Introduction to 3D Dynamic Rupture and Seismic Wave Propagation Simulations with SeisSol

Website: www.seissol.org

GitHub: https://github.com/SeisSol/SeisSol

Documentation: https://seissol.readthedocs.io

SeisSol's Code of Conduct

Instructors & Assistance: Alice-Agnes Gabriel, Fabian Kutschera, Gabrielle Hobson, Vikas Kurapati, Mathilde Marchandon, David Schneller, Thomas Ulrich

Observationally constrained fully dynamic, physics-based earthquake simulation of the 2016 Kaikoura, NZ, multi-fault earthquake (Ulrich et al., Nat. Comms., 2019.)

Welcome!
3D Dynamic Rupture and Seismic Wave Propagation Simulations with SeisSol

www.seissol.org

Morning Session, 9:00 - 10:00 am

- **Hands-on Introduction to SeisSol**

  (Prof. Dr. habil. Alice-Agnes Gabriel, UC San Diego & LMU Munich)

- **SeisSol Installation tutorial**

  (David Schneller & Vikas Kurapati, PhD students, TUM)

- **How to Quakeworx - A Science Gateway running the SeisSol App**

  (Fabian Kutschera, PhD student, UC San Diego)

Welcome!
3D Dynamic Rupture and Seismic Wave Propagation Simulations with SeisSol

www.seissol.org

Afternoon Session, 12:00 - 14:00

- **Meshing using GMSH** (Dr. Gabrielle Hobson, PostDoc, UC San Diego)
- **My First 3D Dynamic Rupture Simulation - a supershear normal faulting earthquake with off-fault plastic deformation (SCEC community benchmark TPV12/13)** (Dr. Mathilde Marchandon, PostDoc, LMU Munich)
- **3D dynamic rupture simulations of the complex 2016, Kaikoura, New Zealand, earthquake** (Dr. Thomas Ulrich, PostDoc, LMU Munich)
- **A fully coupled, 3D earthquake-tsunami simulation, "The Tsunami Problem Version" TTVP1 & TTPV2** (Fabian Kutschera, PhD student, UCSD)

Welcome!
3D Dynamic Rupture and Seismic Wave Propagation Simulations with SeisSol

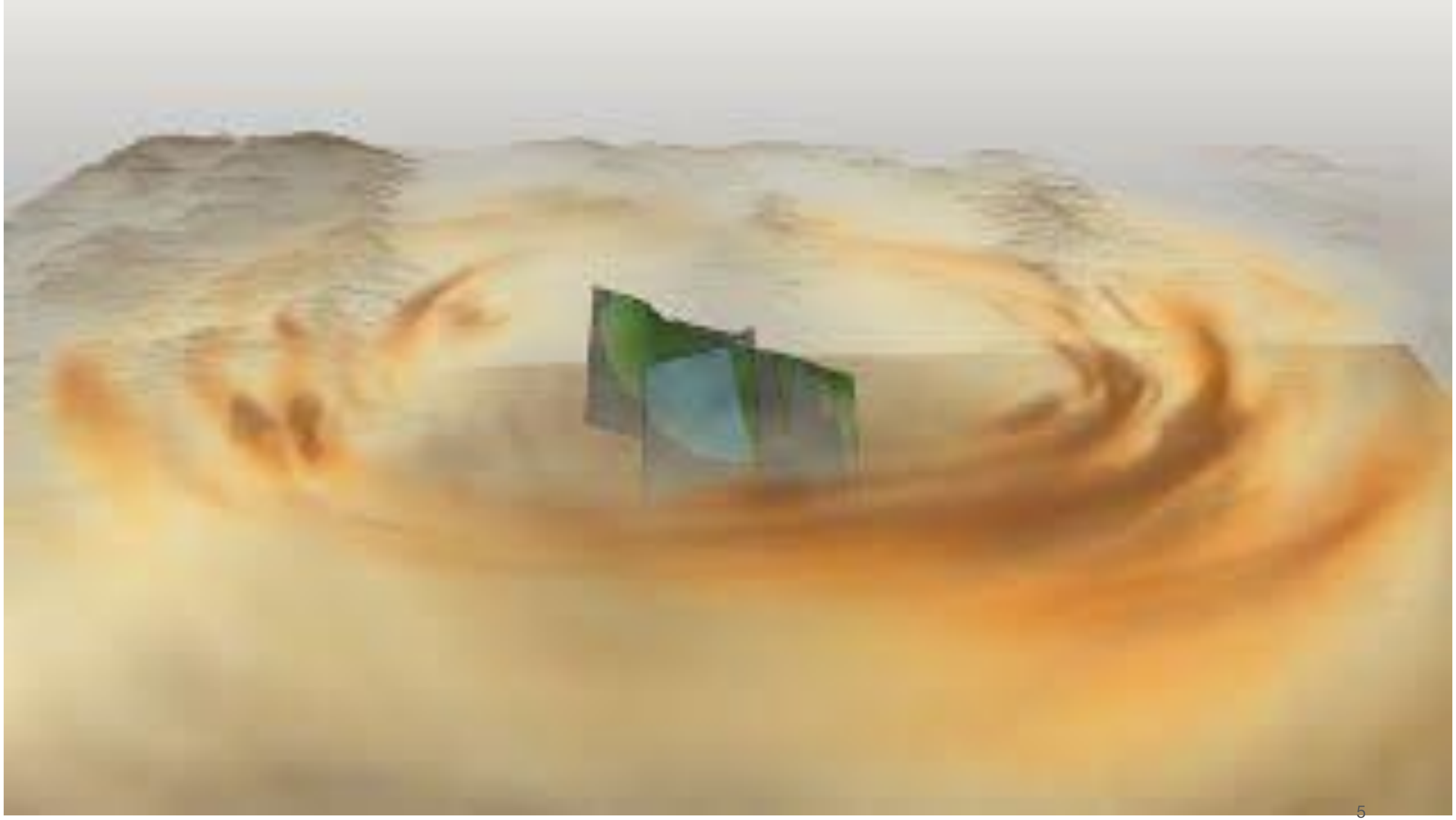
www.seissol.org

You can find many more & longer SeisSol tutorials online, incl. YouTube recordings!

- Quakeworx Kick-off: Advancing Earthquake Science and Cybertraining in Seismology: https://quakeworx.org/events/quakeworx-kick-off-workshop
- HPS/Scoped Cybertraining 2024: https://www.youtube.com/watch?v=0hI18PQmdr0&list=PLNj_YjCqziQJZU2DbrL3Ac-6weGUWsEhd&pp=iAQB
- HPS/SCEC Cybertraining 2023: https://www.youtube.com/watch?v=WeZ2vJxBuTg&list=PLNj_YjCqziQJXCJ5Mw4DlCZ2iuGvnY6fJ&pp=iAQB
- Past ChEESE SeisSol trainings: https://cheese-coe.eu/sites/default/files/uploaded/2021_SeisSol_ChEESE_Training.pdf

Hands-on Introduction to SeisSol

3D dynamic rupture models of the 2019 Ridgecrest earthquakes Taufiqurrahman et al., Nature'23 . Visualization of 15 TB of volumetric simulation data on unstructured tetrahedral meshes on Frontera, Abrams et al., SC'23, [YouTube]



Installation Pathways

- Manual (“from source”)
    - Compile all dependencies by yourself
    - Full control, but takes time
    - Good option for development
    - Works well on supercomputers as a fallback

For more infos, see the docs:

https://seissol.readthedocs.io/en/latest/build-overview.html

If you want to build SeisSol from source during the training, **let us know and we’ll help.**

Installation Pathways

- Spack
  - One command to install SeisSol and dependencies
    - `spack install seissol equations=elastic convergence_order=4`
  - Sometimes tricky to use
    - Duplicate or system dependencies
    - Outdated dependencies
    - Compiler/package issues
  - Good middle ground for supercomputers

See the spack package documentation (https://packages.spack.io/package.html?name=seissol) for a full list of options.

Installation Pathways

- Docker
  - “Quickstart” option
  - (mostly) OS independent
  - Good for testing on local machines
  - Contains sample scenarios
  - Not available on supercomputers – apptainer as alternative
    - Performance can reach native level

Main method for today (after Quakeworx).

Download ("pull") the Docker container

docker pull seissol/training

```
vikas-kurapati@vikas-kurapati-ThinkPad-X13-Yoga-Gen-4:~$ doc^C
vikas-kurapati@vikas-kurapati-ThinkPad-X13-Yoga-Gen-4:~$ docker pull seissol/training
Using default tag: latest
latest: Pulling from seissol/training
803207f66405: Pull complete
ffac0f7790db: Pull complete
089cc1dca342: Pull complete
3bcbf93ac676: Pull complete
beeead80e404: Pull complete
4191750dc5c4: Pull complete
4f4fb700ef54: Pull complete
99babba68895: Pull complete
c495697e1d91: Pull complete
a0b9fa226597: Pull complete
976a914490c5: Pull complete
1505ba9d53f1: Pull complete
a173928edc34: Pull complete
b588815469eb: Pull complete
eb8e437e9299: Pull complete
3061e67614d9: Pull complete
369ff36b0acb: Pull complete
5961ca8625cd: Pull complete
18c1e33d91a5: Pull complete
c0275150c013: Pull complete
17e401ceb864: Pull complete
f4e0b958b358: Pull complete
a8b1c5f80c2d: Pull complete
e106ccb783fb: Pull complete
dd8a85d9fbed: Pull complete
78edef37b6d7: Pull complete
335a8f3a959e: Pull complete
b8716e757863: Pull complete
6459c66149e8: Pull complete
716620322542: Pull complete
364e3c75f43c: Pull complete
133c7e33fb75: Pull complete
1825a8c4e022: Pull complete
9c70983ec031: Pull complete
43375ca4591b: Pull complete
1fbed59d76a7: Pull complete
800afe8ccf41: Pull complete
3e845f54f97f: Pull complete
Digest: sha256:873abe5198ae0708b82999ac31d69dd4144d0527645d7dc9cd393d125097fdaa
Status: Downloaded newer image for seissol/training:latest
docker.io/seissol/training:latest
vikas-kurapati@vikas-kurapati-ThinkPad-X13-Yoga-Gen-4:~$
```

Run the Docker container

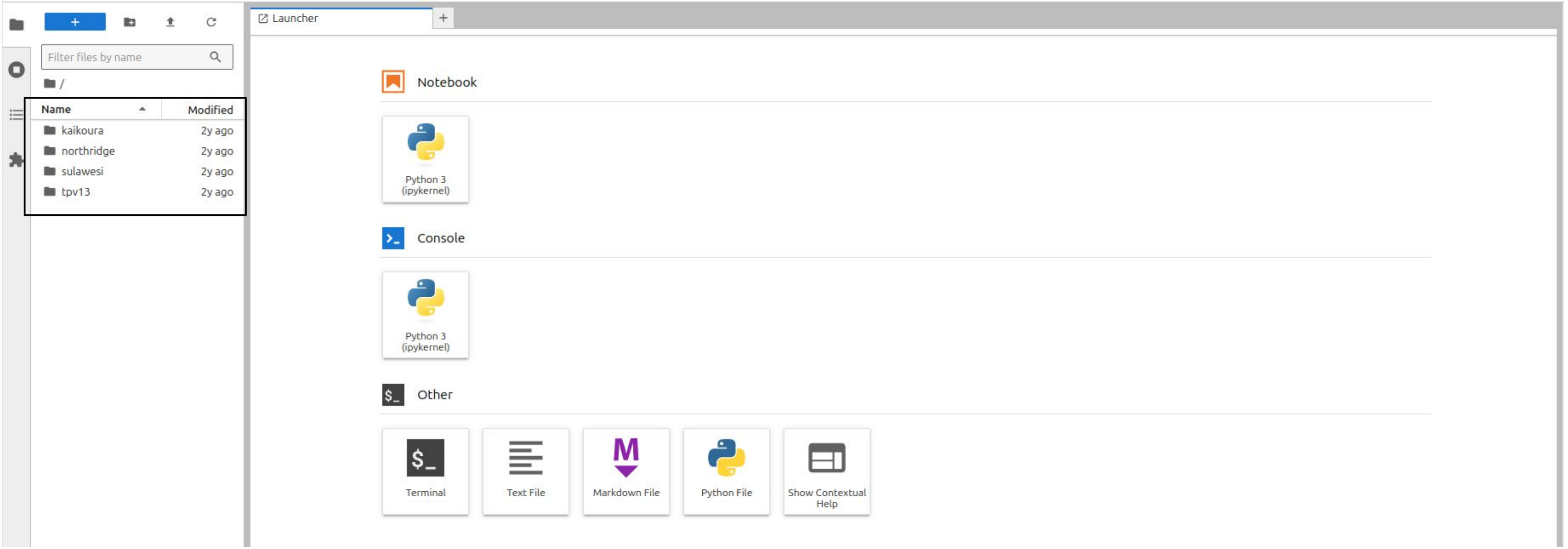
docker run -p 53155:53155 seissol/training

```
vikas-kurapati@vikas-kurapati-ThinkPad-X13-Yoga-Gen-4:~$ docker run -p 53155:53155 seissol/training
+ export DISPLAY=:99.0
+ DISPLAY=:99.0
+ export PYVISTA_OFF_SCREEN=true
+ PYVISTA_OFF_SCREEN=true
+ export PYVISTA_USE_IPYVTK=true
+ PYVISTA_USE_IPYVTK=true
+ which Xvfb
/usr/bin/Xvfb
+ sleep 3
+ Xvfb :99 -screen 0 1024x768x24
+ set +x
[I 2026-06-08 08:50:52.781 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2026-06-08 08:50:52.784 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2026-06-08 08:50:52.787 ServerApp] jupyterlab | extension was successfully linked.
[I 2026-06-08 08:50:52.790 ServerApp] notebook | extension was successfully linked.
[I 2026-06-08 08:50:52.792 ServerApp] Writing Jupyter server cookie secret to /home/scoped/.local/share/jupyter/runtime/jupyter_cookie_secret
[I 2026-06-08 08:50:52.919 ServerApp] notebook_shim | extension was successfully linked.
[I 2026-06-08 08:50:52.919 ServerApp] panel.io.jupyter_server_extension | extension was successfully linked.
[I 2026-06-08 08:50:52.938 ServerApp] notebook_shim | extension was successfully loaded.
[I 2026-06-08 08:50:52.942 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2026-06-08 08:50:52.943 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2026-06-08 08:50:52.946 LabApp] JupyterLab extension loaded from /opt/conda/lib/python3.10/site-packages/jupyterlab
[I 2026-06-08 08:50:52.946 LabApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 2026-06-08 08:50:52.946 LabApp] Extension Manager is 'pypi'.
[I 2026-06-08 08:50:52.963 ServerApp] jupyterlab | extension was successfully loaded.
[I 2026-06-08 08:50:52.967 ServerApp] notebook | extension was successfully loaded.
[I 2026-06-08 08:50:52.969 ServerApp] panel.io.jupyter_server_extension | extension was successfully loaded.
[I 2026-06-08 08:50:52.969 ServerApp] Serving notebooks from local directory: /home/training
[I 2026-06-08 08:50:52.969 ServerApp] Jupyter Server 2.14.0 is running at:
[I 2026-06-08 08:50:52.969 ServerApp] http://12d997be3762:53155/lab?token=7f42627c410434b9f098a53625c503480299ff6e65ccae8b
[I 2026-06-08 08:50:52.969 ServerApp]     http://127.0.0.1:53155/lab?token=7f42627c410434b9f098a53625c503480299ff6e65ccae8b
[I 2026-06-08 08:50:52.969 ServerApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 2026-06-08 08:50:52.972 ServerApp]

    To access the server, open this file in a browser:
        file:///home/scoped/.local/share/jupyter/runtime/jpserver-19-open.html
    Or copy and paste one of these URLs:
        http://12d997be3762:53155/lab?token=7f42627c410434b9f098a53625c503480299ff6e65ccae8b
        http://127.0.0.1:53155/lab?token=7f42627c410434b9f098a53625c503480299ff6e65ccae8b
[I 2026-06-08 08:50:53.578 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
```

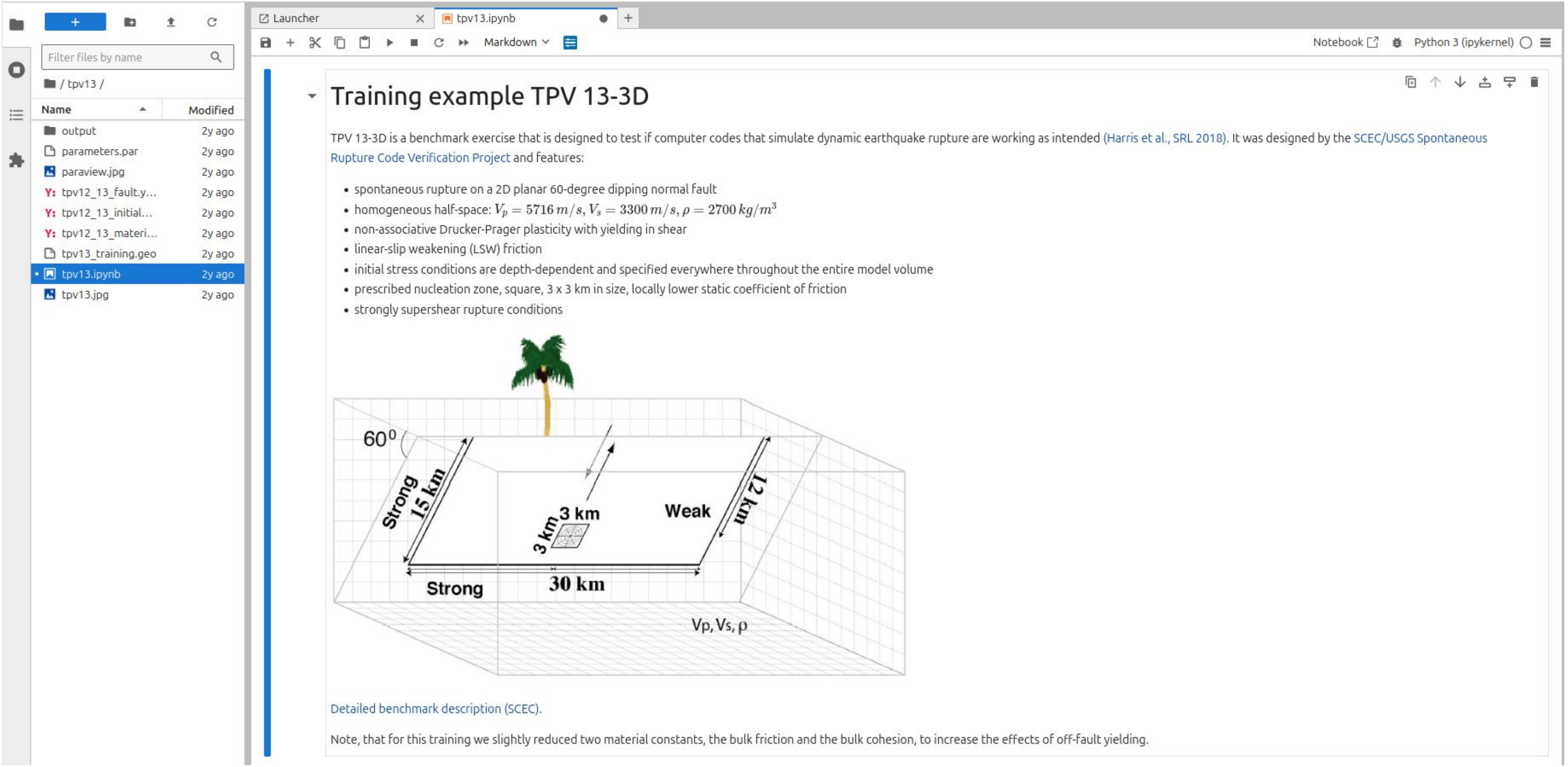
Jumping to the notebooks…

Enter in your webbrowser (cf. the previous screen—leave that one also running!):

http://127.0.0.1:53155/lab?token=<HASH>



Example Notebook: TPV13



Expanding Earthquake Science & Education With Quakeworx Interactive Web-based Tools

slides adapted from Rekoske et al., SSA 2026

*Rekoske, J., **Kutschera, F.**, Yun, J., Oryan, B., Gabriel, A., May, D., Chourasia, A., Youn, C., Silva, F., Zhao, C., Tainpakdipat, N., Bhatthal, A., Serafini, F., Ronquillo, J., Winjum, B., Maechling, P. J., Elbanna, A., & Ben-Zion, Y. (2026). Expanding Earthquake Science Education With Quakeworx Interactive Web-based Tools. Presented at the 2026 SSA Annual Meeting, Pasadena, CA, 14-18 April. https://seismosoc.secure-platform.com/a/gallery/rounds/47/details/14531*

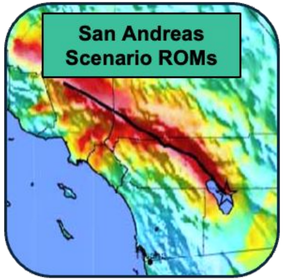
Quakeworx is a gateway for earthquake science applications

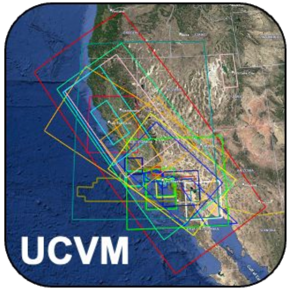
Quakeworx Project Goals			
Reduce barriers to access, expertise, software, compute, and data resources.	**Catalyze & empower** community wide reuse & sharing of research products.	**Scale** your research, education, and workforce development efforts.	**Advance Science** Rupture forecasts, Earthquake physics, and hazard estimates.

Quakeworx supports a range of earthquake simulation and visualization tools

SeisSol

pyCSEP





SCEC
Broadband platform

MOOSE

QuakeNN

UCERF3-ETAS

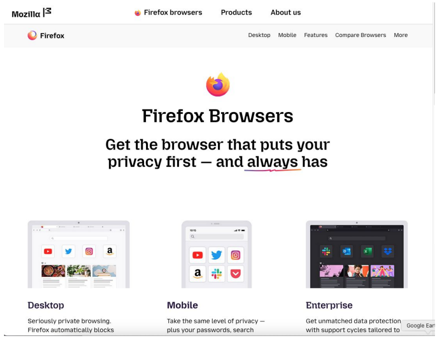
TANDEM

jupyter

.. and more!

Overview of a science gateway

A science gateway, like Quakeworx, typically provides access to curated research software, and access to significant computing and the storage.



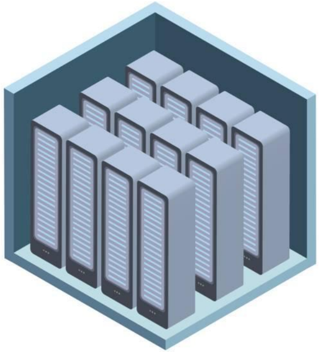
Quakeworx
Web-browser
User Interface

OneSciencePlace
Science Gateway
Cyberinfrastructure

Curated
Science
Applications

Research Data

Quakeworx Website Hosted by SDSC



SDSC Expanse
Computer and Storage
and Cloud Resources

QUAKEWORX

QUAKEWORX A science gateway for seismic simulations

Apps+ Earthquake Pages News & events Publications Overview About+

✨ Early Users Invited

Join the Quakeworx Gateway (staging) for early testing and feedback. Explore curated *Apps* and help shape the future of Quakeworx for the earthquake science community.

