Speed, Hardware, and AI

What's new for ZSoil Power Users

Matthias Preisig
Geomod SA, Geodev Sàrl
August 28th 2026

Overview

- Speed benchmarks and hardware requirements
- Running ZSoil on the cloud
- ZSoil github page

Speed benchmarks

- ZSoil v2026 can improve speed considerably, due to parallelization of matrix assembly routines
  → What is the optimal number of threads?
- Comparison of architectures
  - Processor type
  - RAM
  - Virtual CPUs
  - I/O, disk types, network, …

Speed benchmarks, thread scaling

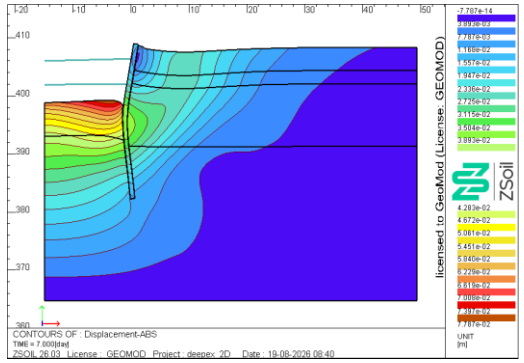
- 5 FE-models on 5 architectures
- FE models

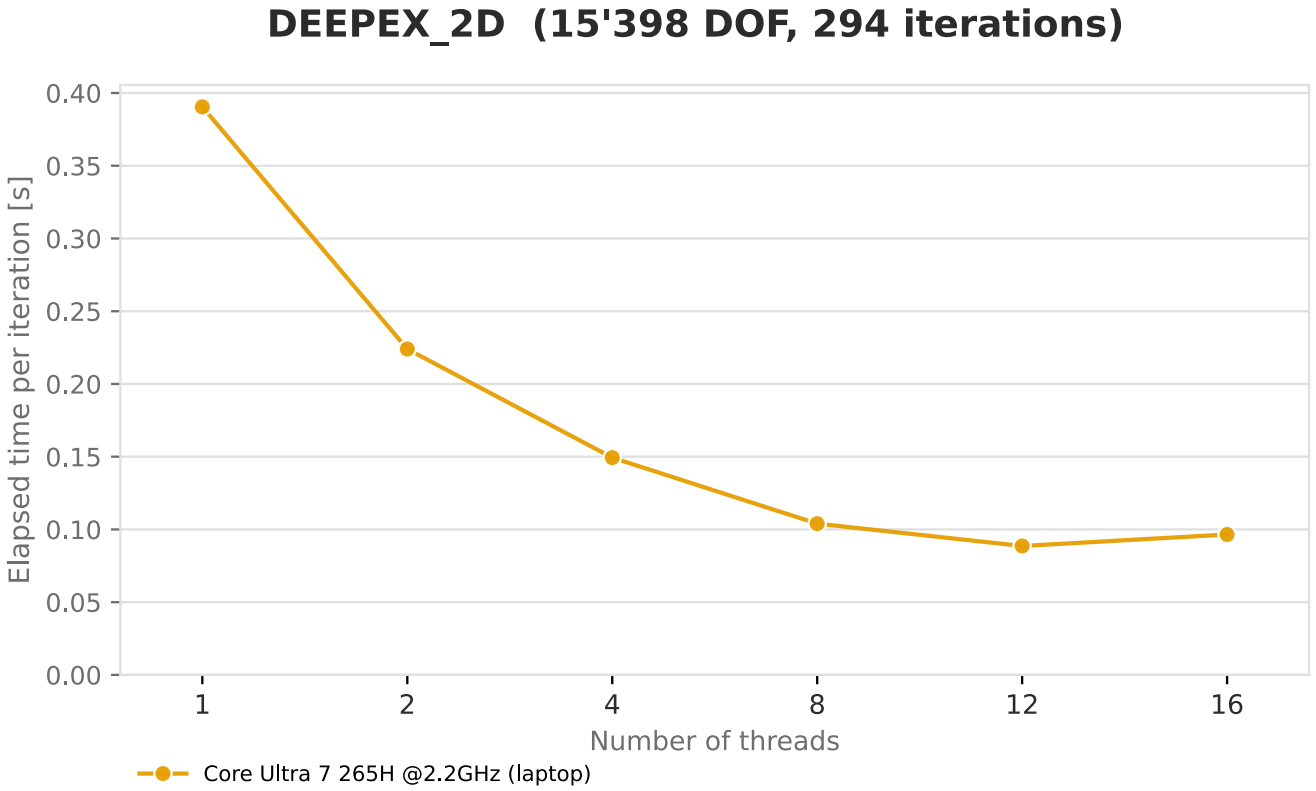
Model	Nb. dofs	Features	Goal
3D_EL_500k	1’536 k	Single iteration	Scaling for linear solver
3D_HSS_100k	296 k	NL iterations (Newton-Raphson)	Scaling for matrix assembly
Deep Ex 2D	7’560 Q4	Multiple solvers, 2-phase, HSS	Scaling for assembly, chunk size
Deep Ex 3D 1	750 k, 213 k B8	Multiple solvers, 2-phase, HSS	Cross-machine comparison
Deep Ex 3D 2	1’000 k, 320 k B8	Multiple solvers, multilaminate	Cross-machine comparison

