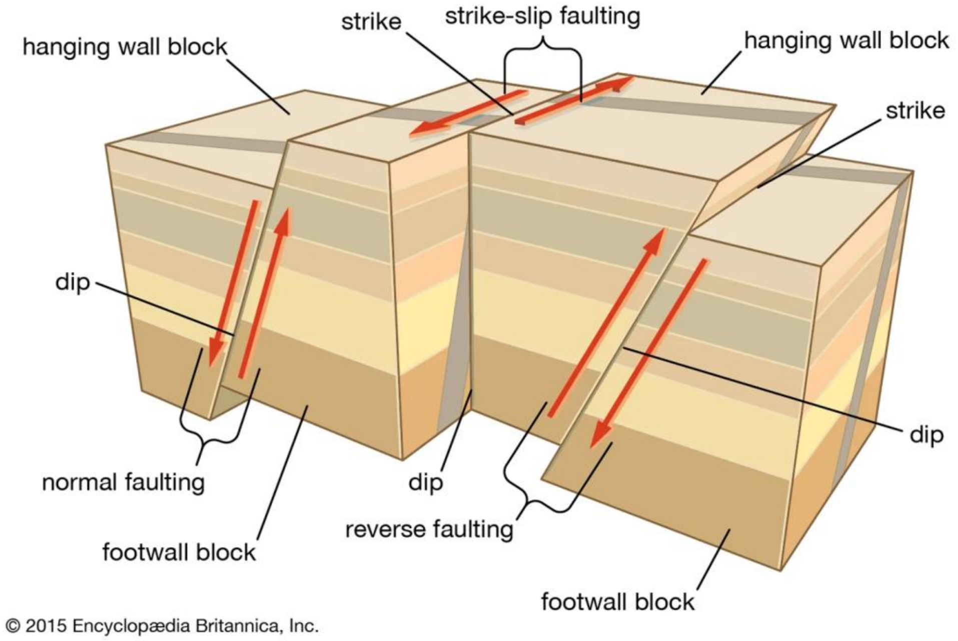
CPU/GPU Implementation of the Okada deformation model due to a finite rectangular earthquake source

Erwin Meza Vega

2025

Context

- Faults are fractures in the Earth's crust where rocks have moved relative to each other,often occurring at plate boundaries. The rapid movement of these faults causes earthquakes, and in the case of undersea faults, Tsunami waves can be formed. (Wikipedia)



Okada deformation model

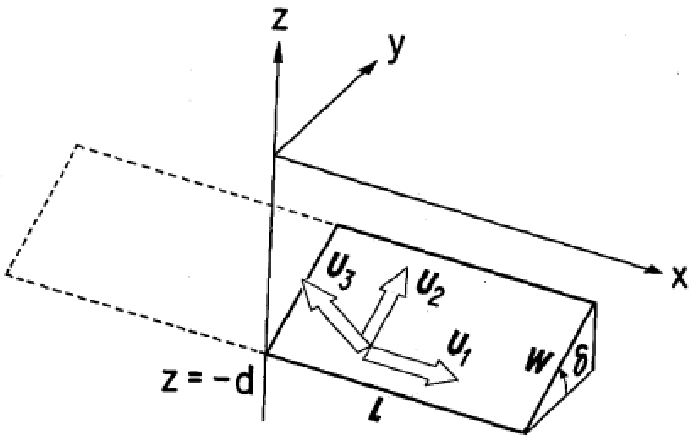


Fig. 1. Source: Okada (1985)

Okada proposes a *suite of analythical expressions for surface displacements, strains ant tilts due to inclined shear and tensile faults in a halfspace for both point and finite rectangular sources*.

Okada (1985, 1992)

Okada deformation model

GeoScienceWorld Journals Books GeoRef Publishers OpenGeoSci Citation Manager Help

Bulletin of the Seismological Society of America

Search... Bulletin of the :

Advanced Search

SSA

Archive Content About The Journal About the Society SSA Member Sign In

Volume 75, Number 4

August 1985

No cover image available

Previous Article Next Article

Article Contents

RESEARCH ARTICLE | AUGUST 01, 1985

Surface deformation due to shear and tensile faults in a half-space

Yoshimitsu Okada

+ Author and Article Information

Bulletin of the Seismological Society of America (1985) 75 (4): 1135–1154.

https://doi.org/10.1785/BSSA0750041135 | Article history

Cite Share Permissions

ABSTRACT

A complete suite of closed analytical expressions is presented for the surface displacements, strains, and tilts due to inclined shear and tensile faults in a half-space for both point and finite rectangular sources. These expressions are particularly compact and free from field singular points which are inherent in the previously stated expressions of certain cases. The expressions derived here represent powerful tools not only for the analysis of static field changes associated with earthquake occurrence but also for the modeling of deformation fields arising from fluid-driven crack sources.

Email alerts

Article activity alert

Early publications alert

New issue alert

Index Terms/Descriptors

deformation dip-slip faults displacements earthquakes effects faults half-space mathematical models practice seismology shear strain strike-slip faults structural geology tensile strength theoretical studies tilt

Project goals

- To develop a CPU and GPU implementation of the deformation model proposed by Okada (1985)
- To compare execution times between the CPU and the GPU implementation

Motivation

- Develop a prototype implementation as the Capstone Project for the GPU Programming Specialization (Coursera – Johns Hopkins University)
- Apply the tools and techniques studied so far to a real world scenario