Log in with your institutional Single Sign-On.

https://quakeworx.org

Featured Apps on Staging

Curated simulators and pipelines you can run instantly on HPC and cloud systems.

SeisSol

SeisSol

High-performance dynamic rupture simulation for complex geometries.

Open in Staging

Tandem

TANDEM

Sequences of earthquakes and aseismic slip (SEAS) modeling.

Open in Staging

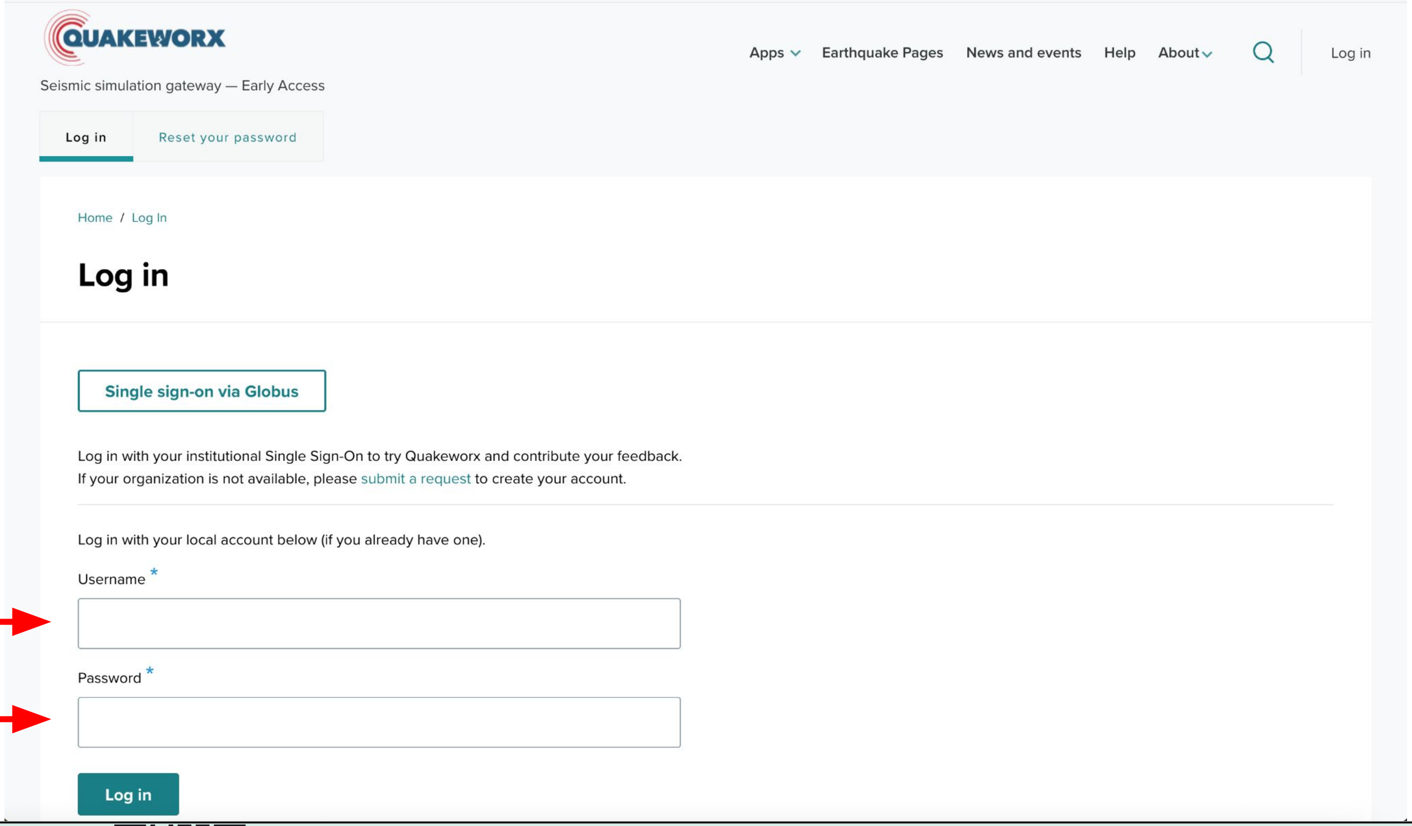
UCERF3-ETAS

UCERF3-ETAS

Operational aftershock modeling and stochastic catalogs for California.

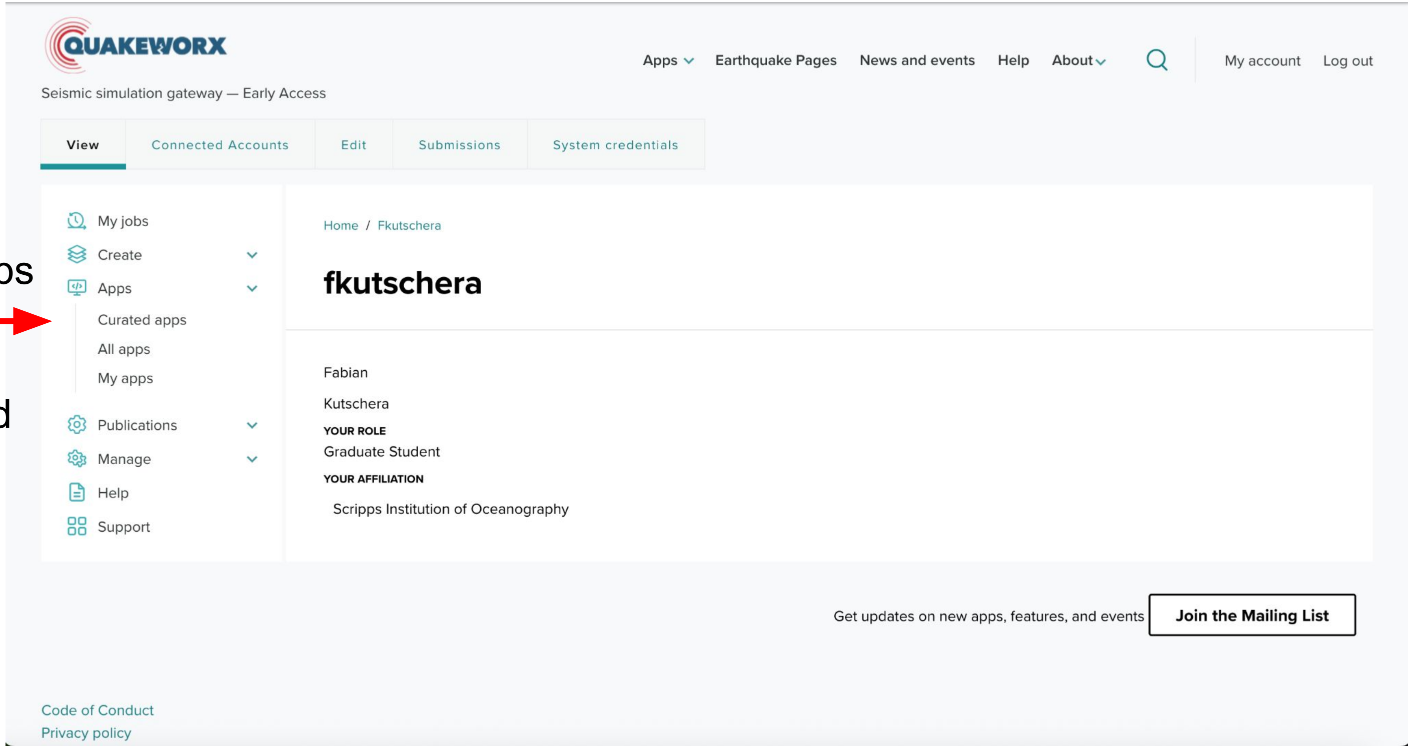
Open in Staging

How to get started?

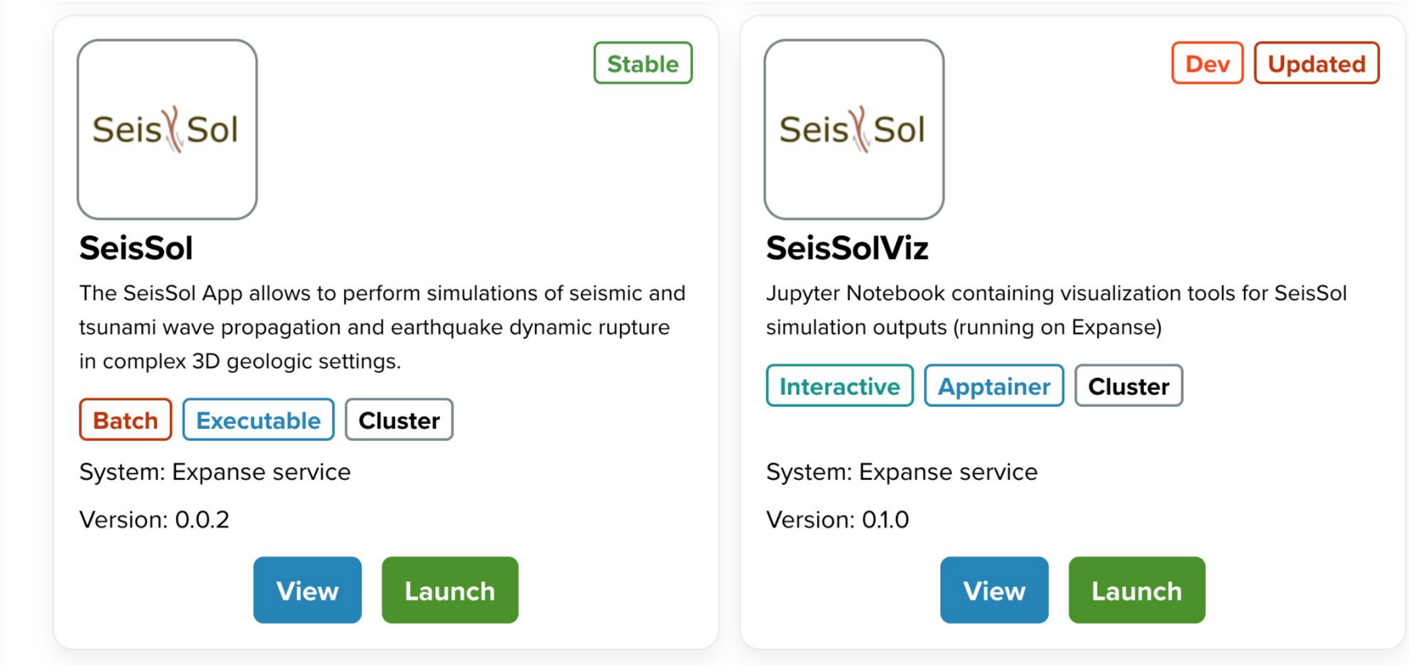


How to get started?

Curated apps
=
Actively maintained



How to get started?



How to get started?

View | Edit | Launch | Revisions | Clone

- My jobs
- Create
- Apps
- Publications
- Manage
- Help
- Support

Home / SeisSol

SeisSol

Seis Sol

BATCH - NON INTERACTIVE COMMAND LINE APPLICATION

Summary: The SeisSol App allows to perform simulations of seismic and tsunami wave propagation and earthquake dynamic rupture in complex 3D geologic settings.

App ID: qwx1.expanse.cyoun.seissol2 | Version 0.0.2

Status: Enabled | **App restrictions**: Public

Owner: choonhan

Launch app

Configure app form

Description

SeisSol is an open-source software for the simulation of seismic, acoustic, and tsunami wave propagation and earthquake rupture dynamics. The software utilizes the arbitrary high-order accurate derivative discontinuous Galerkin method (ADER-DG) to allow high-order accuracy in space and time. SeisSol supports complex 3D model geometries using tetrahedral meshes and handles various material models, including acoustic, elastic, viscoelastic, poroelastic, and viscoplastic rheologies. SeisSol is optimized for running on large-scale parallel computing architectures.

Get Quakeworx Training Material:

SeisSol training material for the application in the Quakeworx Gateway can be downloaded from here: https://github.com/SeisSol/Training

How to get started?

Home / Node / SeisSol / Launch

Launch

SeisSol app type

SeisSol   SeisSol nonlinear materials   SeisSol earthquake-tsunami

Job name *

SeisSol_Example

Specify a name for this job

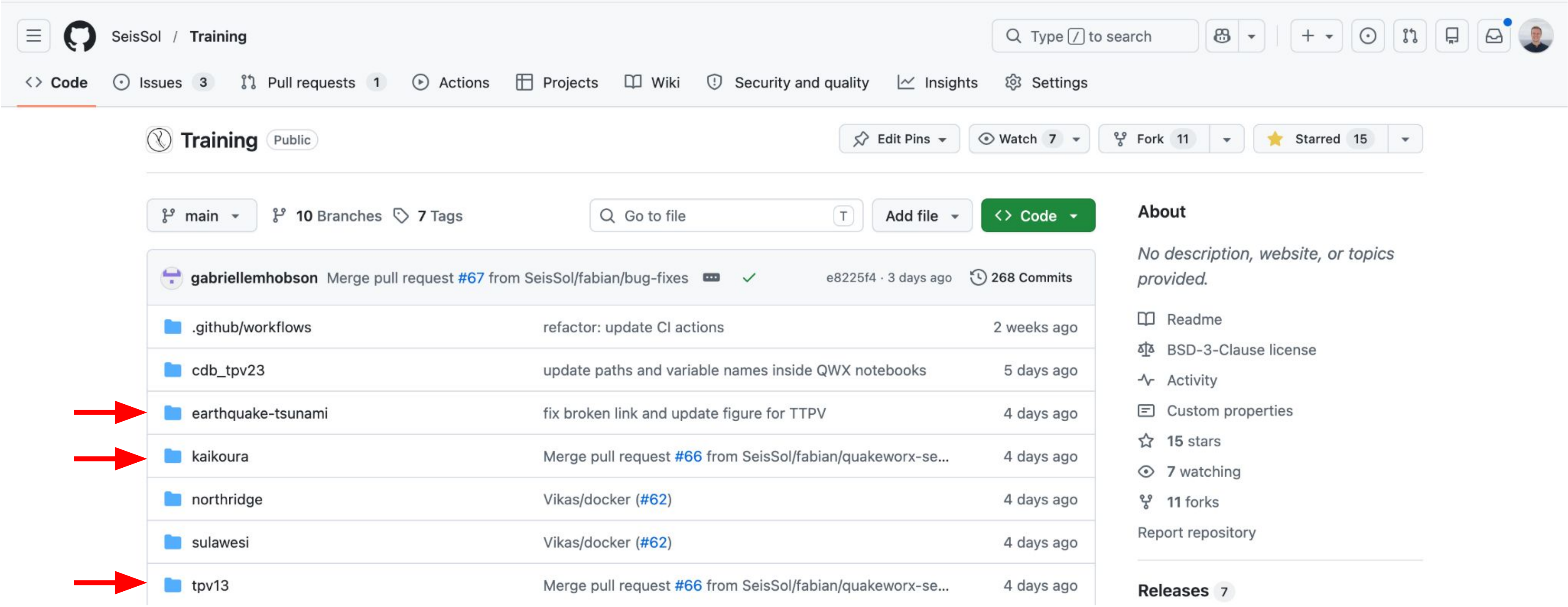
Parameter file (.par)

The results must be directed to a directory named *outputs* in the parameter file. e.g. prefix = "outputs/tpv33"

Browse... No file selected.

https://github.com/SeisSol/Training

https://github.com/SeisSol/Training



Quakeworx is a gateway for earthquake science applications

★ A platform for running, sharing, and publishing earthquake simulation & data-analysis tools.

★ Supports reproducibility and reduces access barriers.

★ Built on the OneScienceBase Platform to facilitate research and education through quick app execution and easy data sharing.

I. Make sure you are able to login to https://quakeworx.org

II. Make sure to download training material from https://github.com/SeisSol/Training

III. Enjoy the Trainings

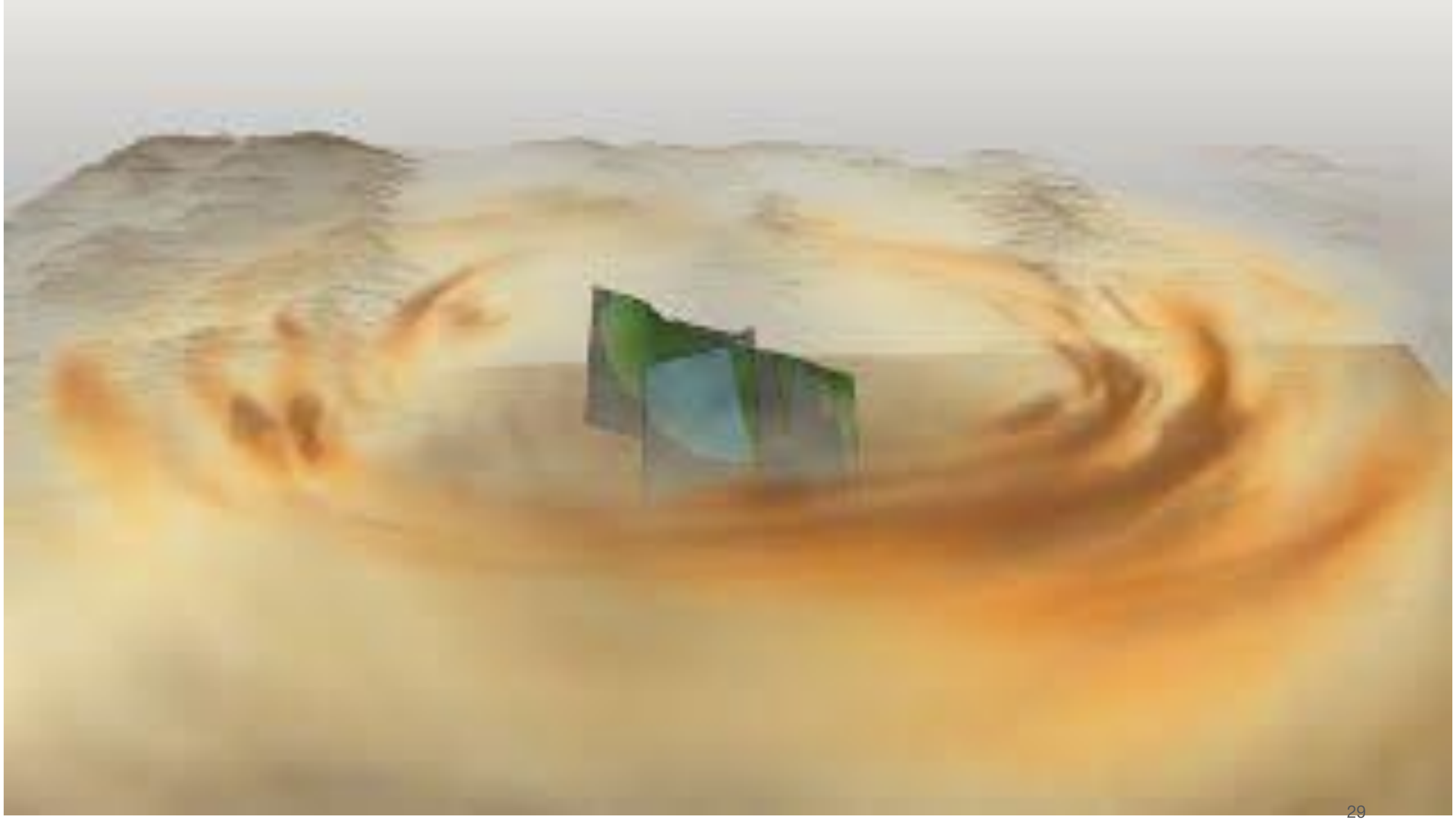
Thank you!

on behalf of the Quakeworx team

SeisSol Training continues at 12:00

Meshing tutorial

3D dynamic rupture models of the 2019 Ridgecrest earthquakes Taufiqurrahman et al., Nature’23 . Visualization of 15 TB of volumetric simulation data on unstructured tetrahedral meshes on Frontera, Abrams et al., SC’23, [YouTube]



Meshing Using GMSH

In this tutorial, we will cover:

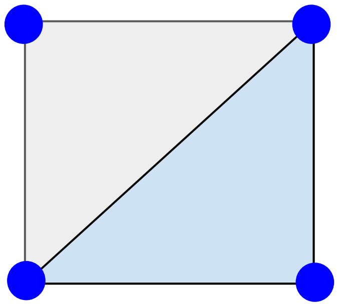
- Creating geometries using elementary entities in GMSH
- Custom tagging of entities to work with SeisSol boundary conditions
- Using both built-in GMSH scripting and the Python API
- Setting the local mesh size
- Mesh quality
- Boolean operations

We will then explore these three demos hands-on in a Jupyter Notebook:

1. A simple 2D rectangle
2. A 3D domain with an embedded fault (TPV12-13)
3. Leveraging the Python GMSH API to generate complex geometries

What is a mesh, and why do we need one?

