Advances in Geo-Energy Research

Vol. 8, No. 3, p. 206-210, 2023

Short communication

APyCE: A Python module for parsing and visualizing 3D reservoir digital twin models

Mateus Tosta[1], Gustavo P. Oliveira[1], Bin Wang[2]*, Zhiming Chen[2], Qinzhuo Liao[2]

[1]*TRIL Lab, Department of Scientific Computing, Federal University of Paraíba, João Pessoa 58000-000, Brazil*

[2]*National Key Laboratory of Petroleum Resources and Engineering, China University of Petroleum, Beijing 102249, P. R. China*

Keywords:
Reservoir modeling
grid processing
3D visualization
digital twin

Cited as:
Tosta, M., Oliveira, G. P., Wang, B., Chen, Z., Liao, Q. APyCE: A Python module for parsing and visualizing 3D reservoir digital twin models. Advances in Geo-Energy Research, 2023, 8(3): 206-210.
https://doi.org/10.46690/ager.2023.06.07

Abstract:
Engineers, geoscientists, and analysts can benefit from fast, easy, and real-time immersive 3D visualization to enhance their understanding and collaboration in a virtual 3D world. However, converting 3D reservoir data formats between different software programs and open-source standards can be challenging due to the complexity of programming and discrepancies in internal data structures. This paper introduces an open-source Python implementation focused on parsing industry reservoir data formats into a popular open-source visualization data format, Visual Toolkit files. Using object-oriented programming, a simple workflow was developed to export corner-point grids to Visual Toolkit-hexahedron structures. To demonstrate the utility of the software, standard raw input files of reservoir models are processed and visualized using Paraview. This tool aims to accelerate the digital transformation of the oil and gas industry in terms of 3D digital content generation and collaboration.

1. Introduction

Data visualization resources are indispensable for any software intended to handle 3D models of oil and gas (O&G) reservoirs. As never seen before, the groundbreaking power of computer graphics and scientific visualization have brought endless capabilities for geoscientists, engineers, and analysts to understand subsurface phenomena. Multidimensional views greatly improve seismic interpretation, reservoir characterization, and real-time monitoring of producing fields, since high-resolution images and 3D models can reconstitute the physical world almost perfectly (Masison et al., 2021).

Despite the rapid spread of new paradigms that head up the O&G digital transformation, such as digital twins, artificial intelligence, and machine learning, numerical simulators are still distinguished assets for reproducing the multiphase dynamics of reservoir fluids, performing history matching, and optimizing well placement strategies (Sircar et al., 2022; Sun and Zhang, 2020).

Many simulators with popularity among the O&G community use discrete formulations based on corner-point grids (Ponting, 1989). Corner-point grids can be considered an international standard to represent all the geologic features found in complex reservoirs, including dips, bends, pinches, and faults. While many tools for post-processing of corner-point grids are available, most of them are independent and have their own internal semantic and object hierarchy, which prevents their full integration. Furthermore, because the interoperability of grid files (in terms of importing/exporting data) among such programs depends on complex programming routines, the representation of modeling entities and features from a given software may be mismatched when read into another.

As seen, data integration and data visualization are permanent challenges for software and algorithm developers working on solutions for the O&G sector. In particular, researchers often need visualization tools to communicate their discoveries, make reports with interactive plots, and easily generate publication-ready graphics of post-processed data.

Yandy Scientific Press

*Corresponding author.
E-mail address: metzkerrr@gmail.com (M. Tosta); gustavo.oliveira@ci.ufpb.br (G. P. Oliveira); bin.wang@cup.edu.cn (B. Wang); zhimingchn@cup.edu.cn (Z. Chen); liaoqz@cup.edu.cn (Q. Liao).

Received May 21, 2023; revised June 3, 2023; accepted June 9, 2023; available online June 12, 2023.