Challenges

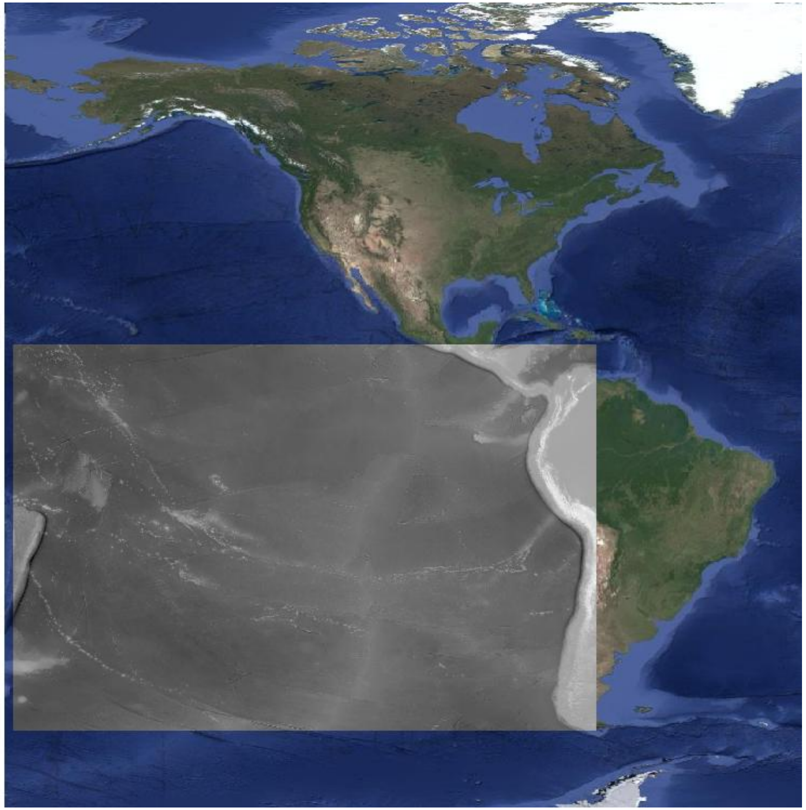
- Individual calculation on each point of the grid. Several equations to be evaluated, potentialli different conditions on each point of the grid.
- Multiple faults: Each fault affects all the points, if several faults occur, their effects are added.

Challenges

- Calculation over huge grids, millions of points to be evaluated

Information from provider

Extent	-178.4166666666666572,-55.8125000000000142 : -68.5541666666666885,16.9624999999999950
Width	26367
Height	17466
Data type	Int16 - Sixteen bit signed integer
GDAL Driver Description	netCDF
GDAL Driver Metadata	Network Common Data Format
Dataset Description	C:/Users/er_mv/Desktop/TunamiF1Cuda/test/Lineal/Grids/gebco_2024_n16.9629_s-55.8105_w-178.418_e-68.5547.nc
Compression	
Band 1	• grid_mapping=crs • long_name=Elevation relative to sea level • NETCDF_VARNAME=elevation • sdn_parameter_name=Sea floor height (above mean sea level) {bathymetric height} • sdn_parameter_urn=SDN:P01::BATHHGHT • sdn_uom_name=Metres • sdn_uom_urn=SDN:P06::ULAA • standard_name=height_above_mean_sea_level



Process

- Analysis of the analythical model
  - Literature review for configuration scenarios
- Implementation
  - Definition of inputs/outputs
  - Coding, testing, refactoring, documenting
  - Visualizing results
- Publishing repo, writing documentation

