COMPUTATIONAL MECHANICS 2021
36th conference with international participation
Srní
November 8 - 10, 2021

Assessment of Python tensor contraction packages in the context of finite element evaluations

R. Cimrman[a,b]

[a] New Technologies Research Centre, University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic
[b] Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic

Fast evaluation of the weak formulation integral forms in finite element calculations is of crucial importance, especially when using a higher order approximation of the unknown and test fields. The integral forms are usually evaluated using nested loops over elements, and over quadrature points. Many such forms (e.g. all linear or multi-linear) can be written using a single tensor contraction expression using the Einstein summation convention. This convention, abbreviated as *einsum* in the following text according to the eponymous function of NumPy [7], allows a concise ways of writing tensor expressions.

Unlike large tensor networks in machine learning or quantum physics calculations, see e.g. [5, 11], the einsum expressions in the FE context consist usually of only several tensors and the contractions have low numerical intensity. This complicates use of modern computer architectures and is subject to an ongoing research [13]. Nevertheless, einsums are successfully used, for example, in the pure finite element package scikit-fem [6].

In the contribution we will comment on our recent results published in [2], where we assess several Python scientific computing packages implementing optimized tensor contractions. The tested packages are

- NumPy [7], with the basic implementation and some optimizations from opt_einsum;
- opt_einsum [12], with state-of-the-art contraction optimization strategies;
- Dask [4], which allows parallel out-of-core calculations with very large data.
- JAX [1], with JIT compilation[1] and possible parallel execution or automatic GPU transfer.

In [2] we proposed a simple "weak form to einsum" transpiler[2] from generalized einsum-like expressions (see Table 1). The transpiler allows simple definitions of multi-linear finite element weak forms as evidenced in Table 2. It supports several einsum evaluation backends implemented using the packages listed above, arbitrary memory layout of operands, easy automatic differentiation due to (multi-)linearity of the considered weak forms and various evaluation modes.

The transpiler based weak forms are available in SfePy from version 2021.1, allowing a rapid prototyping of multi-physical finite element models and subsequent calculations in diverse fields such as biomechanics of liver [10] or solid state physics [9]. All data used in [2] are available online [3].

[1] JIT compilation = Just in time compilation.

[2] A transpiler translates input in a language to another language that works at approximately the same level of abstraction, unlike a traditional compiler that translates from a higher level programming language to a lower level programming language.

Table 1. The generalized einsum-like notation

symbol	meaning	example
`0`	scalar	p
`i`	i-th vector component	u_i
`i.j`	gradient: derivative of i-th vector component w.r.t. j-th coordinate component	$\frac{\partial u_i}{\partial x_j}$
`i:j`	symmetric gradient	$\frac{1}{2}(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i})$
`s(i:j)->I`	vector storage of symmetric second order tensor, I is the vector component	Cauchy strain tensor $e_{ij}(\underline{u})$

Table 2. Examples of multi-linear weak form definitions

description	definition	weak form expression
vector dot product	`('i,i', v, u)`	$\int_T v_i u_i$
weighted vector dot product	`('ij,i,j', M, v, u)`	$\int_T v_i M_{ij} u_j$
weak Laplacian	`('0.i,0.i', v, u)`	$\int_T \frac{\partial v}{\partial x_i} \cdot \frac{\partial u}{\partial x_i}$
Navier-Stokes convection	`('i,i.j,j', v, u, u)`	$\int_T v_i \frac{\partial u_i}{\partial x_j} u_j$
Stokes coupling	`('i.i,0', v, p)`	$\int_T \frac{\partial v_i}{\partial x_i} p$
transposed Stokes coupling	`('i.i,0', u, q)`	$\int_T q \frac{\partial u_i}{\partial x_i}$
divergence operator	`('i.i', v)`	$\int_T \frac{\partial v_i}{\partial x_i}$
linear elasticity	`('IK,s(i:j)->I,s(k:l)->K', D, v, u)`	$\int_T D_{ijkl}\, e_{ij}(\underline{v}) e_{kl}(\underline{u})$
Cauchy stress	`('IK,s(k:l)->K', D, u)`	$D_{ijkl}\, e_{kl}(\underline{u})$

The calculation speed and memory requirements of the transpiled weak forms, in comparison with the original C implementation of those forms as available in SfePy, will be presented. To provide a broader context, the serial performance of the reference SfePy implementation will be also compared to the finite element framework FEniCS [8].

Acknowledgement

The work was supported from European Regional Development Fund — Project "Application of Modern Technologies in Medicine and Industry" (No. CZ.02.1.01/0.0/0.0/17_048/0007280).

References

[1] Bradbury, J., Frostig, R., et al., JAX: Composable transformations of Python+NumPy programs, https://github.com/google/jax, 2021. Ver. 0.2.9

[2] Cimrman, R., Fast evaluation of finite element weak forms using python tensor contraction packages, Advances in Engineering Software 159 (2021) No. 103033.

[3] Cimrman, R., Performance measurements of Python tensor contraction packages in the finite element context, Data set, Zenodo, 2021, doi: 10.5281/zenodo.4750560.

[4] Dask Development Team, Dask: Library for dynamic task scheduling, 2016.

[5] Gray, J., Kourtis, S., Hyper-optimized tensor network contraction, Quantum 5 (2021) No. 410.

[6] Gustafsson, T., McBain, G., Scikit-fem: A Python package for finite element assembly, Journal of Open Source Software 5 (52) (2020) No. 2369.

[7] Harris, C. R., Millman, K. J., et al., Array programming with NumPy, Nature 585 (7825) (2020) 357-362.

[8] Logg, A., Mardal, K. A., Wells, G. (editors), Automated solution of differential equations by the finite element method: The FEniCS book, Lecture Notes in Computational Science and Engineering, Berlin Heidelberg, Springer-Verlag, 2012.

[9] Novák, M., Vackář, J., Cimrman, R., Evaluating Hellmann–Feynman forces within non-local pseudopotentials, Computer Physics Communications 250 (2020) No. 107034.

[10] Rohan, E., Turjanicová, J., Lukeš, V., Multiscale modelling and simulations of tissue perfusion using the Biot-Darcy-Brinkman model, Computers & Structures 251 (2021) No. 106404.

[11] Schindler, F., Jermyn, A. S., Algorithms for tensor network contraction ordering, Machine Learning: Science and Technology 1 (3) (2020) No. 035001.

[12] Smith, D. G. A., Gray, J., opt_einsum – a python package for optimizing contraction order for einsum-like expressions, Journal of Open Source Software 3 (26) (2018) No. 753.

[13] Świrydowicz, K., Chalmers, N., et al., Acceleration of tensor-product operations for high-order finite element methods, The International Journal of High Performance Computing Applications 33 (4) (2019) 735-757.