

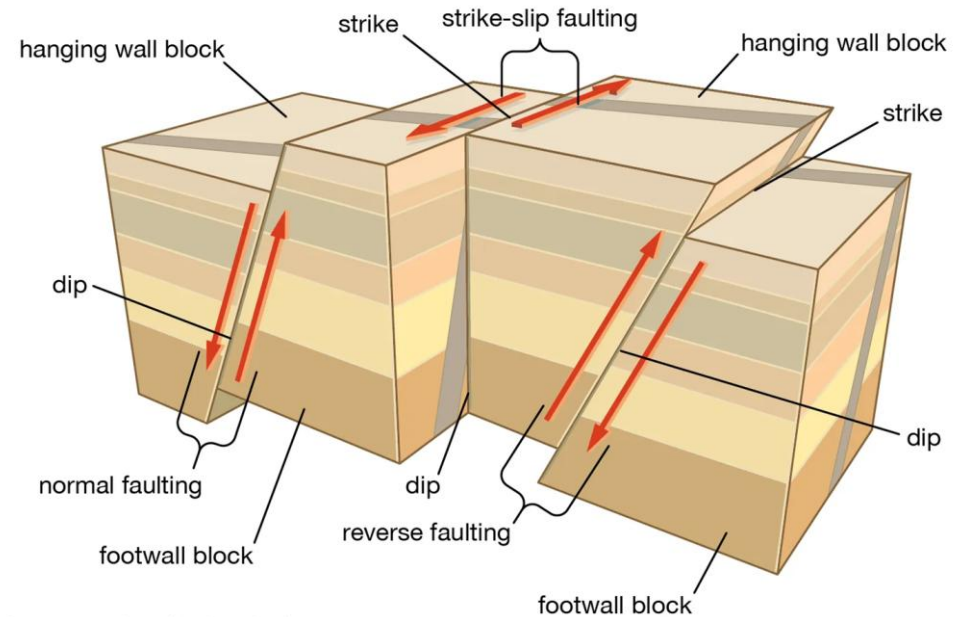
CPU/GPU Implementation of the Okada deformation model due to a finite rectangular earthquake source

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)



© 2015 Encyclopædia Britannica, Inc.

Okada deformation model

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

Okada (1985, 1992)

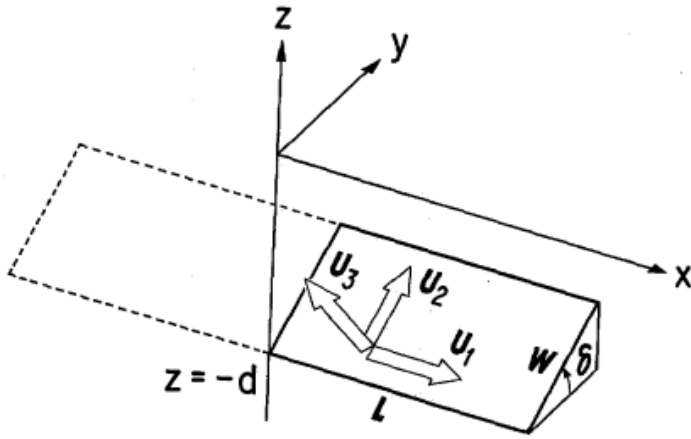


Fig. 1. Source:
Okada (1985)

Okada deformation model



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

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

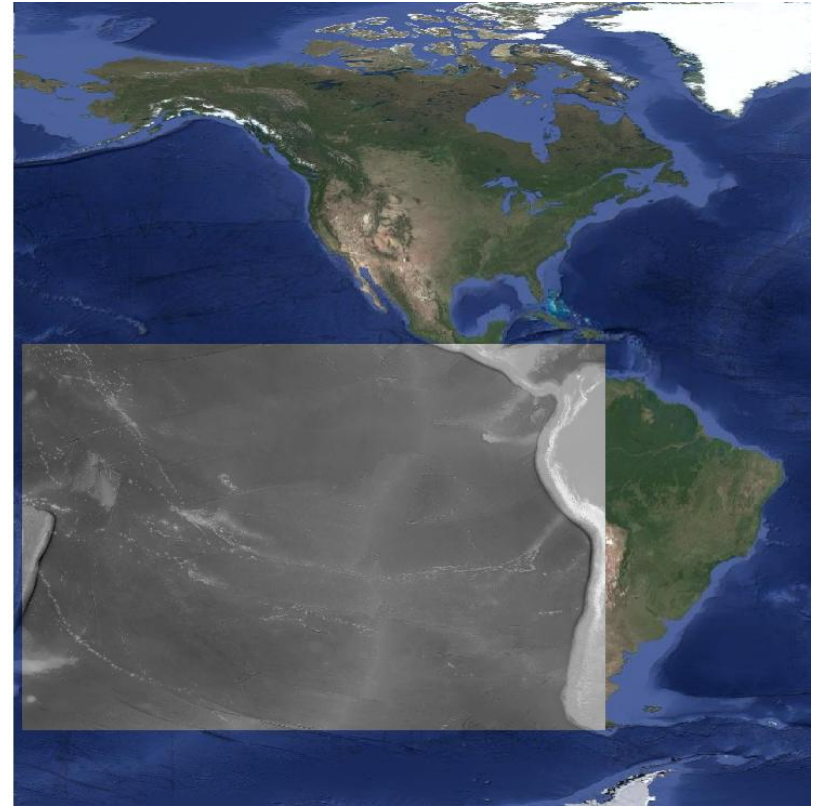
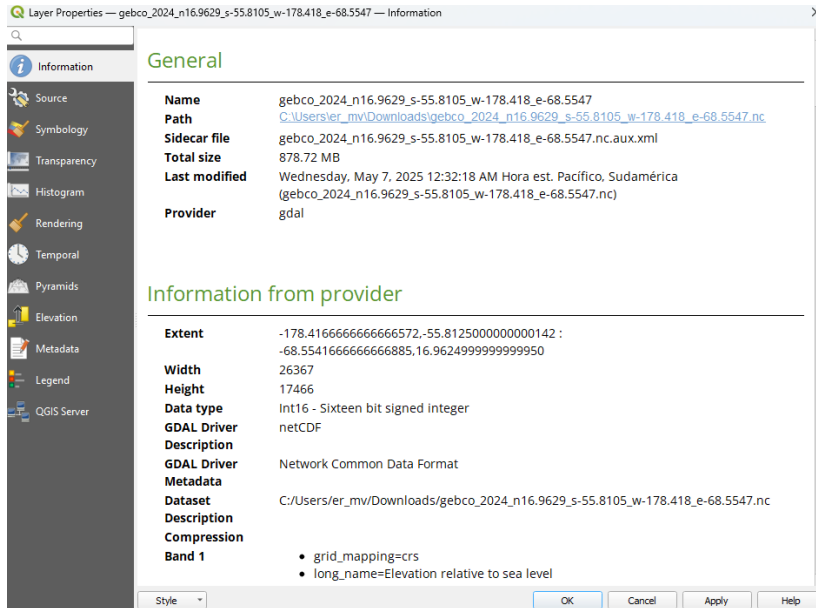
- Develop a prototype implementation as the Capstone Project for the GPU Programming Specialization (Coursera – Johns Hopkins University)
- Apply the tools and techniques studied so far to a real world scenario