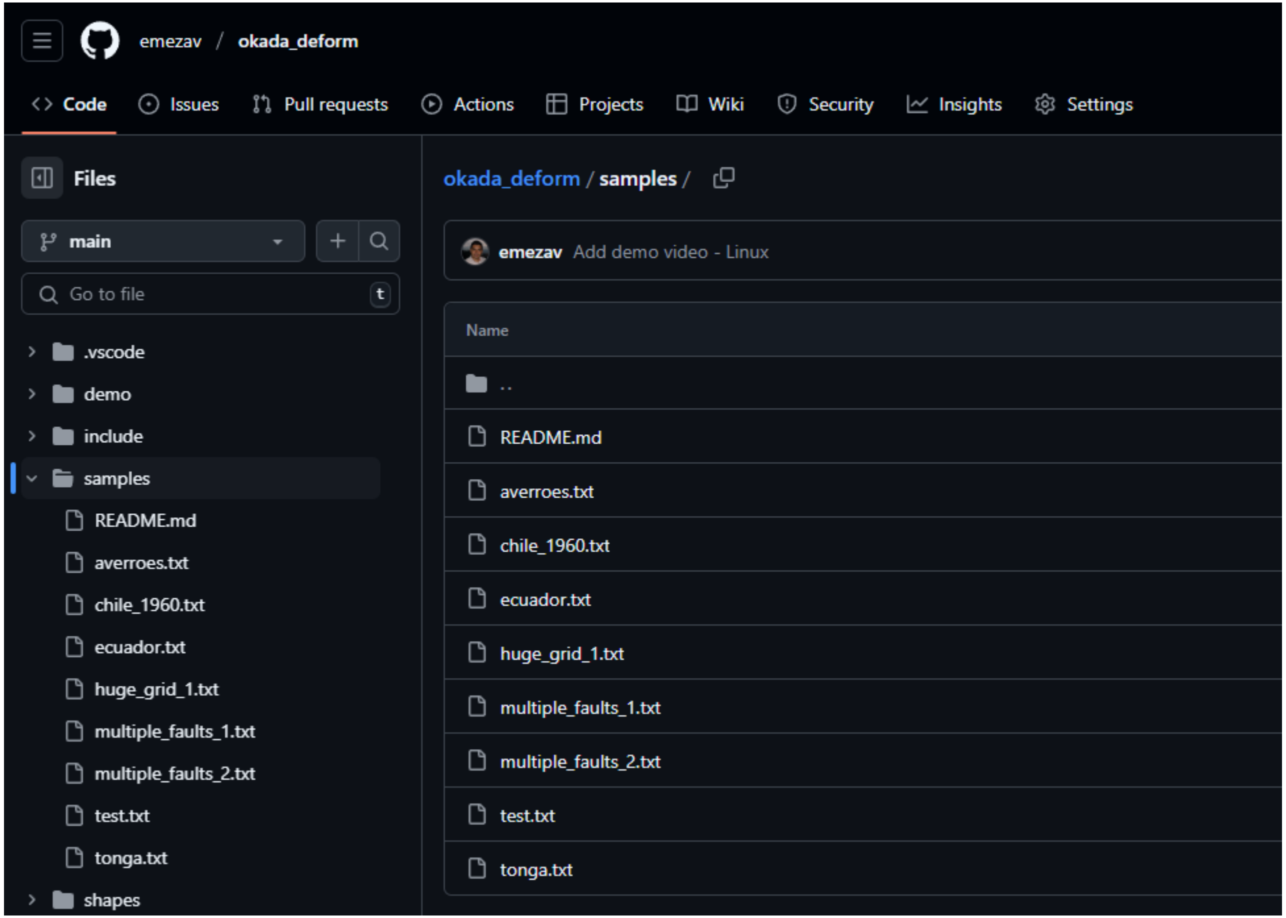
Tools and techniques

- Tools and techniques from the GPU specialization
  - Memory management: host/device memory, constant memory, pointers, copy to host, copy to device
  - CUDA kernel design: blocks per grid / threads per block, blockDim, blockIdx, treadIdx, flattened 2D arrays
  - CUDA Toolkit, development environment, Makefiles
  - Advanced libraries
  - Vs Code Action buttons setup
- Other tools used
  - Cmake, multiple platforms (Windows/Linux), Vs Code actions, GIS software, command line tools

Repo


github.com/emezav/okada_deform/tree/main
README
MIT license
CPU/GPU Implementation of the Okada deformation model due to a finite rectangular earthquake source
According to Wikipedia, faults are fractures in the Earth's crust where rocks have moved relative to each other, often occurring at plate boundaries [1]. The rapid movement of these faults causes earthquakes, and in the case of undersea faults, Tsunami waves can be formed.
In order to study the source and effects of earthquakes (and tsunamis), several mathematical and computational models have been developed, being Okada's 1985 (and 1992 revision) [2] one of the most widely used.
The deformation of the earth surface due to a finite rectangular earthquake source is a complex physical problem that can be expressed as a set of equations to be evaluated on every one of the points of a rectangular grid, representing a portion of the Earth surface.
TL;DR (Summary)
Comparison between CPU version and GPU version of Okada model over a region (grid) given one or more fault events.
Sample input is provided on the samples/ folder.
No input bathymetry is required. However, the resulting grids are correctly located using WGS84 and the obtained results can be used to deform the bathymetry on the same region of the grids.
NOTES:
Default execution stores the resulting grids on disk, see Execution and data input section for details.
Execution time of the CPU time is orders of magnitude larger than the GPU time for huge grids or several fault events.
Resulting grids can use a lot of disk space. Set the parameters with caution.
Based on your tech stack
SLSA Generic generator
Configure
Generate SLSA3 provenance for your existing release workflows
CMake based, multi-platform projects
Configure
Build and test a CMake based project on multiple platforms.
MSBuild based projects
Configure
Build a MSBuild based project.
More workflows
Dismiss suggestions

Samples


emezav / okada_deform
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Files
main
Go to file
t
.vscode
demo
include
samples
README.md
averroes.txt
chile_1960.txt
ecuador.txt
huge_grid_1.txt
multiple_faults_1.txt
multiple_faults_2.txt
test.txt
tonga.txt
shapes
okada_deform / samples /
emezav Add demo video - Linux
Name
..
README.md
averroes.txt
chile_1960.txt
ecuador.txt
huge_grid_1.txt
multiple_faults_1.txt
multiple_faults_2.txt
test.txt
tonga.txt