- Architectures

Processor	Clock speed	RAM	Details
Xeon E5-2650v4	2.2 GHz	256 GB	2 sockets (workstation)
Xeon Gold 6136	3.0 GHz	256 GB	2 sockets (workstation)
Core Ultra 7 265H	2.2 GHz	64 GB	(laptop PC)
EPYC-Rome	2.0 GHz	48 GB	12 vCPU (Infomaniak cloud, perf2 disk tier)
Xeon Platinum 8275CL	3.0 GHz	32 GB	16 vCPU (AWS c5.4xlarge)

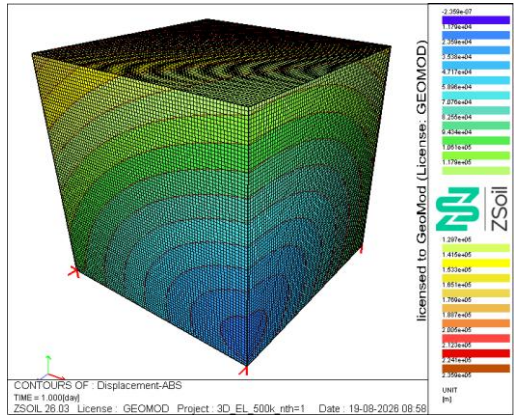
Thread Scaling, realistic 2D model

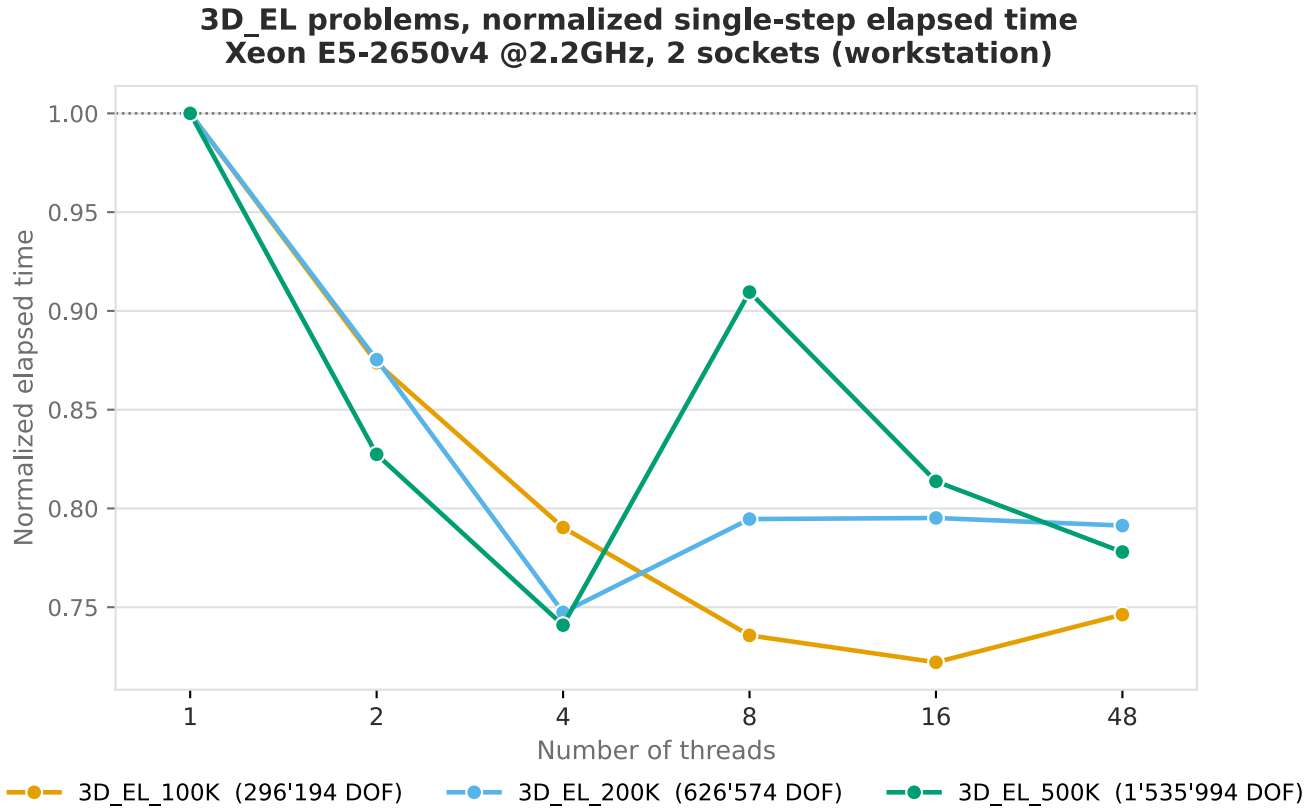




- *Small 2D problem with many NL iterations, therefore matrix assembly is dominant*
- *Good speedup up to 4-8 threads (Laptop PC)*

