Open Review of "Expressing general constitutive models in FEniCSx using external operators and algorithmic automatic differentiation"

Andrey Latyshev, Jérémy Bleyer, Corrado Maurini, Jack S Hale, Thomas Helfer, Chris Richardson, Alexander Popp

Open Review
Journal of Theoretical, Computational and Applied Mechanics

Identifiers
Open Review
OAI hal-05082559
Reviewed Article
DOI 10.46298/jtcam.14449

Review of "Expressing general constitutive models in FEniCSx using external operators and algorithmic automatic differentiation"

Andrey Latyshev[1,2], Jérémy Bleyer[3], Corrado Maurini[2], Jack S. Hale[1], Thomas Helfer[4,R], Chris Richardson[5,R], and Alexander Popp[6,E]

[1] Institute of Computational Engineering, Department of Engineering, Faculty of Science, Technology and Medicine, Université du Luxembourg, Luxembourg
[2] Institut Jean Le Rond d'Alembert, Sorbonne Université, UMR CNRS 7190, France
[3] Laboratoire Navier, École des Ponts ParisTech, Université Gustave Eiffel, UMR CNRS 8205, France
[4] CEA, DES, IRESNE, DEC, Cadarache F-13108 Saint-Paul-Lez-Durance, France
[5] BP Institute, University of Cambridge, Cambridge, United Kingdom
[6] Institute for Mathematics and Computer-Based Simulation, Universität der Bundeswehr München, Werner-Heisenberg-Weg 39, 85577, Neubiberg, Germany
[R] Reviewer
[E] Editor

Review of version 1

Permalink: hal-04735022v1

Authors Thank you for reviewing our manuscript.

In summary, the three reviewers were largely positive towards the first version of the article and have requested some minor corrections/clarifications, and some discussion around the issues of integrating legacy constitutive models and executing constitutive models on GPUs.

In the case of the optional comments from Reviewer 1, we decided to mention some of the conversation points raised in a new version of the Conclusions, and we have acknowledged this contribution at the end of the article.

Thank you in advance for your further efforts reviewing the revised version of our manuscript.

1 Reviewer 1 (Thomas Helfer)

Reviewer **General overview**

The paper is very well written, and the reviewer shares much of the authors' enthusiasm: the prospects opened up by the introduction of external operators in FEniCS are indeed rich in promise, and enable this platform:

- to interface with the very large number of material libraries available, especially those with the UMAT interface democratized by the Abaqus solver, which is the case in a very large number of cases. One may refer to the UMAT4COMSOL project (Lucarini and Martínez-Pañeda 2024) or, in the context of FEniCS, to the work of Rosenbusch *et al.* (Rosenbusch et al. 2024) to highlight how interesting this approach is.

- to use new approaches based on the wide range of scientific libraries (optimization, machine learning, AI) available in en python, who is de facto the lingua franca of scientific computations. This points totally justifies the data centric approach of FEniCS highlighted in the paper.

Authors We have added references to both of these papers, briefly in the introduction and also added some discussion along the points you have made in your "specific questions" section.

Reviewer However, the reviewer thinks that the choice of examples is questionable:

the authors discuss implementing complex laws (the term "complex" appears 16 times in the paper), but the two examples in question can hardly be considered as complex.

Authors We have scaled back the overuse of the word complex, both for readability and also because the definition is subjective – nonetheless we have kept our definition "Typical examples of such complex constitutive models…" in the introduction.

Reviewer The authors insist on the need to be able to implement behavior laws that do not fit into a variational framework (the term "variational" appears 12 times in the three first pages). However, these two laws do fit into the framework of generalized standard materials. The choice of a non-associated plastic model would have been relevant and would exhibit an example closer to a common application case in geomechanics.

Authors We have removed the variational before form where it is not necessary.

Although the plasticity laws mentioned in the article fit the framework of generalized standard materials (GSM), in practice, such laws are usually still implemented via numerical algorithms like return-mapping. Our work specifically focuses on those problems where researchers choose a purely numerical implementation of their constitutive models even if a variational formulation is available.

Within the model of Mohr-Coulomb described in the article, switching between the associative model and the non-associative one is done by varying the friction and dilatancy angles. From the implementation perspective, this does not change a lot. Indeed, demonstrating that our implementation works in the non-associative case as well is a good idea, so we have added a new result for the apex-smoothed Mohr-Coloumb model where the parameters are set for non-associativity.

Reviewer The authors insist on the effectiveness of external operators. It is a pity that this aspect is not discussed at all in the examples presented.

