FQSHA: An open-source Python software for Fault-Based Seismic Hazard Assessment

Nasrin Tavakolizadeh[a,c], Hamzeh Mohammadigheymasi[a,b], Nuno Pombo[a,c]

[a] *Department of Computer Sciences, University of Beira Interior, n.tavakolizadeh@ubi.pt*
[b] *Atmosphere and Ocean Research Institute (AORI), The University of Tokyo, Japan*
[c] *Instituto de Telecomunicações (IT), Covilhã, Portugal*

Abstract

The PyQt framework facilitates the development of desktop applications, offering an effective environment for implementing scientific algorithms while leveraging the flexibility of Python. In this paper, we introduce FQSHA, a PyQt5-based application designed to streamline the workflow of fault-based Seismic Activity Rate (SAR) calculation and seismic hazard assessment. The primary aim is to provide a unified and user-friendly interface that makes these processes more accessible. FQSHA enables users to perform hazard calculations and map the results from scratch using minimal input data and requiring only basic knowledge of hazard computation engines. The SAR calculation core (FaultQuake) is seamlessly integrated with the hazard assessment engine (OpenQuake), offering additional flexibility to customize input parameters for hazard analysis. This integration supports an end-to-end workflow within a single platform. We present the architecture of FQSHA and demonstrate its capabilities through a hands-on example, which is publicly available on GitHub.

Keywords: Graphical user interface, Seismic Activity rate, Fault-Based Seismic Hazard assessment

Code metadata

Current code version	gv01
Permanent link to Reproducible Capsule	https://github.com/GeoSignalAnalysis/FQSHA
Legal Code License	All software and code must be released under one of the pre-approved licenses listed in the Guide for Authors, such as Apache License, GNU General Public License (GPL) or MIT License. Write the name of the license you've chosen here.
Code versioning system used	Git
Software code languages, tools, and services used	Python v3.11
Compilation requirements, operating environments & dependencies	Microsoft Windows, Linux
If available Link to developer documentation/manual	https://github.com/GeoSignalAnalysis/FQSHA/Documentation.pdf
Support email for questions	n.tavakolizadeh@ubi.pt

1. Motivation and significance

The existing software for Probabilistic Seismic Hazard Assessment (PSHA) are categorized in two different categories, intermediate software and hazard modelers. The intermediate software are used for Seismic Activity Rates (SAR) calculation. Such as FiSH [1], SHERIFS [2], SUNFiSH, and FRESH [3], serve as intermediary tools for processing kinematic and geometric fault parameters to estimate a fault's Activity Rate (AR) and these activity rates are then used as inputs for a hazard assessment tool. The most recent tool, FaultQuake [4] incorporates Seismic Coupling Coefficient (SCC) into the SAR calculation process. A significant feature of this software is its ability to discriminate between seismic and aseismic deformation, a critical factor in understanding the behavior of active faults [5]. Although, this tool calculates SAR, it is not a versatile solution for seismic hazard calculations and mapping the hazard. On the hazard calculation side, the OpenQuake engine [6], promoted by the Global Earthquake Model (GEM) Foundation, is a platform that not only facilitates fault-based hazard modeling but also serves as a multi-purpose software for seismic hazard assessment.

These different platforms necessitate the familiarity with compiling at least two different software on different languages and different input formats (text, xml, .ini, etc.) and files. Mapping the output results and comparing them with the fault's shape files is the final step that the user need to do and this steps are a challenging and time-consuming process for students, educators, and researchers.

The motivation for developing our workflow, FQSHA, embedded with a Graphical User Interface (GUI), stems from the need for a versatile and flexible tool that expertly processes the geometry and kinematics of active faults to calculate the SAR, generate the necessary hazard calculation parameters, and efficiently incorporate them into hazard engines like OpenQuake and maps the results. We present an advanced open-source Python tool equipped with a user-friendly GUI to handle fault SAR calculations and fault hazard assessments in a simple and straightforward manner. The GUI synthesizes this data intuitively, allowing users to quickly explore different scenarios and outcomes, thereby enhancing the speed and efficiency of seismic hazard modeling and interpretation for both experts and newcomers in the field of Seismic Hazard Assessment with minimal effort.

