological model
 - Easily updatable
- Fully conforming mesh generated
- Including topography



Task 1: Meshing



Task 2: NN integration



Task 2: NN integration

- Currently, 4 implementations in the 'nn_salvus' package (+ ANNA)
 - Two integration methods:
 - On pre-defined integration points
 - On the GLL points from the mesh
 - Two formalisms:
 - Split form
 - Divergence
 - Divergence formalizm requiers the definition of every interfaces in the mesh for computation (complicated to automate)
 - Undergoing internal (ULiège) review before release to collaboration
- In solver:
 - Volume integration of the split-form implemented
 - Pending IP → implementation of ANNA directly in Salvus

Task 2: NN integration

Output formatting

- Exercised by the basic shared cavity example notebook and the more “realistic” examples.

```
combined_path = export.write_combined_output(
    filename=output_directory / "combined.h5",
    test_masses=test_masses,
    times=times_td,
    accelerations=acc_td,
    frequencies=FREQS,
    spectra=spec_fd,
    surface_file=surface_path,
    point_file=receiver_path,
    meta_file=output_directory / "meta.json",
    simulation=w,
    formulation="split",
    overwrite=True,
)
```

- Now includes the full input file of the solver including time series of source time functions.

```
# Reading the file manually.
with h5py.File("./output_box/combined.h5", mode="r") as f:
    # Full input file to the solver.
    solver_input = json.loads(bytes(f["input"]["meta.json"]))[
        "forward_run_input"
    ]
```

surface data

receiver data

volume data

meta data

newtonian noise

```
combined.h5
├── attrs
│   ├── FILE_FORMAT          "NN_SALVUS_COMBINED_OUTPUT"
│   ├── FILE_FORMAT_VERSION  1
│   ├── NN_SALVUS_VERSION    nn_salvus version that wrote the file
│   └── salvus_version        installed salvus version (salvus._version_)
├── # Salvus surface output, verbatim (cavity-wall displacement):
│   ├── coordinates_ELASTIC_surface (n_facets, n_nodes_facet, 3)
│   ├── connectivity_ELASTIC_surface visualization connectivity
│   └── surface/
│       ├── attrs: start_time_in_seconds, sampling_rate_in_hertz,
│       │           sampling_interval_in_time_steps,
│       │           reference_time_in_seconds
│       └── <field> (n_snap, n_facets, n_cmp, n_nodes_facet)
├── # Salvus receiver output, verbatim (points in the medium):
│   ├── coordinates_ELASTIC_point (n_rec, 3)
│   ├── names_ELASTIC_point (n_rec,) strings "NET.STA.LOC"
│   ├── receiver_ids_ELASTIC_point (n_rec,)
│   └── point/
│       ├── attrs: as above
│       └── <field> (n_rec, n_cmp, n_time) -- note the time axis is LAST here
├── # Salvus volume output, verbatim (only when requested):
│   ├── coordinates_ELASTIC (n_elem, n_nodes, 3)
│   ├── connectivity_ELASTIC visualization connectivity
│   └── volume/
│       ├── attrs: as above
│       └── <field> (n_snap, n_elem, n_cmp, n_nodes)
├── # The solver input:
│   ├── input/
│   │   ├── meta.json the run's meta.json as raw bytes
│   │   └── <stf name> each custom source-time function, copied verbatim from its file: (n_time, n_cmp) plus the attrs sampling_rate_in_hertz, start_time_in_seconds, and filename / dataset_name / referenced_by (where it came from and which input path(s) use it)
└── # Written by nn_salvus:
    ├── newtonian_noise/
    │   ├── attrs
    │   │   ├── gravitational_constant scipy.constants.G, so the accelerations are absolute
    │   │   ├── formulation e.g. "split" (if recorded)
    │   │   ├── start_time_in_seconds float64[1], time domain only
    │   │   ├── sampling_rate_in_hertz float64[1], time domain only
    │   │   ├── spectral_convention "exp(-i*omega*t)", frequency domain only
    │   │   └── solver_time_step_in_seconds float64[1], frequency domain only: the solver time step the spectra are scaled by, read from the run's meta.json
    │   ├── coordinates_test_masses (n_tm, 3) [m]
    │   ├── time_domain_acceleration (n_tm, n_t, 3) [m/s^2]
    │   ├── frequencies_in_hertz (n_freq,) [Hz]
    │   └── frequency_domain_acceleration (n_tm, n_freq, 3) complex [m/s^2 / Hz]
```

Task 3: Frequency output in GPU

- On pause (lower priority)
 - Not urgently needed for the current work
 - Can always be done as post-processing (smaller output to save)

Task 4: Full workflow

- Work ongoing
 - Meshing mostly solved
 - NN integration still under validation

Current work and timeline

- Implementation of the ET collaboration validation cases
- Full meshing pipeline automation (smoothing of the cavity mesh)
- Addition of the surface term of the internally computed split form (should be small if boundaries are far away sufficiently)
- Validation of the tools by ULiège

All expected to be done by mid-September.

Papers

- To be discussed in details next time
 1. NN simulations: from wavefields to NN integration in an integrated framework
 2. NN estimation for the EMR (first for a single cavity: e.g. Beusdael)
 3. Full size estimation of NN at the scale of the triangle
 - First full analysis of NN and the potential of the null stream at scale for the triangle within a complex geology (likely more decorrelation of the wavefield)
 4. Others...
- Contributions are welcome! 😊