Authors We have added an additional sentence to the caption of Figure 2 that emphasises how External-Operator allows the seamless expression of this type of model structure in UFL – this is the only place where we actually show ExternalOperator as code in the paper. The examples in the supplementary material have far more examples of usage.

There is an example in the documentation with an external operator taking two operands https://a-latyshev.github.io/dolfinx-external-operator/demo/demo_nonlinear_heat_equation_part2.html. We are currently experimenting with implicit constitutive models and mixed formulations of mechanics problems, but this seemed like too much for this first paper.

Reviewer Last but not least, neglecting the use of existing material libraries to focus on the implementation of these laws, the authors focus on a niche of users.

Authors This is a fair point, and consequently we have added the discussion and reference to Rosenbusch et al. 2024 which goes much further in this direction of supporting existing libraries.

Reviewer These two examples concern plasticity laws that differ in the choice of criterion: von Mises in one case, Mohr-Coulomb in the other. The first case is without doubt the simplest example imaginable.

Authors Indeed, and this was our intent – everyone knows this model and thus it serves as a didactic example. This is pointed out at the start of Section 4.1.

Reviewer The second is complicated only by the non-regularity of the Mohr-Coulomb criterion, a difficulty evacuated by a regularization introduced in the works of Abbo and Sloan (Abbo and Sloan 1995; Abbo et al. 2011). De Souza Neto et al. 2008 are critical of this regularization and propose an alternative solution in the space of principal stresses that deals exactly with the singularities of the load surface. However, their implementation is much more involved.

This regularization does indeed lead to a relatively complex expression of the yield surface involving the three stress invariants whose derivatives, required to form the Jacobian of the implicit system, are difficult to calculate, allowing the authors to highlight the valuable aid of automatic differentiation. However, the reviewer's experience is that the time spent on the

implementation itself is negligible compared to other considerations: unit test verification; documentation, etc.

In fact, it certainly takes at least as long to obtain Figure 6 (page 16) as it does to calculate the derivatives in question.

Authors Indeed, AD automates only one important part of the model development pipeline. Our experience is that by using automated tools users are less prone to making mistakes, leaving more time for activities such as unit testing and documentation, which are still often forgotten or executed poorly.

The code for generating figure 6 is in the supplementary material – it did not take us a long time (development, or execution) to make this once we had built the working constitutive model in JAX.

Reviewer The strategy put forward by many material libraries is to offer pre-established frameworks that enable quick and easy implementation of complex constitutive equations. For example, the ZMAT library enables to implement complex (visco-)plasticity laws in the formalism F_eF_p from a minimal description (without code), see Abatour et al. 2021 for details. This greatly relativizes the simplicity of the solutions put forward by the authors.

Authors In terms of domain specific languages for writing constitutive models, we referenced MFront in the first draft, but some of the authors were not aware of ZMAT – we have now added a reference to ZMAT as software and to its use in (Abatour et al. 2024, published).

Reviewer The two laws presented here are very simple and ultimately quite similar. As a result, the presentation of the first law doesn't add much to the paper. By giving an example of crystalline plasticity, among others, (Rosenbusch et al. 2024) is more convincing.

In short, these toy examples are only meant to give an idea of the potential of external operators, which will no doubt be explored in future work.

Overall, this paper deserves to be published in JTCAM. The remarks made so far are intended to give feedbacks to the authors and perhaps give them the opportunity to rebalance their remarks by putting more emphasis on the possibilities of interfacing with established material libraries, a possibility of interest to an engineering community.

Authors We agree that the first version of the paper was very focused on 'novel approaches to constitutive modelling'. We have added a paragraph to the discussion which restores the balance, acknowledging that there has already been a great deal of work done in the community using established techniques, and that (Rosenbusch et al. 2024) contains work linking FEniCS to the things you mention (UMATs, standard programming languages, DSLs for constitutive modelling etc.).

In the future we would like to explore:

Better interfaces/examples to existing libraries using established techniques in the style of (Rosenbusch et al. 2024).

An in-depth exploration of what is actually possible with JAX (and other AD tools) with respect to the most complex constitutive models.

Reviewer **Specific questions**

This section describes some of the reviewers's question and proposals to improve the paper. Answers to those questions are not required to be integrated to the final text.

External operators and GPU

GPU are mentioned once with references to the work of Blühdorn. The reviewer wonders if the strategy described in the paper can be extended to GPU programming, in particular regarding the memory transfer between FEniCS and numpy's arrays.

Authors At the moment we are using the Python interface to DOLFINx which is CPU oriented, so we need to do memory transfers of the CPU evaluated operands to the GPU memory using JAX https://jax.readthedocs.io/en/latest/_autosummary/jax.device_put.html to evaluate the constitutive model. Then, the external operators can be computed on the GPU. At this point with a GPU assembler (others are working on this) we could potentially avoid the transfer of the external operator back to the CPU.