Challenges

- Individual calculation on each point of the grid. Several equations to be evaluated, potentially 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



Process

- Study of the analytical 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, threadIdx, 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

The screenshot shows a GitHub repository page for 'github.com/emezav/okada_deform/tree/main'. The page has a dark theme. At the top, there's a navigation bar with 'README' and 'MIT license' tabs. The main content area features a title 'CPU/GPU Implementation of the Okada deformation model due to a finite rectangular earthquake source'. Below the title, there are three paragraphs of text explaining the model and its application. A 'TL;DR (Summary)' section follows, containing a bulleted list of key points and notes. On the right side, there's a sidebar titled 'Based on your tech stack' which recommends three tools: 'SLSA Generic generator', 'CMake based, multi-platform projects', and 'MSBuild based projects', each with a 'Configure' button. At the bottom of the sidebar, there are links for 'More workflows' and 'Dismiss suggestions'.

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

The screenshot displays the GitHub web interface for the repository 'okada_deform' by user 'emezav'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The left sidebar shows the file explorer with the 'samples' folder selected, listing files: README.md, averroes.txt, chile_1960.txt, ecuador.txt, huge_grid_1.txt, multiple_faults_1.txt, multiple_faults_2.txt, test.txt, and tonga.txt. The main content area shows the 'okada_deform / samples' directory view, featuring a commit by 'emezav' titled 'Add demo video - Linux' and a list of files: .., README.md, averroes.txt, chile_1960.txt, ecuador.txt, huge_grid_1.txt, multiple_faults_1.txt, multiple_faults_2.txt, test.txt, and tonga.txt.

emezav / okada_deform

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

main + 🔍

Go to file 🔍

- > .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:

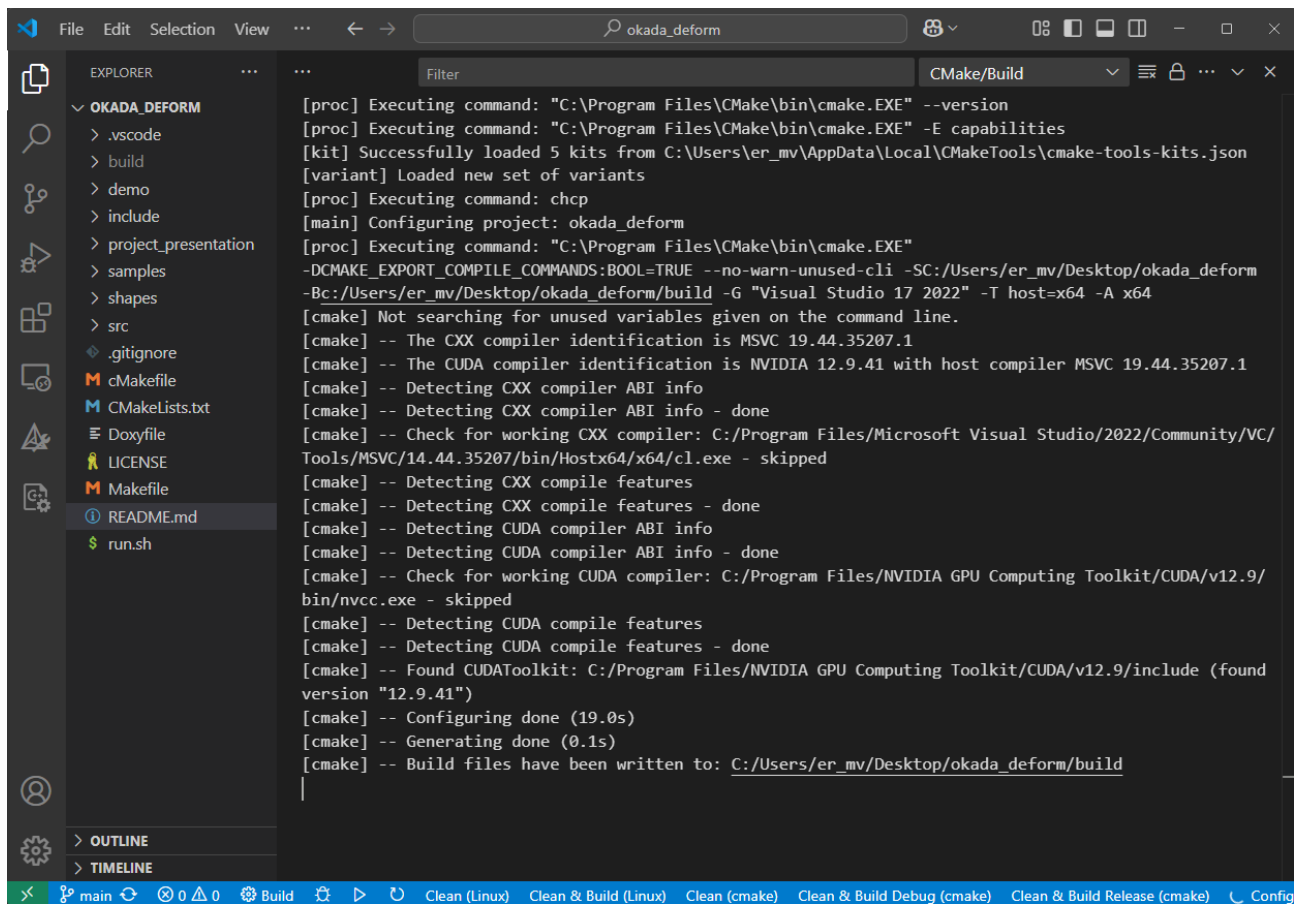
- make -f cMakefile dev-vs
- cd build
- cmake --build . --target Release
- build\Release\okada_deform.exe ..\samples\test.txt

- VS code: Use the action buttons at the bottom toolbar

- Visual Studio solution is created via cmake.

- Open the .sln under the build folder
- Use the Visual Studio GUI to run/debug.