2. Software description

The significance of the FQSHA tool in the context of current literature lies in its ability to unify the workflow for fault-based seismic hazard analysis. Its main contributions include:

- Eliminating the need for time-consuming activities like analyzing preparation of inputs for different tools and packages.
- Integrating SAR computation directly within the framework, removing the dependency on intermediate software to calculate seismic activity rates and exporting the ARs to be compatible by another software.
- Enhancing accessibility to fault-based hazard analysis by enabling direct computation of seismic hazard from fault models within a single environment.
- Providing built-in functionality to generate fault-specific hazard maps, thereby accelerating the evaluation of multiple hazard scenarios.

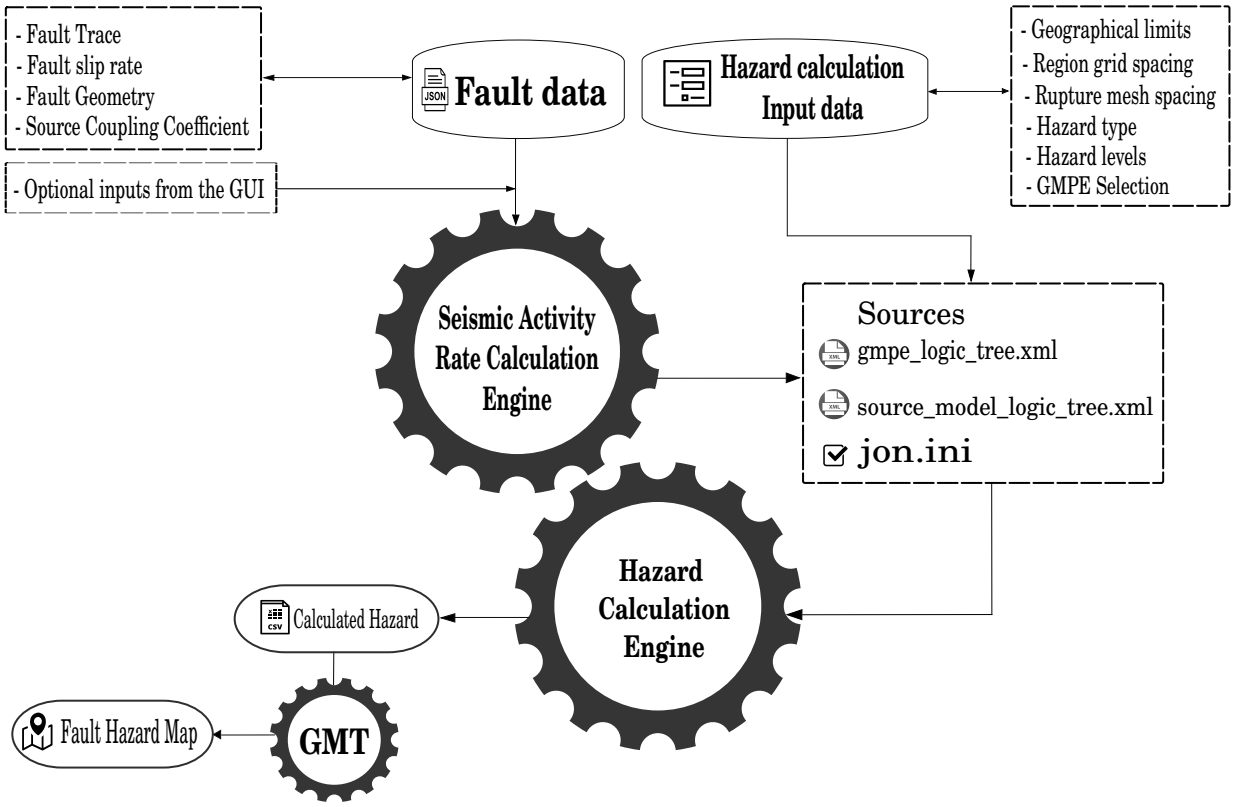


Figure 1: FQSHA features and functionalities are presented in a schematic display framework. As illustrated, the tool integrates three core engines for Seismic Activity Rate (SAR) and Hazard calculation, and Generic Mapping Tools (GMT) mapping toolkit.

2.1. Software Functionality

This section presents the theoretical background and structural design of the FQSHA tool. The core functionality of this Python-based graphical interface lies in the integration of fault SAR modeling—based on the methodology introduced by [4]—with the OpenQuake hazard computation engine [6], allowing for the direct generation of fault-based seismic hazard maps (Fig. 1).