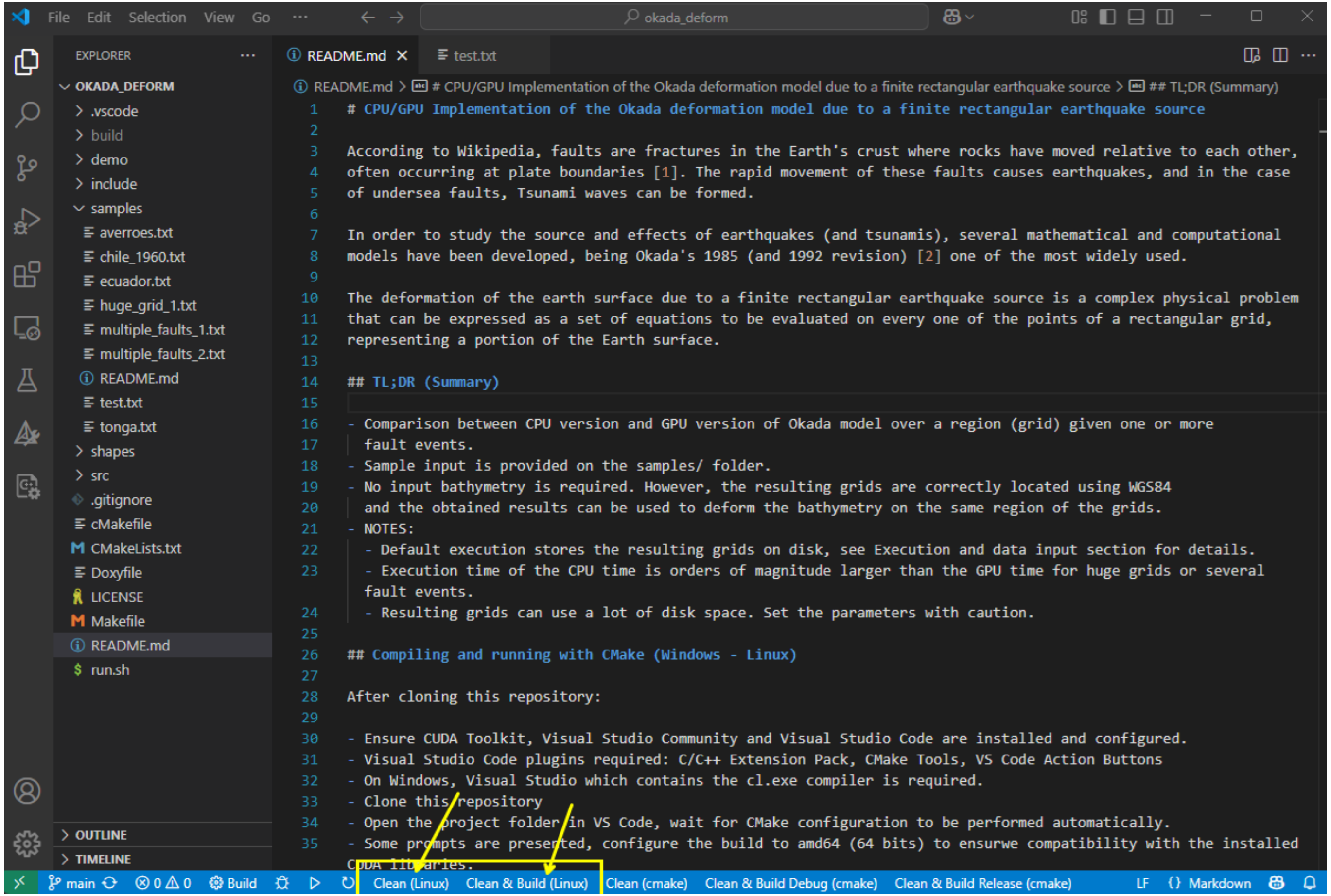
Execution (Linux)

- Usual process from the GPU specialization:
  - make clean build
  - ./run.sh samples/test.txt

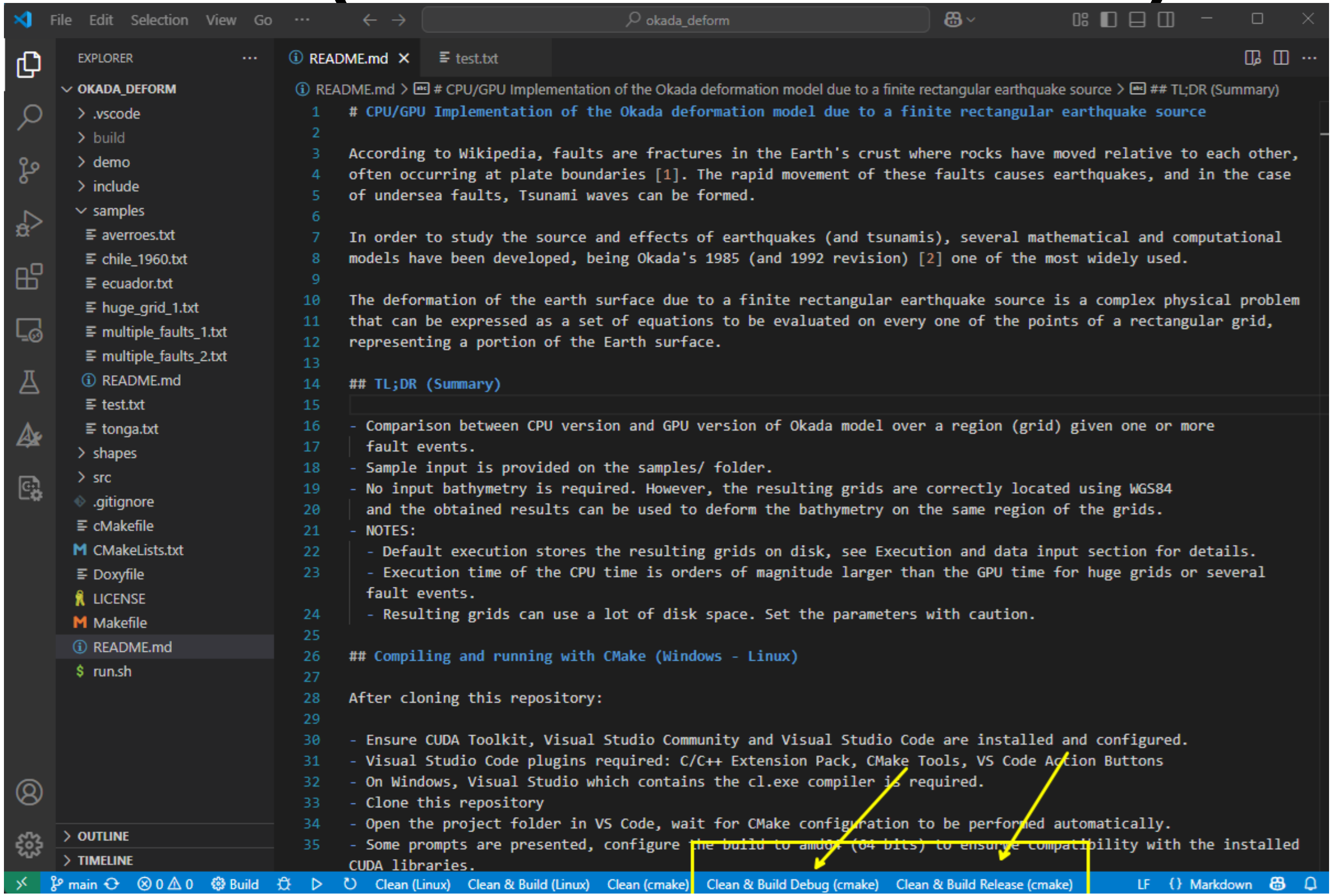
Execution (Windows)

- Cmake:
  - cd build
  - cmake –build . --target Release
  - buiild\Release\okada_deform.exe ..\samples\test.txt
- VS code: Use the action buttons at the bottom toolbar
- Visual Studio solutions is created via cmake.
  - Use the Visual Studio GUI to run/debug.