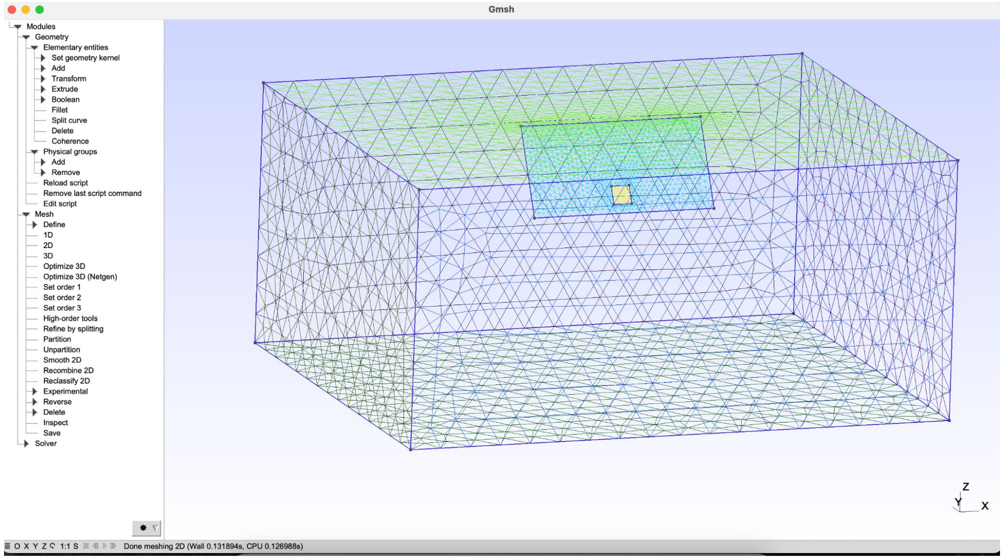
- To perform finite element calculations, we must discretize the domain of our geometry into cells to allow for elements and nodes to be defined
- The **geometry** for our problem of interest is the representation of a 2D surface or 3D volume with a specific **shape** (including how external and internal boundaries and surfaces are positioned and connected)
- The geometry's domain is **discretized** into cells with vertices
- The **mesh** is the collection of cells and vertices



Creating geometries and meshes for SeisSol

GMSH

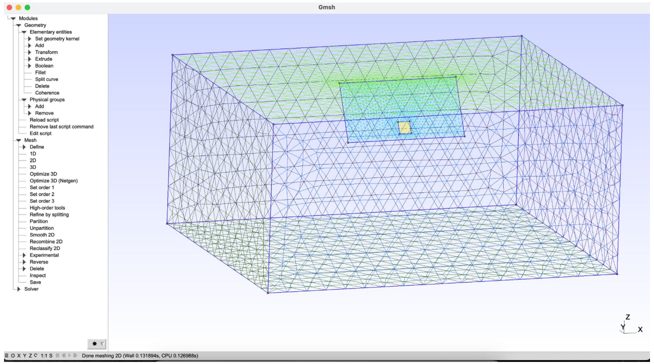
- Open-source software
- Many tutorials are available
- Useful for most geometries and everyday mesh sizes
- Limitations when working with highly complex geometries
- Serial mesh generation



Creating geometries and meshes for SeisSol

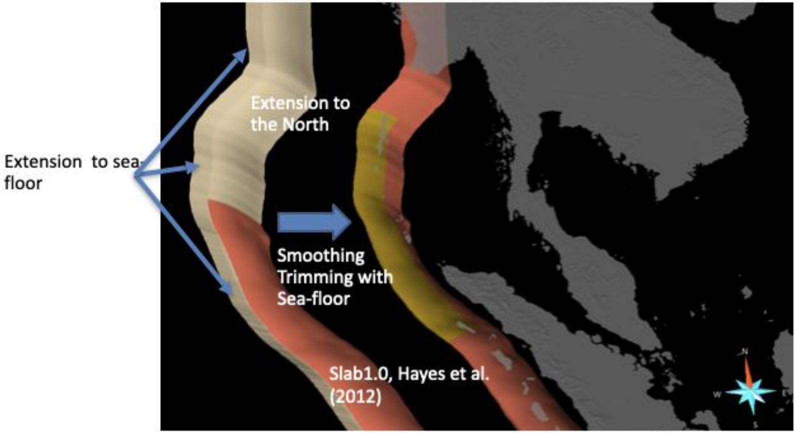
GMSH

- Open-source software
- Many tutorials are available
- Useful for most geometries and everyday mesh sizes
- Limitations when working with highly complex geometries
- Serial mesh generation



Simmodeler

- Requires a license (free for academics)
- Useful for large meshes / complex geometries
- Customized GUI for SeisSol
- Pumgen library for parallel meshing on HPC
- Check out this tutorial in the docs



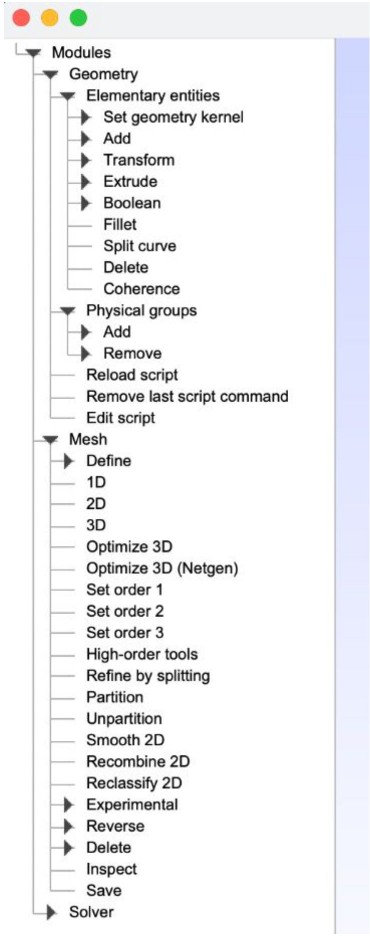
47

GoCAD interface for complex geometries

Working with GMSH

You can use GMSH in multiple ways:

- The **GUI** is useful for getting familiar with GMSH, building simple geometries, and debugging
- GMSH has a **built-in scripting language**, and commands get added to .geo files which can be meshed from the command line (or opened with the GUI)
- GMSH has **APIs** for C++, C, **Python**, Julia or Fortran which permit (i) a programmatic approach to building geometries and (ii) interfacing with other code



Creating geometries using elementary entities in GMSH

GMSH builds geometries using what they call “elementary entities”. Each entity has a type (Point, Curve, Surface, Volume) and a tag (an integer, usually starting at 1).

Points

- Point(3) = {0.3, 0.2, 0, h_min};
- gmsh.model.geo.addPoint(0.3, 0.2, 0, h_min, 3);

Lines/Curves

Creating geometries using elementary entities in GMSH

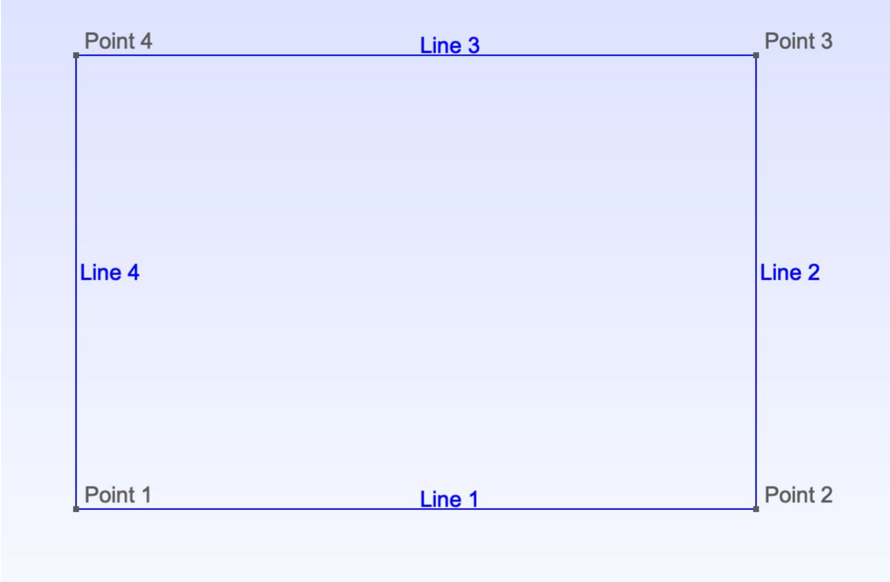
GMSH builds geometries using what they call “elementary entities”. Each entity has a type (Point, Curve, Surface, Volume) and a tag (an integer, usually starting at 1).

Points

- Point(3) = {0.3, 0.2, 0, h_min};
- gmsh.model.geo.addPoint(0.3, 0.2, 0, h_min, 3);

Lines/Curves

- Line(1) = {1, 2};
- gmsh.model.geo.addLine(1, 2, 1)



Creating geometries using elementary entities in GMSH

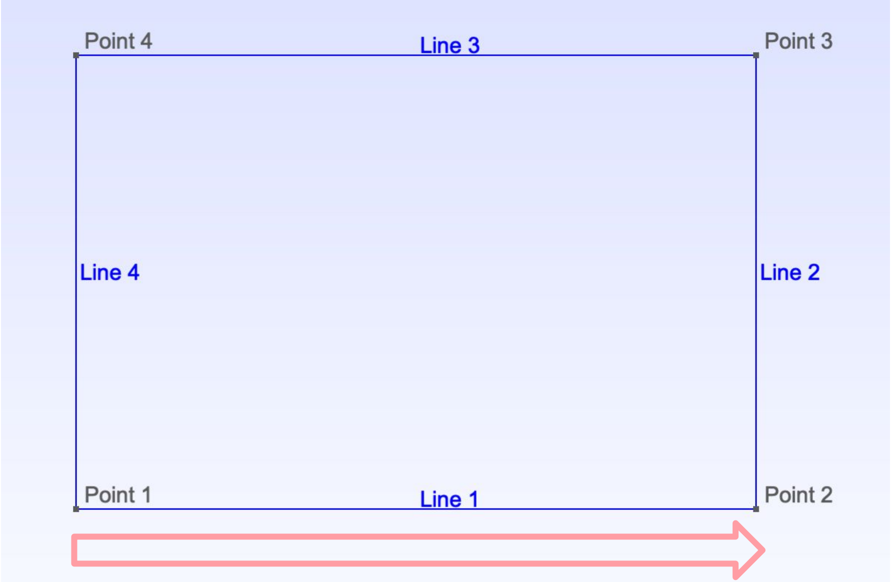
GMSH builds geometries using what they call “elementary entities”. Each entity has a type (Point, Curve, Surface, Volume) and a tag (an integer, usually starting at 1).

Points

- Point(3) = {0.3, 0.2, 0, h_min};
- gmsh.model.geo.addPoint(0.3, 0.2, 0, h_min, 3);

Lines/Curves

- Line(1) = {1, 2};
- gmsh.model.geo.addLine(1, 2, 1)



Creating geometries using elementary entities in GMSH

GMSH builds geometries using what they call “elementary entities”. Each entity has a type (Point, Curve, Surface, Volume) and a tag (an integer, usually starting at 1).

Points

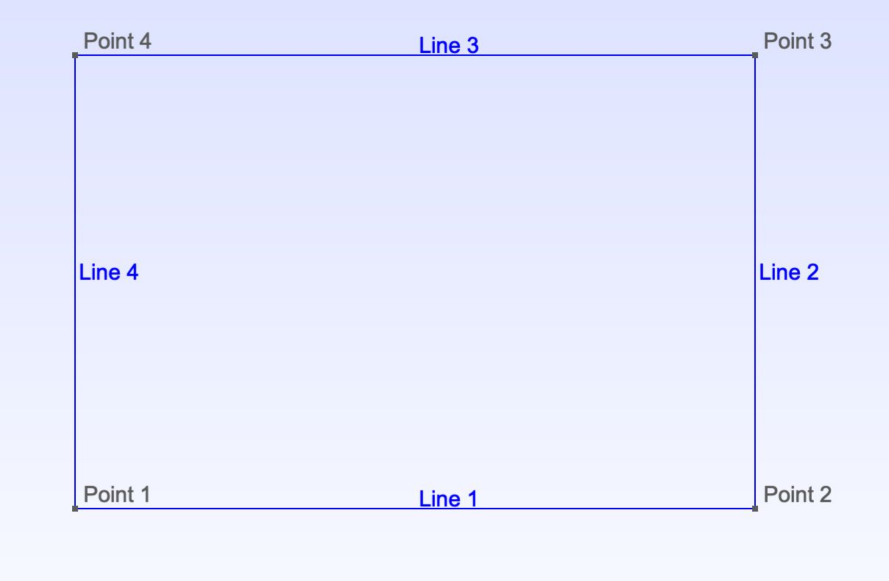
- Point(3) = {0.3, 0.2, 0, h_min};
- gmsh.model.geo.addPoint(0.3, 0.2, 0, h_min, 3);

Lines/Curves

- Line(1) = {1, 2};
- gmsh.model.geo.addLine(1, 2, 1)

Curve Loops

- Curve Loop(111) = {4, 1, -2, 3};
- CL = gmsh.model.geo.addCurveLoop([4, 1, -2, 3], 111)



Creating geometries using elementary entities in GMSH

GMSH builds geometries using what they call “elementary entities”. Each entity has a type (Point, Curve, Surface, Volume) and a tag (an integer, usually starting at 1).

Points

- Point(3) = {0.3, 0.2, 0, h_min};
- gmsh.model.geo.addPoint(0.3, 0.2, 0, h_min, 3);

Lines/Curves

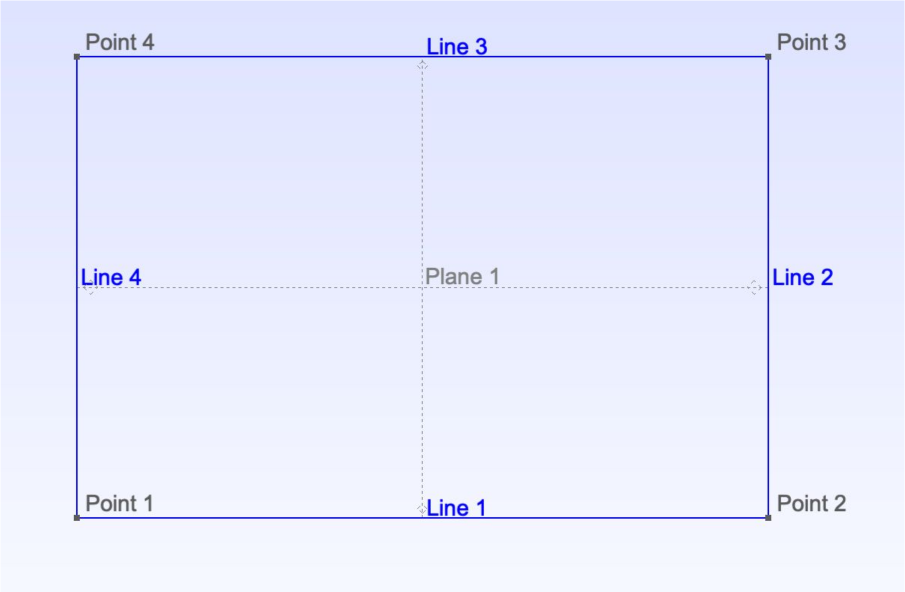
- Line(1) = {1, 2};
- gmsh.model.geo.addLine(1, 2, 1)

Curve Loops

- Curve Loop(111) = {4, 1, -2, 3};
- CL = gmsh.model.geo.addCurveLoop([4, 1, -2, 3], 111)

Surfaces

- Plane Surface(1) = {111};
- PS = gmsh.model.geo.addPlaneSurface([CL], 1)



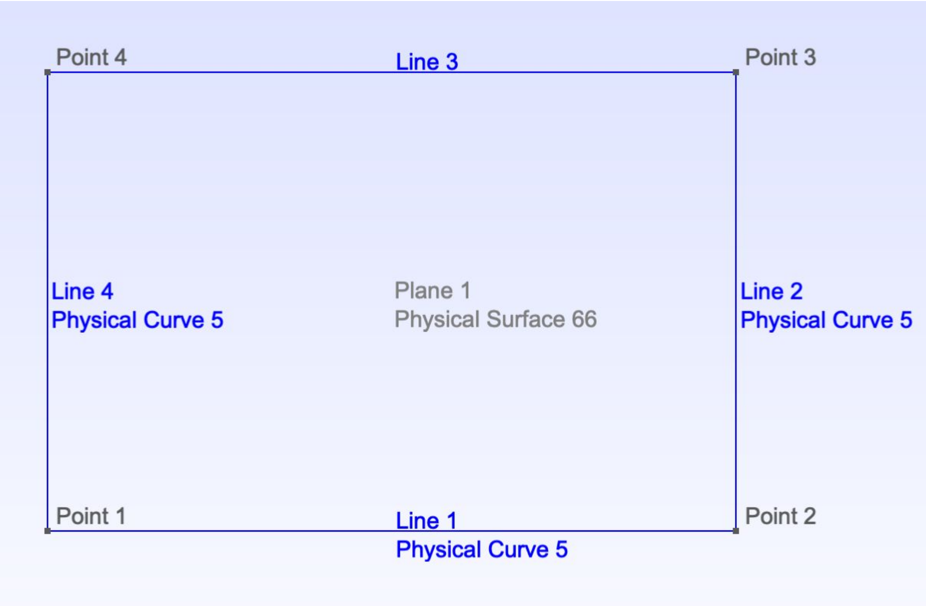
Tagging physical entities

Using GMSH scripting:

```
Physical Curve(5) = {1, 2, 4};

Physical Surface(66) = {1};
```

Using the Python API:

```
gmsh.model.addPhysicalGroup(1, [1, 2, 4], 5)

gmsh.model.addPhysicalGroup(2, [1], 66)
```



Generating the mesh

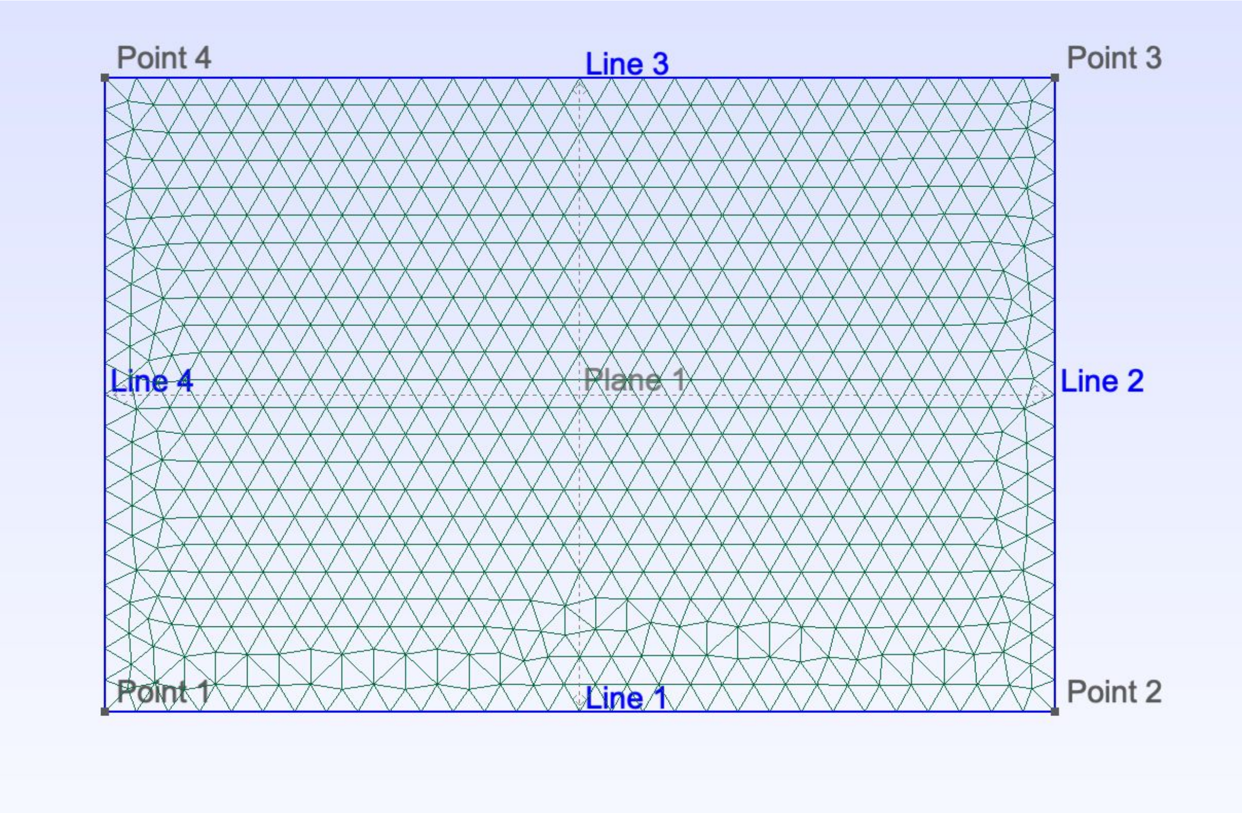
Using the command line:

gmsh -2 tutorial_1.geo

Using the Python API:

gmsh.model.mesh.generate(2)

gmsh.write("t1.msh")



Setting the mesh size

In addition to setting the mesh size at points, GMSH uses mesh size field functions to control local mesh size.

- Distance Field: returns the distance from points or curves (points equidistantly spaced along curves)
- Threshold Field: sets the mesh size as a function of distance

```
// SizeMax -                     /------------------
//                              /
//                             /
//                            /
// SizeMin -o----------------/
//          |                |    |
//        Point         DistMin  DistMax
```

Setting the mesh size

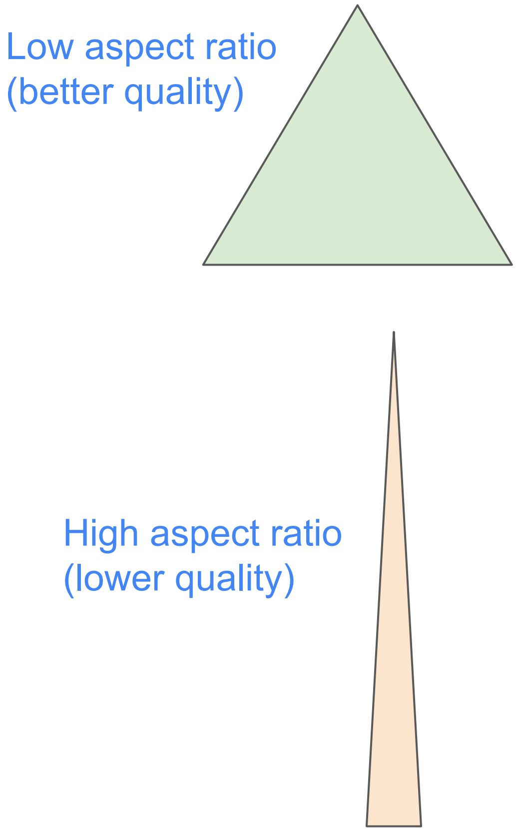
In addition to setting the mesh size at points, GMSH uses mesh size field functions to control local mesh size.

- Distance Field: returns the distance from points or curves (points equidistantly spaced along curves)
- Threshold Field: sets the mesh size as a function of distance
- MathEval Field: modulates mesh size as a mathematical function of spatial coordinates or distance
- Box Field: imposes a step change in element size inside a box
- …and much more

Mesh quality

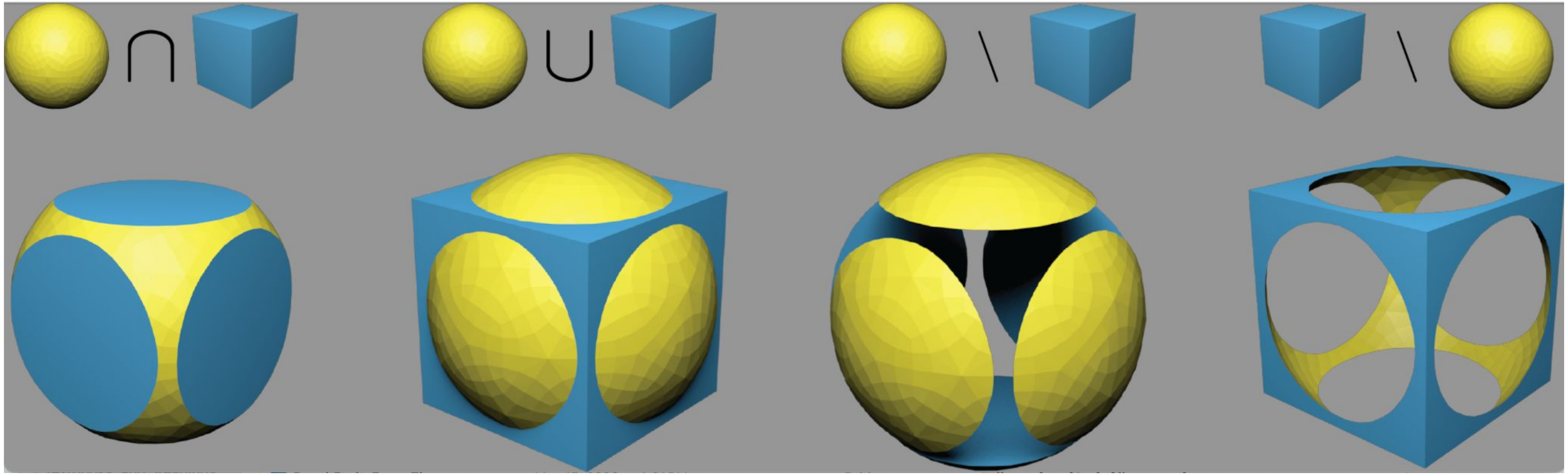
Our goal is to create meshes that are **valid** and of acceptable **quality**.