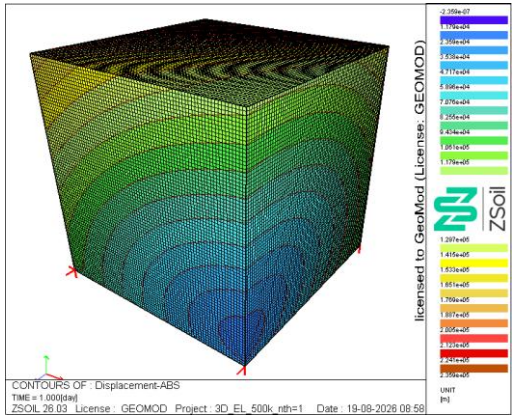
Thread Scaling, dependency on model size

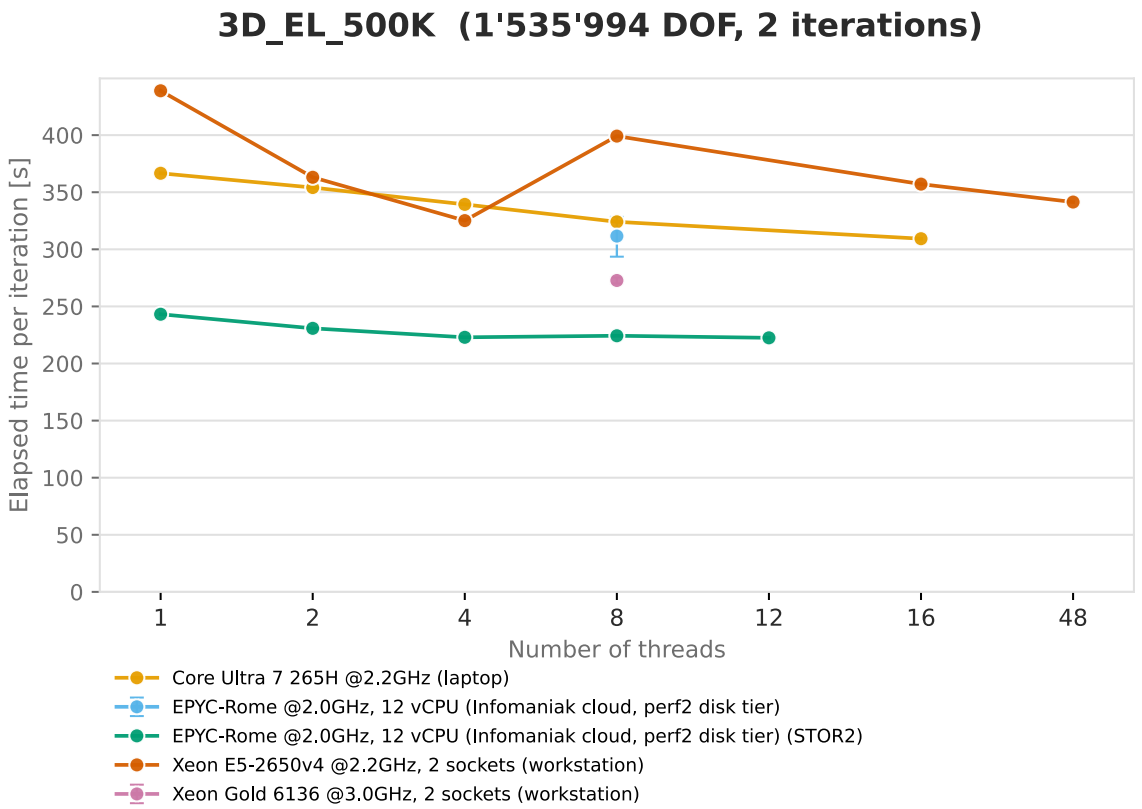




- *Good speedup for small models*
- *For larger models linear equation solver, which is already parallelized independent of parallel setting, probably becomes dominant*
- *Possible bottleneck with memory or I/O operations for larger models*