Automatic Project setup (Vs Code)



The screenshot displays the Visual Studio Code interface with the 'okada_deform' project open. The Explorer sidebar on the left shows the project structure, including files like .vscode, build, demo, include, project_presentation, samples, shapes, src, .gitignore, CMakefile, CMakeLists.txt, Doxyfile, LICENSE, Makefile, README.md, and run.sh. The main editor area shows the output of CMake commands, indicating the successful configuration of the project. The status bar at the bottom shows the current build configuration as 'Clean (Linux)'.

```
[proc] Executing command: "C:\Program Files\CMake\bin\cmake.EXE" --version
[proc] Executing command: "C:\Program Files\CMake\bin\cmake.EXE" -E capabilities
[kit] Successfully loaded 5 kits from C:\Users\er_mv\AppData\Local\CMakeTools\cmake-tools-kits.json
[variant] Loaded new set of variants
[proc] Executing command: chcp
[main] Configuring project: okada_deform
[proc] Executing command: "C:\Program Files\CMake\bin\cmake.EXE"
-DCMAKE_EXPORT_COMPILE_COMMANDS=BOOL=TRUE --no-warn-unused-cli -SC:/Users/er_mv/Desktop/okada_deform
-Bc:/Users/er_mv/Desktop/okada_deform/build -G "Visual Studio 17 2022" -T host-x64 -A x64
[cmake] Not searching for unused variables given on the command line.
[cmake] -- The CXX compiler identification is MSVC 19.44.35207.1
[cmake] -- The CUDA compiler identification is NVIDIA 12.9.41 with host compiler MSVC 19.44.35207.1
[cmake] -- Detecting CXX compiler ABI info
[cmake] -- Detecting CXX compiler ABI info - done
[cmake] -- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/
Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
[cmake] -- Detecting CXX compile features
[cmake] -- Detecting CXX compile features - done
[cmake] -- Detecting CUDA compiler ABI info
[cmake] -- Detecting CUDA compiler ABI info - done
[cmake] -- Check for working CUDA compiler: C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.9/
bin/nvcc.exe - skipped
[cmake] -- Detecting CUDA compile features
[cmake] -- Detecting CUDA compile features - done
[cmake] -- Found CUDAToolkit: C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.9/include (found
version "12.9.41")
[cmake] -- Configuring done (19.0s)
[cmake] -- Generating done (0.1s)
[cmake] -- Build files have been written to: C:/Users/er_mv/Desktop/okada_deform/build
```

Manual project setup (Windows)

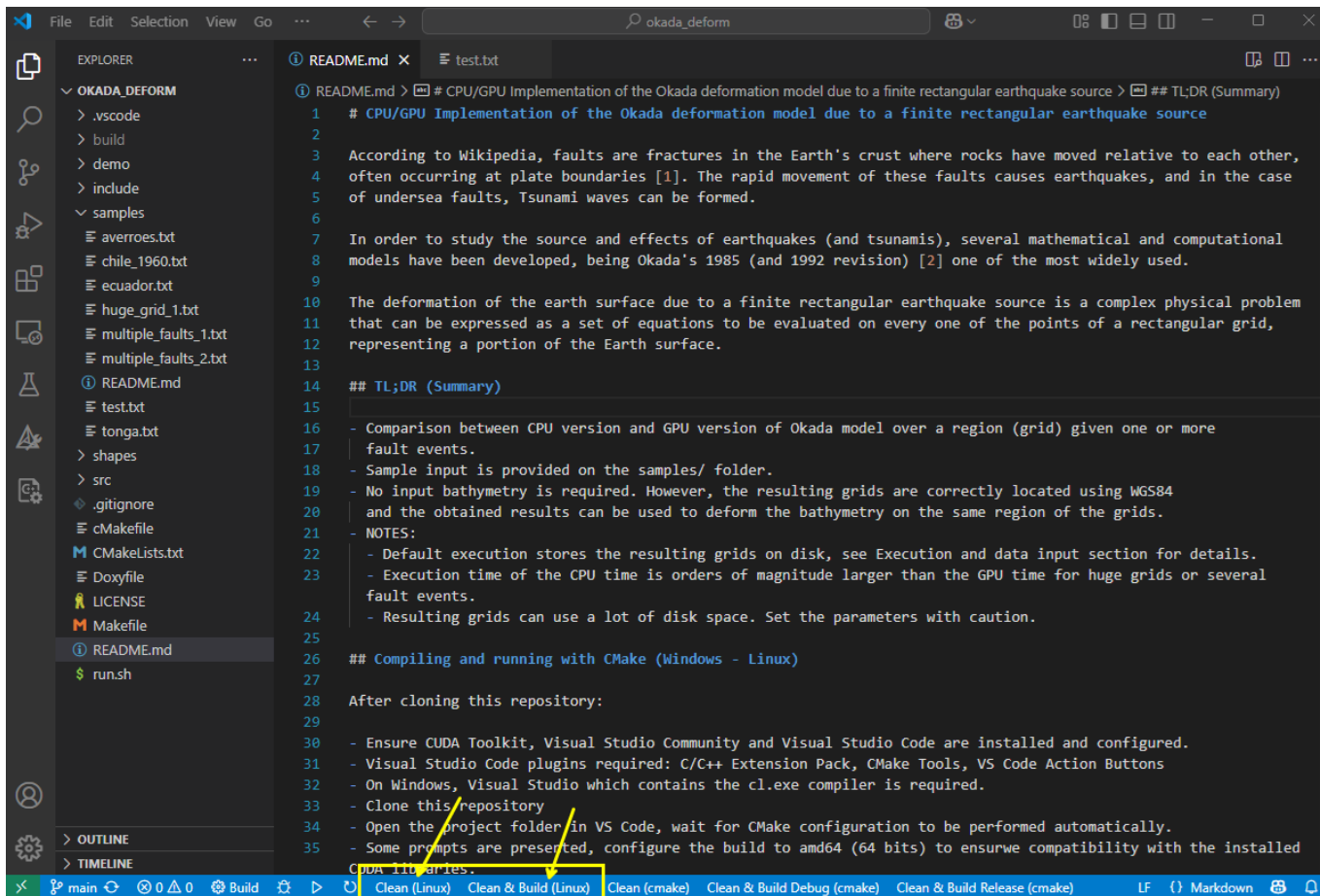
make is required

```
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\er_mv\Desktop\okada_deform>make -f cMakefile dev-vs
rm -rf build
mkdir -p build
cd build;cmake -G "Visual Studio 17 2022" -A x64 ..
-- The CXX compiler identification is MSVC 19.44.35207.1
-- The CUDA compiler identification is NVIDIA 12.9.41 with host compiler MSVC 19.44.35207.1
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Detecting CUDA compiler ABI info
-- Detecting CUDA compiler ABI info - done
-- Check for working CUDA compiler: C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.9/bin/nvcc.exe - skipped
-- Detecting CUDA compile features
-- Detecting CUDA compile features - done
-- Found CUDAToolkit: C:/Program Files/NVIDIA GPU Computing Toolkit/CUDA/v12.9/include (found version "12.9.41")
-- Configuring done (19.0s)
-- Generating done (0.1s)
-- Build files have been written to: C:/Users/er_mv/Desktop/okada_deform/build

C:\Users\er_mv\Desktop\okada_deform>
```