- **Invalid mesh**: a mesh with “inverted” cells, where edges have crossed each other so the cell has a self-intersecting geometry, or where a cell edge length is zero.
- **Mesh quality** can be measured using metrics like the Scaled Jacobian or the Aspect Ratio (AR).
- The range of acceptable mesh quality is application-dependent.
- Typically for SeisSol simulations we aim for meshes with max AR between 20 to 40 (this is a rule of thumb) - the lower the AR value, the better.



Boolean Operations

- Boolean operations are applied to curves, surfaces and volumes to calculate their intersections, unions, and differences.
- They usually involve a “tool” applied to an “object”
- They are crucial for obtaining a conforming or “watertight” mesh



https://pymesh.readthedocs.io/en/latest/mesh_boolean.html

Converting to SeisSol-compatible formats with Pumgen

- Once we have meshed our geometries and generated .msh files, we must convert them to SeisSol-compatible formats
- For this task we use PUMGen, an open-source tool for creating and exporting meshes in an efficient format
- The basic syntax on the command line:

```
pumgen -s msh4 tpv13_training.msh
```

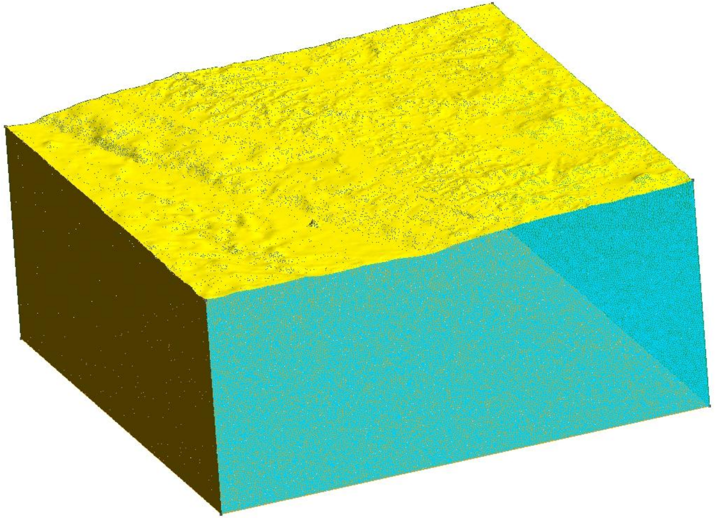
or to run in parallel:

```
mpirun -n 1 pumgen -s msh4 tpv13_training.msh
```

- This creates a file in the efficient, open-source HDF5 format, which supports large, complex, heterogeneous data

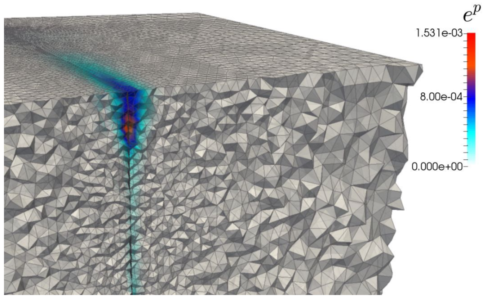
Topography and other complex meshing tasks

- Adding topography to a mesh is possible in GMSH (see the tutorial here)
- However, for topography and other more complex meshing tasks, we recommend using SimModeler
- A beginning SimModeler tutorial can be found here
- A tutorial on the SimModeler CAD workflow can be found here , which shows how to build the structural model of the Palu earthquake dynamic rupture scenario (Ulrich et al., 2019)



Adding topography using GMSH

My first dynamic rupture model - TPV13 community benchmark
3D normal-faulting earthquake with off-fault plasticity



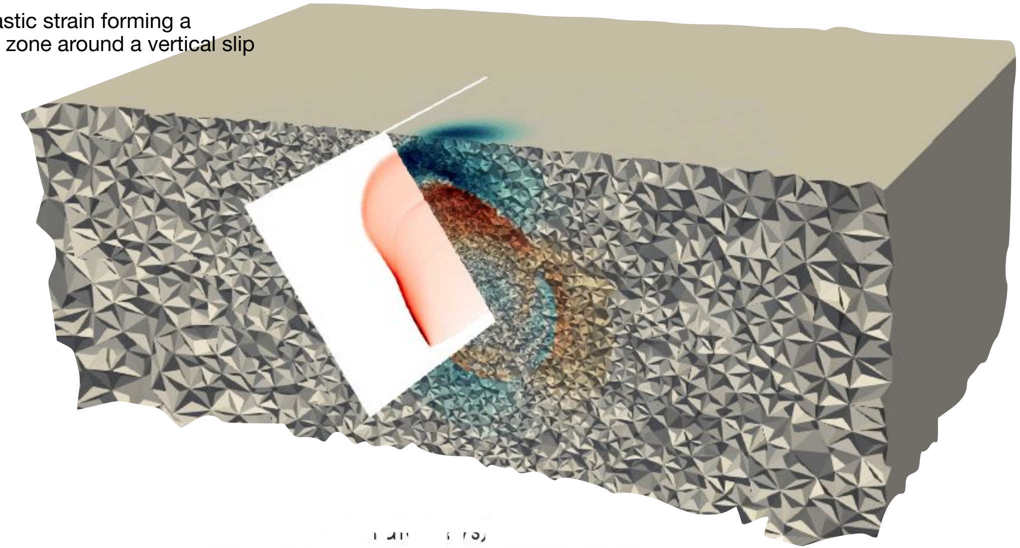
Accumulated plastic strain forming a ‘flower-like’ fault zone around a vertical slip interface

A Suite of Exercises for Verifying Dynamic Earthquake Rupture Codes

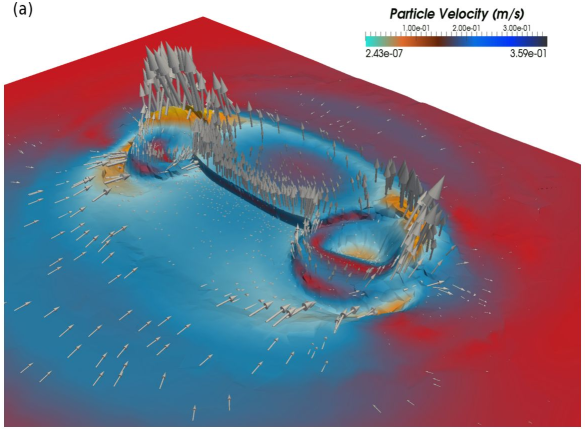
Ruth A. Harris; Michael Barall; Brad Aagaard; Shuo Ma; Daniel Roten; Kim Olsen; Benchun Duan; Dunyu Liu; Bin Luo; Kangchen Bai; Jean-Paul Ampuero; Yoshihiro Kaneko; Alice-Agnes Gabriel; Kenneth Duru; Thomas Ulrich; Stephanie Wollherr; Zheqiang Shi; Eric Dunham; Sam Bydlon; Zhenguo Zhang; Xiaofei Chen; Surendra Nadh Somala; Christian Pelties; Josué Tago; Victor Manuel Cruz-Atienza; Jeremy Kozdon; Eric Daub; Khurram Aslam; Yuko Kase; Kyle Withers; Luis Dalguer

Seismological Research Letters (2018) 89 (3): 1146–1162.

https://doi.org/10.1785/0220170222 Article history



(a)



My first dynamic rupture model

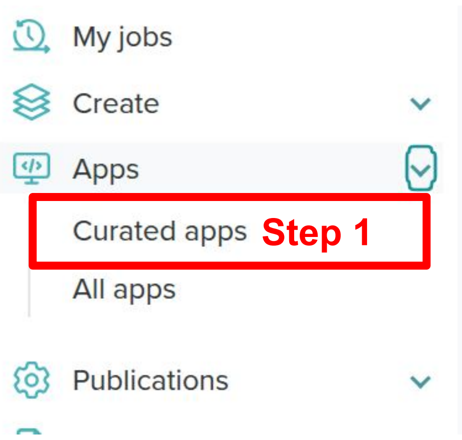
TPV13 community benchmark:

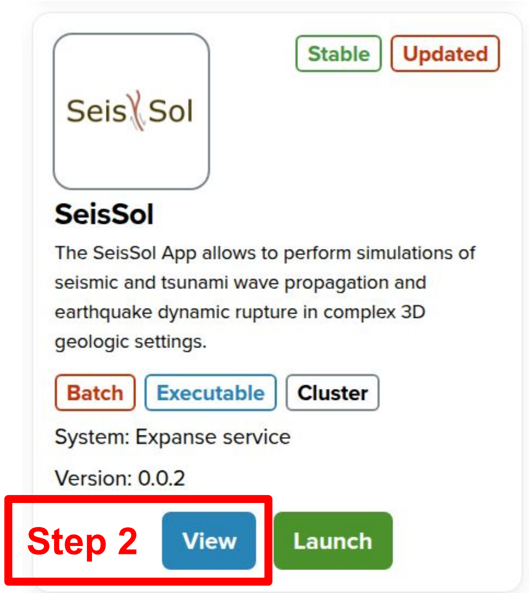
3D normal-faulting earthquake with off-fault plasticity

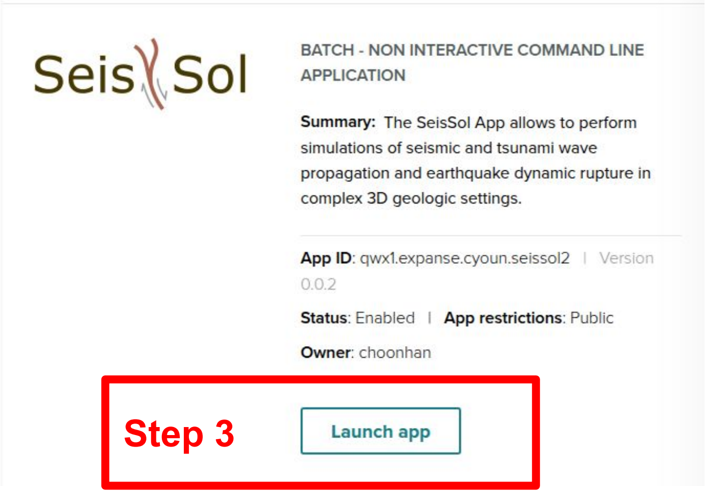
- **Task 1 - Run your first SeisSol simulation! Model a 3D dynamic rupture earthquake scenario with off-fault plastic deformation**
    - Task 1a - Running SeisSol (TPV13 benchmark) on Gateway
    - Task 1b - Understanding SeisSol input files
- **Task 2 - Visualization**
- **Task 3 - Understand parameter choices in the benchmark**
- **Task 4 - Rupture speed analysis: what are the conditions for supershear rupture?**
- **Task 5 - Turn-off plasticity and analyze the difference between the two models**

Task 1a - Running SeisSol on Gateway

- Log in through the Gateway home page
  https://quakeworx.org







https://github.com/SeisSol/Training ← To download the input files

Task 1a - Running SeisSol on Gateway

- Upload input files for SeisSol

SeisSol app type

SeisSol
SeisSol nonlinear materials

Job name *

TPV13

Specify a name for this job

Parameter file (.par)

parameters.par (2.59 KB) **Remove**

Mesh file (.msh, .xdmf, .puml.h5)

Choose Files No file chosen

tpv13_training.puml.h5 (1.58 MB)

Other files (csv, dat, txt, yaml, nc)

Choose Files No file chosen

tpv12_13_fault.yaml (953 bytes)

tpv12_13_material.yaml (296 bytes)

tpv12_13_initial_stress.yaml (823 bytes)

Number of nodes *

1

The number of nodes to use for running this app.

Cores per node *

32

The number of cores to use per node.

System *

Expanse service

The system to run this app on.

Batch Logical Queue *

shared

The batch logical queue to run this app on.

Save Draft **Submit**

https://github.com/SeisSol/Training ← **To download the input files**

Task 1a - Running SeisSol on Gateway

- The job is running

Data retention (scratch): Action required: Job data is stored on the scratch filesystem and will be automatically purged **90 days after job creation** per Expanse cluster policy. Download any job data you want to keep **before** the purge date; after that, data may be permanently deleted and not recoverable.
Job created: 2026-06-08 14:52 CEST
Purge date: 2026-09-06 14:52 CEST
Expanse scratch storage policy

Last message: Setting job status to running.

Remote job ID: 50516168

Remote job submitted: 2026-06-08T12:53:09

Remote job started: 2026-06-08T12:53:55

Job usage details

Resource details

Processing unit: CPU
Node count: 1
Cores per node: 32
Memory: 64000 MB

App: SeisSol

System: Expanse service

Owner: mmarchandon

Created: Jun 08, 2026

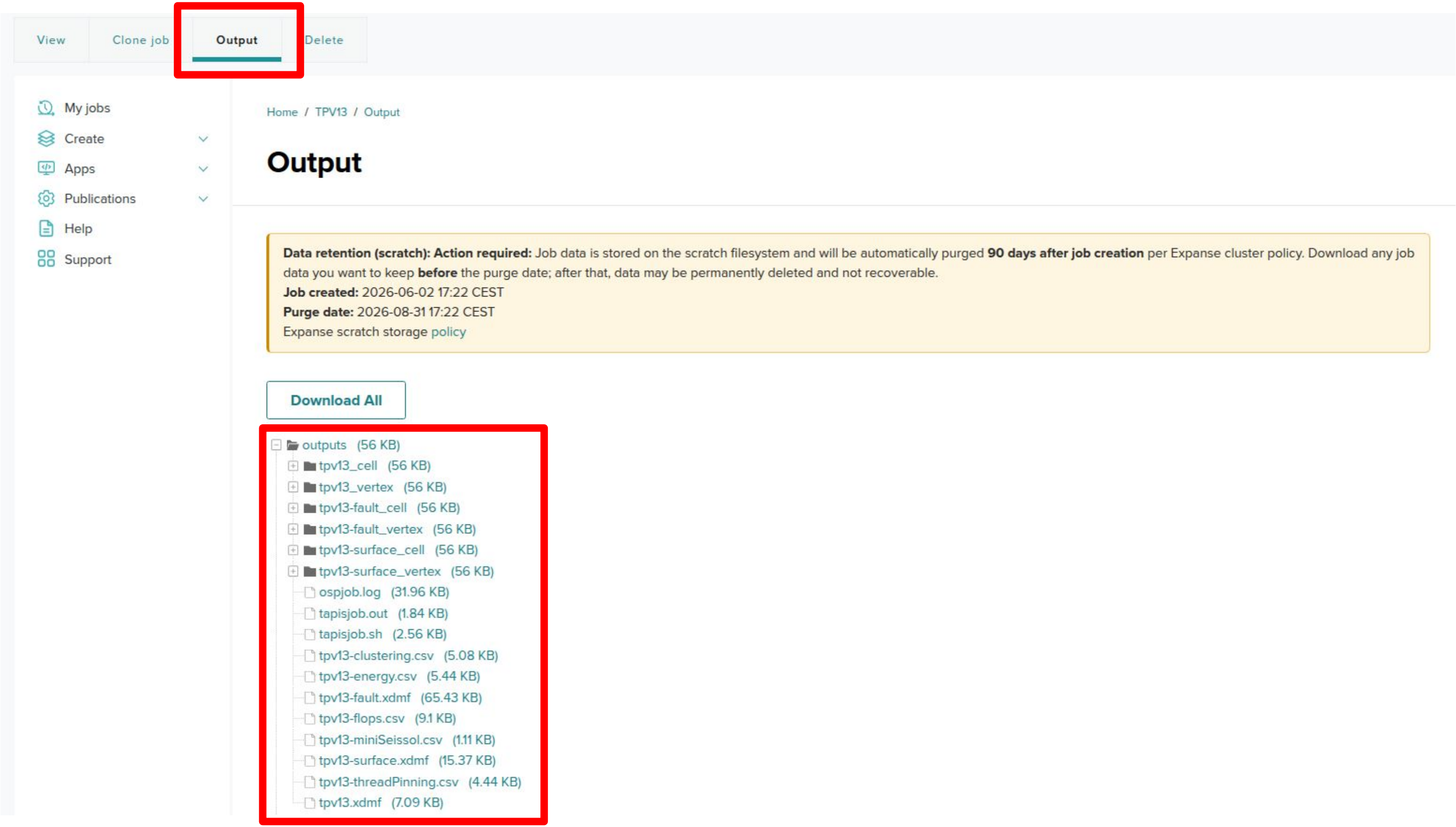
Tapis UUID: e3a1d0d3-95c3-4a24-9d3f-e22e83716aa9-007

- Running
- Queued
- Staging job
- Staging inputs
- Processing inputs

Terminate Job

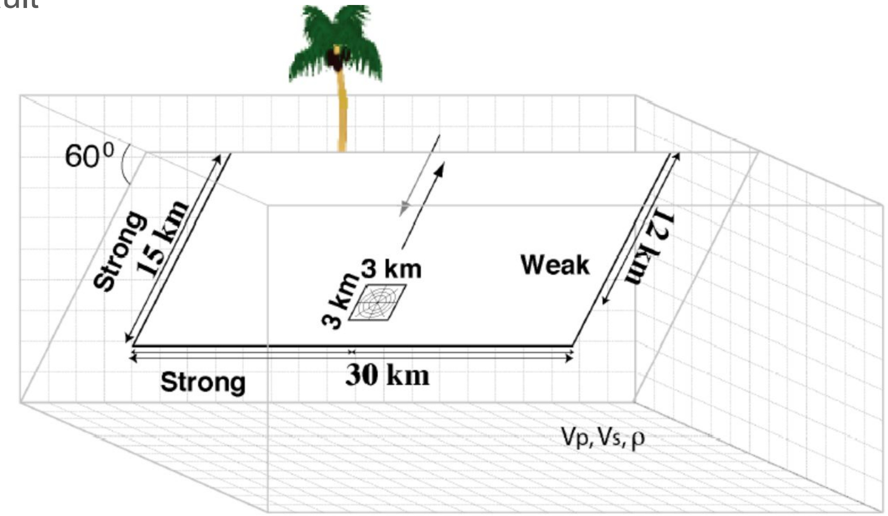
Task 1a - Running SeisSol on Gateway

- Where to find the outputs



TPV13 community benchmark
3D normal-faulting earthquake with off-fault plasticity

- Spontaneous rupture on a 60°-dipping planar normal fault
- Homogeneous half-space
  Vp=5716 m/s, Vs=3300 m/s, rho=2700 km/m3
- Non-associative Drucker-Prager plasticity
- Linear-slip weakening friction (LSW)
- Initial stress conditions are depth-dependent
- Prescribed nucleation: low-friction square patch
- Supershear initial conditions



full benchmark description:
https://strike.scec.org/cvws/tpv12_13docs.html

Task 1b - While SeisSol is running, let's checkout SeisSol's input files

- ## Main parameter file: *parameters.par*

```
&equations
!yaml file defining spatial dependence of material properties
MaterialFileName = 'tpv12_13_material.yaml'
!enable off-fault plasticity (ignored in Plasticity=0)
Plasticity = 1
Tv = 0.03
numflux = 'rusanov' ! The numerical flux. Values: 'godunov', 'rusanov'
numfluxnearfault = 'rusanov' ! The numerical flux for faces of cells adjacent to the fault (on cells which have at least one DR face),
excluding the DR faces themselves. Values: 'godunov', 'rusanov'
/

&IniCondition
/

&DynamicRupture
! linear slip-weakening friction
FL = 16                              ! Friction law
!yaml file defining spatial dependence of fault properties
ModelFileName = 'tpv12_13_fault.yaml'

!characteristic time of the forced nucleation
t_0 = 0.5

!reference vector for defining strike and dip direction
XRef = 0.0
YRef = -1.0
ZRef = 0.0
refPointMethod = 1

OutputPointType = 4
/
```

```
tpv13
├── output
├── parameters.par
├── paraview.jpg
├── tpv12_13_fault.yaml
├── tpv12_13_initial_stress.yaml
├── tpv12_13_material.yaml
├── tpv13.ipynb
├── tpv13.jpg
└── tpv13_training.geo
1 directory, 8 files
```

For a detailed description of this file:
https://seissol.readthedocs.io/en/latest/parameter-file.html

Task 1b - SeisSol's input files

- ## Material parameters

parameters.par:

```
&equations
!yaml file defining spatial dependence of material properties
MaterialFileName = 'tpv12_13_material.yaml'
!enable off-fault plasticity (ignored in Plasticity=0)
Plasticity = 1
Tv = 0.03
/
```

tpv12_13_material.yaml:

```
!Switch
[rho, mu, lambda, plastCo, bulkFriction]: !ConstantMap
  map:
    rho:          2700
    mu:       2.9403e+010
    lambda:      2.941e+010
    plastCo:      1.0e+06
    bulkFriction:     0.60
[s_xx, s_yy, s_zz, s_xy, s_yz, s_xz]: !Include tpv12_13_initial_stress.yaml
```

```
tpv13
├── output
├── parameters.par
├── paraview.jpg
├── tpv12_13_fault.yaml
├── tpv12_13_initial_stress.yaml
├── tpv12_13_material.yaml
├── tpv13.ipynb
├── tpv13.jpg
└── tpv13_training.geo
1 directory, 8 files
```

easi: library for the **Eas**y **I**nitialization of models in 3D domains.

https://easyinit.readthedocs.io/en/latest/

Task 1b - SeisSol's input files

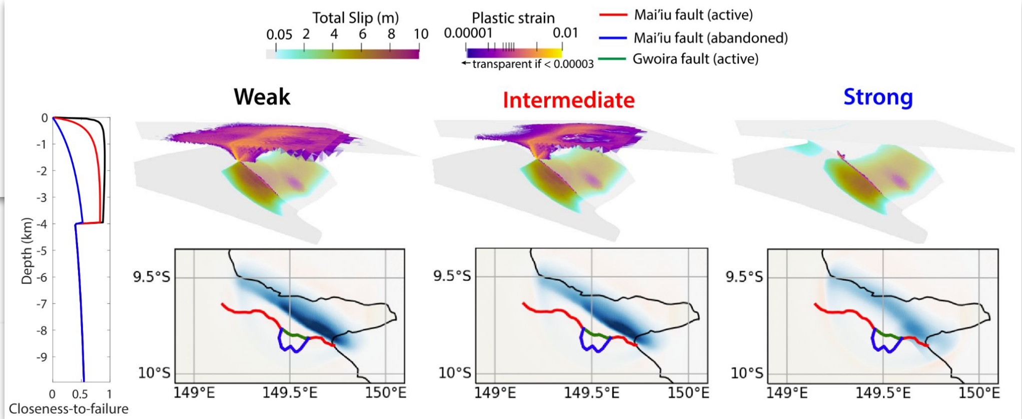
- ## Material parameters

parameters.par:

```
1 &equations
2 !yaml file defining spatial dependence of material properties
3 MaterialFileName = 'tpv12_13_material.yaml'
4 !enable off-fault plasticity (ignored in Plasticity=0)
5 Plasticity = 1
6 Tv = 0.03
7 /
```

tpv12_13_material.yaml:

```
1 !Switch
2 [rho, mu, lambda, plastCo, bulkFriction]: !ConstantMap
3   map:
4     rho:          2700
5     mu:        2.9403e+010
6     lambda:       2.941e+010
7     plastCo:      1.0e+06
8     bulkFriction:      0.60
9 [s_xx, s_yy, s_zz, s_xy, s_yz, s_xz]: !Include tpv12_13_initial_stress.yaml
```

