The Journal of Open Source Software

PyToughReact – A Python Package for automating reactive transport and biodegradation simulations.

Temitope Ajayi[*1] **and Ipsita Gupta**[1¶]

1 Louisiana State Univeristy, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 21 March 2022
Published: unpublished

Statement of need

The suite of TOUGH simulators by the Lawrence Berkeley National Laboratory (LBNL) are well known simulators for flow and transport simulations. In order to model chemical reactions coupled to flow for isothermal and non-isothermal systems, the TOUGHREACT simulator exists. In addition, the TMVOC software exists for the modeling of multicomponent mixtures of volatile organic compounds suitable for contamination problems. When biodegradation is added to the process, the tool used is known as TMVOC-BIO. These reaction simulators as is now do not provide a tool to automatically run many simulations for tasks such as uncertainty quantification or sensitivity analysis. Users need to create many simulation folders to run every sensitivity they intend to run with the simulator. This makes it cumbersome to use and users could get lost on the purpose of each folder if not correctly labeled. While PyTOUGH and TOUGHIO exists for regular flow and transport simulations, no such tool exists for chemical and biodegradation reactions. `PyToughReact` fulfills this need for users familiar with python.

Summary

TOUGH based simulators consist of simulators that require the user to write or modify a text file and parse the file to an executable to run the simulation. Results from simulations are subsequently also saved to text files for interpretation by third party softwares. These makes it cumbersome for sensitivity analysis and uncertainty studies to be conducted as this would require a user to make multiple edits to files and manually extract required data. As would be imagined, this is also prone to human errors. Further, coupling the simulator with other simulators would be more involving. As a result of this, numerous tools have been created that wrap around the executables. Examples of such include PetraSim (Yamamoto, 2008) for running and viewing simulation results, TOUGH2VIEWER (Bondua et al., 2012) and TECPLOT (Tecplot, 2013) for postprocessing and visualizations and IGMESH for both building and integrating visualization tools suited especially for irregular gridding (Hu et al., 2016). For scripting purposes, PyTOUGH (Croucher, 2011) and TOUGHIO (Luu, 2020) are used for pre and post processing of TOUGH2 simulations. So far, no tool exists for scripting of the reaction softwares of TOUGH. Reactive processes as with other processes are subject to uncertainties and thus need to be adequately accounted for in engineering studies. To enable this with the TOUGHREACT simulator, it is essential to create a scripting tool to accomplish this. This tool extends the work done by PyTOUGH in creating an automatic platform for running TOUGH simulations by creating a concurrent tool for automatically running chemical and biodegradation reactions from any python enabled terminal or IDE. The software is partitioned into two parts for either running biodegradation or chemical reactions. In the biodegradation partition, a t2bio file/class is created containing classes and functions which perform the heavy lifting of reading and writing to files. To complement this file, the t2component file is

[*] Co-first author

created to read in components and biomasses while the t2process file is available to create biodegradation processes. In the reaction component, a t2react file is created to write and read information into the flow.inp present in TOUGHREACT. In addition, this file modifies the t2grid file from PYTOUGH to t2reactgrid in PYTOUGHREACT to account for chemical zones present in reactive domains. In addition, a t2chemical file exists to read and write in results and date from and into the chemical.inp file and a t2solute file to read and write in results and date from and into the solute.inp file. New classes are also created in the reaction components that are essential for successfully writing to a file. The classes created are the Water and WaterComp classes for the composition of fluids and fluid zones, PrimarySpecies class for the primary species present in the domain, Mineral and MineralComp classes to account for minerals and mineral zones, Dissolution and Precipitation classes to account for dissolution and precipitation of minerals, pHDependenceType1 and pHDependenceType2 classes to account for pH dependence, ReactGas class for gases in the domain, PermPoro and PermPoroZone classes for permeability-porosity relationships and zones. For obtaining results from the simulations, a t2result class is used to achieve this objective. The t2result class contains functions used to automatically retrieve data from the simulation for use in python. The simulator can thereafter be coupled with other tools using python to achieve various tasks.