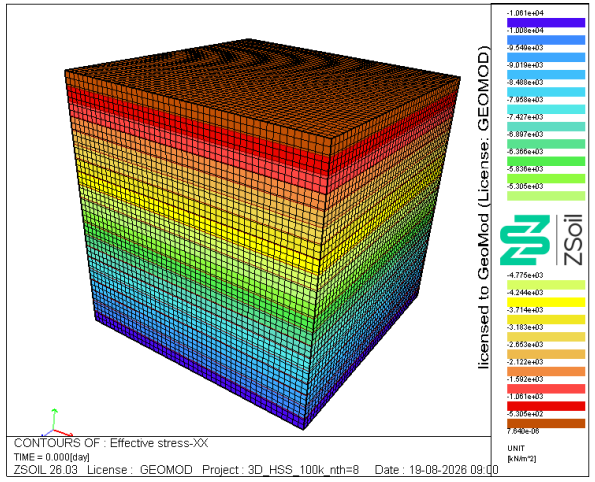
Thread Scaling & Cross-Machine Speed

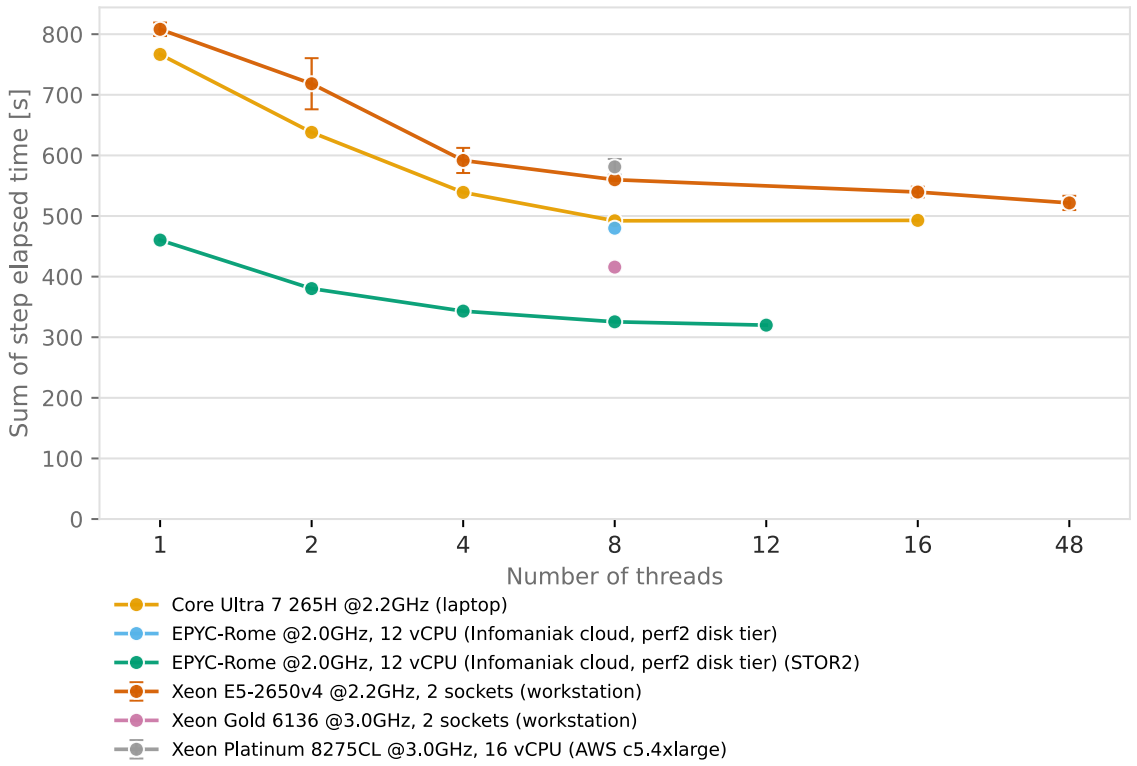




- *Small thread scaling, heavily dependent on sparse linear solver which is parallelized independently (Intel MKL's Pardiso)*
- *Bottlenecks for specific machines*
- *Cloud instance with optimized storage is the clear winner*