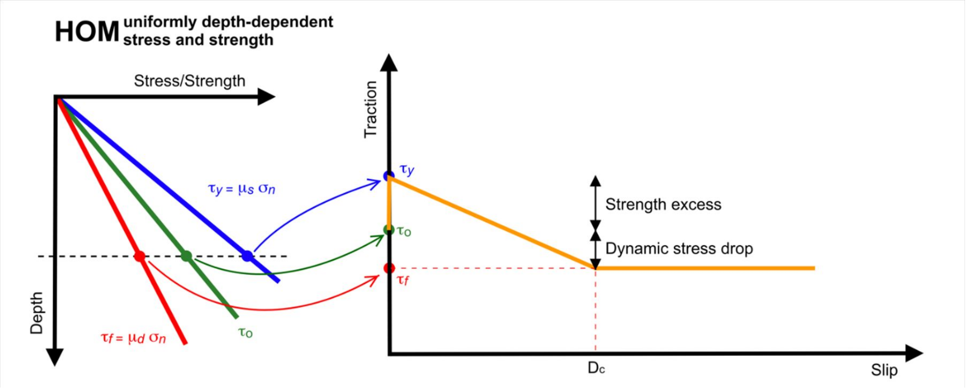


Impact of plastic rock strength on on-fault slip and accumulated off-fault deformation (modified from Biemiller et al., 2023)

Task 1b - SeisSol's input files

- **Friction law**

```
&DynamicRupture
! linear slip-weakening friction
FL = 16                                  ! Friction law
!yaml file defining spatial dependence of fault properties
ModelFileName = 'tpv12_13_fault.yaml'

!characteristic time of the forced nucleation
t_0 = 0.5

!reference vector for defining strike and dip direction
XRef = 0.0
YRef = -1.0
ZRef = 0.0
refPointMethod = 1

OutputPointType = 4
/
```



Tinti et al., 2021

```
tpv13
├── output
├── parameters.par
├── paraview.jpg
├── tpv12_13_fault.yaml
├── tpv12_13_initial_stress.yaml
├── tpv12_13_material.yaml
├── tpv13.ipynb
├── tpv13.jpg
└── tpv13_training.geo
1 directory, 8 files
```

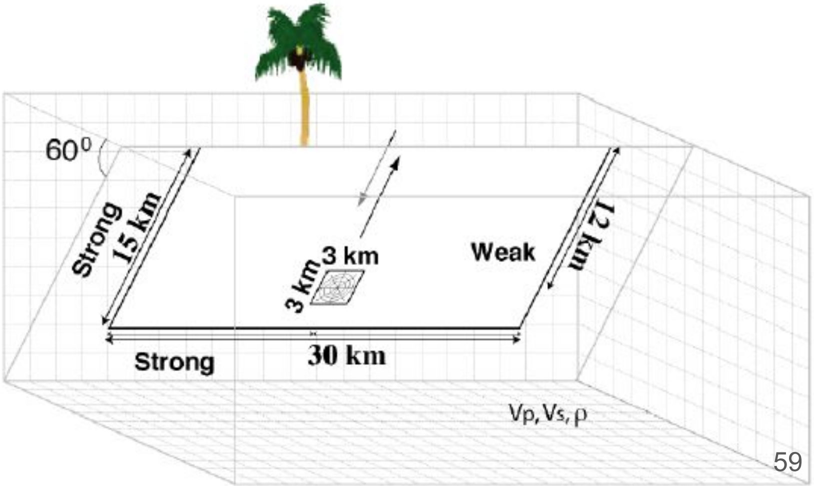
For more info about implemented friction laws in SeisSol →
https://seissol.readthedocs.io/en/latest/dynamic-rupture.html#friction-laws

Task 1b - SeisSol's input files

- ## Friction parameters & nucleation

```yaml
!Switch
[s_xx, s_yy, s_zz, s_xy, s_yz, s_xz]: !Include tpv12_13_initial_stress.yaml
[mu_d, d_c, cohesion]: !ConstantMap
      map:
        mu_d:        0.10
        d_c:         0.50
        cohesion: -200000
[mu_s]: !LuaMap
  returns: [mu_s]
  function: |
    function f(x)
        mu_s = 0.7
        nucleation_size = 3000.0
        along_width_nucleation_center = 12000.0
        sinDip = math.sin(60 * math.pi / 180)
        x_in_nucleation = (math.abs(x["x"]) <= 0.5 * nucleation_size)
        z_in_nucleation = (
            math.abs(x["z"] + along_width_nucleation_center * sinDip) <=
            0.5 * nucleation_size * sinDip
        )
        if x_in_nucleation and z_in_nucleation then
            mu_s = 0.54
        end

        return { mu_s = mu_s }
    end

[Tnuc_n, Tnuc_s, Tnuc_d, forced_rupture_time]: !ConstantMap
    map:
        Tnuc_n: 0
        Tnuc_s: 0
        Tnuc_d: 0
        forced_rupture_time: 1e10
```

```
tpv13
├── output
├── parameters.par
├── paraview.jpg
├── tpv12_13_fault.yaml
├── tpv12_13_initial_stress.yaml
├── tpv12_13_material.yaml
├── tpv13.ipynb
├── tpv13.jpg
└── tpv13_training.geo
1 directory, 8 files
```

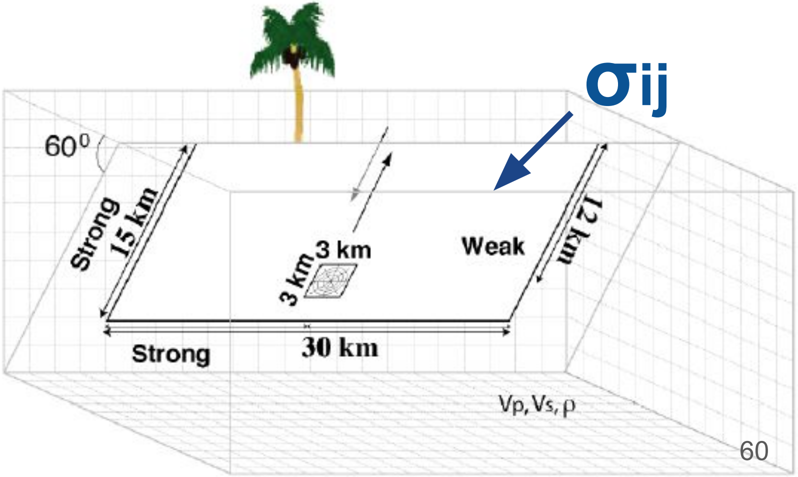


Task 1b - SeisSol's input files.

- Initial stress as a cartesian stress tensor

```
!Switch
[s_xy, s_yz, s_xz]: !ConstantMap
  map:
    s_xy: 0
    s_yz: 0
    s_xz: 0
[s_xx, s_yy, s_zz]: !LuaMap
  returns: [s_xx, s_yy, s_zz]
  function: |
        function f (x)
          depth = math.abs(x["z"])
          s_max_minus_Pf = 9.8 * (2700.0 - 1000.0);
          if (depth<=11951.15) then
            s3_to_s1 = 0.3496
            -- math.floor(a+0.5) is round
            s_xx =  -0.01 * math.floor(100.0 * (0.5 * (1.0 + s3_to_s1) * s_max_minus_Pf)
            s_yy =  -0.01 * math.floor(100.0 *              (s3_to_s1 * s_max_minus_Pf)
            s_zz = -s_max_minus_Pf * depth
          else
            s_xx = -s_max_minus_Pf * depth
            s_yy = s_xx
            s_zz = s_xx
          end
          return {
            s_xx = s_xx,
            s_yy = s_yy,
            s_zz = s_zz,
          }
        end
```

```
tpv13
├── output
├── parameters.par
├── paraview.jpg
├── tpv12_13_fault.yaml
├── tpv12_13_initial_stress.yaml
├── tpv12_13_material.yaml
├── tpv13.ipynb
├── tpv13.jpg
└── tpv13_training.geo
1 directory, 8 files
```



Task 1b - SeisSol's input files

- **Simulation output**

```
!see: https://seissol.readthedocs.io/en/latest/fault-output.html
!parameterize paraview fault output
&Elementwise
printtimeinterval_sec = 1.0 ! Time interval at which output will be written
OutputMask = 1 1 1 1 1 1 1 1 1 1 1 ! turn on and off fault outputs
refinement_strategy = 2
refinement = 1
!vtkorder = 3
/
```

```
tpv13
├── output
├── parameters.par
├── paraview.jpg
├── tpv12_13_fault.yaml
├── tpv12_13_initial_stress.yaml
├── tpv12_13_material.yaml
├── tpv13.ipynb
├── tpv13.jpg
└── tpv13_training.geo
1 directory, 8 files
```

```
&Output
OutputFile = 'outputs/tpv13'
Format = 6                         ! Format (10=no output, 6=hdf5 output)
!             |stress     |vel
iOutputMask = 0 0 0 0 0 0 1 1 1
!                  |strain     |eta
iPlasticityMask = 0 0 0 0 0 0 1
TimeInterval =  4.                 ! Index of printed info at time
refinement = 1

! Free surface output
SurfaceOutput = 1
SurfaceOutputRefinement = 0
SurfaceOutputInterval = 2.0
surfacevtkorder = 2

pickdt = 0.005                     ! Pickpoint Sampling
RFileName = 'tpv12_13_receivers.dat'      ! Record Points in extra file

! xdmfWriterBackend = 'posix' ! (optional) The backend used in fault, wavefield,
! and free-surface output. The HDF5 backend is only supported when SeisSol is compiled with
! HDF5 support.

EnergyOutput = 1 ! Computation of energy, written in csv file
EnergyTerminalOutput = 1 ! Write energy to standard output
EnergyOutputInterval = 0.5

Checkpoint = 0
/
```

Task 2 - Visualization of SeisSol output

For an explanation of each output field:
https://seissol.readthedocs.io/en/latest/fault-output.html

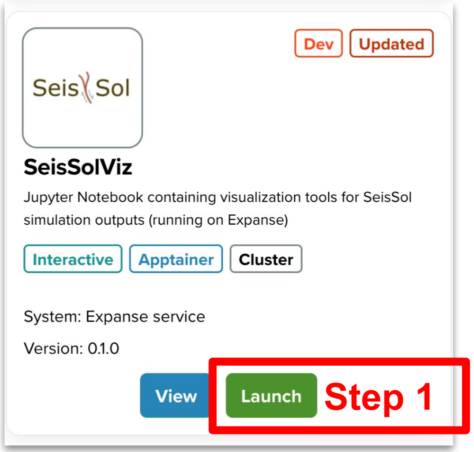
OutputMask

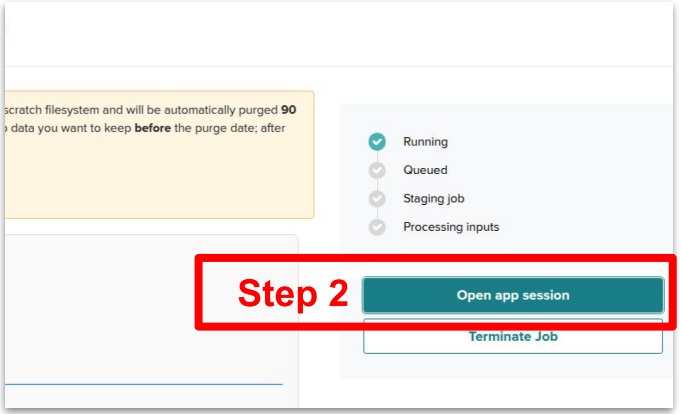
The **OutputMask** variable allows you to select specific unknowns for visualization. You can toggle the output writing of each unknown by changing its corresponding bit in the OutputMask array.

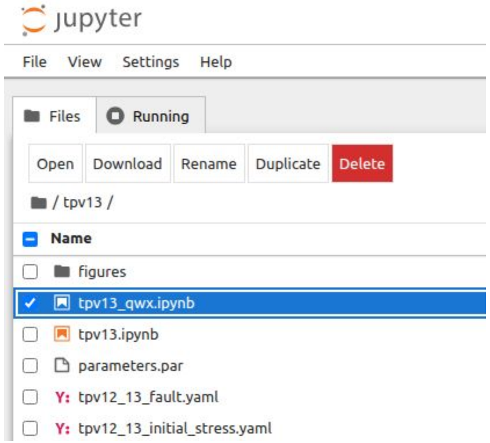
Here's what each bit in the OutputMask array represents:

1. **SRs** and **SRd**: Slip rates in along-strike and along-dip directions
2. **T_s**, **T_d**: Shear stress in strike and dip directions, **P_n**: Normal stress
3. **u_n**: Fault normal velocity (Note: SeisSol does not allow for fault opening (mode I))
4. **Mud**: Current effective friction coefficient, **StV**: State variable for RS friction
5. **Ts0**,**Td0**,**Pn0**: Total shear and normal stresses, including initial stresses
6. **Sls** and **Sld**: Fault slip in along-strike and -dip directions
7. **Vr**: Rupture velocity, computed from the spatial derivatives of the rupture time
8. **ASl**: Accumulated slip
9. **PSR**: Peak slip rate
10. **RT**: Rupture time
11. **DS**: Dynamic stress time. With LSW, the time at which ASl>D_c. With RS, the time at which mu <= (f0 + mu_w). DS can be used to evaluate the process zone size.
12. **P_f** and **Tmp**: Only with thermal pressurization, pore pressure and temperature

Task 2 - Visualization of SeisSol output using SeisSolviz on QUAKEWORX







Step 3: Navigate to the *tpv13* folder and open *tpv13_qwx.ipynb*

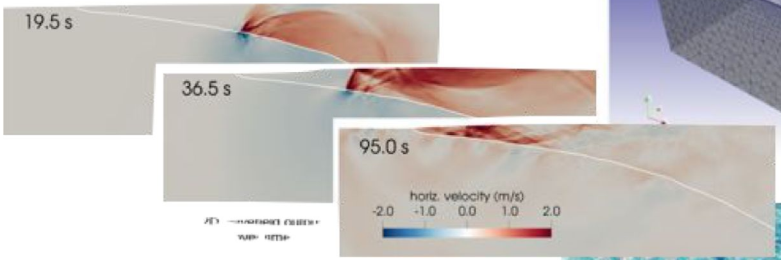
Let's move to the Jupyter Notebook :)

In this notebook, we will skip the meshing part, which was already covered by Gabrielle in the previous section, and move directly to the visualization part.

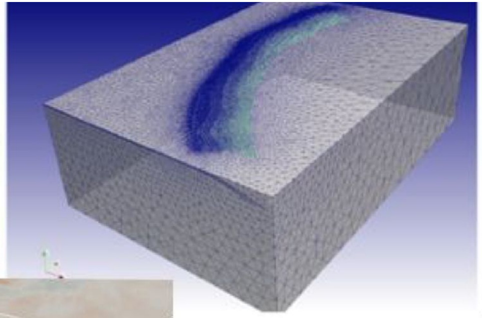
Task 2 - Visualization of SeisSol output using Paraview

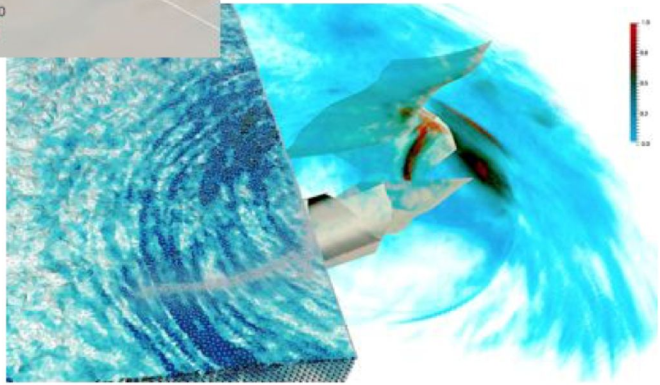
You can also visualize the SeisSol output in Paraview on your local machine. For that, you need to 1) download Paraview and 2) download the output files from your Quakeworx SeisSol simulation.

ParaView for visualizing your results



2D SeisSol (Laptop)





Kaikoura: 29 mio elements, 90 sec., 2 hours on 3000 Sandy Bridge cores

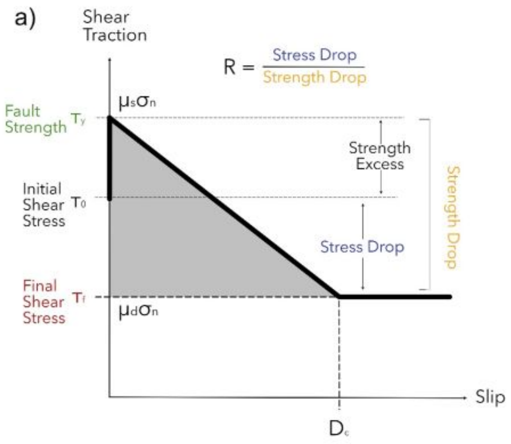
- time series (seismograms), 2D (fault, free surface) and 3D output fields
- paraview filters for simple post-processing

Task 3 - Understand parameter choices in the benchmark

Consider this code block within the notebook. By computing the relative pre-stress ratio R in the nucleation area, verify that the nucleation area is critically stressed (R > 1).

```python
import sympy as sp
import numpy as np

# Define symbolic variables
s3_to_s1, sigma_max, dip, mu_s, mu_d = sp.symbols("s3_to_s1 sigma_max dip mu_s mu_d")

# stress tensor
stress = sp.diag((1 + s3_to_s1) * sigma_max / 2, s3_to_s1 * sigma_max, sigma_max)
# normal and along dip unit vectors
u_n = sp.Matrix([0, -sp.sin(dip), sp.cos(dip)])
udip = sp.Matrix([0, sp.cos(dip), sp.sin(dip)])

# tractions
traction = stress * u_n

sigma_n = traction.dot(u_n)
print("sigma_n:", sigma_n)
sigma_d = traction.dot(udip)
print("sigma_d:", sigma_d)

R = (sigma_d - mu_d * sigma_n) / ((mu_s - mu_d) * sigma_n)
print("relative prestress ratio R:", R)

params = {"dip": 60 * np.pi / 180, "s3_to_s1": 0.3495, "mu_s": 0.7, "mu_d": 0.1}
print(
    f"relative prestress ration R evaluated with {params}:",
    R.subs(params),
)

print("sigma_n:", sigma_n.subs(params))
print("sigma_d:", sigma_d.subs(params))
```

$$R = \frac{\tau_0 - \mu_d \sigma_n}{(\mu_s - \mu_d)\,\sigma_n}$$

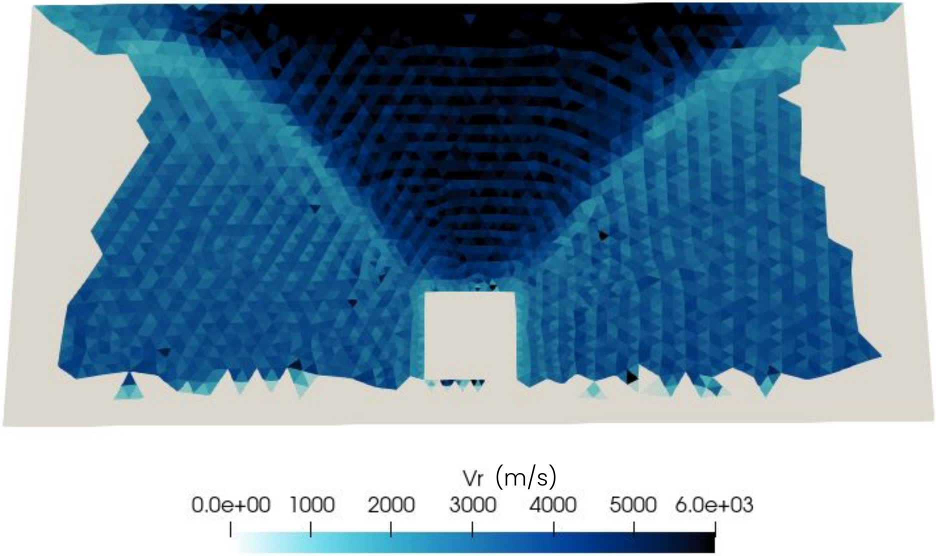


modified from Yin et al. (2025)

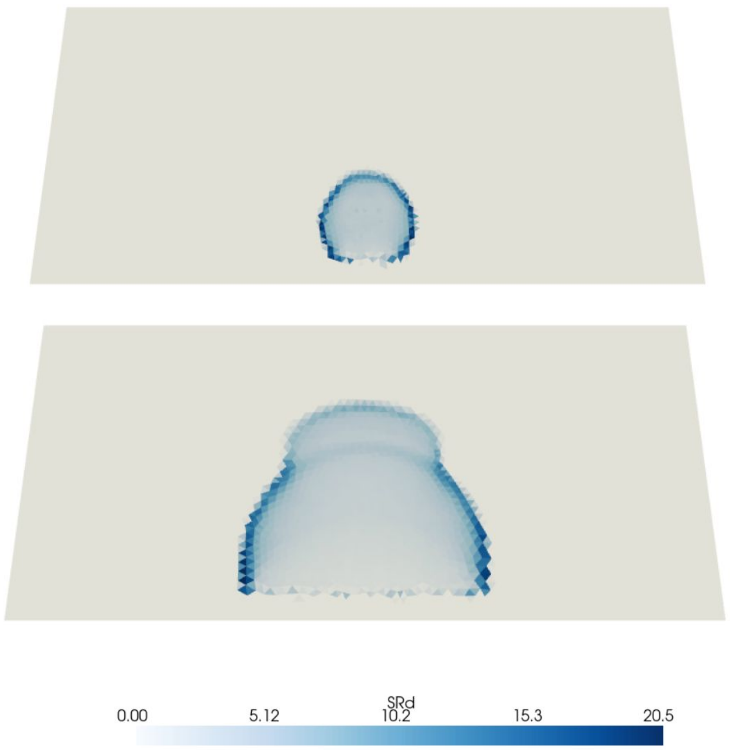
Task 4 - Rupture speed analysis

We analyze the mechanism causing supershear in this setup and disable it.