Build with VS Code (Linux)



```
File Edit Selection View Go ... okada_deform
EXPLORER
OKADA_DEFORM
  .vscode
  build
  demo
  include
  samples
    averroes.txt
    chile_1960.txt
    ecuador.txt
    huge_grid_1.txt
    multiple_faults_1.txt
    multiple_faults_2.txt
    README.md
    test.txt
    tonga.txt
  shapes
  src
  .gitignore
  CMakefile
  CMakeLists.txt
  Doxyfile
  LICENSE
  Makefile
  README.md
  run.sh
OUTLINE
TIMELINE
main 0 Build Clean (Linux) Clean & Build (Linux) Clean (cmake) Clean & Build Debug (cmake) Clean & Build Release (cmake) LF Markdown
```

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

2

3 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.

4

5 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.

6

7 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.

8

9

10

11

12

13

14 ## TL;DR (Summary)

15

16 - Comparison between CPU version and GPU version of Okada model over a region (grid) given one or more fault events.

17

18 - Sample input is provided on the samples/ folder.

19

20 - 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.

21

22 - NOTES:

23

24 - Default execution stores the resulting grids on disk, see Execution and data input section for details.

25

26 - Execution time of the CPU time is orders of magnitude larger than the GPU time for huge grids or several fault events.

27

28 - Resulting grids can use a lot of disk space. Set the parameters with caution.

29

30

31 ## Compiling and running with CMake (Windows - Linux)

32

33 After cloning this repository:

34

35

36 - Ensure CUDA Toolkit, Visual Studio Community and Visual Studio Code are installed and configured.

37

38 - Visual Studio Code plugins required: C/C++ Extension Pack, CMake Tools, VS Code Action Buttons

39

40 - On Windows, Visual Studio which contains the cl.exe compiler is required.

41

42 - Clone this repository

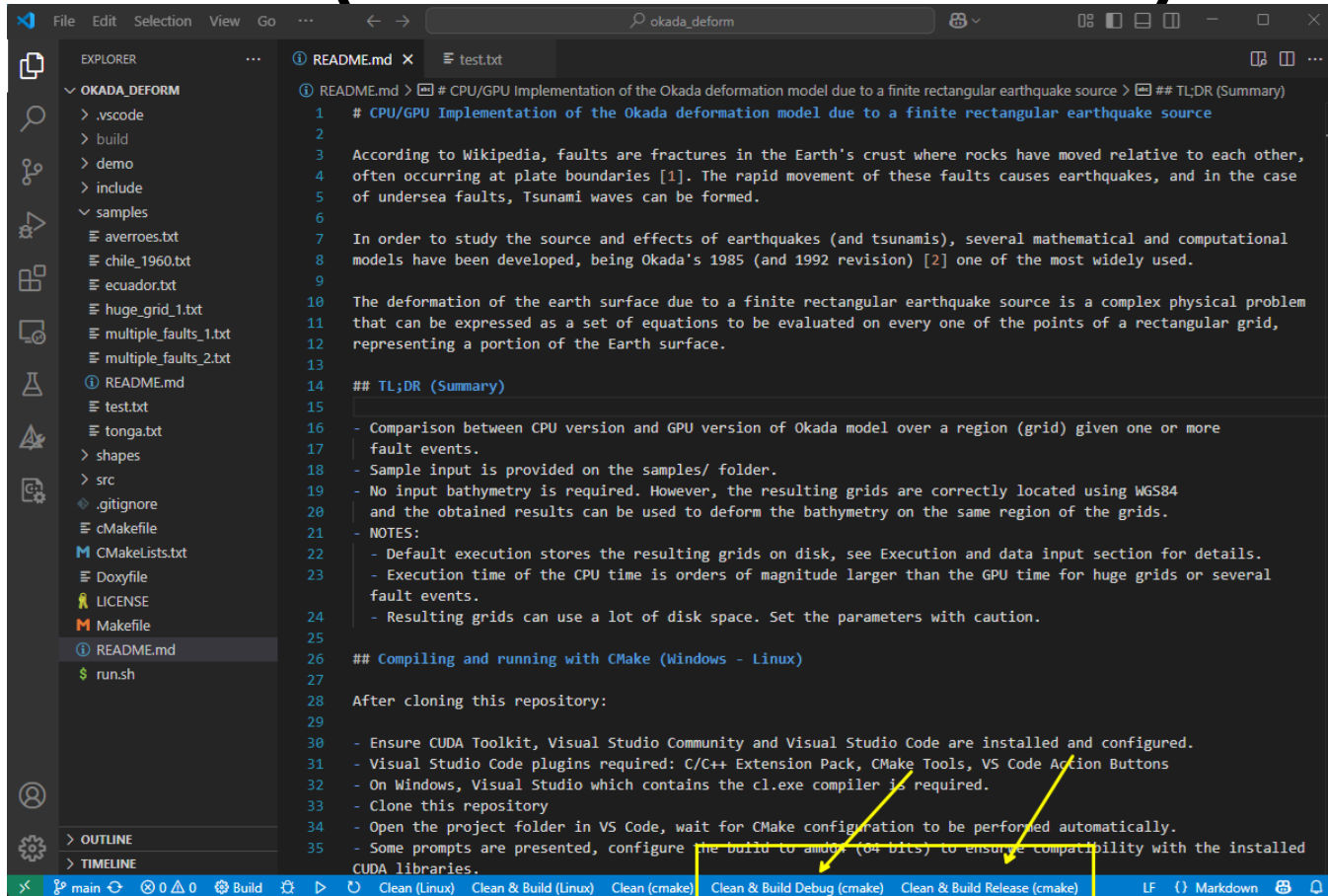
43

44 - Open the project folder in VS Code, wait for CMake configuration to be performed automatically.