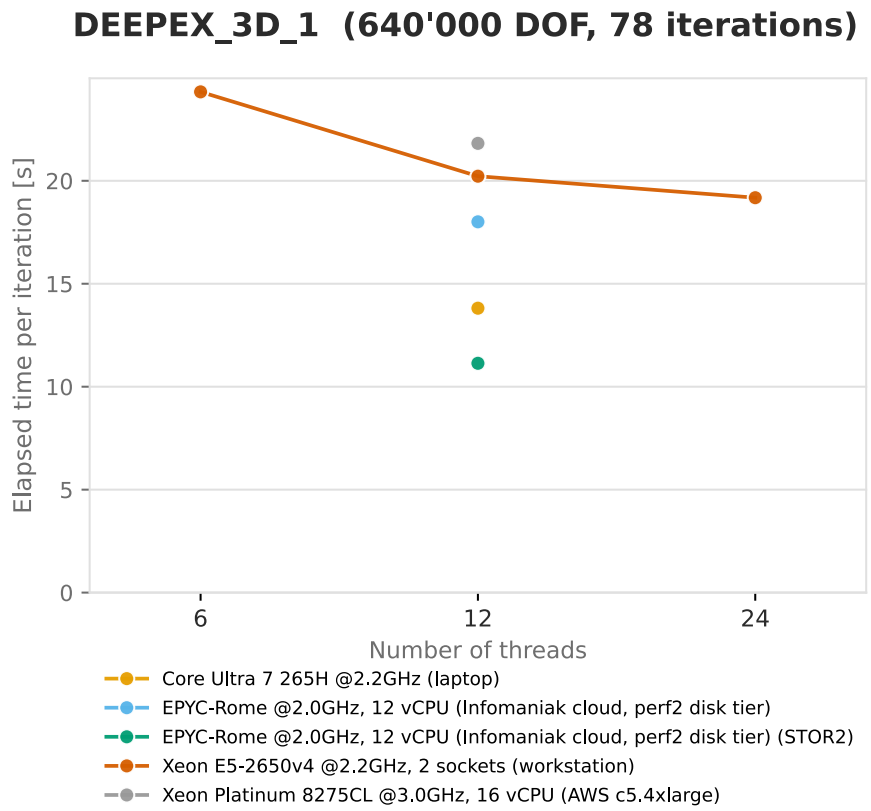
Thread Scaling & Cross-Machine Speed

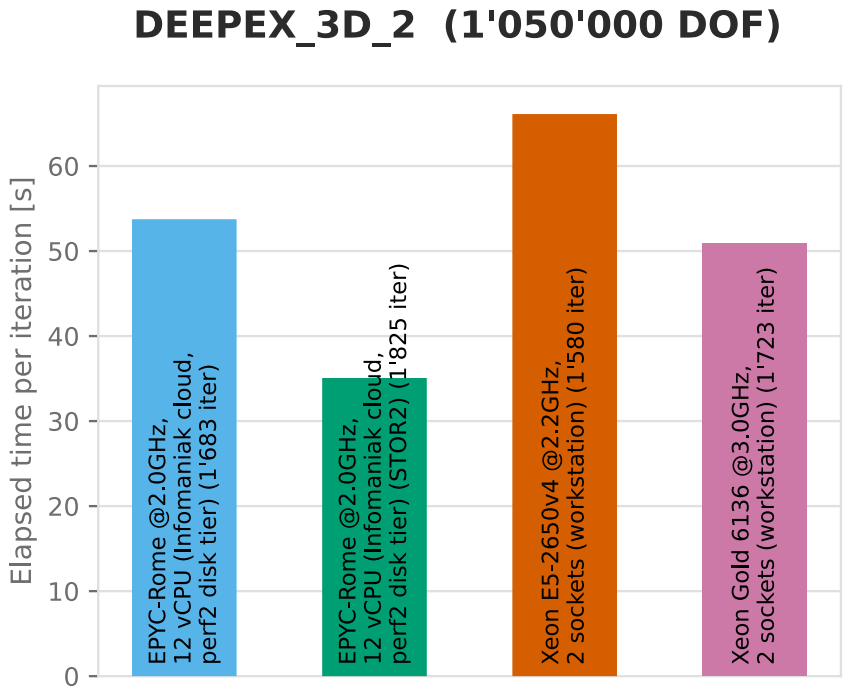




- *Best performance with 4 - 8 threads, 4 threads appear to be optimal*
- *Again, cloud instance with optimized storage is the clear winner*

Cross-Machine Speed, Realistic 3D Models





- *Confirmation that more than 12 threads is not efficient*
- *Cloud instance with optimized storage is the clear winner*
- *Normalized by no. of iterations because iteration count can differ between machines*

Summary

- Cloud computing is an efficient alternative to on-premise PCs
- Hardware requirements, ZSoil v2026:
  - AMD-processors can be used!
  - RAM: min 32, better 64 GB
  - Storage: Best on local HDD or SSD
- Parallel settings:
  - 8 to 16 threads for smaller 2D models
  - 4 to 8 threads for larger 3D models
  - No difference observed for chunk size setting
  - Linear solver is parallelized independently (Intel MKL's Pardiso)!

Cloud computing

- Infomaniak public cloud (www.infomaniak.ch)
  - Perf2 instances (higher iops), with storage volumes CEPH_1_perf2
  - Effort to be Eco-friendly, hosted in Switzerland
  - Cost (estimates):
    - Compute: up to CHF 0.10862 / hour for a16_ram64_disk80_perf2 → CHF 78 / month for uninterrupted use
    - Storage: CHF 0.00024 / GB / hour → CHF 17 / month for 100 GB
- AWS EC2 (Amazon Web Services)
  - Lager number of available instances
  - Excellent setting up guide: https://docs.itascacg.com/3dec700/common/docproject/source/manual/program_guide/setup/aws.html
  - Hosted in US
  - Cost (estimates):
    - Compute: e.g. USD 1.229 / hour for a16_ram32 → USD 885 / month for uninterrupted use (no saving for stopped instances)
    - Storage: depending on volume, IOPS, performance, … → probably cheaper than Infomaniak

ZSoil open source material

- Brand new github page: ZSoil-FEM (https://github.com/ZSoil-FEM)
- Repositories:

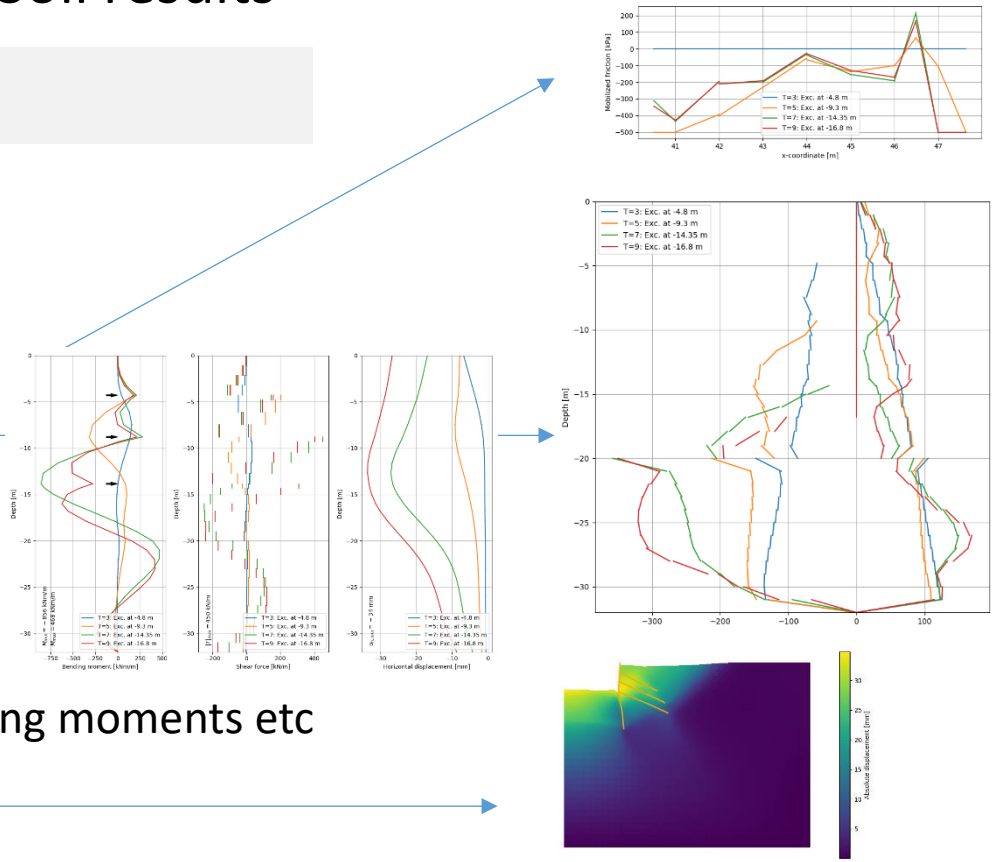
Repo	What it's for
zsoil-output	Python SDK for reading ZSoil model and result files (formerly ZSoilPy3). MIT licensed.
zsoil-output-examples	Ready-to-run scripts for postprocessing ZSoil results with zsoil-output.
documentation	Reference documentation for ZSoil's file formats and I/O. Work-in-progress
(zsoil-input)	Python SDK for input manipulation → Not yet available, more details TBA