Reviewer This question is relevant as some authors (Lewandowski et al. 2023) claims that the classical structure of mechanical solvers, that this work allows to reproduce, is not compatible with GPUs.

Thanks for pointing out this very interesting paper – essentially the authors construct a method where the plasticity problem is re-written as a block diagonal system and then a field split preconditioner is applied to "effectively" eliminate the plastic deformation fields at the global algebraic level. We would remark that in practice (Lewandowski et al. 2023) still assembles sparse matrices and does not yet execute on 'unusual' hardware.

In the same spirit, matrix-free solvers would require either to (Brown et al. 2021):

Recompute the consistent tangent operators and the stresses during the iterative process (which is feasible for simple hyperelastic behaviours for instance (see Brown et al. 2021).

Store the consistent tangent operator and the stresses of all quadrature points on the GPUs.

Could the authors make some comment about this?

Authors Although the framework we have developed is innovative in terms of expressing full models, what is produced is still in the legacy style of solid mechanics solvers (low-order polynomial approximation and sparse matrix assembly).

In the context of high-order polynomial matrix-free solvers, the current structure of the library is moving in the direction of your second bullet point (store the consistent tangent operator and the stresses of all quadrature points on the GPUs.) How feasible this is in practice in terms of memory, particularly in three dimensions, and also in minimising memory motion, is currently unclear. Clearly there are some interesting things to explore here, perhaps including low precision or compressed computation/storage and also partial/batched computations.

Inspired by your question we have added a paragraph in the conclusion which mentions this open point, and the references to the reader the high quality papers you mentioned for some ideas.

Reviewer **Numerical efficiency**

The paper does not discuss if the implementation of the Mohr-Coulomb plastic behaviour is efficient. Experiences with automatic differentiation of plastic yield surface using ADOL-C in MOFEM (Kaczmarczyk et al. 2020) lead to disappointing results.

Authors We remark on some more promising results using Enzyme in https://arxiv.org/pdf/2204.01722. The wider point on benchmarking generated AD code from different tools is an interesting one and worth exploring further.

Reviewer This question could be treated in the paragraph "Solution of slope stability problem" which could compare the time spent in the behaviour integration to the total computational time and the time spent in the linear solver of example.

Could the author comment on this?

Authors As we guess is already clear, in terms of scaling, the linear solve will at some point dominate the assembly of the operators (including the constitutive model evaluation). To show this we have added strong scaling experiment in Appendix A1 – when the Mohr-Coloumb model is solved on a reasonably large mesh the constitutive model evaluation takes on the same order of time as the linear solve. This suggests that JAX is doing a reasonable job here, but this is hardly a rigorous performance analysis, hence why we have put this in the appendix.

Reviewer **Robustness**

According to the reviewer experience on a similar behaviour (using the Hosford criterion to approximate the Tresca criterion), the robustness of the Newton algorithm used to solve the implicit system strongly depends on the requlariation angle of the yield surface: if this angle is too low, the flow direction may oscillate strongly between two iterations of the Newton algorithm leading to divergence of the algoritm.

Indeed, the relatively high value (hidden in Appendix A) chosen (on purpose?) by the authors for the transition angle shall probably eliminate this problem.

Authors The high value of the transition angle is chosen as advised in the original paper introducing this particular model of Mohr-Coulomb: "in practice, should not be too near 30° to avoid

ill-conditioning… and the typical value is 25°" (Abbo and Sloan 1995, p. 429)".

Reviewer This lead to the following question: in pratice, complex mechanical behaviours may lead to failure in the behaviour integration. How are integration error supported, in particular in an MPI context?

Authors JAX/XLA can raise Exceptions of type XLARuntimeError across to Python, see e.g. https://stackoverflow.com/questions/77381356/idiomatic-ways-to-handle-errors-in-jax-jitted-functions.

In terms of handling these in an MPI context, if the Python exception remains unhandled, and if the special `python -m mpi4py` launcher is used, mpi4py will clean up on COMM_WORLD, avoiding any deadlocks with processes where the exception was not raised.

https://mpi4py.readthedocs.io/en/stable/mpi4py.run.html#exceptions-and-deadlocks

In terms of actually handling integration errors, this should probably not be done with Exceptions but instead handled through standard control flow logic in JAX – only if this logic fails should an Exception be raised.

Reviewer **About Section "Mohr-Coulomb model without AD", page 16**

In practice, the analytical expressions of these derivatives are very complex and so their translation into programming code can create dozens or even hundreds of lines of code.

