Response to the reviewers of "Open Chemistry, JupyterLab, REST, and Quantum Chemistry"

Marcus D. Hanwell[1,2], Wibe Albert de Jong[3], and Johannes Hachmann[4]

[1]Kitware
[2]Brookhaven National Lab
[3]Lawrence Berkeley National Laboratory
[4]University at Buffalo – State University of New York

September 2, 2020

The following provides excerpts from the reviews of the manuscript along with responses to them from the authors of the paper (italicized).

Referee #1 (Report openly available here after publication of the article)

Detailed Report

The authors discuss a variety of serialization formats (JSON, msgpack, JSONb), but this section would be strengthened by several additional points:

- All of the formats above describe data at rest/transit (serialized) and not the description of the data in the various programs (dictionaries in Python, maps or ptrees in C++, etc). It is important to separate these ideas as JSON isn't a formal part of any language beyond JS and the ideas should describe when this data is in use as well.
- The positives and negatives of the serialization formats were not discussed, why was JSON chosen as the format of choice? For example, Arrays are common in quantum chemistry, but are known to be slow, lossy, and larger through JSON compared to binary formats.

The introductory paragraph for "Handling Data and Metadata" introduces the need for formats that can be "used from a number of programming languages ranging from compiled languages such as C, C++, and Fortran on supercomputers/high performance computing resources to perform quantum chemistry calculations through to interpreted languages such as Python for data analysis and JavaScript/TypeScript in web frontends or C/C++ in desktop applications." The second paragraph introduces several standards considered, and mentioned "Large data is preferably stored in binary formats, which is where HDF5 was seen as one strong contender and more recently MessagePack has gained traction due to its JSON-like structure and wide language support thanks to its simple binary specification." We have added paragraph 4 to that section to more fully explain our reasoning, and future paths forward.

General comments:

- It may be worth highlighting programs like CCLib, RDKit, ASE, and more when it comes to translating in addition to Open Babel.

References added, the authors primarily used Open Babel but others were used in places.

- According to the QCSchema documentation (https://molssi-qc-schema.readthedocs.io/en/latest/auto_topology.html), there is a definition for connectivity available.

Pull request https://github.com/MolSSI/QCSchema/pull/23 remains unmerged as of August 18, 2020, but I see the merge of connectivity in May of 2018. Thank you for spotting this, it is only in one example at this stage. Removed language about no agreed upon representation for bonding, and added in an example of equivalent bonding to the QCSChema JSON.

- QCSchema basis sets appear to have also been released which supports the new Basis Set Exchange format (https://github.com/MolSSI/QCSchema/pull/62).

Added a comment to highlight that recent work has added support for the basis set exchange format.

- It would be good to cite Jupyter, Jupyter Lab, Pub Chem, and ChemSpider as they are used in the paper.

Jupyter had already been cited (Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, Jupyter development team. Jupyter Notebooks ? a publishing format for reproducible computational workflows. 87–90 In Positioning and Power in Academic Publishing: Players, Agents and Agendas. *IOS Press, 2016), added citation to jupyter.org as well, added missing citations to JupyterLab, PubChem and ChemSpider, thank you for spotting these.*

- Many of the audience are unlikely to be aware of REST interfaces or what they empower, it would be good to describe this in more detail.

Added some further explanation of what a RESTful interface is to the first paragraph of "Flexible Data Server Platform"

- The reasons of selecting Girder could be indicative of common Python web frameworks such as Django, Flask, FastAPI, and more. Are there specific capabilities Girder supplies over these general frameworks?

It reuses a number of Python frameworks, it provides some capabilities that can be found in Django, while offering a smaller more focused codebase with more convenience functions than lower level frameworks such as Flask and FastAPI. Added a further sentence explaining some of these advantages such as authentication, HPC queuing integration, etc.

- In the *Molecule(Resource)* demonstration, the *MoleculeModel* is never explained. It appears to be an ORM, but I believe is unclear to readers. The second example also has lines for pagination and index setting which are not explained.

It is a simple ORM, added some text but at its core it maps from the underlying database modelto and from the RESTful API providing a clear separation from the RESTful endpoint code. They were mentioned in the text, added some additional detail to avoid vagueness.

- Containers are not typically available on the majority of supercomputing platforms so the exclusive choice of this seems limiting. Perhaps the authors could comment on this choice and availability of this platform without containers.

This platform has been developed as a forward looking platform, and we chose to integrate with NERSC early on where containers have been available for some time in the form of Shifter. A little more detail added to highlight this choice, while the platform could be adapted to function without containers it is our belief that next generation supercomputers will offer first class support for containers.

- Binder and QCArchive should be cited, TorchANI's GitHub should be cited in leui of a paper. (It appears ANI is in the references, but not in the paper)

ANI is mentioned in the third paragraph of the "Machine Learning" section, and cited there. A TorchANI paper has since been published, added that to the citations. Added Binder and QCArchive citations that were missing, thank you for pointing out the omissions.

Referee #2 (Report openly available here after publication of the article)

Summary

The manuscript describes the features of a platform developed by the OpenChemistry consortium that bridges many prominent technologies in data sharing, analysis, and visualization. The platform unifies a web-based GUI with few computational backends and can be used to visualize and analyze pre-existing data or newly produced computational results.