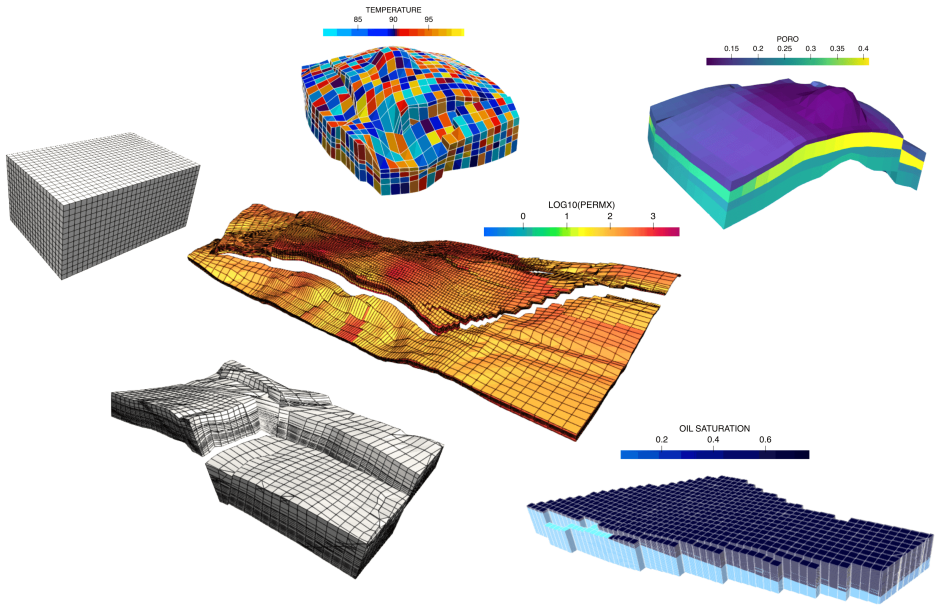

Fig. 1. Several models of reservoirs are shown and analyzed using APyCE.

A considerable number of commercial and open-source software capable to deal with corner-point grids for visualization purposes are known. Either they have embedded viewers as secondary functionality or are primarily tailored for visualization. In the first group, CMG Results© (https://www.cmgl.ca/results), Schlumberger Petrel© (https://www.software.slb.com/products/petrel), ESSS Kraken© (https://oilandgas.esss.co/technologies/kraken), and Amarile RE-Studio© (https://www.amarile.com/-RE-Studio-.html); in the second group, SINTEF's MRST (Lie, 2019) and Ceetron Solution's ResInsight (https://resinsight.org) are a few options. Minor open-source software projects have been proposed by scholars and independent scientists with an interest in O&G data, but they are usually focused on flow simulations instead of visualization and corner-point grid manipulation.

This paper introduces a lightweight Python-based package for the plotting of reservoir models discretized through Cartesian or corner-point grids called `APyCE`. The objective of this software is to provide a simple pipeline for post-processing through the PyVista module (Sullivan and Kaszynski, 2019) and Visual Toolkit (VTK) exporting for 3D visualization inside Kitware Paraview software (Ayachit et al., 2015).

`APyCE` is an enhanced version of the early project PyGRDECL (Wang, 2018), developed to handle Schlumberger Eclipse© (Schlumberger, 2014) deck files for visualization. `APyCE` is multi-platform and easy to use, being prepared to fulfill its objectives with only 4 lines of code and requiring no background in Python to be run. It is recommended for researchers who need to render high-quality figures for inclusion into scientific papers, reports, presentations, handouts, interactive notebooks, and general documents for teaching purposes.

Next, the computational framework is discussed in Section 2, examples of use are provided in Section 3, and lastly, some concluding remarks and perspectives for future development are drawn.

2. Background

`APyCe` is developed on top of Eclipse deck files' structure (Schlumberger, 2014). However, one should stress that similar topologies are found in most of the input files managed by competing software. Such files are internally divided into sections. Each section admits a broad set of keywords that reflect the model's complexity. The higher is the number of keywords to be processed, the greater is the level of details on stratigraphy, fluid properties, well control, numerical settings, and so on.

In particular, Eclipse's more general files that contain grid specifications have a `.GRDECL` extension. Initially, we implemented a parser routine that recognizes only the essential keywords to build either a corner-point or block-centered grid (Ponting, 1989). All keywords of the file are identified by a regular expression like `^[A-Z][A-Z0-9]{0,7}` (Fig. 2).

2.1 Corner-point grids

Corner-point is a grid format for 3D geometries that enables us to describe reservoirs with geological realism. Sometimes it is referred to as "pillar grid" because of ensembles of pillars that span from the top to the bottom of the model. The grid cells are then defined by eight paired nodes encountered over four adjacent pillars (Fig. 3). Since the nodes are free to slide along the pillars, several features, such as pinch-outs, and folds are accurately and consistently represented, even with degenerated or nonconform cells.