This statement seems a bit exaggerated. In the cited example 1, the computation of the derivatives takes less than twenty lines of code (14 according to the reviewers' count) and does not seem that complex.

Authors We have scaled back our remarks here.

Reviewer Moreover, this page also exhibits an alternative implementation that does not require to write a single line of code based on some sort of JSON description of the behaviour. The ZMAT library provides a similar catalog named gen_evp allowing to easily implement (really) complex behaviours without a single line of code ("`Z-Set | z-Mat`" n.d.).

Authors This looks very interesting – we mentioned MFront in the introduction as a kind of "domain specific language" for constitutive modelling, and as already discussed we've now also added ZMAT.

Plain JAX works at a lower level, which means it will never be as elegant as these tools you mention. Where we see things going in the future perhaps is building constitutive modelling libraries directly on top of general differentiable programming language such as JAX.

Reviewer If the number of line of codes, or the fact that one does not have a single derivative to implement are valid criteria, those two solutions are superior to the one proposed in the solution.

Authors Indeed, for the same reasons we would also avoid writing a finite element Laplace kernel in C and would instead use UFL (domain specific language) + FFCx (compiler) to generate it. Nevertheless, it is sometimes necessary to escape from these abstractions and work at a lower level. JAX looks promising, if this is necessary. We hope the revised comments in the paper now give sufficient weight to other well-established processes for writing constitutive models.

Reviewer Mostly a detail, but having a Section 3.1 without a Section 3.2 seems strange.

Authors This has been fixed by removing the subsection 3.1.

Reviewer **Section 3.1, page 10** where the constitutive model is defined by hundreds of internal state variables, e.g. in crystal plasticity (Méric and Cailletaud 1991). In the reviewer's experience, the Méric-Cailletaud have dozens of state variables, not hundreds. We may however have thousands of state variables for homogenized behaviours treating hundred of grains each described by the Méric-Cailletaud' law.

Authors This has been corrected.

Reviewer About reference "(Helfer, Michel, Proix, Sercombe, et al. 2024, MohrCoulomb.md)"

This reference seems useless. As stated before, googling "MFront Mohr-Coulomb", gives the following page as the first answer: https://thelfer.github.io/tfel/web/MohrCoulomb.html and

seems more appropriate.

Authors The reference has been modified.

2 Reviewer 3 (Chris Richardson)

Reviewer Overall, this is an excellent contribution, that shows that by composing together different software packages (FEniCSx, Numba, JAX etc.), a variety of difficult non-linear problems can be solved.

The abstract clearly sets out the problem at hand, and there is an articulate explanation of the need for constitutive models provided by general programming languages. Previous work is cited and referred to appropriately. Apart from a few minor clumsy expressions: "by-hand derivation" might be better just as "derivation by hand", and probably "expressions swell" should just be "expression swell", the paper is very well written. It was easy to understand, and although technical in places, the reader does not feel they are being swamped with too much jargon.

Authors We have made the modifications suggested.

Reviewer The supplementary material (demo code) was easy to download, and I could run the demos and regenerate the figures from the paper. What more could one ask?

Authors We're pleased to hear that everything worked well. We have continued to update and improve the library and examples, and are making releases against the upstream releases of DOLFINx.

3 Reviewer 4 (Anonymous)

Reviewer Review of "Expressing general constitutive models in FEniCSx using external operators and algorithmic automatic differentiation".

The paper describes the dolfinx implementation of the external operator introduced to UFL by the Firedrake team. The paper is well-written and interesting and I recommend its publication.

I do have a few comments:

In eq (14) they introduce the local problem. However, it is not really clear that it is local except that it is local to that iteration. Is it spatially local? Are these local problems solved for each index? Not really clear from the description.

Authors Indeed, spatially local, this has now been clarified immediately after eq (14).

Reviewer Page 8: It is stated that "FEMExternalOperator… efficiently manage memory allocation and data transfer by optimizing…". I understand the memory allocation part but could the authors elaborate on the steps taken for optimizing the data transfer?

Authors We have removed the word optimizing which was used incorrectly.

Reviewer Page 10: "the evaluation of external operators… is an embarrassingly parallel…". I understand this, but how is this exploited in practice. Please explain…

Authors In the current examples there is already the potential to exploit CPU instruction-level parallelism (e.g. SIMD) via Numba's use of LLVM and JAX's use of XLA for compilation, although we have not done analysis of the generated code. We also have not explored e.g. parallelism on wider architectures like GPUs/TPUs, or higher levels of parallelism such as threading at this stage. We have clarified this statement about the embarrassingly parallel nature of the problem, with the limits of what we have achieved in practice today, and performed a strong scaling analysis using DOLFINx's existing MPI functionality of the Mohr-Coulomb model. The results of the analysis can be found in the appendix A1.