The development of this platform is timely and the description in the manuscript is clear and compelling. I recommend the paper be **published with minor revisions**, suggested in the following.

I hereby give permission to publicly associate my name to this referee report.

Detailed Report

I would first like to thank the authors for describing their work in a language approachable also to those that are not steeped into novel web-based technologies. The submission is timely and is extremely well-suited for publication in an interactive format. Finally, I couldn't agree more with the last statement in your conclusions: *As a community it is important to embrace open source, open data, open standards, and open access to reproducible research.* I think your work does represent an important step forward in this direction.

This said, I have some questions and suggestions which the authors should consider in revising the manuscript:

- The use of the Chemical JSON format seems essential for the inner workings of the platform described in the manuscript. It is my impression, however, that the format is not as widely known as it should be. Are there standard examples in your repositories of how to write a Chemical JSON file from commonly used compiled languages, such as C++, C, and Fortran? Are there any quantum chemistry codes that can already emit their output in this format? Or is there an intermediate Python layer that translates from, *e.g.* a checkpoint file, to Chemical JSON?

There is JavaScript/TypeScript in the web client code, C++ in AvogadroLibs, and wrapped C++ from AvogadroLibs capable of going from checkpoint files to Chemical JSON. I think it is fair to say it is essential for the inner workings of the platform, but the intent is to offer that in addition to other formats for import/export of data. The Chemical JSON GitHub repository (https://github.com/openchemistry/chemicaljson) is mentioned when discussing examples.

- Is there a formal standardization process for the Chemical JSON format in place? Who is participating? Could you describe the workflow used in the definition of the open standard?

There is nothing formal, there is a repository referenced in the paper. It is a working format developed to support several projects that can move quite quickly. The authors (Hanwell and de Jong) helped organize an initial workshop, and started discussions with MolSSI shortly after MolSSI was founded to spur standardization.

- How widespread is the adoption of Chemical JSON so far? Could it be merged with the QCSchema efforts of the MolSSI?

It is already in a number of codebases, ultimately it could be merged but in efforts to standardize it became clear that the velocity of QCSchema was too slow to accommodate projects with a shorter timeframe for development. The primary effort is to ensure QCSchema/similar support everything in Chemical JSON. We have added text to make this approach clearer - thank you for the questions. Please see the final paragraph of "Handling Data and Metadata" for some further detail elicited from this line of questioning.

- Are there any limitations to the format? I could think that storing basis set and MO coefficients information for very large molecules would make it rather impractical. Is this the case? If yes, how do you plan to solve this problem?

Absolutely, as discussed in the early part of "Handing Data and Metadata", and a concluding remark in the new paragraph making it clear our belief is that binary formats are essential for large calculations. JSON offers valuable room to explore the space before mapping to binary formats that share similar structures such as MessagePack or the more traditional HDF5.

- What about QM/MM simulations? How would the format need to be extended, if at all?

These are not addressed at this stage, they would of course be useful additions to support in the platform without doubt. For QM/MM one of the format extensions will be to label atoms as quantum or classical, and force field description. Some of these are currently being looked at in the QCSchema.

- Are there any plans/thoughts to support output from non-GTO-based codes? For example, numerical basis sets, plane waves, multiwavelets.

Not at this time.

- Are there guides available on how to deploy the platform on local HPC infrastructure?

This is the biggest gap right now, we have guides for local single machine developer deployments, AWS deployments with cloud clusters, and a SPIN-based deployment using NERSC at LBNL. This should be possible, but the team has not explored it focusing primarily on local development and the NERSC/AWS deployments. As with many things, this could certainly be accomplished given further development time.

- The MolSSI QCArchive initiative has an overlap of functionalities with the platform described in this paper, or so it seems to me. Would you clarify the relationship between your platform and QCArchive? What are the use cases for which your platform is specifically designed? Could the platform benefit from integration with QCArchive? Would QCArchive benefit from integration with your platform?

MolSSI and QCArchive arrived after we began working on this project, and there are some distinct differences. During early discussions it was clear that we focused on individual calculations, and retain more data so that electronic structure can be visualized for example. We also enable search based on name, InChI, SMILES etc which is (or at least was) difficult in QCArchive. They focus on large parameter sweeps, and generating inputs for machine learning/MD potentials.

- I think the manuscript would benefit from a short description of the code development workflow used by the developers.

Good point, added some description to the introduction in the final paragraph.

- The authors should put more emphasis on the fact that quantum chemical program packages in the backend are accessed through Docker/Singularity/Shifter images. Containers make it possible to share the code used to generate computational results without violating licenses. Even without the (undue, in my opinion) barrier imposed by licenses, some authors find unpleasant any obligation to share their research code. The use of containers removes a significant barrier not only for reproducibility, but also

for collaboration. This is, to me, an extremely compelling feature of the platform and I think it deserves to be highlighted more in the manuscript.

Thank you, we agree on their importance, and the section on "Containers for Chemistry Codes" discusses the various containers, and why more than one kind is needed at this time. I added two paragraph to highlight some of these points at the end of the introduction section, and thank you for your suggestions and highlighting these points. We wholeheartedly agree.