The Journal of Open Source Software

toughio: Pre- and post-processing Python library for TOUGH

Keurfon Luu[1]

1 Energy Geosciences Division, Lawrence Berkeley National Laboratory, Berkeley, CA, USA

DOI: 10.21105/joss.02412

Software

- Review
- Repository
- Archive

Editor: Jeff Gostick

Reviewers:

- @rreinecke
- @ma-sadeghi

Submitted: 16 June 2020
Published: 27 July 2020

Statement of Need

TOUGH is a widely used general purpose numerical simulator designed for fluid and heat flows of multiphase, multicomponent fluid mixtures in porous and fractured media, which has been applied to many real-world problems such as underground geological storage, geothermal reservoir engineering or nuclear waste disposal, to name a few.

When setting up a model, the most time-consuming part is preparing the input data required to run a TOUGH simulation due to its error-prone text-based fixed-format input files and the lack of proper built-in meshing tool to model complex geological structures. In addition, a TOUGH mesh is optimally represented by a Voronoi graph as TOUGH uses an integral finite-difference formulation (Narasimhan & Witherspoon, 1976) to solve the coupled fluid and heat flow equations. A TOUGH mesh is only represented as a set of elements and connections without any reference to a coordinate system usually required for post-processing with common visualization softwares (e.g. ParaView, Tecplot and VisIt).

Summary

In the recent years, many softwares have been developed and published to address the lack of pre- and post-processing features of TOUGH family of codes, mostly in the form of Graphical User Interfaces (GUIs) such as (free) TIM (Yeh, Croucher, & O'Sullivan, 2013), TOUGH2Viewer (Bondua, Berry, Bortolotti, & Cormio, 2012), (commercial) Leapfrog (Newson et al., 2012), mView (Avis, Calder, Walsh, & Engineering, 2012) or PetraSim (Yamamoto, 2008). While GUIs provide a convenient integrated working environment since they do not require to have any programming knowledge, users are often limited to the features implemented in the softwares and pre- and post-processing outputs are hardly reproducible due to closed or proprietary formats. Besides, automation of runs or coupled simulations (e.g. with a mechanical simulator) cannot be carried out through a GUI. All of the aforementioned issues can be addressed by using a high level scripting language such as Python.

`toughio` is a lightweight, object-oriented and vectorized Python library that aims to provide user-friendly routines to facilitate pre- and post-processing of a TOUGH simulation. Currently, to the best of our knowledge, only PyTOUGH (Croucher, 2011) offers an exhaustive list of features to carry out a complete TOUGH simulation using a scripting language. `toughio` and PyTOUGH share the same objectives, yet with different approaches. On the one hand, a PyTOUGH mesh is represented as a *MULGRAPH* geometry where elements can be unstructured horizontally but only layered vertically (usually referred to as 2.5D). On the other hand, although it provides basic meshing features, `toughio` mostly relies on common third-party softwares (e.g. Abaqus, FLAC3D, Gmsh (Geuzaine & Remacle, 2009), LaGriT) to generate the mesh by importing and converting it to a TOUGH mesh, which also conveniently facilitates the coupling of TOUGH with any other simulator that also supports the same

mesh formats. In addition, `toughio` mainly targets the latest version TOUGH3 (Jung, Pau, Finsterle, & Pollyea, 2017) and supports most of its new features such as the new input data blocks, the new output formats, and variable length element names. Nevertheless, `toughio` is backward compatible with TOUGH2 (Pruess, Oldenburg, & Moridis, 2012) and can read/write TOUGH2 input/output files.

Figure 1 shows the result of a sample CO2 sequestration simulation where supercritical CO2 is continuously injected during 3 years in a reservoir near a fault modeled as a finite-thickness element with high permeability. The model has been entirely set up using `toughio` with a mesh generated by Gmsh imported and pre-processed in Python thanks to `meshio` (Schlömer et al., 2020). The conversion from a finite-element mesh to its dual-graph representation as required by TOUGH is automatically handled by `toughio` when exporting the mesh for TOUGH. Outputs of the TOUGH simulation have been imported and remapped into the original finite-element grid and directly visualized in Python thanks to `pyvista` (Sullivan & Kaszynski, 2019).