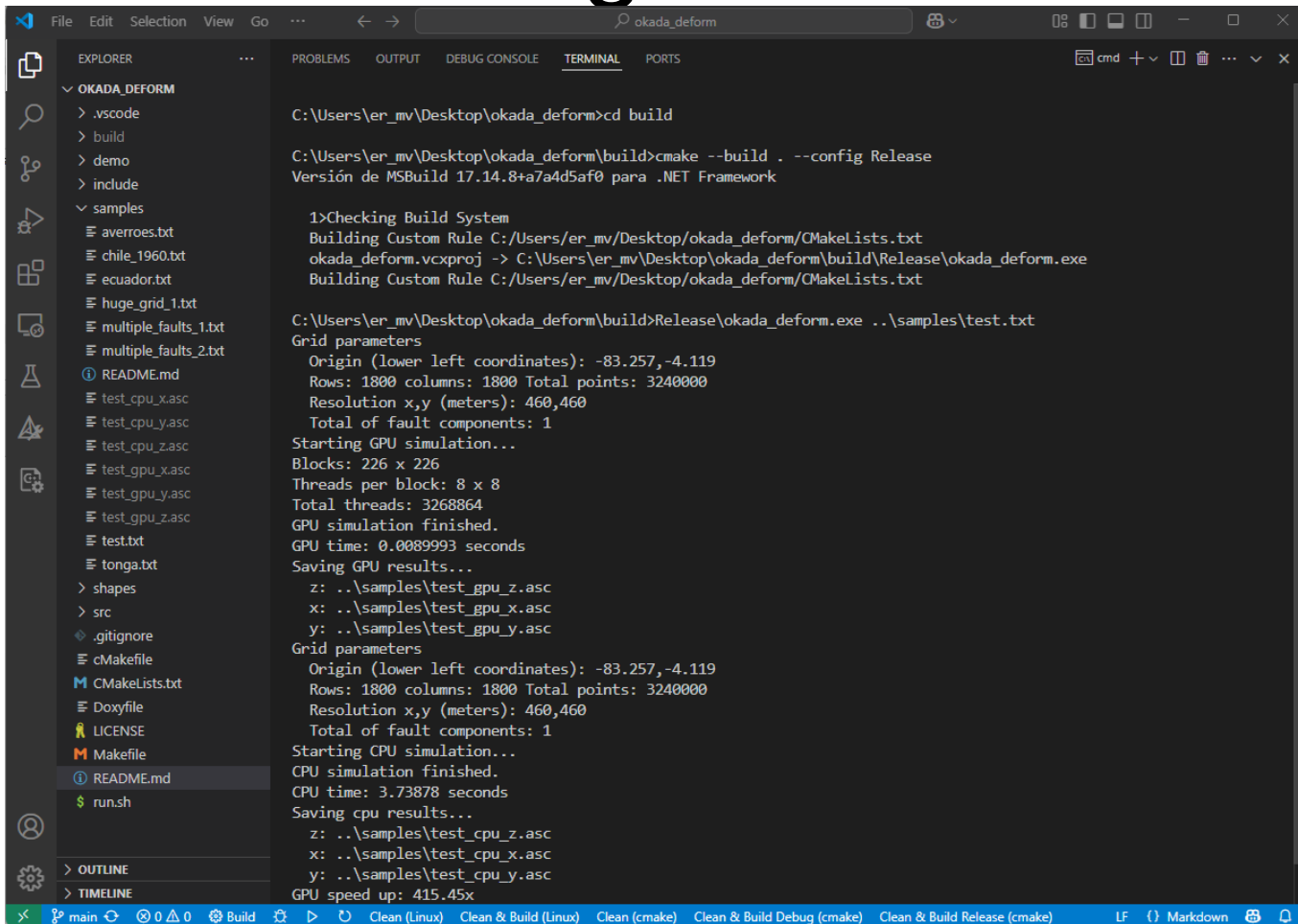
45

46 - Some prompts are presented, configure the build to amd64 (64 bits) to ensure compatibility with the installed CUDA libraries.

Build with VS Code (Windows/Linux)



Running scenarios



The screenshot shows a Visual Studio Code interface with a terminal window open. The Explorer sidebar on the left shows a project named 'OKADA_DEFORM' with various files and folders. The terminal window displays the following commands and output:

```
C:\Users\er_mv\Desktop\okada_deform>cd build

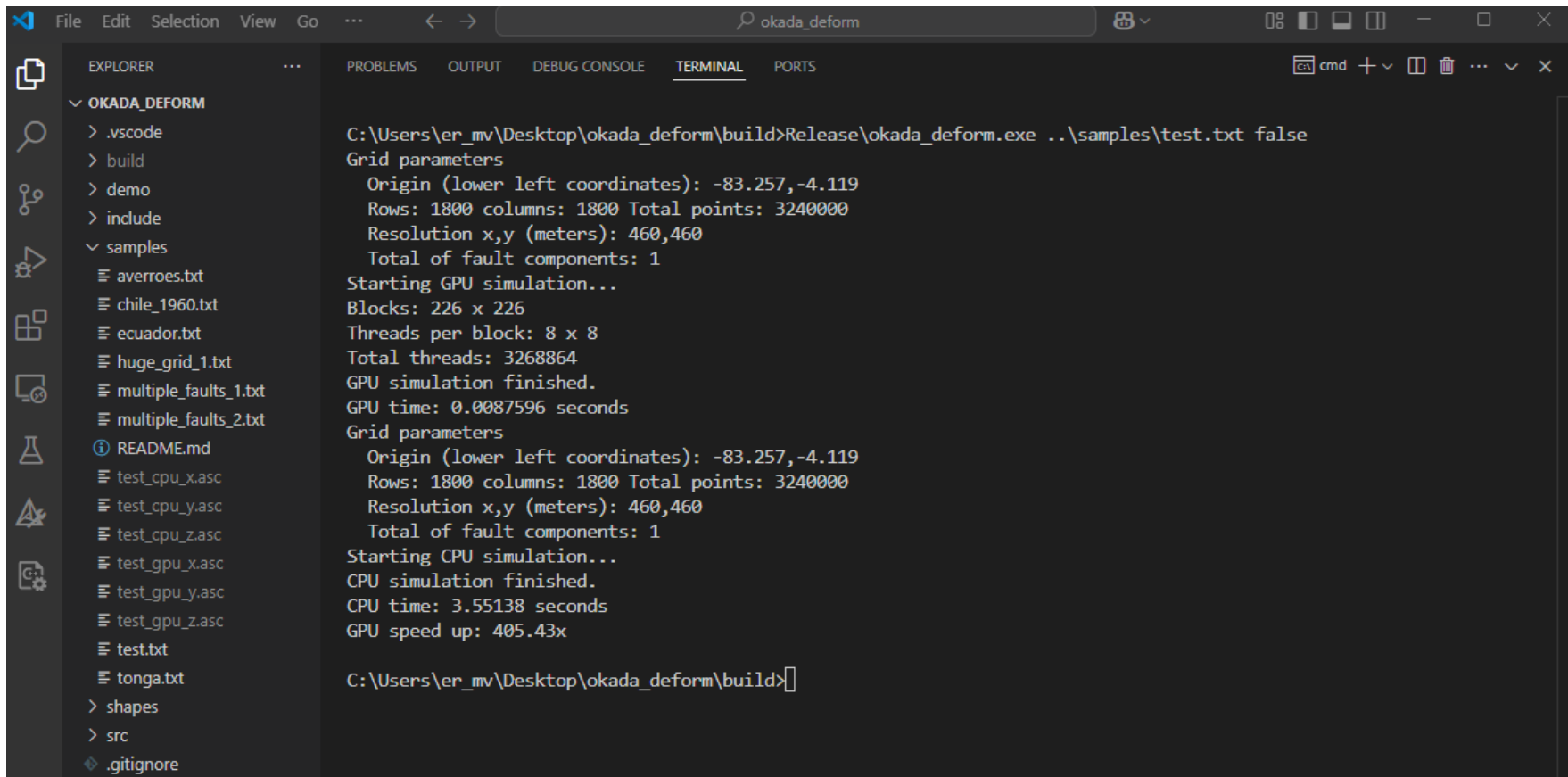
C:\Users\er_mv\Desktop\okada_deform\build>cmake --build . --config Release
Versión de MSBuild 17.14.8+a7a4d5af0 para .NET Framework

1>Checking Build System
Building Custom Rule C:/Users/er_mv/Desktop/okada_deform/CMakeLists.txt
okada_deform.vcxproj -> C:/Users/er_mv/Desktop/okada_deform/build/Release/okada_deform.exe
Building Custom Rule C:/Users/er_mv/Desktop/okada_deform/CMakeLists.txt

C:\Users\er_mv\Desktop\okada_deform\build>Release\okada_deform.exe ../samples/test.txt
Grid parameters
Origin (lower left coordinates): -83.257,-4.119
Rows: 1800 columns: 1800 Total points: 3240000
Resolution x,y (meters): 460,460
Total of fault components: 1
Starting GPU simulation...
Blocks: 226 x 226
Threads per block: 8 x 8
Total threads: 3268864
GPU simulation finished.
GPU time: 0.0089993 seconds
Saving GPU results...
z: ../samples/test_gpu_z.asc
x: ../samples/test_gpu_x.asc
y: ../samples/test_gpu_y.asc
Grid parameters
Origin (lower left coordinates): -83.257,-4.119
Rows: 1800 columns: 1800 Total points: 3240000
Resolution x,y (meters): 460,460
Total of fault components: 1
Starting CPU simulation...
CPU simulation finished.
CPU time: 3.73878 seconds
Saving cpu results...
z: ../samples/test_cpu_z.asc
x: ../samples/test_cpu_x.asc
y: ../samples/test_cpu_y.asc
GPU speed up: 415.45x
```