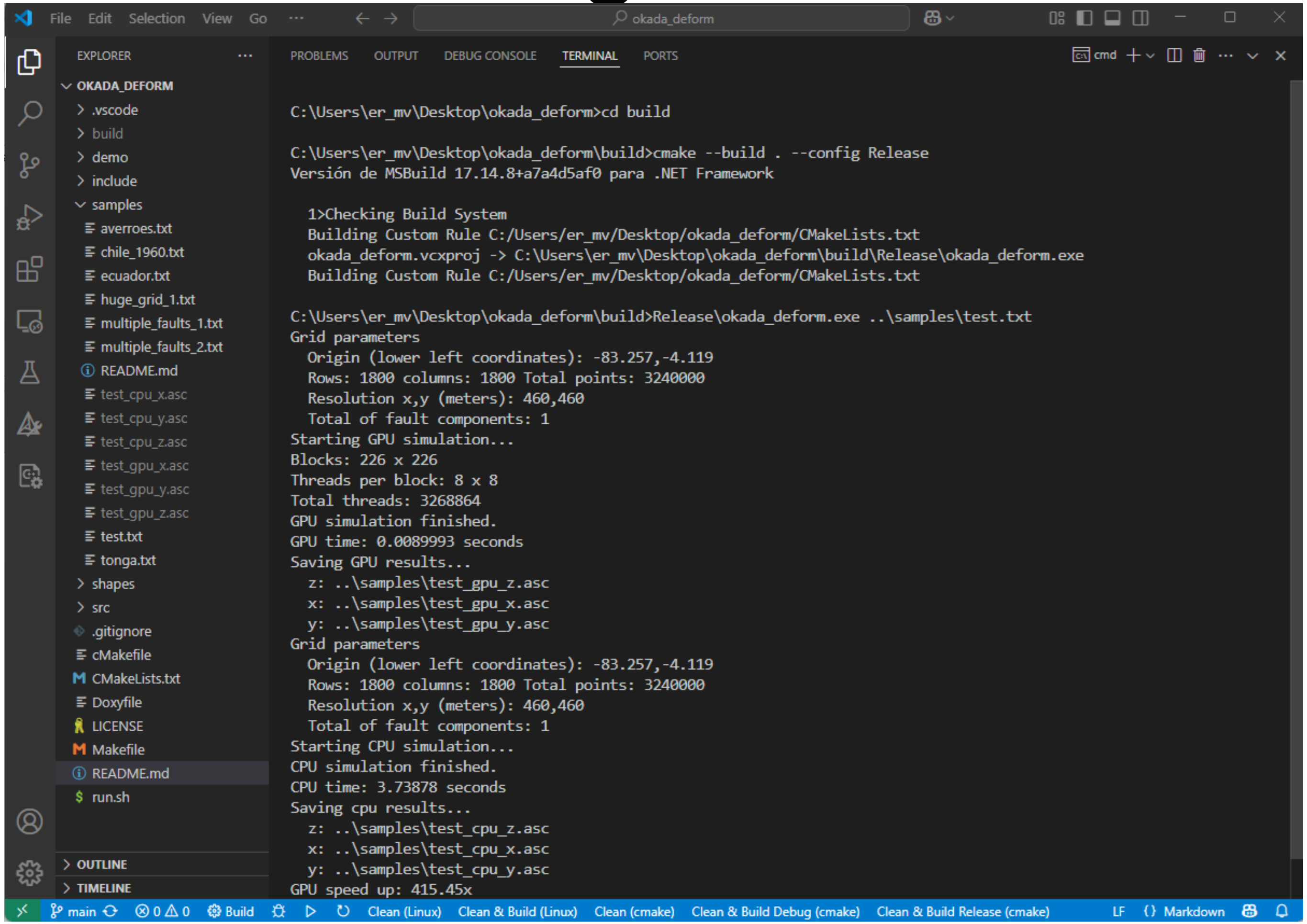
Build with VS Code (Linux)