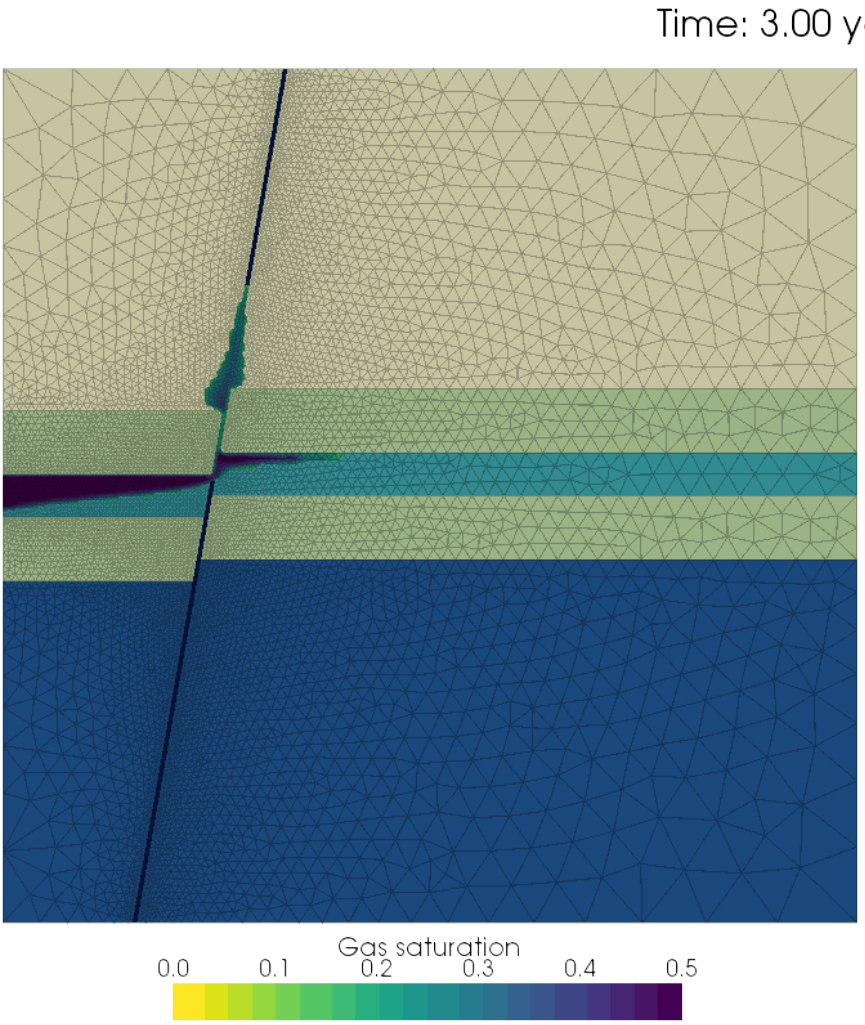

Figure 1: Example of simulation of CO2 upward leakage along a fault completely developed with `toughio`. Mesh has been generated with Gmsh and imported in Python by `meshio`. Output figure has been prepared and exported by `pyvista`.

`toughio` offers a complete set of features to pre- and post-process a TOUGH simulation in Python. Finite-element meshes generated by third-party softwares can be imported and

converted to a Voronoi graph for TOUGH, simulation parameters can be defined using a human-readable and jsonable dictionary automatically converted to a fixed-format input file for TOUGH, and simulation results can be imported in Python for post-processing and visualization.

Acknowledgements

This material is based upon work supported by Lawrence Berkeley National Laboratory under U.S. Department of Energy Award No. DE-AC02-05CH11231.

References

Avis, J., Calder, N., Walsh, R., & Engineering, G. (2012). mView - A powerful pre- and post-processor for TOUGH2, 8.

Bondua, S., Berry, P., Bortolotti, V., & Cormio, C. (2012). TOUGH2Viewer: A post-processing tool for interactive 3D visualization of locally refined unstructured grids for TOUGH2. *Computers & Geosciences*, *46*, 107–118. doi:10.1016/j.cageo.2012.04.008

Croucher, A. (2011). PyTOUGH: A Python scripting library for automating TOUGH2 simulations. *New Zealand*, 6.

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. doi:10.1002/nme.2579

Jung, Y., Pau, G. S. H., Finsterle, S., & Pollyea, R. M. (2017). TOUGH3: A new efficient version of the TOUGH suite of multiphase flow and transport simulators. *Computers & Geosciences*, *108*, 2–7. doi:10.1016/j.cageo.2016.09.009

Narasimhan, T. N., & Witherspoon, P. A. (1976). An integrated finite difference method for analyzing fluid flow in porous media. *Water Resources Research*, *12*(1), 57–64. doi:10.1029/WR012i001p00057

Newson, J., Mannington, W., Sepulveda, F., Lane, R., Pascoe, R., Clearwater, E., & O'Sullivan, M. J. (2012). Application of 3D modelling and visualization software to reservoir simulation: Leapfrog Geothermal and TOUGH2, 6.

Pruess, K., Oldenburg, C., & Moridis, G. (2012). TOUGH2 User's guide, (September). Retrieved from http://esd.lbl.gov/TOUGHPLUS/manuals/TOUGH2_V2_Users_Guide.pdf

Schlömer, N., McBain, G., Luu, K., Li, T., Tsolakis, C., Mataix Ferrándiz, V., Barnes, C., et al. (2020). *Nschloe/meshio v4.0.15*. Zenodo. doi:10.5281/zenodo.3888325

Sullivan, C., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. doi:10.21105/joss.01450

Yamamoto, H. (2008). PetraSim : A Graphical User Interface for the TOUGH2 Family of Multiphase Flow and Transport Codes. *Ground Water*, *46*(4), 525–528. doi:10.1111/j.1745-6584.2008.00462.x

Yeh, A., Croucher, A. E., & O'Sullivan, M. J. (2013). Tim – Yet Another Graphical Tool for TOUGH2. *35th New Zealand Geothermal Workshop: 2013 Proceedings*, (November).