The primary objective of a fault-based seismic hazard assessment is to realistically incorporate the physical characteristics of active faults into hazard models. This involves leveraging available geological and geophysical data—such as fault geometry (length, dip, seismogenic thickness), slip rates (minimum and maximum), and earthquake history (observed magnitude (M_{obs}) and its associated standard deviation, Standard Deviation (SD))—to estimate the likelihood and impact of future seismic events. In this context, FQSHA utilizes an advanced SAR calculation engine that also accepts optional inputs such as the Seismic Coupling Coefficient (SCC) to partition the total deformation into seismic and aseismic components (see Table 1). These geometrical and kinematic attributes are used to derive five different empirical magnitude estimates (Empirical Magnitudes (EMs)). Through Conflation of Probabilities (CoP) that also incorporates M_{obs}, the maximum expected earthquake magnitude (Maximum Magnitude (M_{max})) and its uncertainty (σ) are computed. The total moment rate is calculated by integrating $M_{\max}$, σ, and the mean recurrence interval T_{mean}, which is derived using the segment seismic moment conservation principle [7]. This moment rate is then balanced across a range of magnitudes between M_{min} and $M_{\max}$, using relative activity models such as the Truncated Gutenberg-Richter (Truncated Gutenberg-Richter (TGR)) or Characteristic Gaussian Distribution (Characteristic Gaussian Distribution (CGD)) models. The result is a complete SAR distribution for each fault, which forms the basis for subsequent

hazard computations. Further methodological details are available in [4].
To perform the seismic hazard analysis, FQSHA interfaces directly with the OpenQuake engine. In the context of fault-based hazard modeling, the OpenQuake engine enables detailed specification of active fault sources using user-defined geometries, slip rates, and magnitude-frequency distributions (MFDs) [8]. Epistemic uncertainties are handled through logic tree frameworks, allowing for probabilistic representation of source model and ground motion model alternatives. These capabilities make OpenQuake engine a powerful and well-suited backend for regional and national hazard modeling applications [9].

Algorithm 1 FQSHA GUI-Based Seismic Hazard Workflow

Input: Fault geometry (JSON), hazard parameters (GUI), user selections
Launch PyQt5 GUI and display fields for model input and configuration
On *Browse* button click:
  Open file dialog and parse fault JSON
On *RUN* button click:
  Collect user inputs from GUI fields
  Validate input fields and apply defaults
  Create output directory tree
Prepare Input Files:
  Export parameters to `inputs.json`
  Export fault traces to `fault_traces.json`
  Call `source_model_logic_tree()` to generate source model logic tree XML
  Call `generate_job_ini()` to create OpenQuake configuration
  Call `gmpe_generate_xml()` to export selected GMPEs
Moment Budget and Scaling:
  Call `momentbudget()` from `SeismicActivityRate.py`
    Uses: `kin2coeff()`, `coeff2mag()`, `conflate_pdfs()`
    Computes: Mmax, $\sigma_{M_{\max}}$, T_{mean}, MoRate
  Generates PDFs and plots for fault-specific magnitude models
Seismic Activity Rate Modeling:
  Call `sactivityrate()` with:
    - Fault behavior (Characteristic Gaussian or Truncated GR)
    - Time window, bin width
    Internally calls:
      `CHGaussPoiss()` or `CHGaussBPT()` for Gaussian faults
      `TruncatedGR()` for GR faults
  Updates fault XML files using `export_faults_to_xml()`
Run Hazard Calculation:
  Call `run_oq_engine()` to run OpenQuake engine with `job.ini`
  Exports CSV outputs to `OutPut/`
Map Visualization:
  Use `find_latest_hazard_map()` to locate latest CSV result
if hazard map is found **then**
  Call `create_contour_map_with_faults()` to plot hazard contours
end if
Display completion message and highlight output folder

Algorithm 2 Seismic Activity Rate Calculation in FQSHA

Input: Fault dictionary from JSON, parameters: Zeta, Khi, Siggma, Fault behaviour, time window w, bin width
Initialize constants $c = 1.5$, $d = 9.1$
for all faults in input **do**
    Extract and convert fault geometry and physical parameters
    Compute fault area from dip and seismogenic thickness
    Derive multiple magnitude estimates:
    - M_{Mo}: from moment equation
    - M_{ASP}: from aspect ratio
    - $M_{\text{RLD}}, M_{\text{RA}}$: from scaling laws via `coeff2mag()`
    if observed magnitude M_{obs} exists **then**
        Conflate M_{obs} with other estimates using `conflate_pdfs()`
        Adjust standard deviation using ζ, ξ
    end if
    Compute $M_{\max}$ and $\sigma_{M_{\max}}$ from conflated PDF
    Compute T_{mean} using moment rate and geometry
    Store $M_{\max}, \sigma_{M_{\max}}, T_{\text{mean}}$ in fault dictionary
    Plot magnitude PDFs and save figures
end for