The status bar at the bottom shows the current build configuration as 'main' and provides buttons for cleaning and building the project in different configurations (Linux, Debug, Release).

Dry run (No output of results)



```
File Edit Selection View Go ... okada_deform
```

EXPLORER

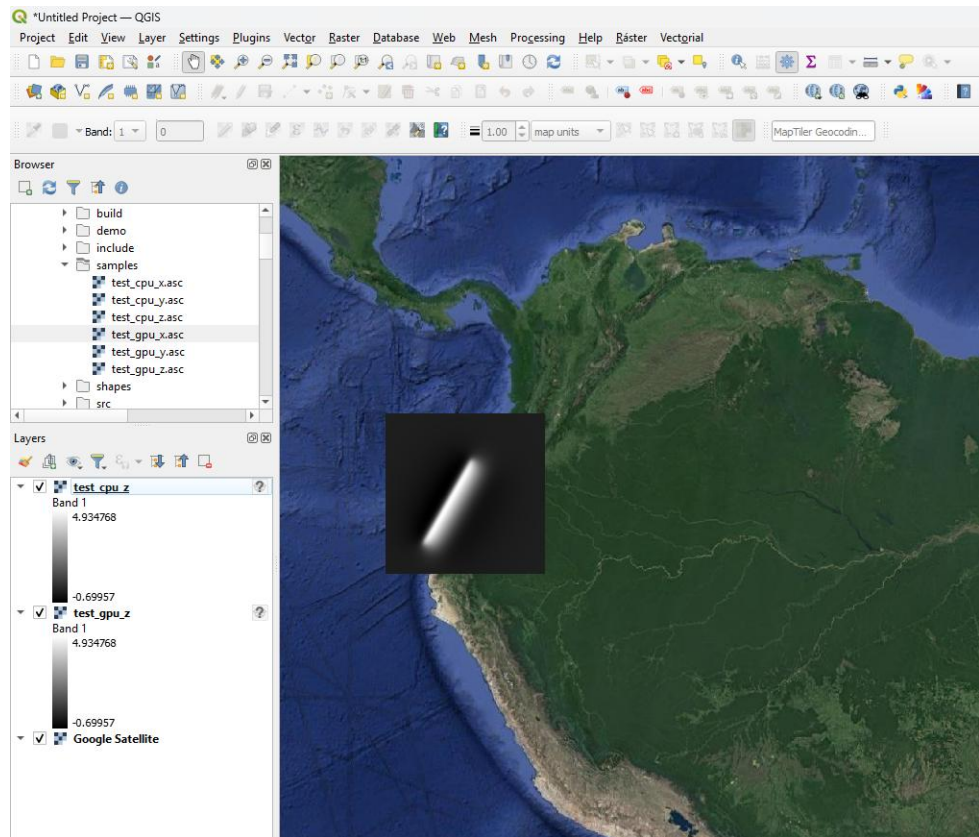
- OKADA_DEFORM
 - .vscode
 - build
 - demo
 - include
 - samples
 - averroes.txt
 - chile_1960.txt
 - ecuador.txt
 - huge_grid_1.txt
 - multiple_faults_1.txt
 - multiple_faults_2.txt
 - README.md
 - test_cpu_x.asc
 - test_cpu_y.asc
 - test_cpu_z.asc
 - test_gpu_x.asc
 - test_gpu_y.asc
 - test_gpu_z.asc
 - test.txt
 - tonga.txt
- shapes
- src
- .gitignore