4 Editor's comments (Alexander Popp)

Editor The manuscript presents a novel framework for incorporating general constitutive models into FEniCSx by utilizing external operators in combination with algorithmic automatic differentiation (AD). Through this approach, the authors enable the integration of complex, non-variational, or algorithmically defined constitutive models—expressed in general-purpose programming

languages—into a finite element solver traditionally constrained by variational formulations. The methodology is demonstrated via two solid mechanics examples: a von Mises plasticity model implemented with Numba, and a more advanced apex-smoothed Mohr–Coulomb plasticity model using JAX. In the latter, AD is leveraged to avoid error-prone manual differentiation, and the implementation is validated through yield surface tracing, a Taylor remainder test, and comparison with literature benchmarks. The manuscript underwent a rigorous peer-review process involving three expert reviewers. All reviews were broadly supportive, highlighting the technical depth, clarity of presentation, and relevance of the topic for the computational mechanics and open-source FEM communities. Reviewers requested only minor revisions, including improved discussion of the integration with legacy material libraries, clarification on GPU execution, and elaboration on performance considerations and scaling behavior. The authors provided a detailed and thoughtful point-by-point response, addressing each comment and expanding the discussion in several key areas. Changes were incorporated into the revised manuscript, and the additional results—such as a strong-scaling analysis—enhanced the overall contribution. Given that all reviewer concerns were resolved comprehensively and no further issues were raised, the revised manuscript was accepted for publication after the second round of external review.

References

Abatour, M., K. Ammar, S. Forest, C. Ovalle, N. Osipov, and S. Quilici (2024). A generic formulation of anisotropic thermo-elastoviscoplasticity at finite deformations for finite element codes. *Computational Mechanics*. [DOI], [OAI].

Abatour, M., K. Ammar, S. Forest, C. O. O. Rodas, N. Osipov, and S. Quilici (2021). "A generic formulation of anisotropic thermo-elastoviscoplasticity at finite deformationsfor Finite Element codes". working paper or preprint.

Abbo, A. J., A. V. Lyamin, S. W. Sloan, and J. P. Hambleton (2011). A C2 Continuous Approximation to the Mohr–Coulomb Yield Surface. *International Journal of Solids and Structures*. [DOI].

Abbo, A. J. and S. W. Sloan (1995). A Smooth Hyperbolic Approximation to the Mohr-Coulomb Yield Criterion. *Computers & Structures*. [DOI].

Brown, J., A. Abdelfattah, V. Barra, N. Beams, J.-S. Camier, V. Dobrev, Y. Dudouit, L. Ghaffari, T. Kolev, D. Medina, W. Pazner, T. Ratnayaka, J. Thompson, and S. Tomov (2021). libCEED: Fast algebra for high-order element-based discretizations. *Journal of Open Source Software*. [DOI], [OAI].

De Souza Neto, E. A., D. Perić, and D. R. J. Owen (2008). *Computational Methods for Plasticity: Theory and Applications*. Wiley.

Kaczmarczyk, Ł., Z. Ullah, K. Lewandowski, X. Meng, X.-Y. Zhou, I. Athanasiadis, H. Nguyen, C.-A. Chalons-Mouriesse, E. J. Richardson, E. Miur, A. G. Shvarts, M. Wakeni, and C. J. Pearce (2020). MoFEM: An open source, parallel finite element library. *Journal of Open Source Software*. [DOI].

Lewandowski, K., D. Barbera, P. Blackwell, A. H. Roohi, I. Athanasiadis, A. McBride, P. Steinmann, C. Pearce, and Ł. Kaczmarczyk (2023). Multifield finite strain plasticity: Theory and numerics. *Computer Methods in Applied Mechanics and Engineering*. [DOI], [OAI].

Lucarini, S. and E. Martínez-Pañeda (2024). UMAT4COMSOL: An Abaqus user material (UMAT) subroutine wrapper for COMSOL. *Advances in Engineering Software*. [DOI], [OAI].

Méric, L. and G. Cailletaud (1991). Single Crystal Modeling for Structural Calculations: Part 2–Finite Element Implementation. *Journal of Engineering Materials and Technology*. [DOI].

Rosenbusch, S. M., P. Diercks, V. Kindrachuk, and J. F. Unger (2024). *Integrating custom constitutive models into FEniCSx: A versatile approach and case studies*. [DOI], [OAI].