An example of a full biodegradation run file simulation is shown below

```
import numpy as np
import pytoughreact as pyt
from pytoughreact import mulgrid, t2grid, Component, Gas, Water_Bio, Biomass, Process, B

second = 1
minute = 60 * second
hour = 60 * minute
day = 24 * hour
year = 365. * day
year = float(year)
simtime = 1 * year

length = 9
xblock = 3
yblock = 1
zblock = 4
dx = [length / xblock] * xblock
dy = [0.1]
dz = [2] * zblock
geo = mulgrid().rectangular(dx, dy, dz, origin=[0, 0, -100])
geo.write('geom.dat')

bio = pyt.t2bio()
bio.title = 'Biodegradation Runs'

bio.grid = t2grid().fromgeo(geo)
bio.grid.delete_rocktype('dfalt')
shale = rocktype('shale', 0, 2600, 0.67, [6.51e-14, 6.51e-14, 6.51e-14], 1.5, 900)
bio.grid.add_rocktype(shale)

for blk in bio.grid.blocklist[0:]:
    blk.rocktype = bio.grid.rocktype[shale.name]

bio.multi = {'num_components': 3, 'num_equations': 3, 'num_phases': 3,
             'num_secondary_parameters': 8}
```

```python
bio.parameter.update(
    {'print_level': 3,
     'max_timesteps': 9999,
     'tstop': simtime,
     'const_timestep': 1.,
     'print_interval': 1,
     'gravity': 9.81,
     'option': np.array([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
     'relative_error': 1e-5,
     'phase_index': 2,
     'default_incons': [9.57e+05, 0, 0, 0, 0, 0, 0, 1e-3, 10.]})

# ,

bio.start = True
toluene = Component(1).defaultToluene()
bio.components = [toluene]

O2_gas = Gas('O2', 2)
bio.gas = [O2_gas]

water = Water_Bio('H2O')

biomass = Biomass(1, 'biom', 0.153, 1.00e-6, 30, 0, 0.e-6)
oxygen_ks = 0.5e-6
oxygen_uptake = 1
water_uptake = -3

process1 = Process(biomass, 2, 1.6944e-04, 0.58, 0)
water.addToProcess(process1, water_uptake)
O2_gas.addToProcess(process1, oxygen_uptake, oxygen_ks)
toluene.addToProcess(process1, 1, 7.4625e-06)

biodegradation = BIODG(0, 1e-5, 0, 0.2, 0.9, 0.9,
                       [process1],
                       [biomass])
bio.biodg = [biodegradation]

bio.diffusion = [
    [2.e-5, 6.e-10, 6.e-10],
    [2.e-5, 6.e-10, 6.e-10],
    [2.e-5, 6.e-10, 6.e-10]
]

bio.write('INFILE')
bio.run('tmvoc.exe')
```

Acknowledgements

Funding for this research is from the National Academy of Sciences, Engineering, and Medicine (NASEM) Gulf Research Program (GRP) grant on “Mitigating Risks to Hydrocarbon Release through Integrative Advanced Materials for Wellbore Plugging and Remediation” under award number 200008863 and the Early Career Research Fellowship of Ipsita Gupta.

References

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

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

Hu, L., Zhang, K., Cao, X., Li, Y., & Guo, C. (2016). A convenient irregular-grid-based pre-and post-processing tool for TOUGH2 simulator. *Computers & Geosciences*. https://doi.org/10.1016/j.cageo.2016.06.014

Luu, K. (2020). toughio: Pre- and post-processing Python library for TOUGH. *Journal of Open Source Software*. https://doi.org/10.21105/joss.02412

Tecplot. (2013). *Tecplot360 user's manual*. Tecplot.

Yamamoto, H. (2008). PetraSim: A graphical user interface for the TOUGH2 family of multiphase flow and transport codes. *Groundwater*. https://doi.org/10.1111/j.1745-6584.2008.00462.x