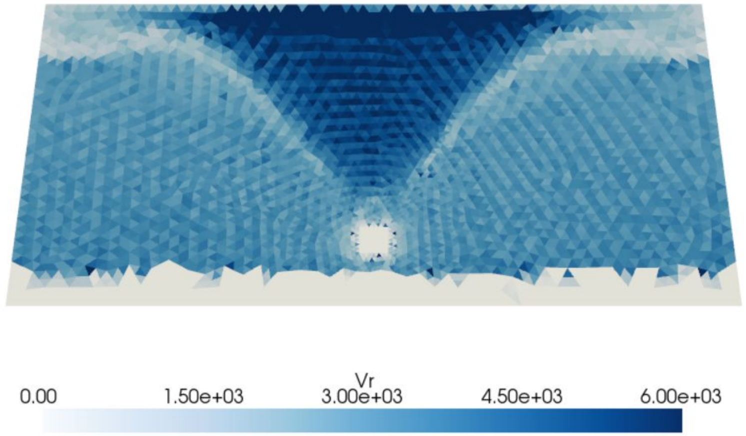
- Examine the earthquake rupture speed and identify the conditions that trigger supershear rupture.
- Disable the current supershear triggering mechanism by changing nucleation size to 1.2km in \`tpv12_13_fault.yaml\`
- Decrease the initial shear stress (that is changing \`s3_to_s1\` from 0.3495 to 0.4 in \`tpv12_13_initial_stress.yaml\`) to suppress the new supershear transition



Task 4 - Rupture speed analysis

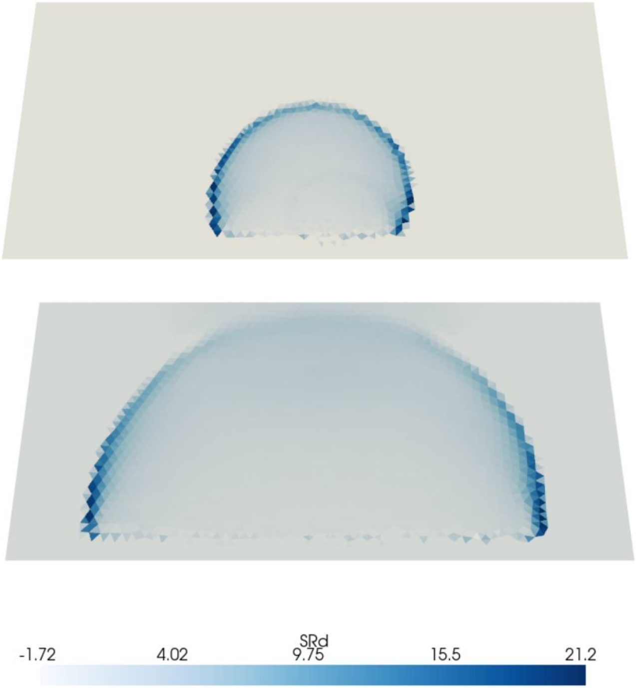


- nucleation size: 1.2km



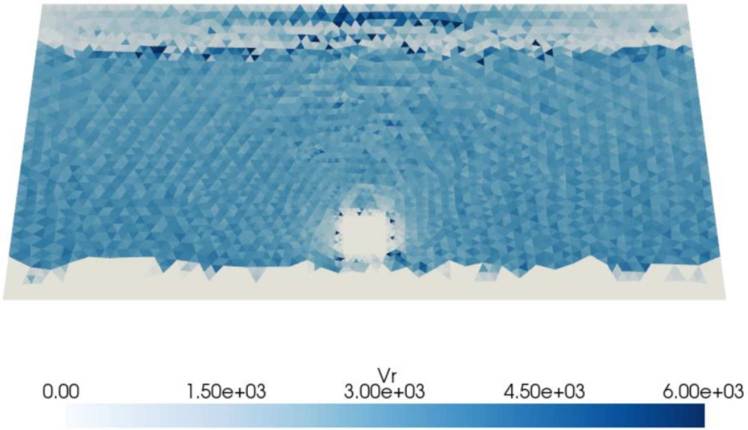
Task 4 - Rupture speed analysis

- s3_to_s1 = 0.4
- nucleation size 1.8km

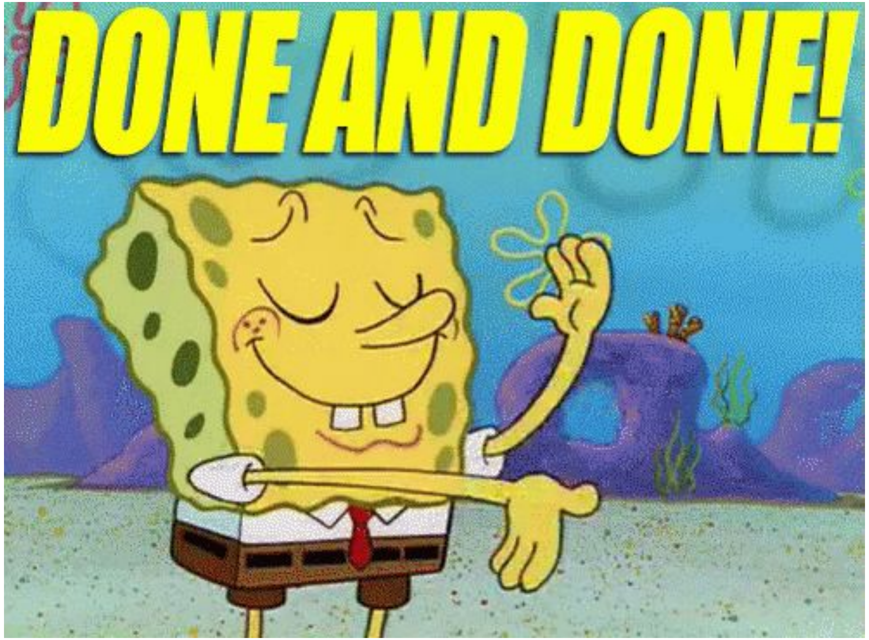


$$\text{Stress excess} = \frac{\tau_s - \tau_0}{\tau_0 - \tau_d}$$

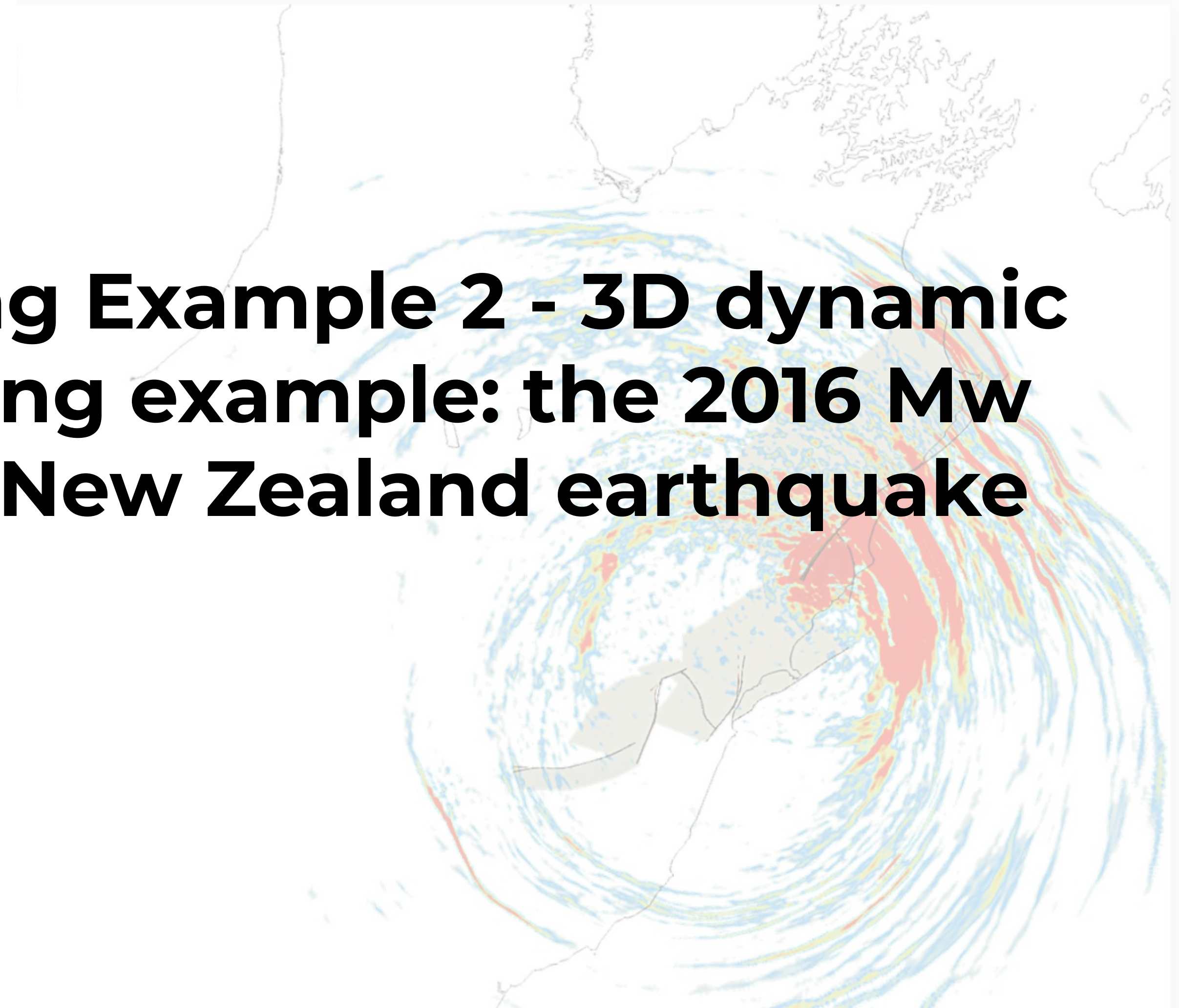
τ_s: static shear strength
τ_d: dynamic shear strength
τ_0: initial shear stress



Task 5 - Turn-off plasticity and compare the elastic and elastoplastic models


DONE AND DONE!

SeisSol Training Example 2 - 3D dynamic rupture training example: the 2016 Mw 7.8 Kaikōura, New Zealand earthquake



Ulrich et al., Nat. Comms., 2019

Let's start by running the SeisSol Kaikoura simulation using the SeisSol App in the Quakeworx Gateway

Launch

SeisSol app type *

Step 1:

SeisSol

SeisSol nonlinear materials

Job name *

Step 2:

SeisSol_Kaikoura

Step 3:

Parameter file (.par)

The results must be directed to a directory named

Browse... No file selected.

One file only.
512 MB limit.
Allowed types: par.

Step 4:

Mesh file (.msh, .xdmf, .puml.h5)

Browse... No files selected.

Unlimited number of files can be uploaded to this field.
512 MB limit.
Allowed types: msh, xdmf, puml.h5.

Step 5:

Other files (csv, dat, txt, $\boxed{\text{yaml, nc)}}$

Upload all other files that are required by the parameter file.

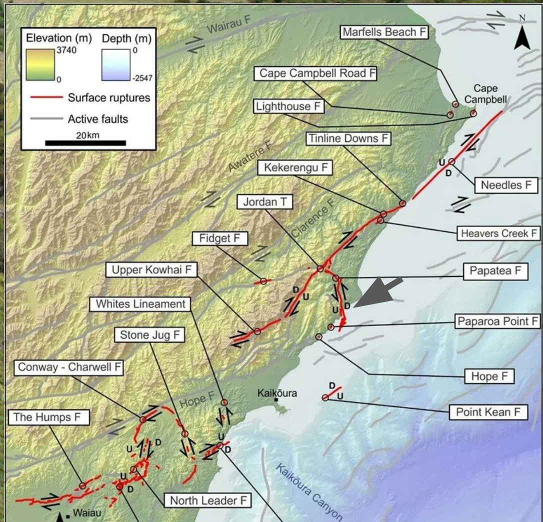
Browse... No files selected.

Unlimited number of files can be uploaded to this field.
512 MB limit.
Allowed types: csv, dat, txt, yaml, nc.

Step 6: Adjust Max runtime to 30 (minutes), but leave all other parameters as is, then submit

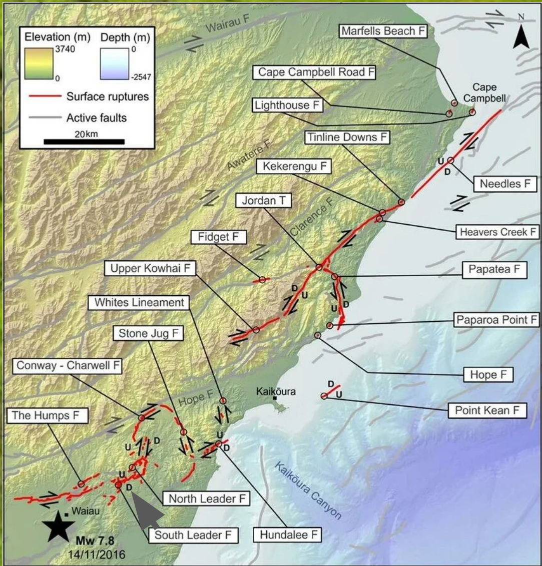
2016 Mw 7.8 Kaikoura earthquake

- one of the most complex earthquake ruptures to-date



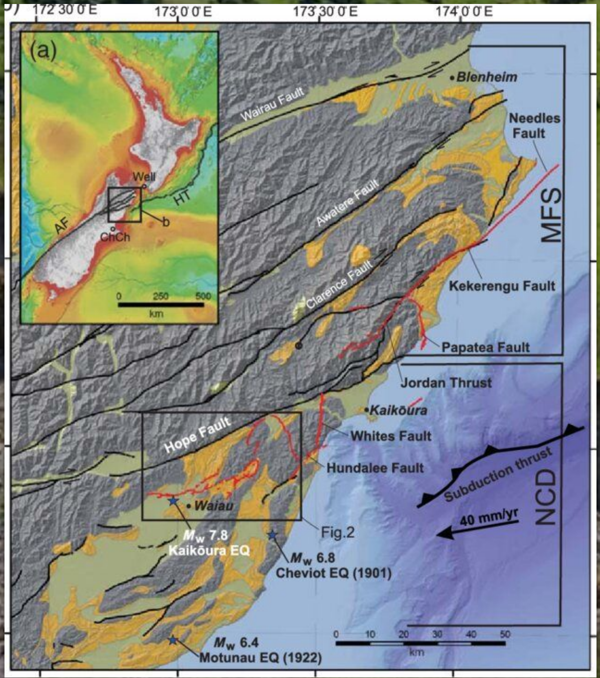
2016 Mw 7.8 Kaikoura earthquake

- one of the most complex earthquake ruptures to-date



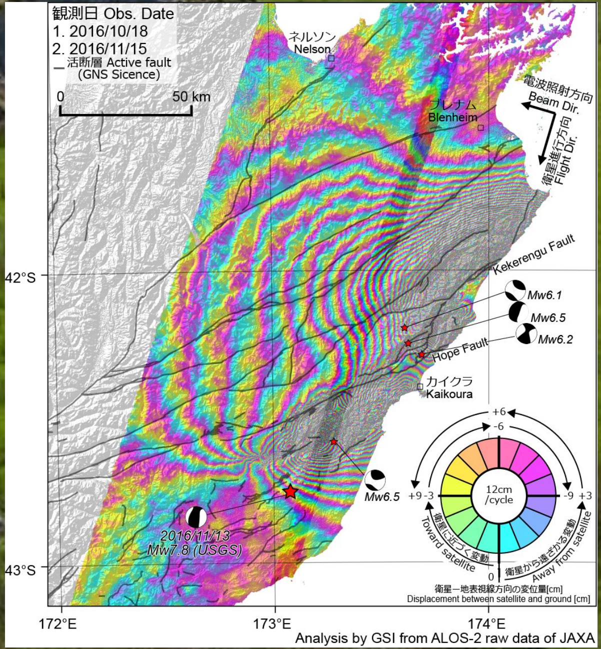
2016 Mw 7.8 Kaikoura earthquake
- one of the most complex earthquake ruptures to-date

Fault networks in the region



A. Nicol et al. (2018)

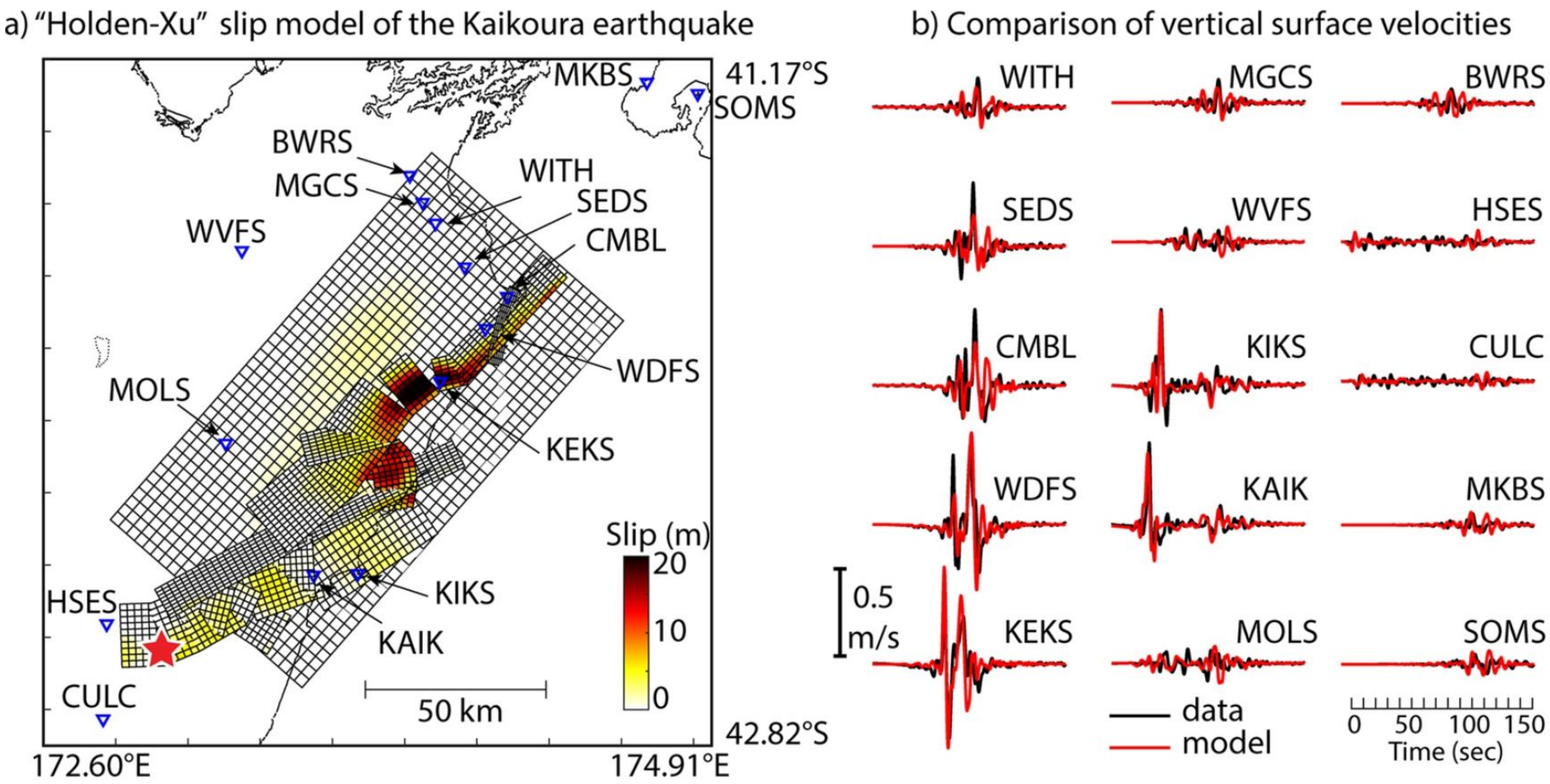
InSAR images



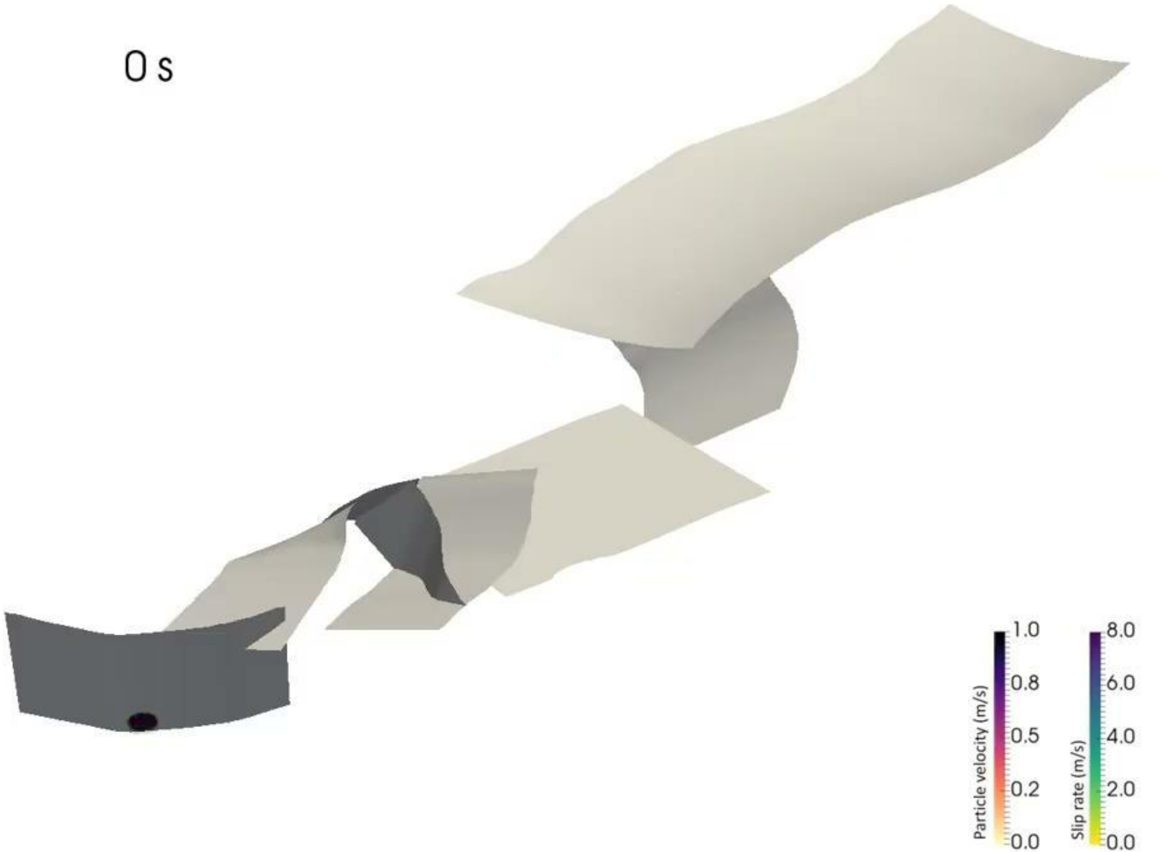
Geospatial Information Authority (Japan) [The 2016 New Zealand Earthquake: Crustal deformation detected by ALOS-2 data]

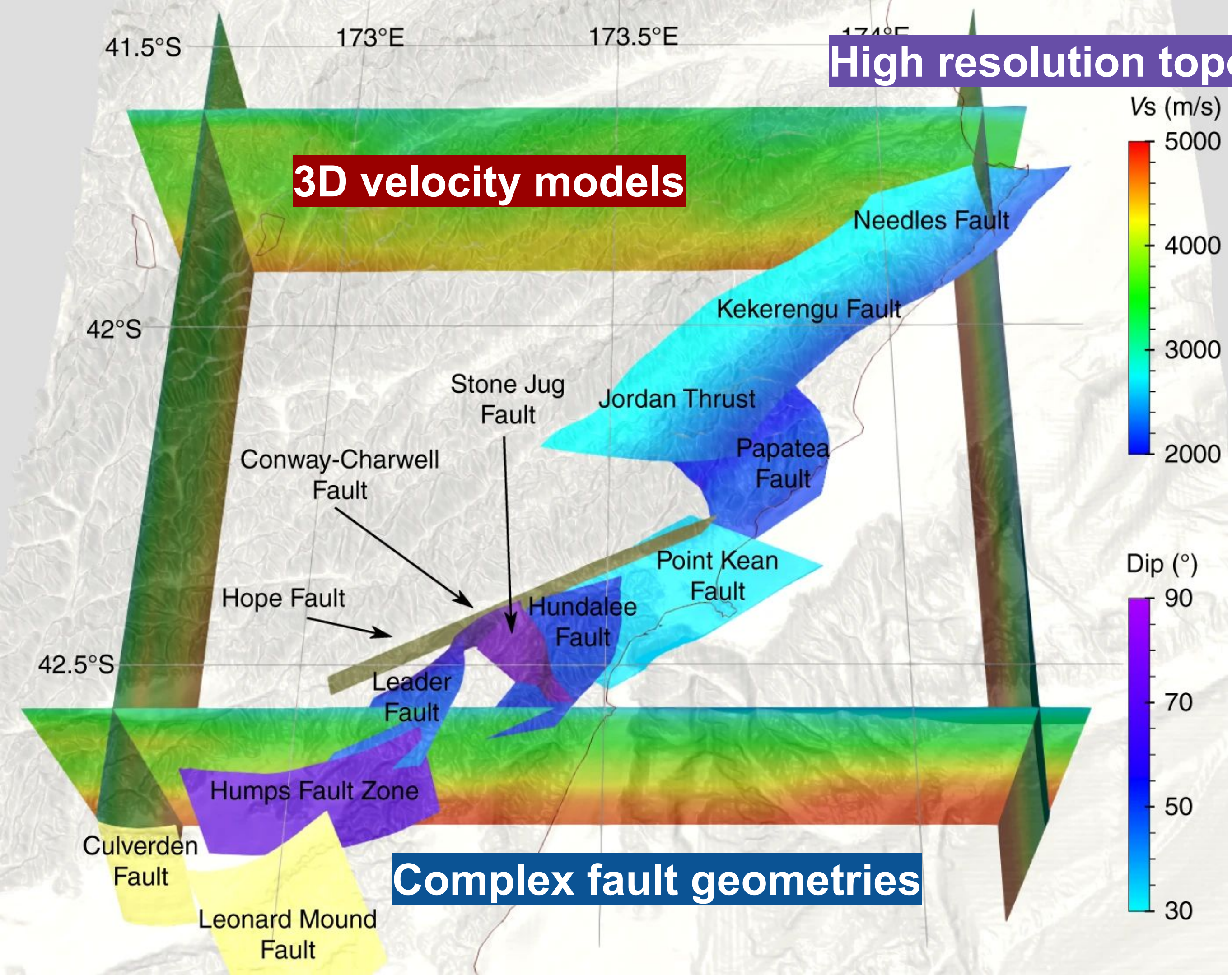
Finite-fault models suggest complex rupture

- Are these rupture dynamic feasible?