2.2 Block-centered grids

Block-centered grids are Cartesian-like domains specified by multidirectional step sizes and top reference coordinates

```
SPECGRID
20 20 4 1 F /
COORD
0.60814319E+03 -0.12195820E+04 0.22497217E+04 0.59895313E+03 -0.12417069E+04 0.24990918E+04
0.65048248E+03 -0.12046404E+04 0.22456648E+04 0.64349963E+03 -0.12271340E+04 0.24927664E+04
0.69211285E+03 -0.11901328E+04 0.22466001E+04 0.68799969E+03 -0.12132166E+04 0.24919016E+04
0.73225159E+03 -0.11736407E+04 0.22484019E+04 0.73173944E+03 -0.11976670E+04 0.24935242E+04
0.78713696E+03 -0.11509646E+04 0.22498828E+04 0.79193066E+03 -0.11765302E+04 0.24963970E+04
...
```

Fig. 2. Excerpt of a `.GRDECL file` for a reservoir model highlighting the `SPECGRID` and `COORD` keywords. The ellipsis are not part of the syntax and here they only illustrate continuity of the file content.

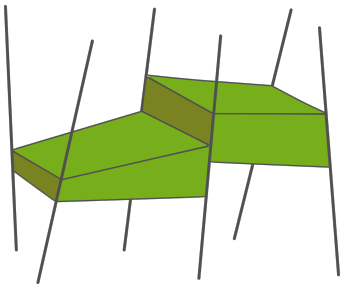

Fig. 3. Simple illustration of adjacent cells in a typical corner-point grid underpinned by arbitrary pillars. Adapted from (Lie, 2019).

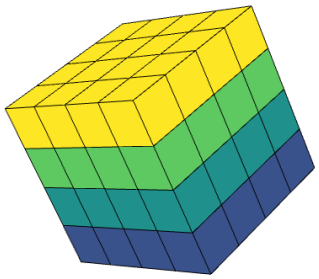

Fig. 4. Simple illustration of a block-centered Cartesian grid.

(keywords `DX`, `DY`, `DZ`, and `TOPS`). In this type of grid, the cells are sugar cuboids that arrange to approximate a regular three-dimensional volumes, i.e., "shoebox" models (Fig. 4).

3. Methodology

3.1 File processing

The processing of the file, where the topology and geometry of the grid will be calculated, takes place in the `process_grid` function. For each of the $NX * NY * NZ$ cells present in the mesh, the algorithm will retrieve the raw corner-point data from the keywords COORD and ZCORN or, it will retrieve the raw block-centered geometry data from the keywords DX, DY, DZ and TOPS.

For corner-point geometries, X and Y values are interpolated from the ZCORN. It is assumed by APyCE that all pillars are straight lines, making linear interpolation sufficient. In special cases where collapsed pillars occur, meaning pillars of the form $(x0\,y0\,z0 : x0\,y0\,z0)$ where the top point of the pillar coincides with the bottom point of the pillar, cells following the ZCORN convention must be recovered. In the case of block-centered geometry, VTK points and cells are created manually. The tool's workflow is illustrated in Fig. 4. The minimum code required to run the tool is to specify the file to process and its origin (Eclipse/Builder functions to get a VTU file).

After creating the VTK points, the software will retrieve the cells that follow the VTK pattern, as this pattern is different from the one followed by the Eclipse software. Fig. 5 shows the VTK pattern for a hexahedron object. Eclipse software uses different indexing than VTK uses. For the software in question, points 2 and 3 in Fig. 5 are "swapped" in position with each other; the same goes for points 6 and 7. When carrying out these changes, a standard method in VTK is adopted, simply invoking the `export_data` function to export the file for later viewing in the Paraview software. Fig. 5. Orientation of the points of a hexahedron within the VTK, the value 12 represents the hexahedron within the VTK (Schroeder et al., 2002).

3.2 Keyword handling

The current version of `APyCE` supports the following keywords.