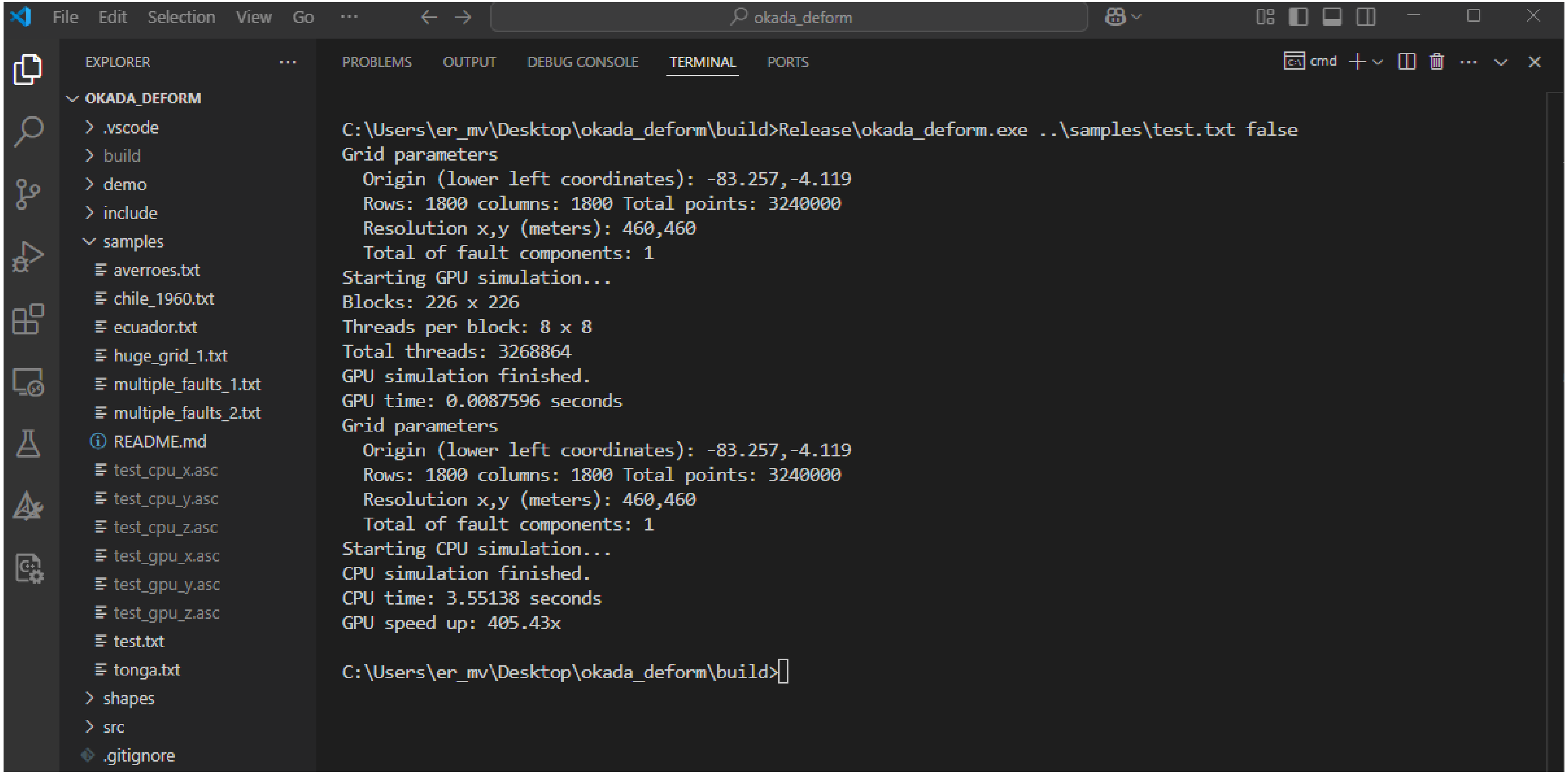
Build with VS Code (Windows/Linux)