SeisSol 3D dynamic rupture training example: the 2016 Mw 7.8 Kaikōura, New Zealand earthquake Ulrich et al. (2019)





150 m element edge length everywhere on the faults

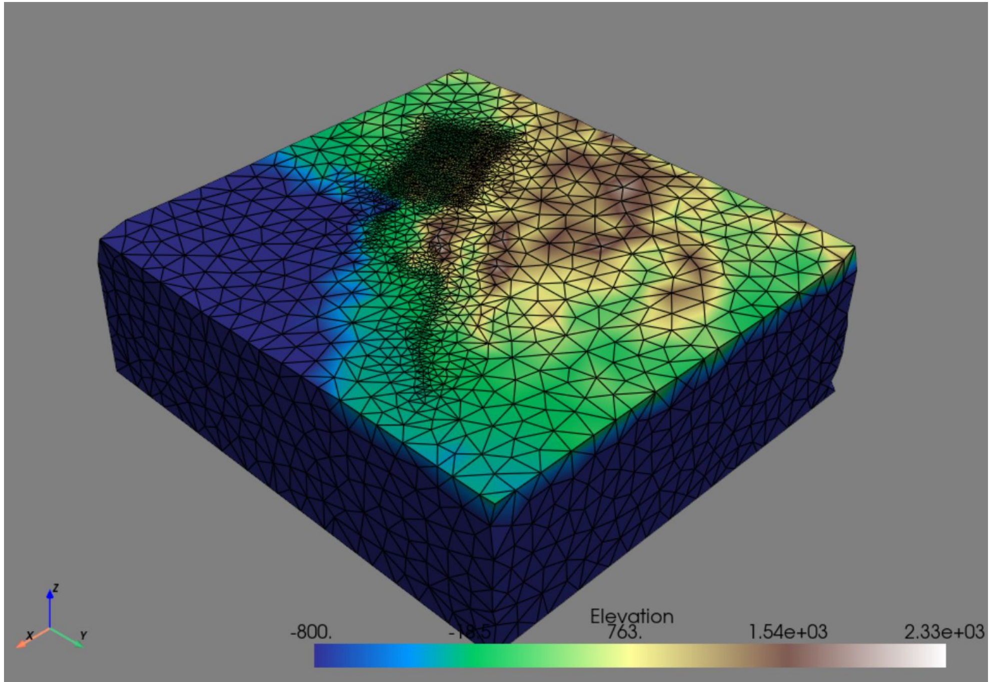
29 million elements in total

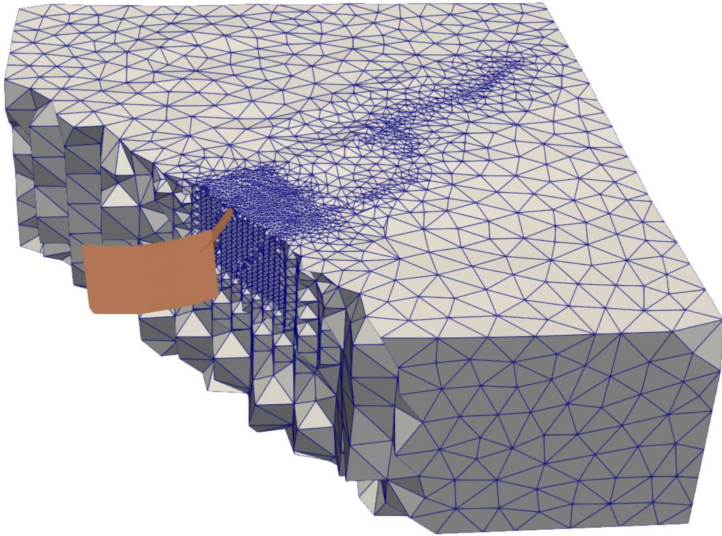
Resolving up to 3 Hz near the fault

Ulrich et al., 2019

Mesh visualization

This training uses a pre-prepared, 280,000-element mesh. It features a 1,500 m resolution along the faults, a box-shaped refinement zone for southern rupture waves, and static coarsening further away.

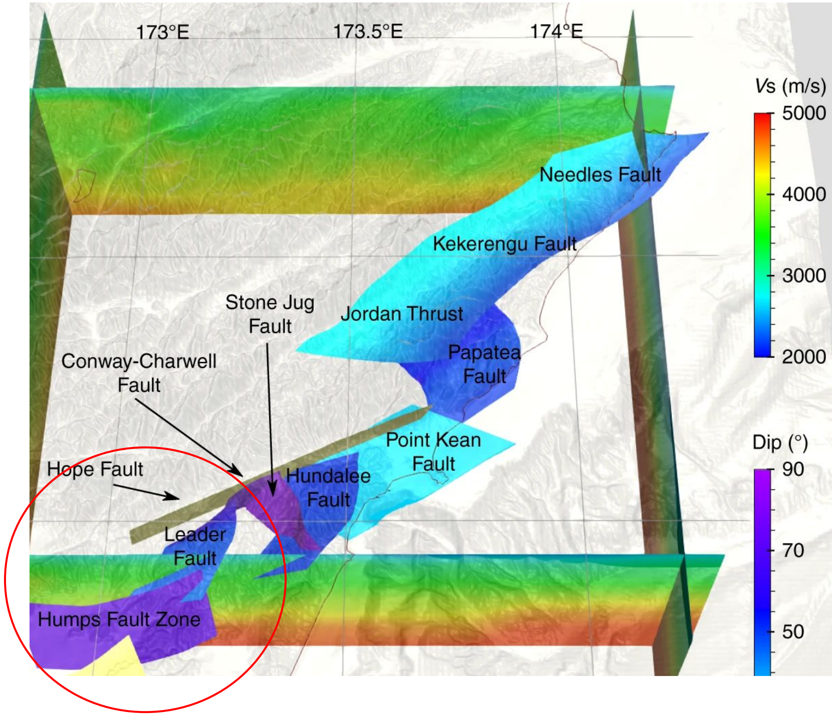




Task 1 - Understanding the dynamics of the first branching using Paraview

We now analyse the dynamic fault traction on the Leader fault from the rupture of the Humps fault zone.

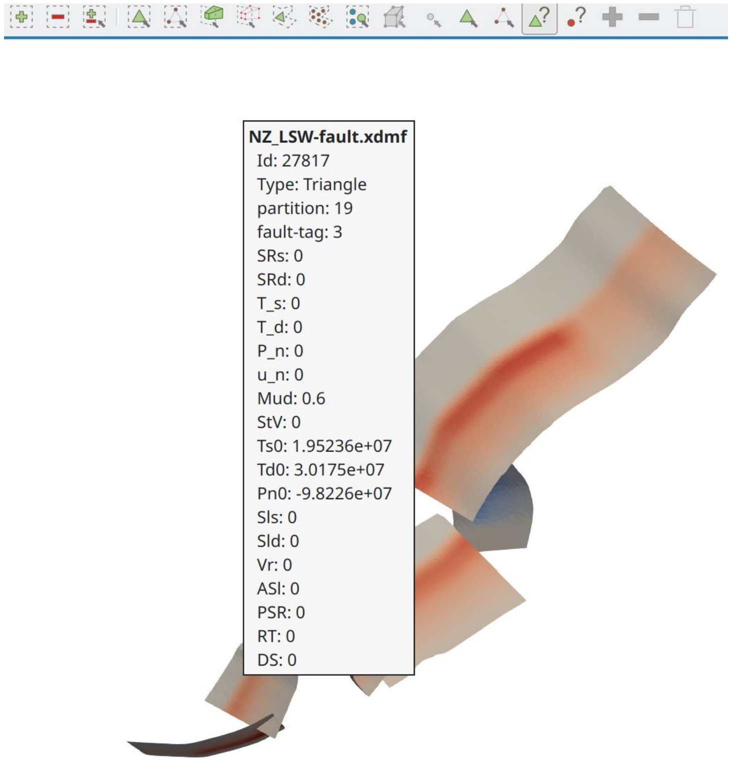
- **Task 1.1**: Download the output files from your Quakeworx SeisSol simulation and open them in Paraview



Task 1 - Understanding the dynamics of the first branching using Paraview

We now analyse the dynamic fault traction on the Leader fault from the rupture of the Humps fault zone.

- **Task 1.1**: Download the output files from your Quakeworx SeisSol simulation and open them in Paraview
- **Task 1.2**: Inspect the initial shear tractions along strike (Ts0) and along dip (Td0). Which component is dominant on the Leader and Humps fault zone?

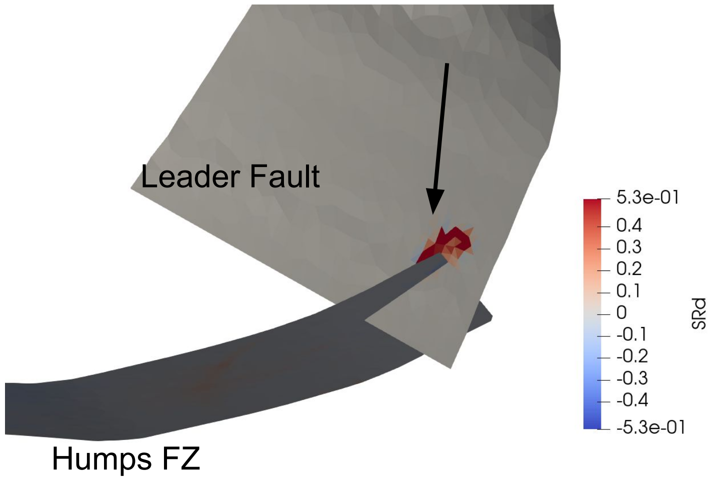


the dip component

Task 1 - Understanding the dynamics of the first branching using Paraview

We now analyse the dynamic fault traction on the Leader fault from the rupture of the Humps fault zone.

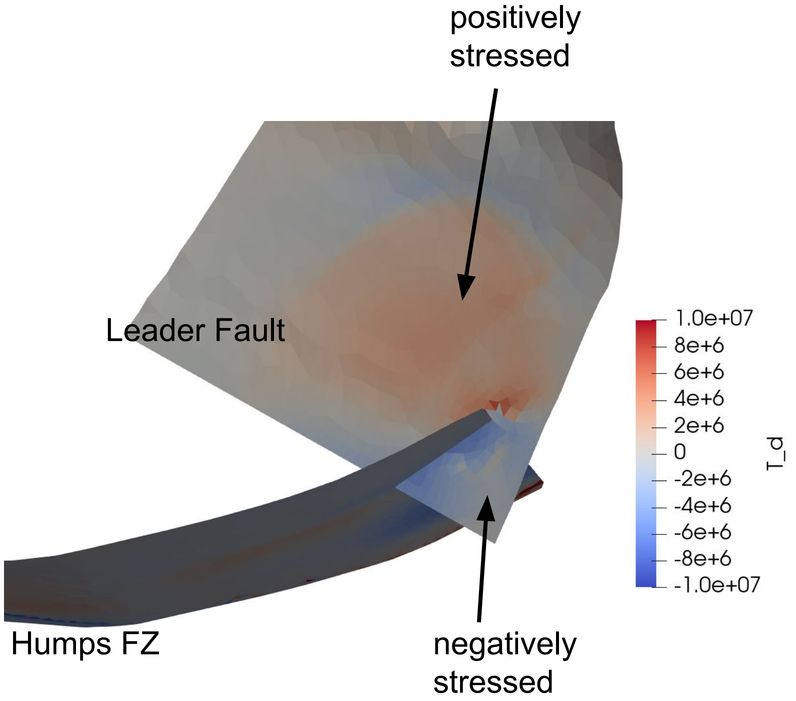
- **Task 1.1**: Download the output files from your Quakeworx SeisSol simulation and open them in Paraview
- **Task 1.2**: Inspect the initial shear tractions along strike (Ts0) and along dip (Td0). Which component is dominant on the Leader and Humps fault zone?
- **Task 1.3**: Inspect the slip-rate along dip (SRd) on the Leader fault: where does the Leader fault get triggered?



Task 1 - Understanding the dynamics of the first branching using Paraview

We now analyse the dynamic fault traction on the Leader fault from the rupture of the Humps fault zone.

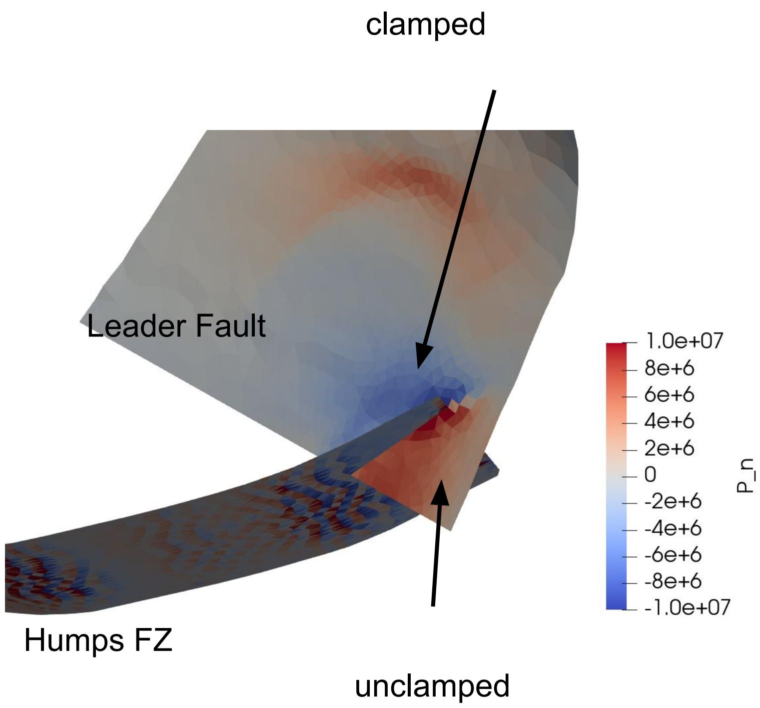
- **Task 1.1**: Download the output files from your Quakeworx SeisSol simulation and open them in Paraview
- **Task 1.2**: Inspect the initial shear tractions along strike (Ts0) and along dip (Td0). Which component is dominant on the Leader and Humps fault zone?
- **Task 1.3**: Inspect the slip-rate along dip (SRd) on the Leader fault: where does the Leader fault get triggered?
- **Task 1.4**: Inspect the transient shear tractions along strike (T_s) and along dip (T_d) right before rupture branching to the Leader fault.
  - Which traction orientation (strike or dip) is the most important for triggering the Leader fault?
  - Which part of the fault is positively stressed by the rupture on the Humps fault?



Task 1 - Understanding the dynamics of the first branching using Paraview

We now analyse the dynamic fault traction on the Leader fault from the rupture of the Humps fault zone.

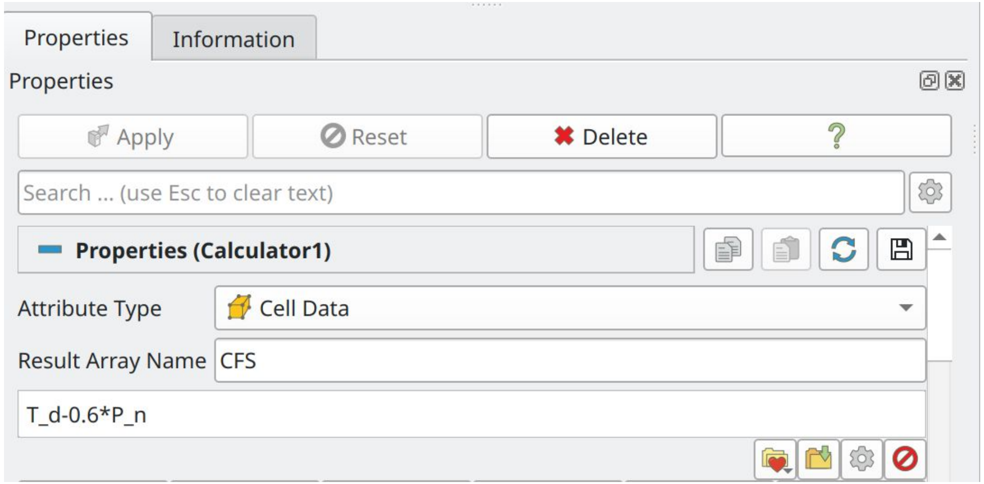
- **Task 1.1**: Download the output files from your Quakeworx SeisSol simulation and open them in Paraview
- **Task 1.2**: Inspect the initial shear tractions along strike (Ts0) and along dip (Td0). Which component is dominant on the Leader and Humps fault zone?
- **Task 1.3**: Inspect the slip-rate along dip (SRd) on the Leader fault: where does the Leader fault get triggered?
- **Task 1.4**: Inspect the transient shear tractions along strike (T_s) and along dip (T_d) right before rupture branching to the Leader fault.
    - Which traction orientation (strike or dip) is the most important for triggering the Leader fault?
    - Which part of the fault is positively stressed by the rupture on the Humps fault?
- **Task 1.5**: Now inspect the transient normal stress (P_n) on the Leader fault. P_n is negative in compression. Which part of the fault is clamped, and which is unclamped?

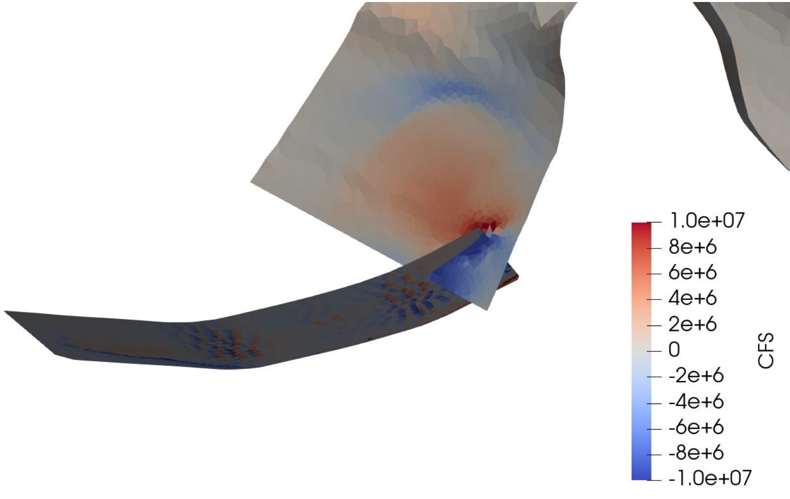


Task 1 - Understanding the dynamics of the first branching using Paraview

We now analyse the dynamic fault traction on the Leader fault from the rupture of the Humps fault zone.

- **Task 1.6**: The combine effect of shear and normal stressing can be evaluated with the dynamic Coulomb stress changes. Use the Calculator filter (Filters>Search>Calculator) to compute the CFS as
  CFS = T_d - 0.6*P_n
  Is it consistent with the Leader Fault getting triggered?