- `SPECGRID`: specifies the number of cells in each direction of a corner-point grid.
- `DIMENS`: ditto for block-centered grids.
- `COORD`: specifies the coordinates `(X,Y,Z)` of the pillars.
- `ZCORN`: indicates the depth of each cell corner over the pillars.
- `PORO`: stores the porosity values per cell.
- `PERMX`: stores the X-direction permeability for each grid cell.
- `PERMY`: ditto for the Y-direction.
- `PERMZ`: ditto for the Z-direction.
- `INCLUDE`: appends supplementary files to the main file.
- `ÅCTNUM`: tags grid cells as active/inactive (boolean value).
- `DX`: specifies the step size over the X-direction for block-centered grids.
- `DY`: ditto Y-direction.
- `DZ`: ditto Z-direction.
- `TOPS`: stores the reference value from the top for each cell in a block-centered grid.

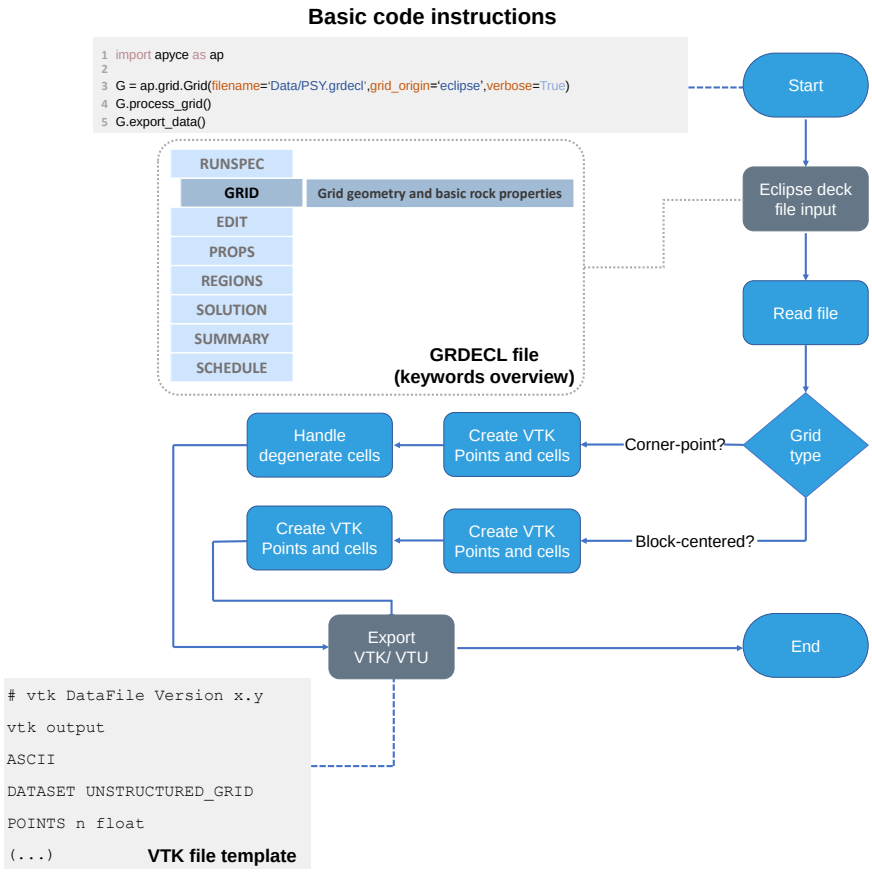

Fig. 5. Workflow of APyCE.

3.3 Code framework

The Python programming language is used because it is interpreted, object oriented, high-level, and with dynamic semantics. In addition, the choice for the Python language for the development of the project is justified by the vast amount of modules available to perform scientific computing, signal processing, and graphic manipulation tasks, such as NumPy, SciPy, and Matplotlib, in addition to having a library for the use of VTK-The Visualization Toolkit, an open-source toolkit for 3D computer graphics, image processing, and visualization, created and maintained by Kitware, Inc. Another point that strengthens the choice for the Python language is its simplicity, which reduces software maintenance. Through the available modules and packages, modularized programming and code reuse is encouraged. The support for programming in an object-oriented paradigm is another factor that contributes to the choice of the Python language, since each object, or class, can be compared to real-world elements and defined based on these elements. Finally, reading and maintaining extensive code developed in an object-oriented paradigm becomes extremely simple when compared to other programming paradigms (functional, imperative, declarative, structural, etc.).