TERMINAL

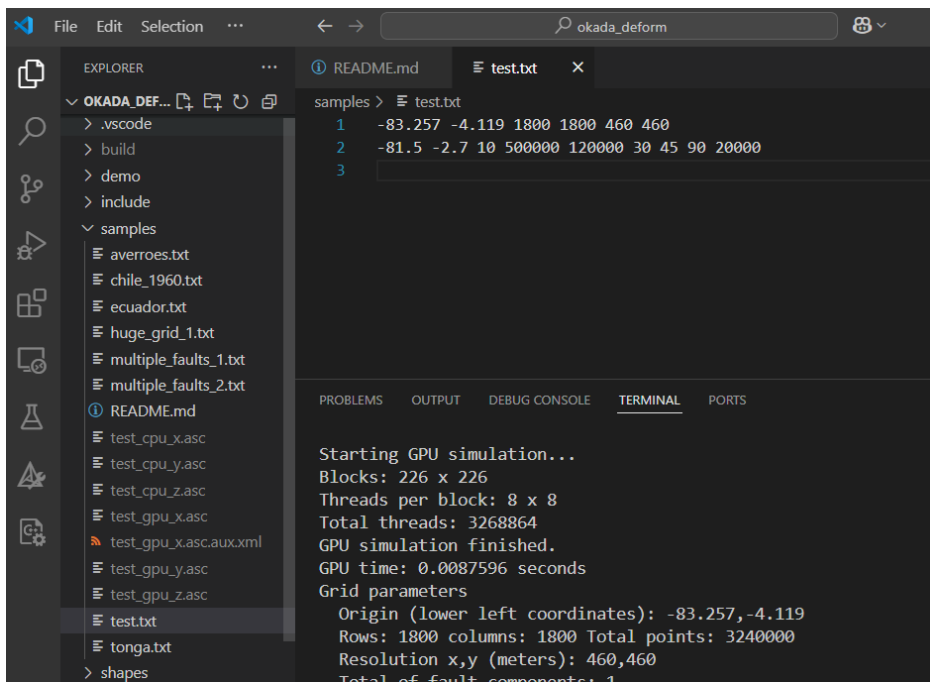
```
C:\Users\er_mv\Desktop\okada_deform\build>Release\okada_deform.exe ..\samples\test.txt false
Grid parameters
  Origin (lower left coordinates): -83.257,-4.119
  Rows: 1800 columns: 1800 Total points: 3240000
  Resolution x,y (meters): 460,460
  Total of fault components: 1
Starting GPU simulation...
Blocks: 226 x 226
Threads per block: 8 x 8
Total threads: 3268864
GPU simulation finished.
GPU time: 0.0087596 seconds
Grid parameters
  Origin (lower left coordinates): -83.257,-4.119
  Rows: 1800 columns: 1800 Total points: 3240000
  Resolution x,y (meters): 460,460
  Total of fault components: 1
Starting CPU simulation...
CPU simulation finished.
CPU time: 3.55138 seconds
GPU speed up: 405.43x

C:\Users\er_mv\Desktop\okada_deform\build>
```

Visualizing the results



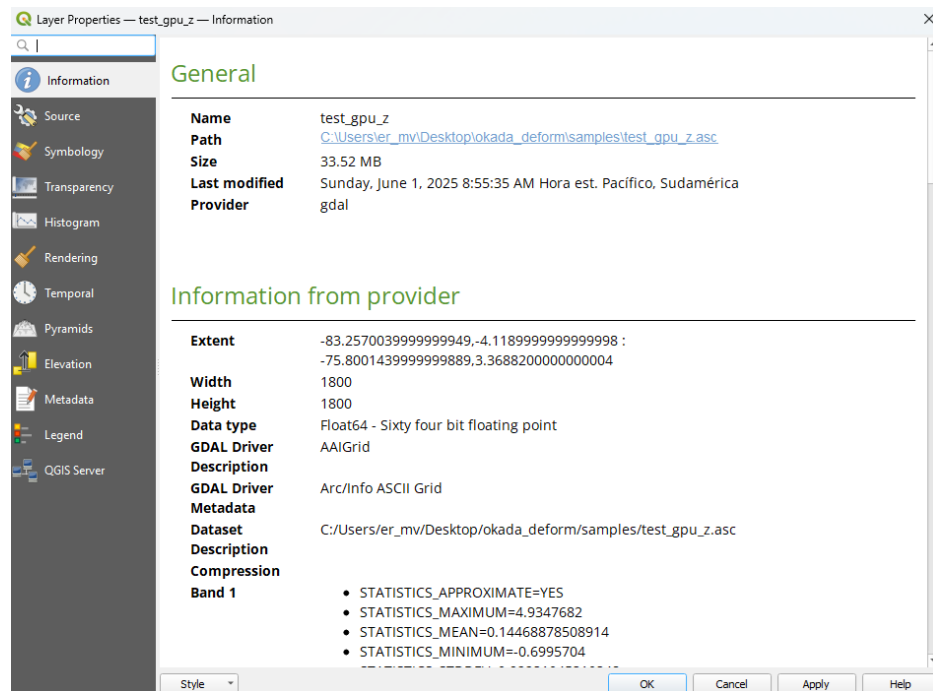
Generated grids



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure with a 'samples' directory containing files like 'averroes.txt', 'chile_1960.txt', 'ecuador.txt', 'huge_grid_1.txt', 'multiple_faults_1.txt', 'multiple_faults_2.txt', 'README.md', 'test_cpu_x.asc', 'test_cpu_y.asc', 'test_cpu_z.asc', 'test_gpu_x.asc', 'test_gpu_x.asc.aux.xml', 'test_gpu_y.asc', 'test_gpu_z.asc', 'test.txt', 'tonga.txt', and 'shapes'. The main editor area shows 'test.txt' with three lines of coordinates. The TERMINAL panel at the bottom displays the output of a GPU simulation.

```
1 -83.257 -4.119 1800 1800 460 460
2 -81.5 -2.7 10 500000 120000 30 45 90 20000
3
```

Starting GPU simulation...
Blocks: 226 x 226
Threads per block: 8 x 8
Total threads: 3268864
GPU simulation finished.
GPU time: 0.0087596 seconds
Grid parameters
Origin (lower left coordinates): -83.257,-4.119
Rows: 1800 columns: 1800 Total points: 3240000
Resolution x,y (meters): 460,460
Total of fault components: 1

