USABILITY GAPS IDENTIFIED FROM REVIEWING PYGIMLI'S ERT MODULE

To understand the problem involved in bridging the gap between enhancing pyGIMLi's usability in teaching, field-based research and rapid data analysis contexts by creating an ERT-specific user-friendly GUI. It is imperative to consider a critical review of the usability gaps in the ERT module.

Following a careful review of the ERT module workflow in the inversion software which provides tools for forward modelling (prediction of electrical resistivity geophysical response from a known subsurface model), inversion (estimation of the values of geological model parameters such as size and depth from a set of observed geophysical data), data handling (data integrity, application of specific processing algorithms such as filtering and migration), and visualisation of electrical resistivity data. Based on documentation, user workflows, and comparisons with similar open-source software tools e.g., BERT and ResIPy, the identified usability gaps generally result from pyGIMLi's script-based nature, technical requirements, and performance characteristics. They are further categorised into:

- Learning Curve and Accessibility
- Workflow Efficiency and Interactivity
- Documentation and Community Support
- Performance and Scalability

LEARNING CURVE AND ACCESSIBILITY

Tasks involved in Electrical Resistivity Tomography e.g., mesh generation, data importation, or inversion setup all require proficiency in Python scripting for all workflows. Core workflows typically rely on classes like *ERTManager* and functions such as *createMesh()* or *invert()*, are all executed via scripts.

There is no built-in support for non-programmers; pyGIMLi's ERT module users typically have to write code for even the most basic tasks like electrode placement or parameter tuning. This is evident in reviews that note that pyGIMLi is designed for technical users with scripting/programming experience contrasting with tools like ResIPy that offer a GUI for similar ERT basic tasks (Doyoro et al., 2022).

These identified gaps technically result in a steep entry barrier for geophysicists or field technicians with zero coding experience. Even documentation resources emphasize script-based examples, with zero point-and-click options. Hence, users have to understand inversion theory, mesh discretisation, and Jacobians before they can be productive.

WORKFLOW EFFICIENCY AND INTERACTIVITY

There is installation and environment friction which generally results in "*works on my machine*" syndrome further leading to reproducibility issues across several machines. Version incompatibility between pyGIMLi and required python data manipulation libraries such as *numpy*, *scipy*, and *matplotlib* e.t.c. is poorly communicated.

There is also a lack of interactive tools for visual setup e.g., dragging electrodes, previewing meshes, or real-time parameter adjustments. Complex steps like mesh

refinement, error estimation, and regularisation typically require manual handling which can lead to trial-and-error iterations. This is evident in *createParaMeshPLC* which requires manual geometry definitions and quality checks as poor meshes can result in inversion failures (Pygimli.Meshtools — pyGIMLi - Geophysical Inversion and Modelling Library, n.d.).

Finally, there is poor error feedback mechanism without diving deeply into code diagnostics.

DOCUMENTATION AND COMMUNITY SUPPORT

While pyGIMLi's documentation is comprehensive and detailed, it is code heavy; examples provided are script-focused with limited beginner-friendly tutorials. This is evident from the provided API references on the website which assume Python programming language familiarity (PyGIMLI API Reference — PYGIMLI - Geophysical Inversion and Modelling Library, n.d.). Usability feedback emphasizes the need for more reproducible, user-friendly workflows.

There is no integrated help for common pitfalls like region decoupling or over-fitting.

DISCUSSION

Would a Graphical User Interface (GUI) significantly resolve these identified gaps?

Identified gaps related to learning curve, accessibility, workflow efficiency, and interactivity can be addressed by a user-friendly GUI by shifting from code-centric to

visual, interactive paradigms. However, gaps related to performance and scalability may not be fully resolved by a GUI.

A GUI would significantly resolve the identified primary usability gaps by making pyGIMLi more intuitive and accessible, potentially expanding its user base beyond programmers to include even field practitioners and students. Evidence from similar tools like the adoption of ResIPy's GUI for ERT in educational settings supports this (Blanchy et al., 2020). A GUI could handle 60-70% of common workflows visually, with code fallback for programming experts. In short, in cases where the overall goal is broader adoption of the pyGIMLi inversion software across teaching, field-based research, and rapid data analysis contexts, implementing a GUI is a high-impact addition as script-only interfaces are noted as barriers in geophysical software reviews.

How would a GUI resolve gaps related to Learning Curve and Accessibility?

Implementation of a GUI would significantly impact learning curve and accessibility as a GUI would democratize access similar to how ResIPy uses a GUI to appeal to non-coders/programmers. A GUI can enable drag-and-drop electrode placement, visual geometry building (e.g., importing topography maps), and wizard-style workflows for inversion setup, reducing the need for coding with Python. Beginners can also load data files, select configurations, and run inversions by simply using buttons/sliders and this would significantly lower the barrier for geophysicists in fieldwork or education.

How would a GUI resolve gaps related to Workflow Efficiency and Interactivity?

Implementation of a GUI would impact workflow efficiency and interactivity as a GUI could cut setup time dramatically, making trial-and-error more intuitive. Visual diagnostics like clickable misfit maps would also improve error handling. Interactive previews (e.g., real-time mesh visualisation, parameter sliders with instant feedback on coverage/misfit), automated error checking, and step-by-step guides could streamline setup and iteration. Tools for visual data preprocessing (e.g., outlier removal via graphs) would reduce manual coding.

How would a GUI resolve gaps related to Documentation and Community Support

Impact of a GUI here would be partial as a GUI would embed guidance directly in the interface, reducing over-reliance on external documentation. However, community issues (e.g., via GitHub) would still need separate handling.

The use of integrated help pop-ups, tutorial overlays, and exportable scripts for sharing could enhance learning. A GUI can also log actions as code snippets for reproducibility with the inversion software CLI-based workflow version.

How would a GUI resolve gaps related to Performance and Scalability?

Impact of a GUI here would also be partial because a GUI would make issues more visible and manageable but would not fix core inefficiencies like slow processing or resolution limits which are purely algorithmic.

In cases where there are large datasets or models which might render pyGIMLi computationally inefficient, using a GUI wouldn't fix so much in this case.

REFERENCES

Blanchy, G., Saneiyan, S., Boyd, J., McLachlan, P., & Binley, A. (2020). ResIPy, an intuitive open source software for complex geoelectrical inversion/modeling. Computers & Geosciences, 137, 104423. https://doi.org/10.1016/j.cageo.2020.104423

Doyoro, Y. G., Chang, P., Puntu, J. M., Lin, D., Van Huu, T., Rahmalia, D. A., and Shie, M. (2022). A review of open software resources in python for electrical resistivity modelling. *Geoscience Letters*, *9*(1). https://doi.org/10.1186/s40562-022-00214-1

PyGIMLI API Reference — PYGIMLI - Geophysical Inversion and Modelling Library. (n.d.). Retrieved from https://www.pygimli.org/pygimliapi/index.html

PyGIMLi.meshtools — pyGIMLi - Geophysical Inversion and Modelling Library. (n.d.). Retrieved from https://www.pygimli.org/pygimliapi/_generated/pygimli.meshtools.html#pygimli.meshtools.createParaMeshPLC