4. Usage and development

On the `ApyCe` repository, one can finds the example script `Getting_Started.py`, which reproduces the basic code instructions as depicted at the top of the flowchart (Fig. 5) and generates multiple VTK files.

Lastly, the outputs can be visually improved in Paraview from a plethora of applicable filters (Fig. 1). The tool is capable of rendering several models of reservoirs with different topologies, even those comprising faults or complex stratigraphy.

Regarding software, the following points are noteworthy:

`APyCE` is written in Python 3.8. The choice for this language is justified by the vast amount of modules available to perform scientific computing, signal processing, and graphic manipulation tasks. Another point that strengthens the choice for Python is its simplicity, which reduces software maintenance. Through the available modules and packages, modularized programming and code reuse is encouraged.

Processing highly refined grids (hundreds or millions of cells) may demand increased RAM and processor power. Yet less complex grids are manageable affordably.

To work properly, APyCE requires the following Python packages: NumPy (v. 1.19.2 or above), VTK (v. 8.2.0 or above), PyVista (v. 0.29 or above), and Matplotlib (v. 3.4.1 or above). The most current version was tested on Ubuntu 20.04 distribution with the 5.11.0-25-generic kernel anw with

PyTest (https://docs.pytest.org/en/latest/). All the workflow for exporting/visualizing corner-point grids are fully workable, but block-centered grids still have partial keyword support.

5. Conclusion

In this paper, we presented a Python module tailored for viewing 3D reservoir models with varying structural complexity. When parsing the input files, the module's routines are able to recognize the main features of corner-point and block-centered grids.

Beyond its open-source nature, easy manipulation and integration with Paraview software are other strong points of this module. One the one hand, it allows that any researcher with minimal knowledge of programming can run the code. On the other hand, it provides high-quality outputs readily usable for general data analysis or publication purposes.

This module can help all professionals working at the oil and gas upstream, mainly reservoir engineers, geologists, and modelers who need to understand spatial distributions or analyze field-scale properties. From this point on, efforts will be made to extend the module's capabilities to parse other file formats, support additional model features, and provide software interchangeability. Code profiling tests in highly refined meshes are needed to verify processing efficiency and bottlenecks.

Acknowledgements

G.P.O and M.M.T acknowledge the National Council for Scientific and Technological Development (CNPq-Brazil) for the financial support granted through PIBIC/UFPB Program 2020-2021 and CT-PETRO Program (No. 405654/2022-7). B.W acknowledge the Natural Science Foundation of China (No. 52204021), Science Foundation of China University of Petroleum, Beijing (No. 2462022BJRC002) for the financial support.

Conflict of interest

The authors declare no competing interest.

Code availability

APyCE can be downloaded directly from its project page on https://github.com/mateustosta/apyce-project.

References

Ayachit, U., Geveci, B., Avila, L. The Paraview Guide: A Parallel Visualization Application. Kitware, New York, USA, 2015.

Lie, K. A. An introduction to reservoir simulation using MATLAB/GNU Octave: User guide for the MATLAB Reservoir Simulation Toolbox (MRST). Cambridge, UK, Cambridge University Press, 2019.

Masison, J., Beezley, J., Mei, Y. et al. A modular computational framework for medical digital twins. Proceedings of the National Academy of Sciences of the United States of America, 2021, 118(20): e2024287118.

Ponting, D. K. Corner point geometry in reservoir simulation. European Association of Geoscientists & Engineers, 1989: cp-234.

Schlumberger. Eclipse Reservoir Simulation Software Reference Manual. Schlumberger, Texas, USA, 2014.

Schroeder, W., Martin, K. M., Lorensen, W. E. The Visualization Toolkit An Object-Oriented Approach to 3D Graphics. Prentice-Hall, New Jersey, USA, 1998.

Sircar, A., Nair, A., Bist, N. et al. Digital twin in hydrocarbon industry. Petroleum Research, 2022, in press, https://doi.org/10.1016/j.ptlrs.2022.04.001.

Sullivan, C., Kaszynski, A. Pyvista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (vtk). Journal of Open Source Software, 2019, 4(37): 1450.

Sun, S., Zhang, T. A 6m digital twin for modeling and simulation in subsurface reservoirs. Advances in Geo-Energy Research, 2020, 4(4): 349-351.

Wang, B. Pygrdecl a python-based grdecl visualization library, 2018.