Activity Rate Modeling (sactivityrate):
for all faults with computed $M_{\max}$ and T_{mean} **do**
    Generate magnitude range: $[M_{\max} - \sigma, M_{\max} + \sigma]$
    Compute moment distribution and normalize PDF
    Calculate total moment and scale to match moment rate
    Derive cumulative rate: $T_m = 1/\lambda$
    if Telapsed is available **then**
        Compute time-dependent probability H_{BPT} using inverse Gaussian
    else
        Use Poissonian probability H_{Pois}
    end if
end for

Distribution Handling:
if Fault behavior is "Characteristic Gaussian" **then**
    if Telapsed exists **then**
        Call `CHGaussBPT()` to compute BPT hazard and export XML
    else
        Call `CHGaussPoiss()` to compute Poisson hazard and export XML
    end if
else if Fault behavior is "Truncated Gutenberg Richter" **then**
    Call `TruncatedGR()` for incremental and cumulative rates
else
    Log warning about unknown fault behavior
end if

2.2. Software architecture

The algorithm in (1), together with the conceptual diagram in Fig. (1), outlines the sequential workflow of the FQSHA software, including its inputs, outputs, and the core functions invoked at each stage. In parallel, Algorithm (2) details the implementation of the seismic activity rate (SAR) modules within FQSHA. Input data and optional parameters provided via the GUI are used to compute $M_{\max}$ and $\sigma_{M_{\max}}$ through conflated probability density functions, as well as T_{mean} based on the moment rate. The subsequent step utilizes the `sactivityrate` module to calculate the total moment and redistribute it across the magnitude range between $M_{\min}$ and $M_{\max}$ for SAR computation. These outputs are then passed to the OpenQuake engine to generate seismic hazard maps corresponding to 10% and 2% probabilities of exceedance (PoE). Finally, the results are visualized using the GMT Python toolkit, which produces detailed hazard map plots.

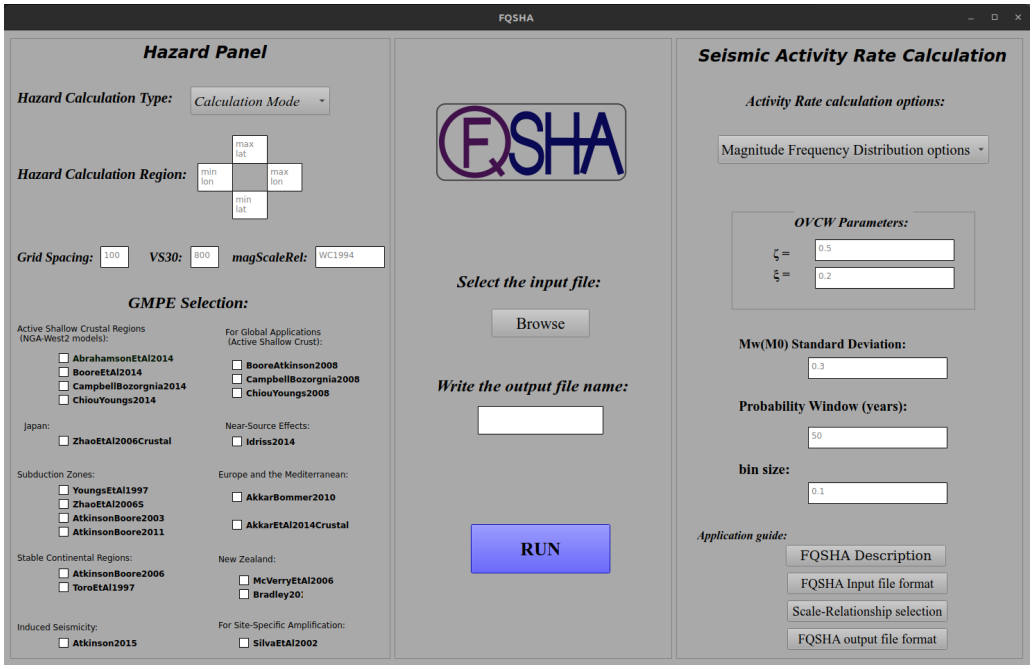


Figure 2: The GUI of FQSHA, displaying the main menus of the software. The left panel shows the input parameters for hazard calculation, including a section for defining the geographical extent for seismic hazard computation, grid spacing, Vs30 values, and the selection of various GMPEs to be incorporated into the logic tree model. The middle panel provides options to browse for the JSON input file and specify the output file name for saving the hazard results. The right panel presents the input parameters for the SAR (Seismic Activity Rate) calculation.