ZSoil open source material

- `zsoil-output`: Reading and manipulating ZSoil results

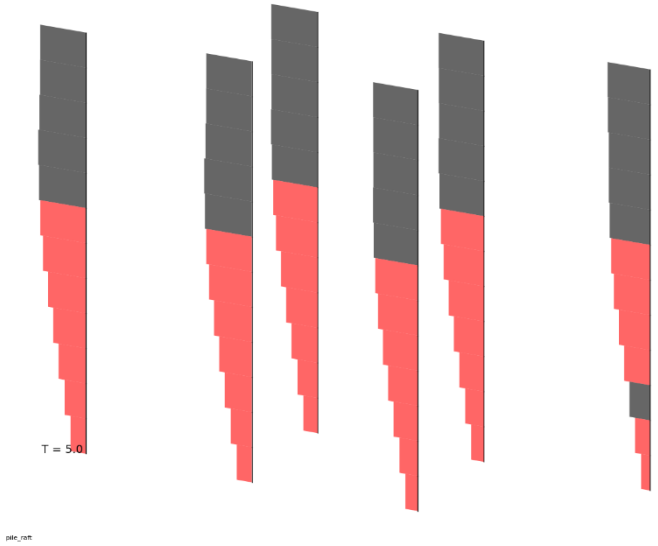
```
pip install zsoil-output
```

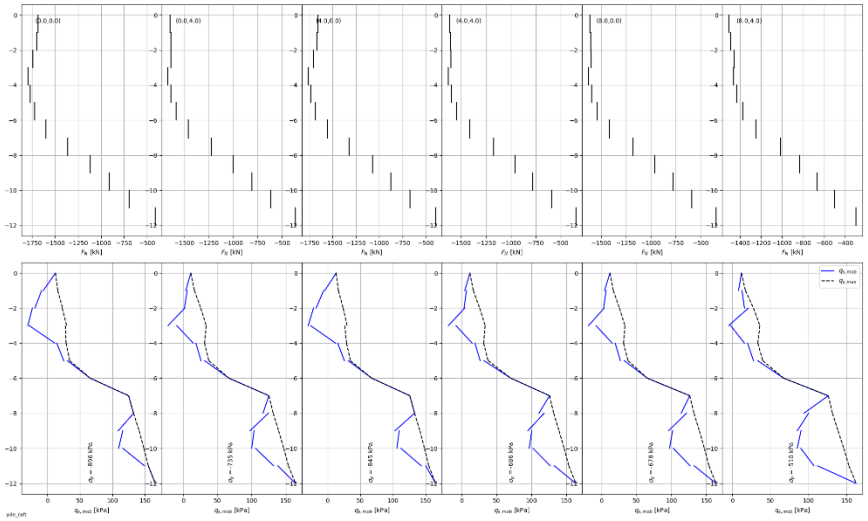
- `zsoil-output-examples`:
  New examples for output manipulation
  - 2D excavation retained by anchored diaphragm wall
    - Results for anchor interfaces (mobilized friction etc.)
    - Earth pressure diagram
    - Bending moment, shear and horizontal displacement diagrams for retaining wall
    - Time histories of displacements, anchor forces, bending moments etc
    - Color map of nodal and element results



ZSoil open source material

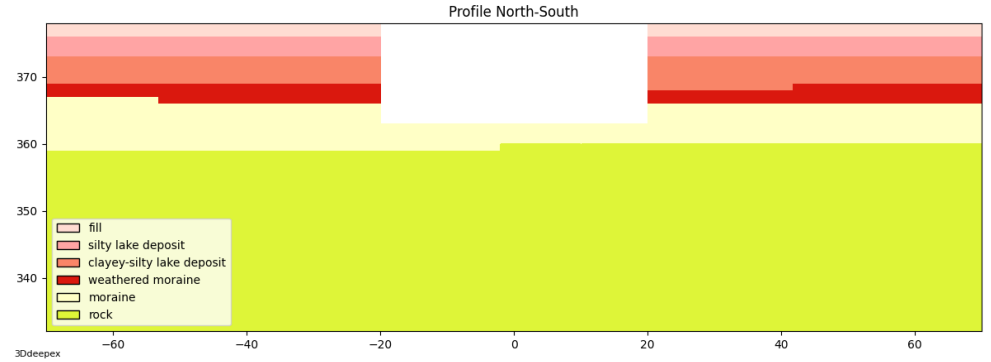
- zsoil-output-examples:
  - 3 x 2 pile group under a raft
    - Normal forces and plastic code along piles, 3D view
    - Normal forces and mobilized friction along shaft, mobilized tip resistance for each pile



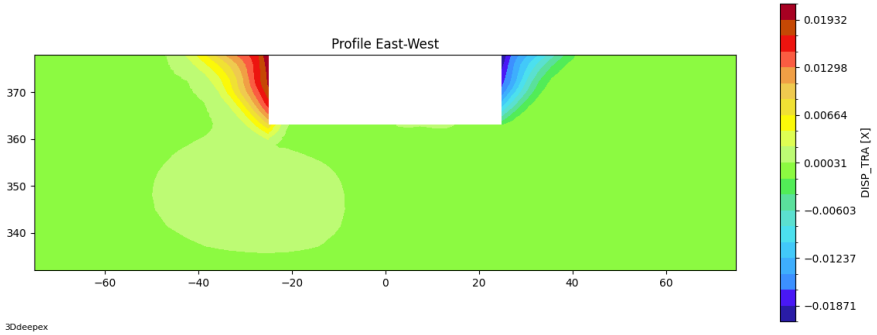


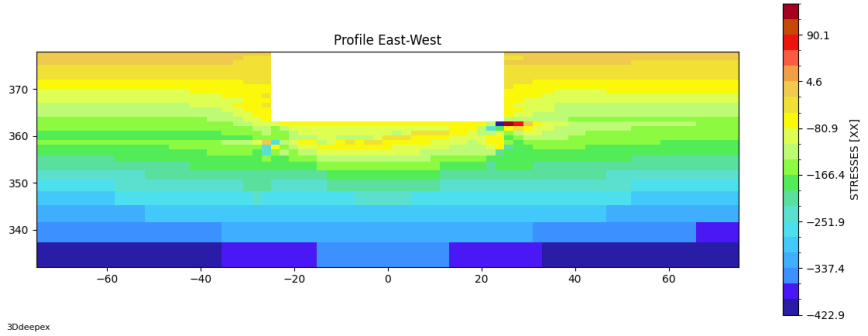
ZSoil open source material

- zsoil-output-examples:
  - Write vtk files using python: `zsoil_to_vtk.py`
  - 3D braced excavation with diaphragm wall
    - `plot_geology_profiles.py`