Task 2 - Understanding how initial loading affects cascading rupture dynamics

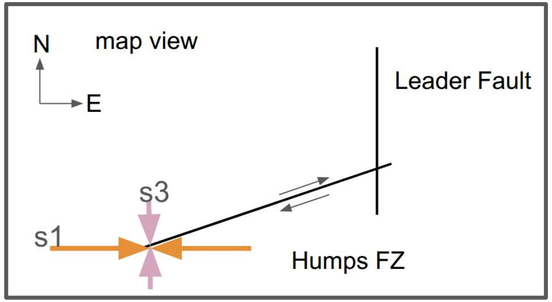
jupyter NZ_initial_stressLSW.yaml✔ a few seconds ago

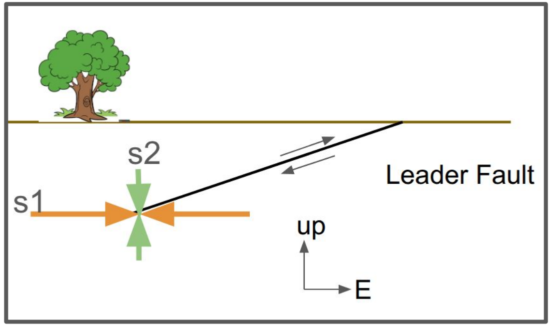
File Edit View Language

```
        [sig_zz]: !Include NZ_sigzz.yaml
      components: !AndersonianStress
        constants:
          mu_d:      0.1
          mu_s:      0.6
          cohesion:  0.0
          s2ratio:   0.15
          S_v: 2
    [sig_zz]: !Include NZ_sigzz.yaml
components: !LuaMap
  returns: [s_xx, s_yy, s_zz, s_xy, s_yz, s_xz]
  function: |
    function f(x)
        return {
            s_xx = x["Omega"] * x["b_xx"] + (1.0 - x["Omega"]) * x["sig_zz"],
            s_yy = x["Omega"] * x["b_yy"] + (1.0 - x["Omega"]) * x["sig_zz"],
            s_zz = x["Omega"] * x["b_zz"] + (1.0 - x["Omega"]) * x["sig_zz"],
            s_xy = x["Omega"] * x["b_xy"],
            s_yz = x["Omega"] * x["b_yz"],
            s_xz = x["Omega"] * x["b_xz"]
        }
    end
```

The stress shape ratio is defined as:

$$\nu = (s_2 - s_3)/(s_1 - s_3),$$

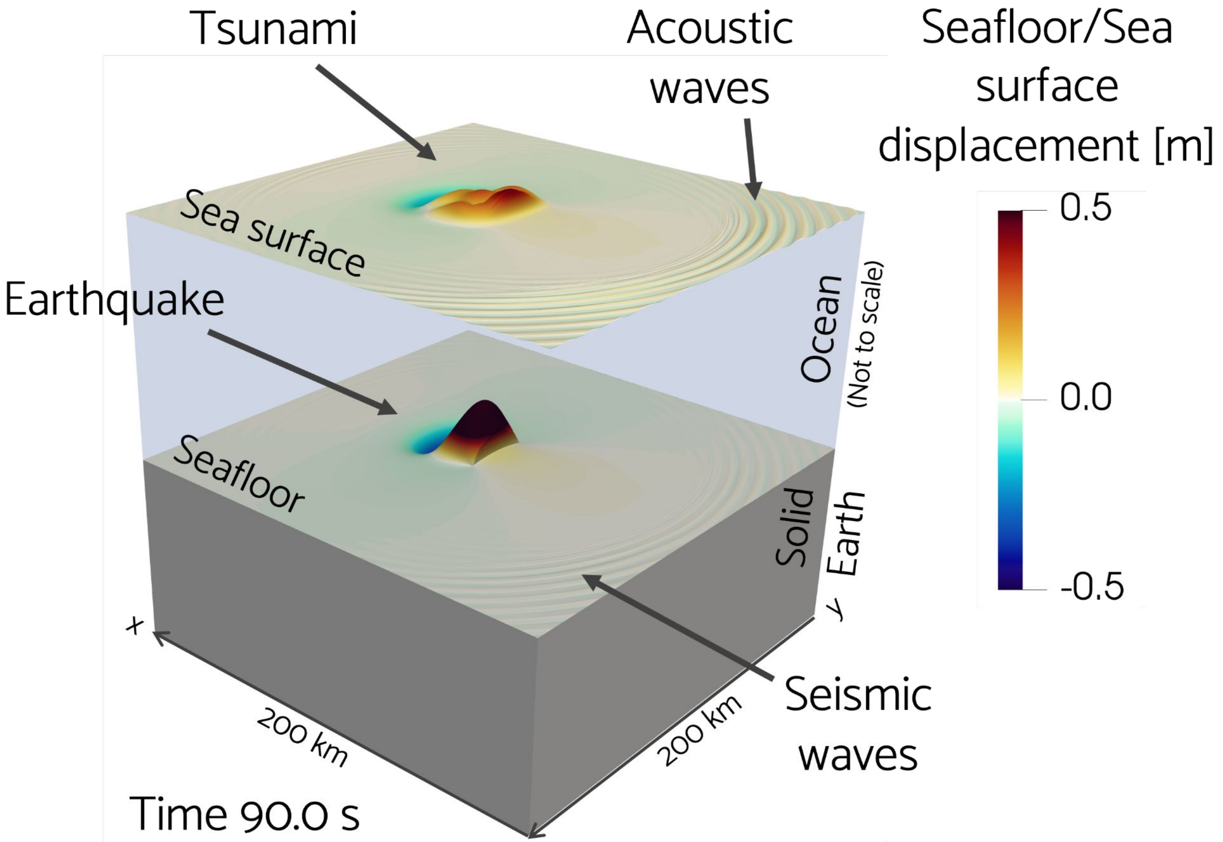




Effect of initial loading on cascading rupture dynamics

Task: Change the stress shape ratio (s2ratio) from 0.15 to 0.20 and run the updated setup to observe how it affects the dynamics. Rename the output folder to compare 3D earthquake dynamics of both setups by opening both fault output files in ParaView.

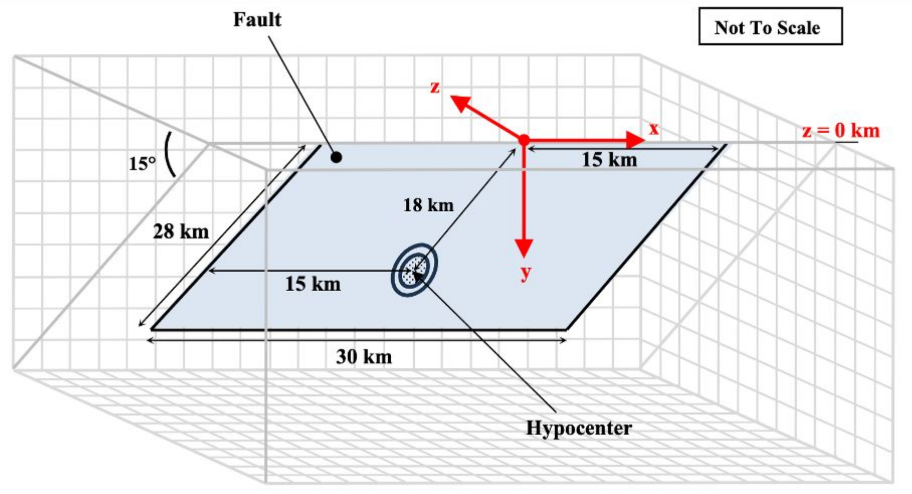
3D fully coupled earthquake dynamic rupture and tsunami benchmarks



Extending an existing benchmark to tsunami generation

The Problem Version 36
(TPV36; Harris et al., 2018)

Problem description for TPV36 and TPV37 - Shallow Dip Thrust Fault.



- Spontaneous 3D earthquake **dynamic rupture** on a 15° dipping thrust fault reaching the Earth's surface
- Rectangular planar fault: 30 km along-strike and 28 km down-dip

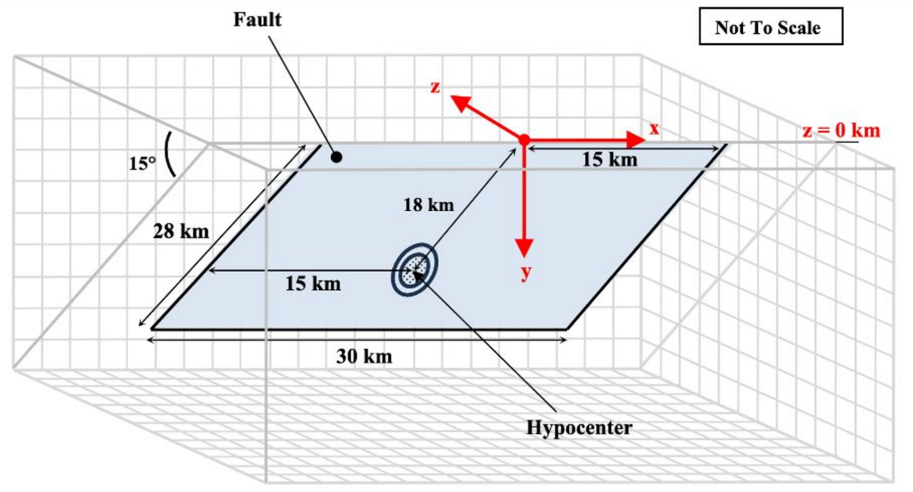
Fabian

Extending an existing benchmark to tsunami generation

The Problem Version 36

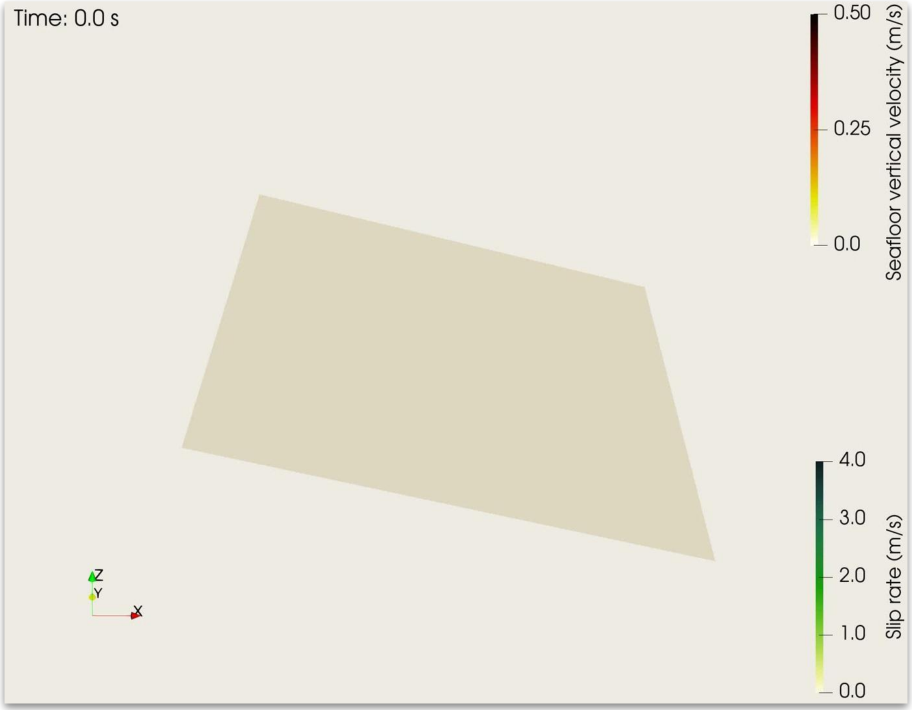
(TPV36; Harris et al., 2018)

[Problem description for TPV36 and TPV37 - Shallow Dip Thrust Fault].



- Spontaneous 3D earthquake **dynamic rupture** on a 15° dipping thrust fault reaching the Earth's surface
- Rectangular planar fault: 30 km along-strike and 28 km down-dip

Movie: First 30 s of spontaneous dynamic rupture for SeisSol with the seafloor vertical velocity

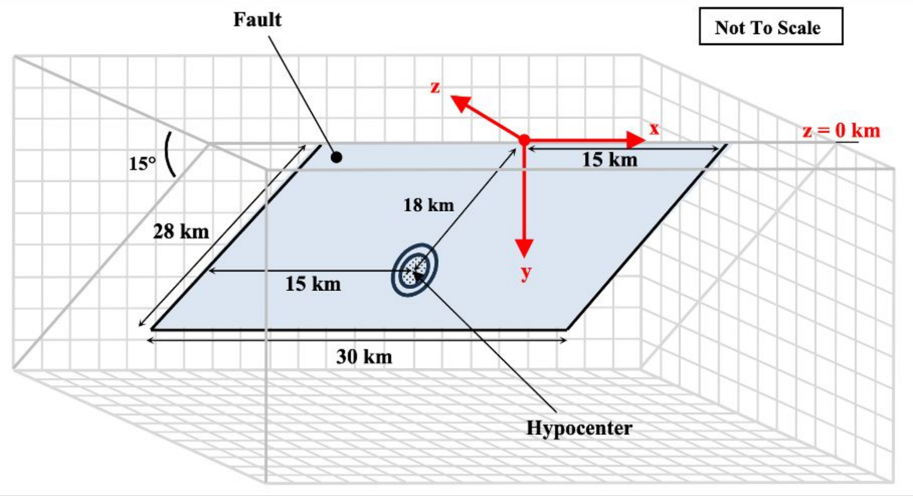


- Linear slip-weakening friction law with (on-fault) cohesion & depth-dependent initial stresses

Extending an existing benchmark to tsunami generation

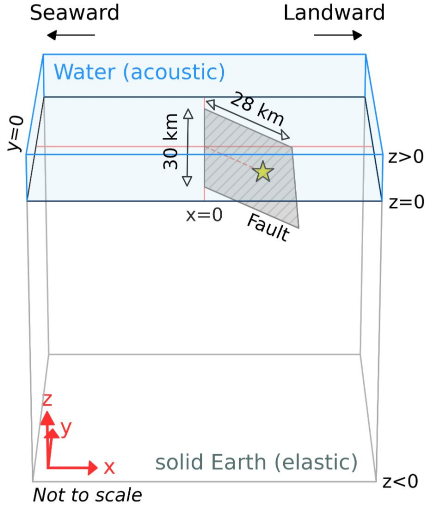
The Problem Version 36
(TPV36; Harris et al., 2018)

Problem description for TPV36 and TPV37 - Shallow Dip Thrust Fault.



The Tsunami Problem Version 1
(TTPV1)

Problem description for TTPV1 and TTPV2 - 3D Earthquake-Tsunami



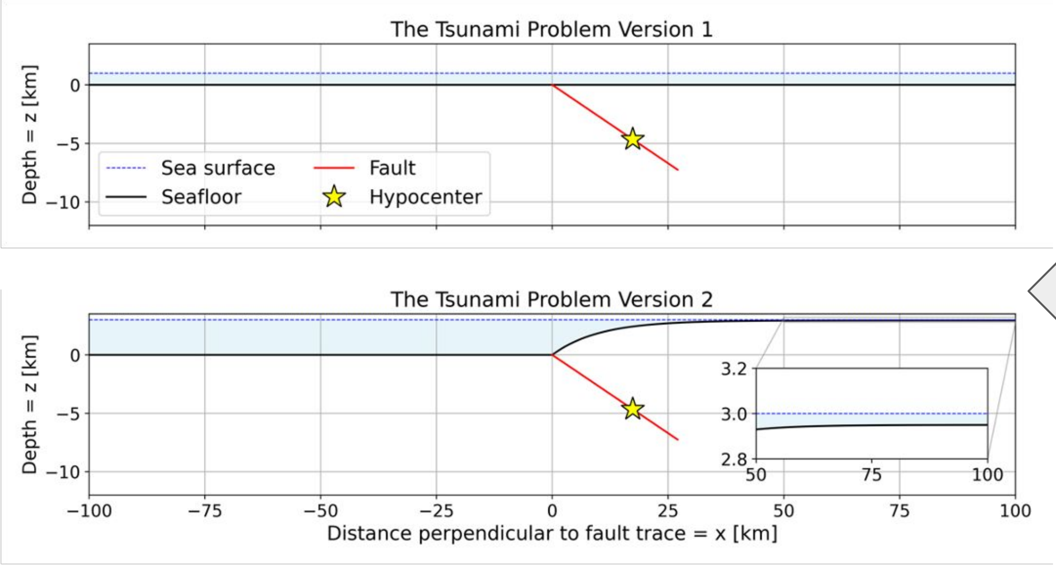
- Spontaneous 3D earthquake **dynamic rupture** on a 15° dipping thrust fault reaching the Earth's surface
- Rectangular planar fault: 30 km along-strike and 28 km down-dip

- The **fully coupled** method combines earthquake dynamic rupture and tsunami generation into a **single simulation**
- We can capture 3D elastic, acoustic, and tsunami waves, including dispersion effects, **simultaneously**

(Maeda & Furumura, 2013; Lotto & Dunham, 2015)

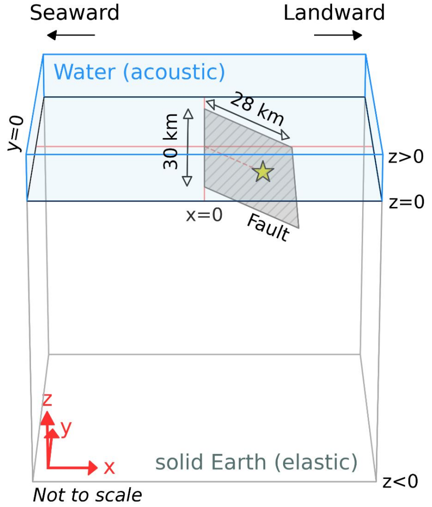
Extending an existing benchmark to tsunami generation

Side view:



The Tsunami Problem Version 1 (TTPV1)

[Problem description for TTPV1 and TTPV2 - 3D Earthquake-Tsunami]

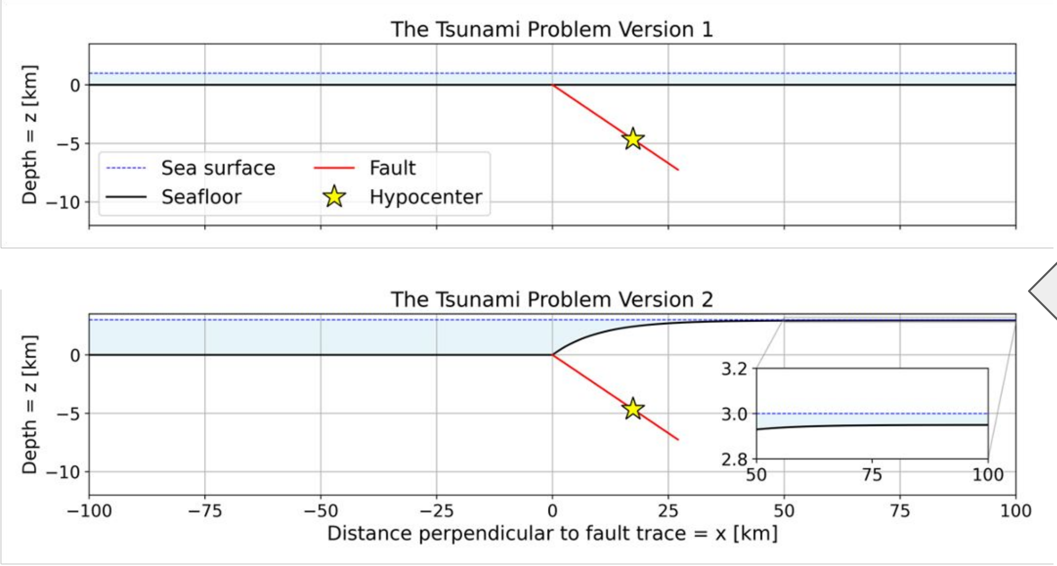


- In TTPV1, a **1 km thick uniform** water layer is added atop the elastic medium
- A water layer with **variable depth** is atop the elastic Earth, with 3 km water depth on the seaward side, sloping to shallow depth (~50 m) landwards

- The **fully coupled** method combines earthquake dynamic rupture and tsunami generation into a **single simulation**
- We can capture 3D elastic, acoustic, and tsunami waves, including dispersion effects, **simultaneously**
(Maeda & Furumura, 2013; Lotto & Dunham, 2015)

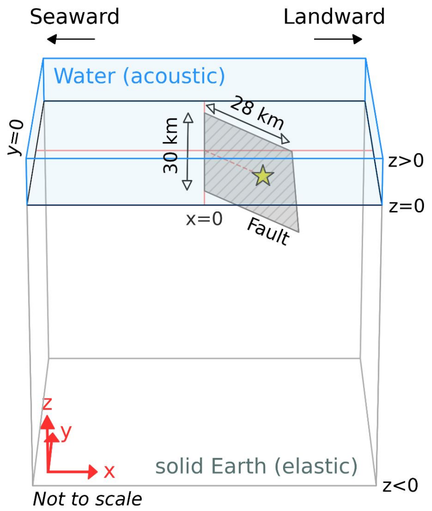
Extending an existing benchmark to tsunami generation

Side view:



The Tsunami Problem Version 1 (TTPV1)

[Problem description for TTPV1 and TTPV2 - 3D Earthquake-Tsunami]



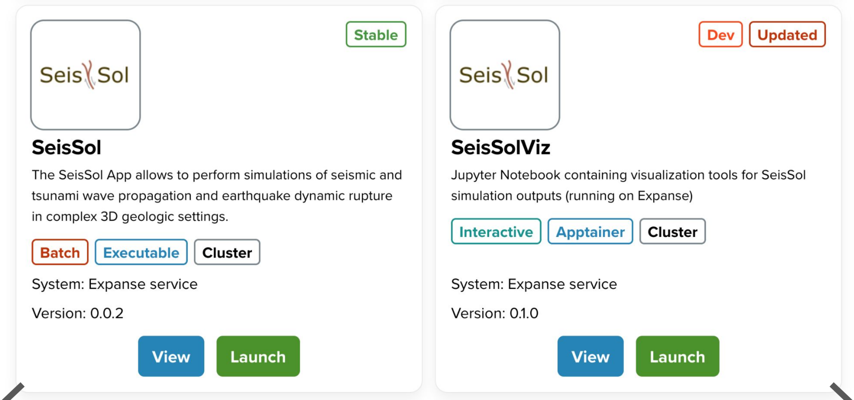
For Training:

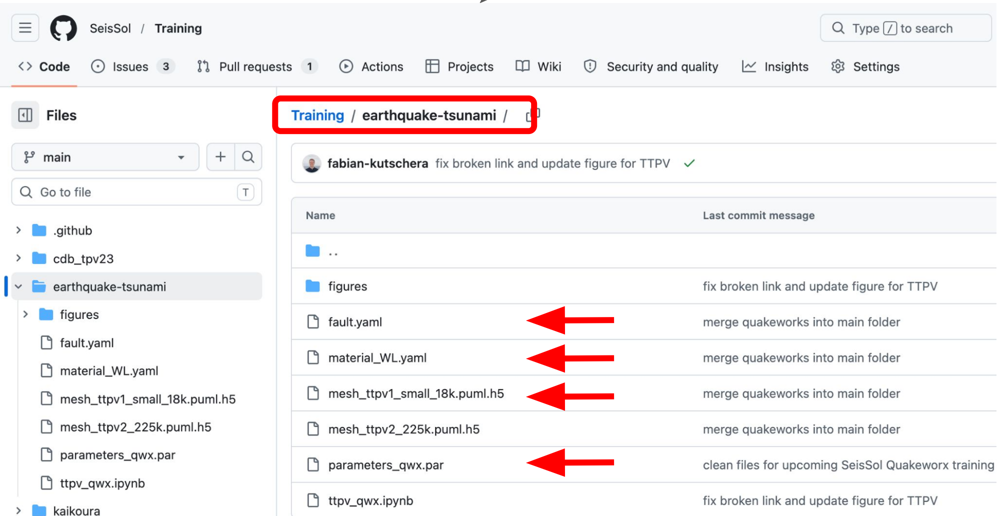
- Very small **105k element mesh for TTPV1** with limited horizontal extent
- **225k element mesh for TTPV2**

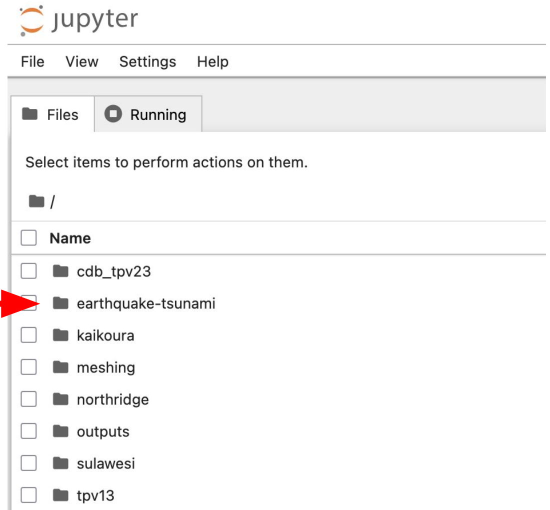
- The **fully coupled** method combines earthquake dynamic rupture and tsunami generation into a **single simulation**
- We can capture 3D elastic, acoustic, and tsunami waves, including dispersion effects, **simultaneously** (Maeda & Furumura, 2013; Lotto & Dunham, 2015)

Start by running TTPV1 or TTPV2

QUAKEWORX







Hands-on: Jupyter Notebook

7000+ commits to SeisSol

Wednesday, 29 July, 2015 06:26:28

Courtesy of David Schneller

Backup