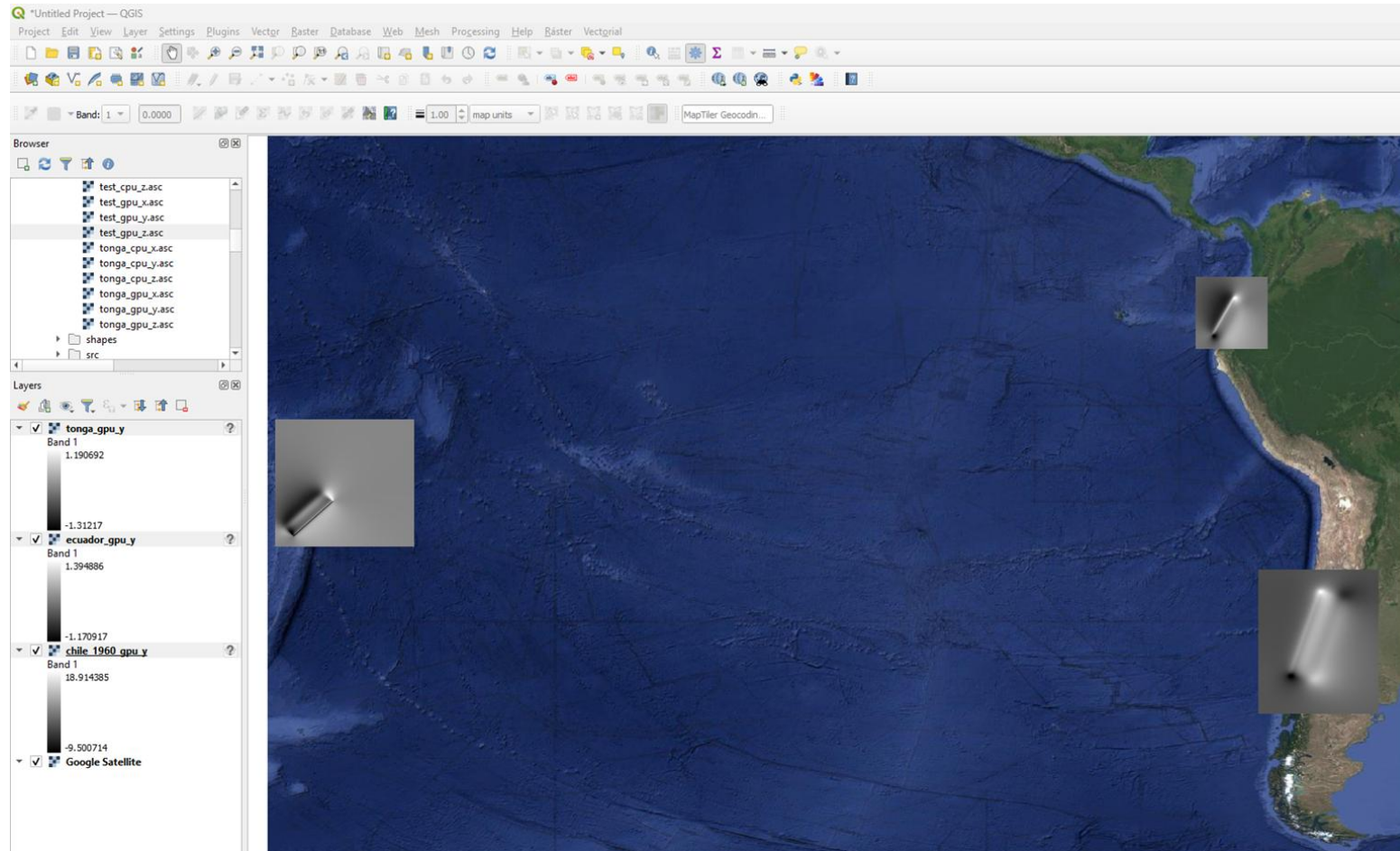


The screenshot shows the 'Layer Properties' dialog for the layer 'test_gpu_z'. The 'General' tab is selected, displaying the following information:

General	
Name	test_gpu_z
Path	C:/Users/er_mv/Desktop/okada_deform/samples/test_gpu_z.asc
Size	33.52 MB
Last modified	Sunday, June 1, 2025 8:55:35 AM Hora est. Pacifico, Sudamérica
Provider	gdal

Information from provider	
Extent	-83.2570039999999949,-4.1189999999999998 ; -75.8001439999999889,3.3688200000000004
Width	1800
Height	1800
Data type	Float64 - Sixty four bit floating point
GDAL Driver	AAIGrid
Description	
GDAL Driver Metadata	Arc/Info ASCII Grid
Dataset	C:/Users/er_mv/Desktop/okada_deform/samples/test_gpu_z.asc
Description	
Compression	
Band 1	<ul style="list-style-type: none">STATISTICS_APPROXIMATE=YESSTATISTICS_MAXIMUM=4.9347682STATISTICS_MEAN=-0.14468878508914STATISTICS_MINIMUM=-0.6995704

Deformation grids (y)

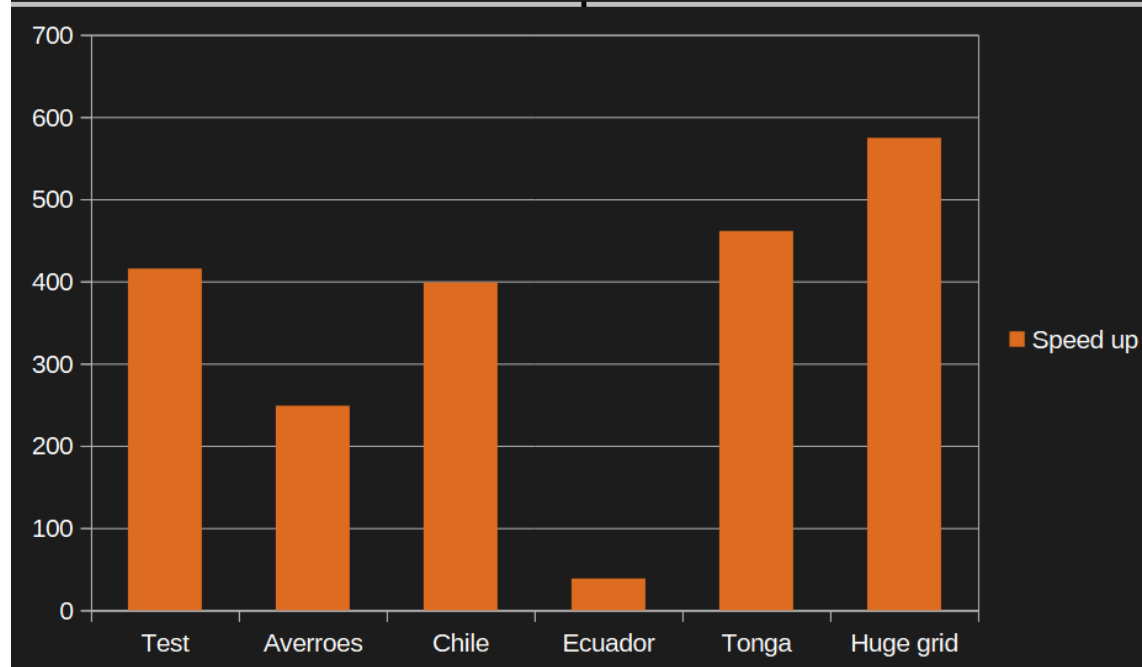


Results

- 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 is 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 after 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