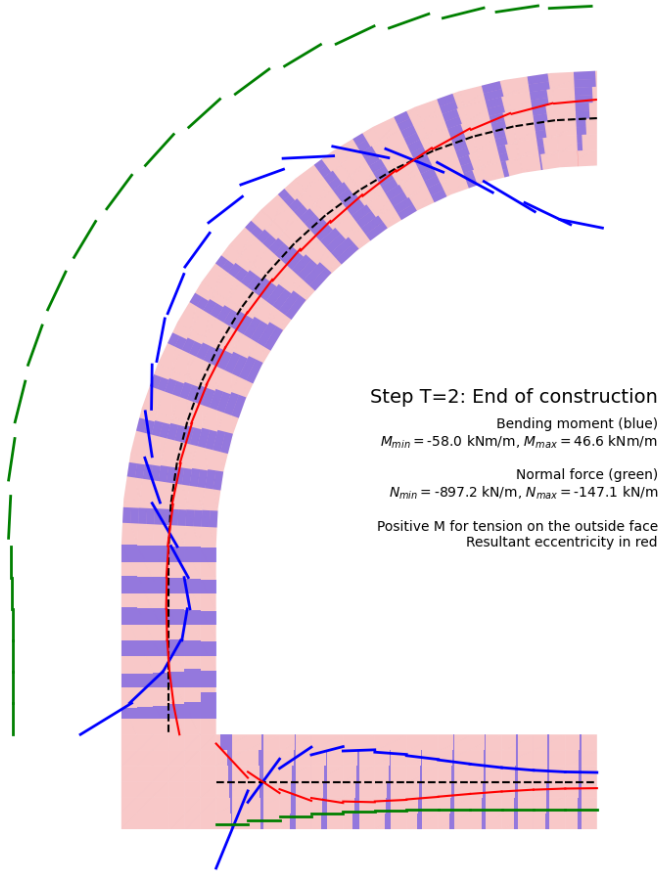
    - `plot_result_profiles.py`
      Plot nodal and element results





ZSoil open source material

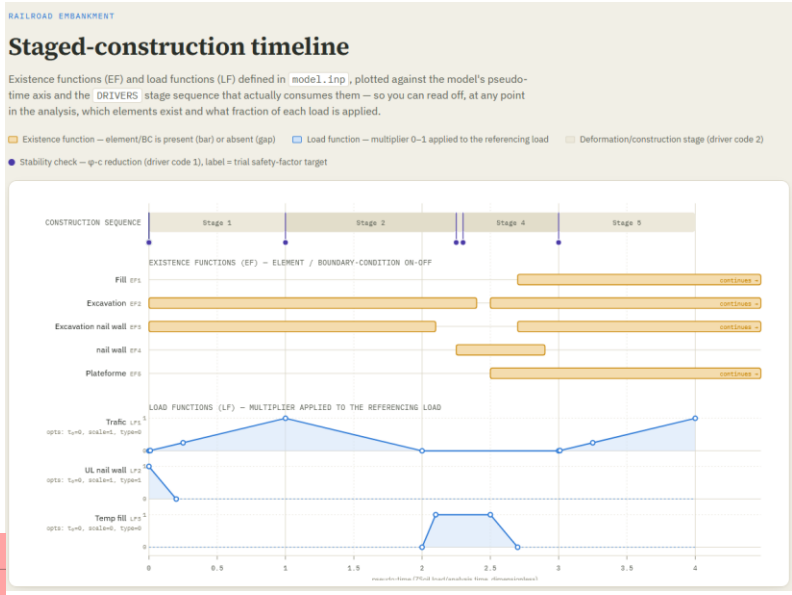
- zsoil-output-examples:
  - Integrate section forces from 3D continuum
    - `integrate_mnt_from_continuum.py`
    - Define neutral axis by shell elements which are never activated (Tstart = 99)
    - Define cross section planes perpendicular to tunnel axis
    - Subdivide neutral axis and cut lining
    - Plot normal stress profiles
    - Integrate stresses, plot moment, normal force and position of resultant normal force

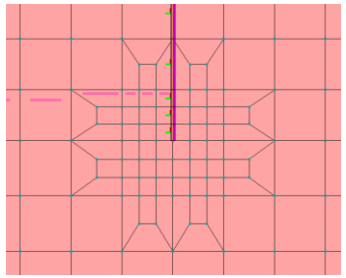


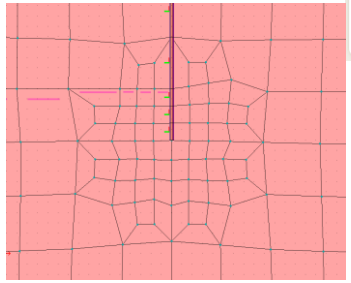
3Dtunnel

ZSoil open source material

- documentation: Inp-file documentation (work-in-progress)
  - Empower users to build AI-powered workflows. E.g.:
    - Compare inp-files and flag differences
    - Extract model information, e.g. timeline of Efs, LFs & drivers
    - Modify inp-files, e.g.:
      - Move anchors, modify loads, materials etc.
      - Simple mesh modifications:
        - Mesh refinement can quickly become complex (time and token consuming).
        - Improving mesh quality seems relatively simple







Conclusions

- Speed benchmarks with recommendations on parallel settings
- Updated hardware requirements
- Check out https://github.com/ZSoil-FEM
- Download it, use it, contribute your own examples and tools
- We’re looking forward learning how you use AI-tools to interact with ZSoil