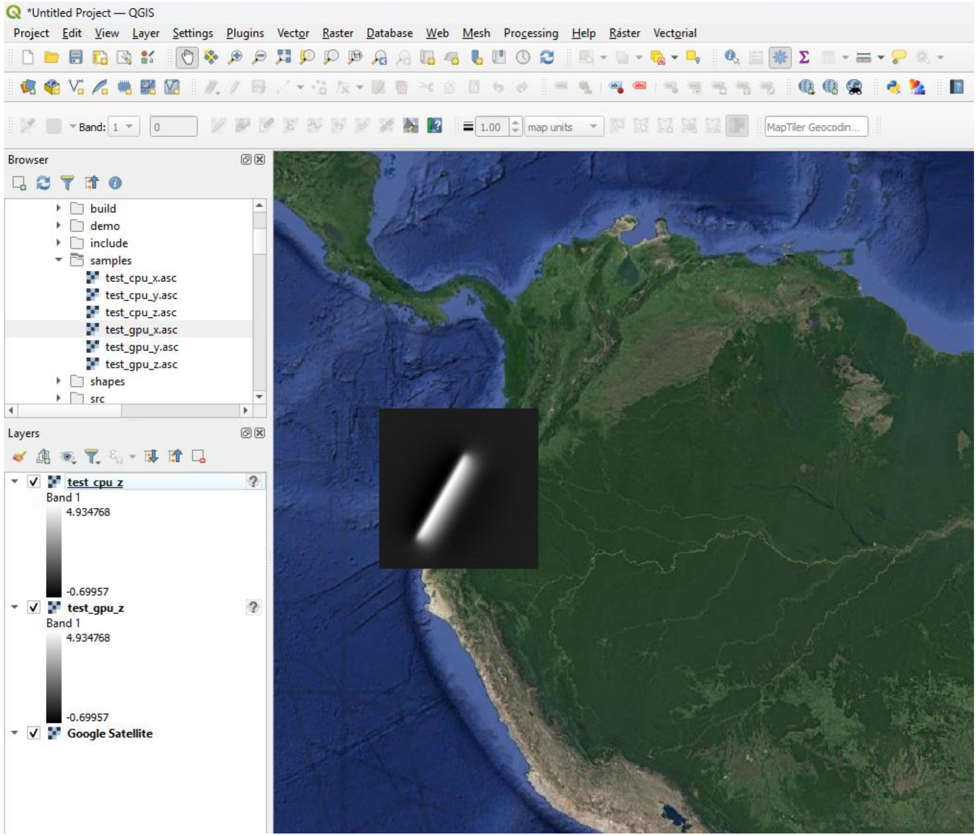
Running scenarios



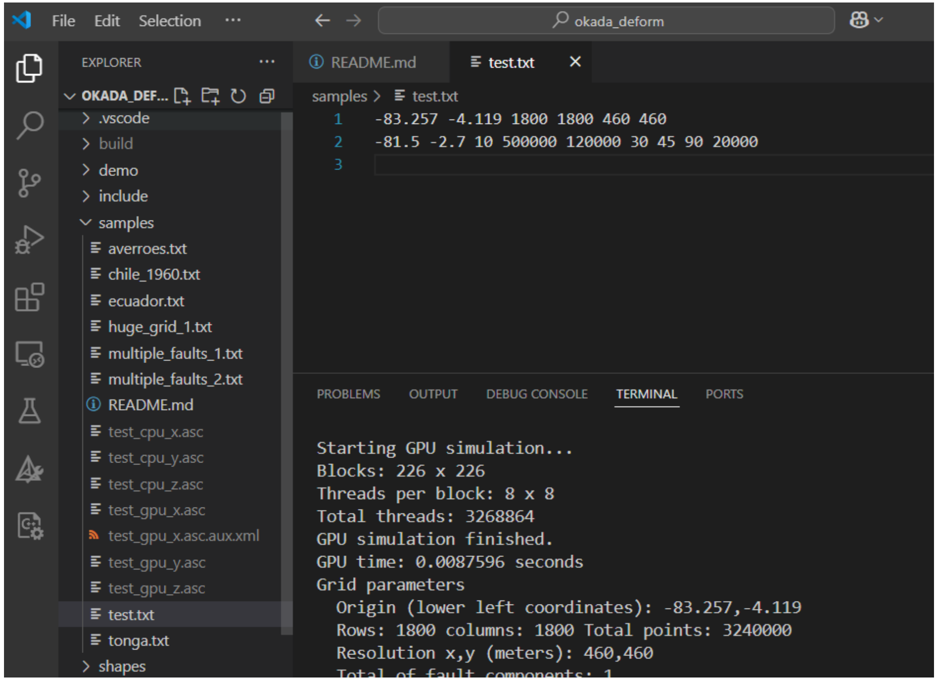
Dry run (No output of results)