3. Illustrative example

The input selection of the application and the functionality is further demonstrated and evaluated through an operational simple case study. Furthermore, the outputs generated by the software is presented. These illustrations include:

- The necessary information and format for the input.
- Defining the inputs and selecting the options in the GUI.
- The intermediate outputs generated by applying the SAR calculation step.
- The generated file structure, necessary for hazard calculation.

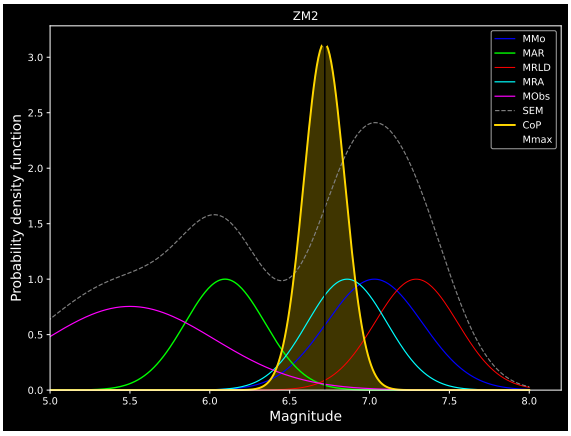
- The generated output hazard maps for a sample study area.

By executing the main code, the graphical user interface (GUI) (Fig. 2)facilitates user input for the fault geometry and kinematic parameters through the utilization of a JSON file (Table. 1). The JSON input file (Table.(1)) supplies the following parameters: ScR, year-for-calculations, Length, Dip, Seismogenic-Thickness, SRmin, SRmax, Mobs, sdMobs, Last-eq-time, SCC, ShearModulus, StrainDrop, Mmin, and b-value, along with the trace of each fault, which consists of a set of coordinate point pairs (Lat, Long) defining the different segments of the modeled fault.
In addition, the GUI includes intuitive buttons and icons that allow users to configure parameters related to both seismic activity rate calculation and seismic hazard assessment.

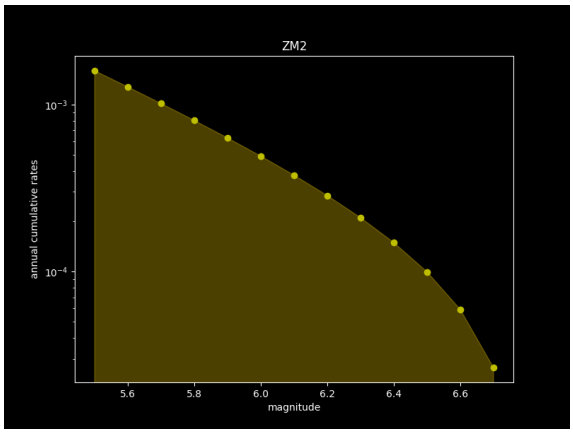
Table 1: Input Fault data in JSON format.

```
{
    "ZFF": {
        "ScR": "WC94-R",
        "year_for_calculations": 2024,
        "Length": 109,
        "Dip": 40,
        "upperSeismoDepth": 3,
        "lowerSeismoDepth": 10,
        "SRmin": 1.36,
        "SRmax": 2.04,
        "Mobs": 6.4,
        "sdMobs": 0.05,
        "Last_eq_time": 1497,
        "SCC": 0.205,
        "ShearModulus": 3,
        "StrainDrop": 3,
        "Mmin": 5.5,
        "b-value": 0.9,
        "fault_trace": [
            [56.8364, 27.3840],
            [56.7842, 27.3923],
            ...
        ]
    },
    "ZM1": {
        "ScR": "WC94-R",
        "year_for_calculations": 2024,
        "Length": 60,
        "Dip": 70,
        "upperSeismoDepth": 2,
        "lowerSeismoDepth": 12,
        "SRmin": 2.2,
        "SRmax": 4.3,
        "Mobs": 5.5,
        "sdMobs": 0.05,
        "Last_eq_time": 1950,
        "SCC": 0.205,
        "ShearModulus": 3,
        "StrainDrop": 3,
        "Mmin": 5.5,
        "b-value": 0.9,
        "fault_trace": [
            [56.9531, 27.6688],
            [56.9761, 27.6329],
            ...
        ]
    }
}
```

The left panel allows users to define parameters for seismic activity rate computation, including ζ, ξ, standard deviation values, and the Magnitude-Frequency Distribution (MFD) balancing parameters such as the probability window and bin size (see [4] for more details). The right panel is dedicated to seismic hazard assessment inputs. Here, users can specify the hazard calculation type, define the geographical region of interest for grid-based computation, input the shear-wave velocity (V_{s30}), select the magnitude scale relationship, and choose the appropriate Ground Motion Prediction Equations (GMPE) based on the characteristics of the study area.