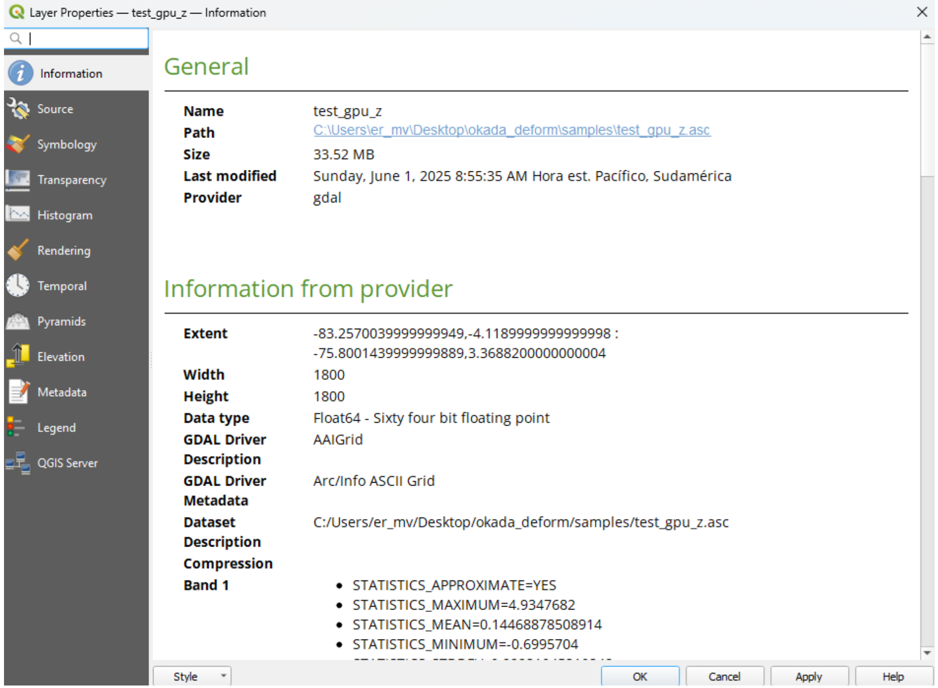


Visualizing the results

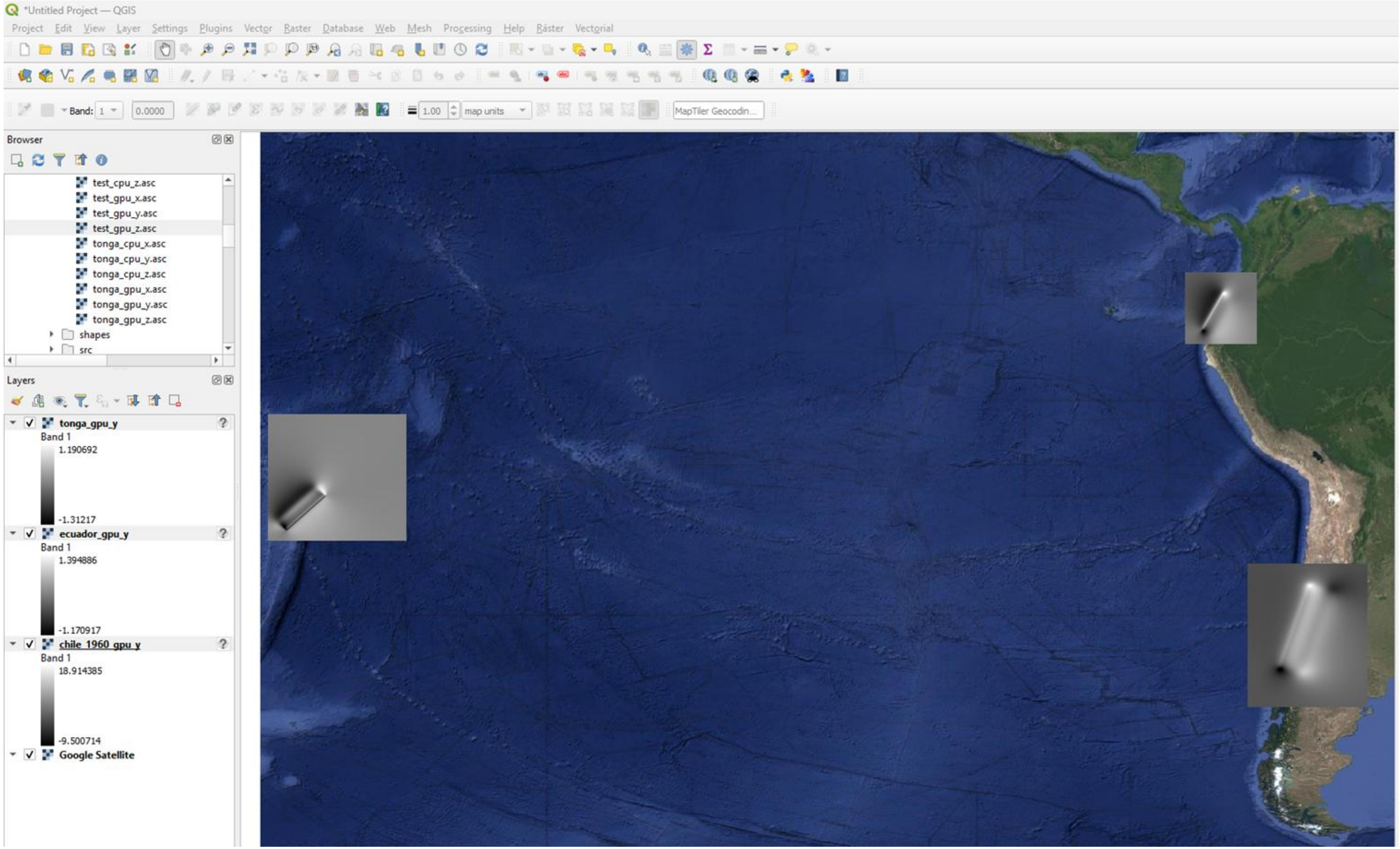


Generated grids





Deformation grids (y)

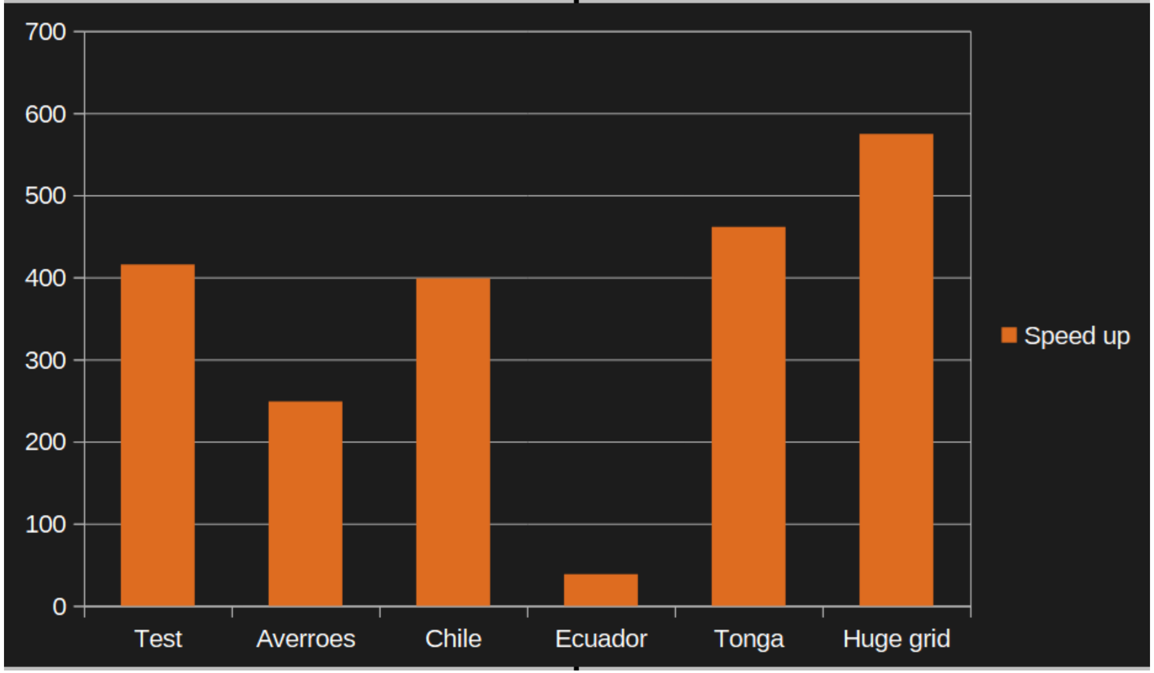


Results

- Successfully developed a CPU and GPU version of the Okada model
- The GPU version reduces the calculation time in orders of magnitude.
- To ensure reproducibility of the results, documentation about the input, output and test scenarios were provided.

Results

Scenario	Rows	Columns	Elements	GPU time	CPU time	Speed up
Test	1800	1800	3240000	0,008993	3,73878	415,743
Averroes	800	1200	960000	0,0042002	1,045	248,798
Chile	2048	1280	2621440	0,0076354	3,04371	398,631
Ecuador	1800	1800	3240000	0,097557	3,76589	38,602
Tonga	1800	1800	3240000	0,0087903	4,0551	461,315
Huge grid	17466	26367	460526022	0,895414	514,355	574,433



Lessons learned

- Time, time time!
  - Start to think about the project right when you enroll in the first course of the specialization (or before if possible!)
- Explore / learn by yourself
- Develop good coding practices
- Ask for help when needed!

Thanks

- Coursera, GPU Programming Specialization – Jhons Hopkins University
- Instructor, Fellow reviewers
- University of Cauca – Colombia