(a) The illustration of potential maximum magnitudes estimations and the final result calculated by conflation methods.



(b) Annual cumulative activity rates of the sample fault.

Figure 3: Two of the outputs by the FQSHA software to monitor the fault behavior and the activity rates based on the MFDs on faults.

The intermediate outputs generated by FQSHA are: the plot of empirical magnitudes and the observed magnitude with the calculated M_{max} by CoP (Fig. 3a), the SAR values of each fault and their corresponding probability curves plots (Fig. 3b), and the source file containing the OpenQuake formatted fault source XML file (Fig. 4).

```
1  xml xmlns="1.0" encoding="utf-8"?>
2  <xnml xmlns="http://opengkorg/xmlns/v0.4' xml:nx=q.opengis,net/gml">
3  <sourceModel name="ZM2" nane="ZM2 Source" present
     <simpleFaultGeometry>
       <gml:PosList id="ZM2" 'ZM2 Source" tectonicReg ion="Acclave Crust">
         <gml.PosList>
           <gnl:LineStringetry>
             57.187168997067 25.949343 26.99334 265934 57.188108 26.993308 27.0284 7.8344
             57.178816707832 25.188816 27.09374 27.0946 76 187230 27.143491 27.5559 1.4525
             57.142646242904 25.618706 27.55840 27.5431 56.982703 27.155902 27.4588 8.5599
             56.838917275599 26.836308 27.37893 27.4159 56.833649 27.551920 27.5559 0.8489
           </gml.LineString>
         </dip>70
         <upperSeismoDepth>12.0uppeSeismoDepth>
         <lowerSeismoDepth>20.0linestevpt>
         <magScaleRel>wC1994>
         <ruptAspectRatio>2.00000000E+0.00>
         <incrementalMFD minMag>5.5 bith>
5  5.390514e-04 4.381574e<04  3.661574e-0344 360514e-04  5.390514e 8.348917e-05
6        <occurKaw5.390514e-04
7  </incrementalMFD>
8  <rake>80>
9  <simpleFaultSource>
10 </nrml>
```

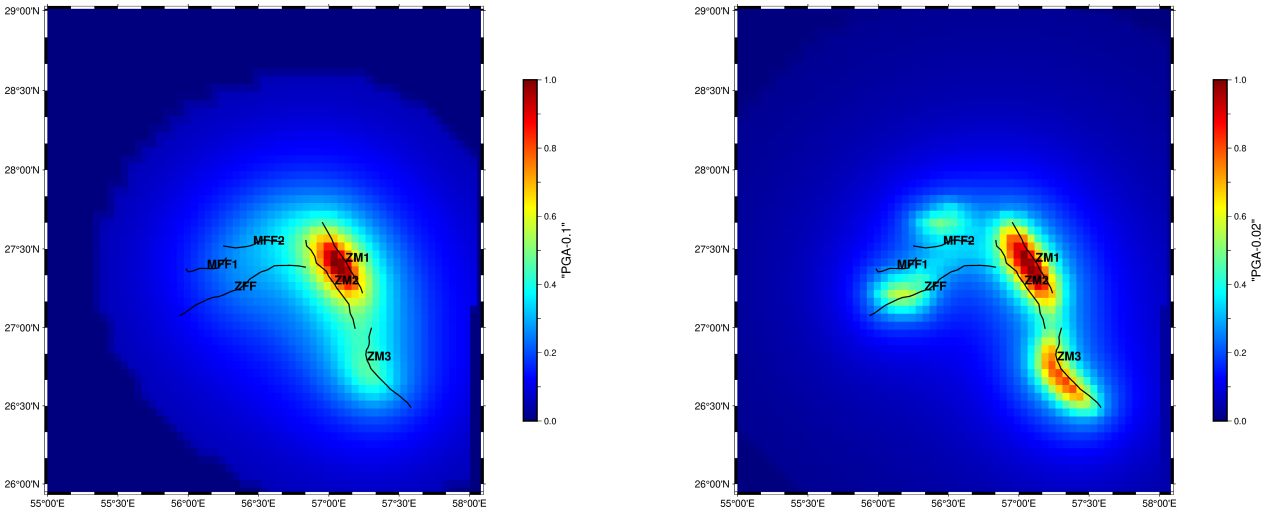
Figure 4: The sample XML output file that can be used by OpenQuake engin for the hazrd calculation.

Based on the inputs provided by the user, in this step it generates the file structure necessary

for hazard calculation by integrating the the input information from the GUI:

```
FQSHA_output/
|-- Sources/
|-- fault_traces.json
|-- gmpe_logic_tree.xml
|-- inputs.json
|-- job.ini
|-- source_model_logic_tree.xml
```

The calculated hazard values are plotted in the defined geographical extent using the GMT toolkit for 10% (Fig. (5a)) and 2% Probabilities of Exceedance (PoE) (Fig. (5b)).



(a) Output hazard map for sample faults with a 10% Probability of Exceedance.

(b) Output hazard map for sample faults with a 2% Probability of Exceedance.

Figure 5: Generated Hazard map by FQSHA.

4. Software Testing and Validation

To ensure the correctness and reliability of FQSHA's core functionalities, we employed Python's built-in `unittest` framework [10] along with the `coverage.py` tool for code coverage analysis. These approaches enhance software stability and minimize the risk of errors and according to [11] is effective in validating Python-based scientific software components.

4.1. Unit Testing for Core Functions

Unit testing was applied to critical functions within FQSHA, particularly those responsible for SAR calculations and input/output data handling. The `pytest` framework was utilized to automate testing, enabling verification of expected outputs under various input conditions. All test cases are implemented as Python classes inheriting from `unittest.TestCase`, and they can be executed using the `unittest` test runner (e.g., via `python -m unittest tests/test_sample.py -v`).

4.2. Automated GUI Testing with pytest-qt

For GUI testing in PyQt5 applications, the `pytest-qt` plugin extends `pytest` to support interaction-based testing. This tool allows developers to simulate user actions, verify UI

Table 2: Code coverage summary for the core functions and data handling functions.

File	Statements	Missed	Coverage
`tests/test_sactivityrate_xmlExport.py`	60	4	93%
Total	**60**	**4**	**93%**

responsiveness, and automate regression testing to maintain software stability across updates [12].

To validate the functionality and stability of the graphical user interface (GUI), automated testing was implemented using the `pytest-qt` framework [12]. Automated GUI testing is particularly beneficial for PyQt5-based applications, as it allows developers to validate the behavior of interactive components without requiring manual testing. These tests simulate user interactions, such as selecting parameters, clicking buttons, and triggering computations, to ensure the GUI responds as expected. The use of `pytest-qt` enhances the overall software robustness by ensuring that visual elements and computational outputs remain consistent across different user interactions.

Table 3: Code coverage summary for the gui test code.

File	Statements	Missed	Coverage
`tests/test_gui.py`	76	3	96%
Total	**76**	**3**	**96%**

4.3. Code Coverage Analysis

To assess the thoroughness of our testing efforts, we used the `coverage.py` tool [13]. This tool measures the percentage of source code executed by the test suite, helping to identify untested code and improve overall robustness.

Our current implementation achieves approximately **96%** code coverage for the gui (2) and **96%** for the tested functions (3). The tool provides detailed reports including statement coverage and highlights untested blocks in an interactive HTML format. Hilton et al. [14] emphasize the importance of coverage analysis for quality assurance in Python, particularly in identifying structural gaps in test suites.

All test scripts and configurations are included in the source code repository. The tests can be executed in any Python environment with minimal setup and can be automatically run using the `coverage.py` test runner (e.g., via `coverage run -m unittest tests/test_guiii.py -v`).

5. Impact

The FQSHA tool serves as a comprehensive and streamlined framework for fault-based seismic hazard analysis, transforming raw fault input data into final hazard estimates with minimal user intervention. By integrating the Seismic Activity Rate (SAR) computation engine (FaultQuake) with the widely used hazard calculation platform (OpenQuake), FQSHA provides a unified environment tailored for Probabilistic Seismic Hazard Assessment (PSHA) directly from fault sources. The tool is fully automated and designed with usability in mind, making it accessible to both experts and new users to the geoscience community. Given its ease of use and efficiency in addressing a critical aspect of seismic hazard assessment, we

expect it to attract a broad range of users and applications. Its utility and user-centered design are expected to encourage frequent use, broad adoption, and noticeable impact within the field.

6. Conclusions

The development of FQSHA is motivated by the growing demand for a transparent and accessible tool to support research and education in fault-based PSHA. Designed to streamline the modeling process, FQSHA enables users to carry out fault-based seismic hazard assessments with minimal manual intervention. It features an intuitive graphical interface for computing fault-based SAR, which is directly coupled with the state-of-the-art OpenQuake engine for hazard computations and the GMT toolkit for visualizing the resulting hazard maps. Owing to its modular and extensible design, the framework is well-suited for continuous development. Future enhancements will include the integration of advanced SAR models and hazard calculation methodologies, further broadening the tool's applicability and robustness in seismic hazard research.

CRediT authorship contribution statement

Nasrin Tavakolizadeh: Methodology, Software, Writing and Editing.
Hamzeh Mohammadigheymasi: Methodology, Validation, Review and Editing, Supervision.
Nuno Pombo: Validation, Review and Editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my codes on GitHub:
https://github.com/GeoSignalAnalysis/FQSHA/

Acknowledgements

This work is funded by FCT/MCTES through national funds and when applicable co-funded by FEDER—PT2020 partnership agreement under the project UIDB/EEA/50008/2020.

References

[1] B. Pace, F. Visini, and L. Peruzza, "Fish: Matlab tools to turn fault data into seismic-hazard models," *Seismological Research Letters*, vol. 87, no. 2A, pp. 374–386, 2016.

[2] T. Chartier, O. Scotti, and H. Lyon-Caen, "Sherifs: Open-source code for computing earthquake rates in fault systems and constructing hazard models," *Seismological Research Letters*, vol. 90, no. 4, pp. 1678–1688, 2019.

[3] F. Visini, A. Valentini, T. Chartier, O. Scotti, and B. Pace, "Computational tools for relaxing the fault segmentation in probabilistic seismic hazard modelling in complex fault systems," *Pure and Applied Geophysics*, vol. 177, pp. 1855–1877, 2020.

[4] N. Tavakolizadeh, H. Mohammadigheymasi, F. Visini, and N. Pombo, "Faultquake: An open-source python tool for estimating seismic activity rates in faults," *Computers & Geosciences*, vol. 191, p. 105659, 2024.

[5] N. Tavakolizadeh, H. Mohammadigheymasi, L. Matias, G. Silveira, R. Fernandes, and N. Dolatabadi, "To what extent do slip rates contribute to the seismic activity of faults?" in *EGU General Assembly Conference Abstracts*, 2022, pp. EGU22–12 893.

[6] M. Pagani, D. Monelli, G. Weatherill, L. Danciu, H. Crowley, V. Silva, P. Henshaw, L. Butler, M. Nastasi, L. Panzeri, M. Simionato, and D. Vigano, "The openquake engine: An open hazard (and risk) software for the global earthquake model," *Seismological Research Letters*, vol. 85, no. 3, pp. 692–702, 2014.

[7] E. H. Field, D. D. Jackson, and J. F. Dolan, "A mutually consistent seismic-hazard source model for southern california," *Bulletin of the Seismological Society of America*, vol. 89, no. 3, pp. 559–578, 1999.

[8] M. Pagani, V. Silva *et al.*, "Openquake-engine documentation, v3.10," https://docs.openquake.org/, 2020, accessed: 2025-04-14.

[9] V. Silva, H. Crowley, and M. Pagani, "Openquake-engine: An open-source software for seismic hazard and risk modeling in developing countries," *Earthquake Spectra*, vol. 34, no. 3, pp. 1319–1337, 2018.

[10] P. S. Foundation, *unittest — Unit testing framework*, 2024, python Standard Library, Version 3.12. [Online]. Available: https://docs.python.org/3/library/unittest.html

[11] M. Gruber, S. Lukasczyk, F. Kroiß, and G. Fraser, "An empirical study of automated unit test generation for python," *Empirical Software Engineering*, vol. 27, no. 5, 2022.

[12] pytest-qt contributors, "pytest-qt: Pytest plugin for qt and PySide/PyQt applications," https://pypi.org/project/pytest-qt/, 2025, accessed: 2025-04-11.

[13] N. Batchelder, *coverage.py: Code coverage measurement for Python*, 2024, version 7.4.0. [Online]. Available: https://coverage.readthedocs.io/

[14] M. Hilton, J. Bell, and D. Marinov, "Test coverage in python programs," in *Proceedings of the 16th International Conference on Mining Software Repositories*. IEEE, 2019, pp. 149–159.