The Journal of Open Source Software

BasicTools: a numerical simulation toolbox

Felipe Bordeu [1¶], **Fabien Casenave** [1], **and Julien Cortial** [1]

1 Safran Tech, Digital Sciences & Technologies Department, Rue des Jeunes Bois, Châteaufort, 78114 Magny-Les-Hameaux, France ¶ Corresponding author

DOI: 10.21105/joss.05142

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @hvonwah
- @sthavishtha

Submitted: 07 February 2023
Published: 23 June 2023

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Numerical simulations of physical phenomena can be computed by many (commercial/free) software packages, but despite the apparent variety, all of them rely on a relatively small set of operations during the preparation, exploitation, and post-processing of these simulations, e.g., handling and modifying meshes and fields. BasicTools is a Python library designed to address these supporting tasks. It features an efficient data model for meshes and field objects, as well as input/output routines compatible with various formats. A finite element engine allows it to assemble abstract variational formulations and integrate fields on volumes and surfaces.

BasicTools is actively used in artificial intelligence and model order reduction (Akkari et al., 2021; Daniel et al., 2020, 2021, 2022), topology optimization (Nardoni et al., 2022), and material sciences (Proudhon, 2013-present) projects.

Statement of need

Industrial design tasks often rely on numerical simulation workflows involving different software packages, each providing its own specific post-processing tools. Common tasks like transferring computed fields from one tool to another must be routinely implemented, with subtle variations. This limits interoperability and increases complexity.

BasicTools is a solution to these concerns. It introduces a data model for meshes and related physical fields that can be populated using different readers and exported using various writers: no new mesh or solution format is forced upon the user. The data-oriented design of BasicTools allows high performance operations using a high-level language (Python with NumPy). BasicTools allows users to convert meshes to other "in-memory" formats (VTK (Schroeder et al., 2006), PyVista (Sullivan & Kaszynski, 2019), MeshIO (Schlömer, 2015-present), CGNS (CGNS contributors, 1994-present), and Gmsh (Geuzaine & Remacle, 2009)), enabling mixing (and reusing) the various treatments available in other frameworks. Other features available in BasicTools include various mesh handling routines, field transfer operators, and a flexible finite element engine.

State of the field

In the computational fluid dynamics community, the CFD General Notation System (CGNS) (CGNS contributors, 1994-present) format is a de-facto standard. However, to the authors' knowledge, no such standard exists for solid mechanics. One may consider VTK and MeshIO for mesh manipulation and file format conversion, respectively, but the post-processing of integration point data, a key requirement in solid mechanics, would not be possible. Most available tools implement the simple, but potentially dangerous, approach of extrapolating the integration point values to the nodes of the mesh or averaging in every cell. This can lead to a misinterpretation of the solution and incorrect engineering decisions. Also, only a few finite

element engines allow assembling abstract variational formulations on arbitrary geometries, like FreeFem++ (Hecht, 2012) or FEniCS (Alnæs et al., 2015), amongst others.

Overview

The main features of BasicTools are:

- Meshes (in the `Containers` module): `ConstantRectilinearMesh` and `UnstructuredMesh` encapsulate respectively the data model for constant rectilinear and unstructured mesh types. Unstructured meshes are efficient: elements are stored using only one array for each element type. Both mesh types can feature nodes and element tags. Many functions are available for creating, cleaning and modifying meshes (e.g., field projection and mesh morphing).
- Filters (in the `Containers` module): Various types of `ElementFilters` and `NodeFilters` allow it to handle subparts of meshes by selecting element- and node-sets using threshold functions, tags, element types, element dimensionality, and masks. Filters can be combined using Boolean operations (union, complementary, ...).
- A finite element engine (in the `FE` module): A general weak formulation engine able to integrate fields over parts of the meshes is available. The `FETools` submodule contains specific functions for Lagrange P1 finite elements, including the computation of stiffness and mass matrices. The domain of integration is defined using `ElementFilters`, making the integration domain flexible. P0 and P2 Lagrange finite element spaces are implemented and tested. The framework is non-isoparametric: the user can write weak formulations mixing P0, P1, and P2 fields on P1 or P2 meshes.
- Input/Output functions (in the `IO` module): Various readers (alternatively, writers) for importing (alternatively, exporting) meshes and solution fields from (alternatively, to) BasicTools' internal data model are available. Supported formats include geo/geof (Z-set (Mines ParisTech & ONERA the French aerospace lab, 1981-present)), VTK, XDMF, SAMCEF, and ABAQUS, and a bridge with MeshIO is provided. Readers for the ABAQUS and SAMCEF proprietary formats are also enabled when properly licensed software is available locally. See BasicTools documentation for more details.
- Implicit geometry engine (in the `ImplicitGeometry` module): Arbitrary subdomains can be defined using implicit geometries (level-set function). Basic shapes (spheres, half-spaces, cylinders, cubes), transformations (symmetry, translation, rotation) and binary operators (union, difference, and intersection) can be used to construct complex shapes. These shapes can be used to select elements (using `ElementFilter`), or be evaluated on point clouds to explicitly construct iso-zero surfaces.
- Linear algebra functions (in the `Linalg` module): Some common operations on linear systems for finite elements are implemented: penalization, elimination, Lagrange multipliers, and the Ainsworth (Ainsworth, 2001) method to impose essential boundary conditions or linear multi-point constraints. The submodule `LinearSolver` offers an abstraction layer for sparse linear solvers, including: Cholesky of the `sksparse` package; factorized, CG, lsqr, gmres, lgmres of the `scipy.sparse.linalg` module; CG, LU, BiCGSTAB, SPQR of the C++ Eigen library; and the AMG solver of `pyamg` package.

The large majority of functions are illustrated in the same file where they are defined, in `CheckIntegrity` functions.

Examples

We present two examples; see BasicTools documentation for complete details.

Pre/post deep learning

Convolution-based deep learning algorithms generally rely on structured data. BasicTools can be used to transfer a field computed on an unstructured mesh using finite elements to a structured grid and vice versa. To validate the operation, the error on the final field is evaluated with respect to the original field.

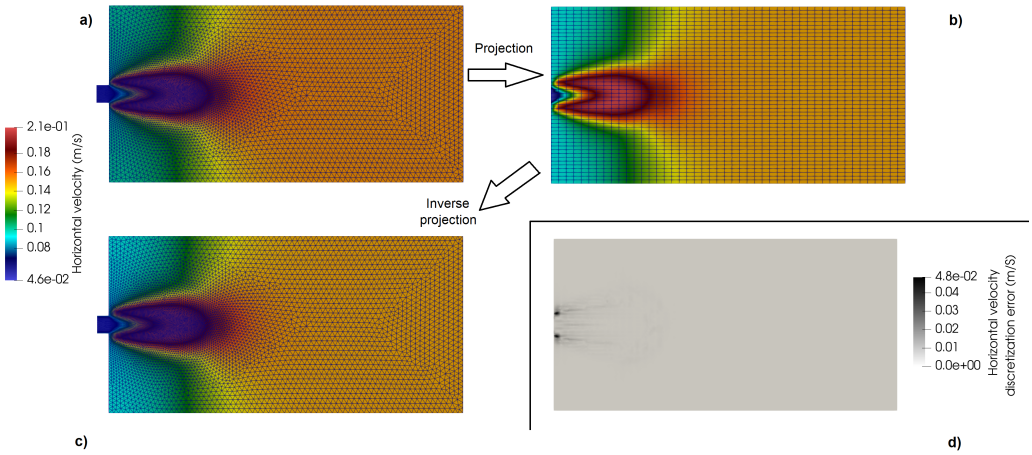

Figure 1: Deep learning workflow coupled to finite element simulator a) Initial field on unstructured mesh, b) transferred field into regular grid (projection step), c) inverse projection into original unstructured mesh, d) projection error on unstructured mesh.

Mechanical analysis: Thick plate with two inclusions

Consider a thick plate with two inclusions, one softer and the other stiffer than the base material. The plate is clamped on the left side with a constant traction applied on the right side. We compute the strain energy on only one inclusion. The linear elasticity problem is solved using P1 Lagrange finite elements on an unstructured mesh.

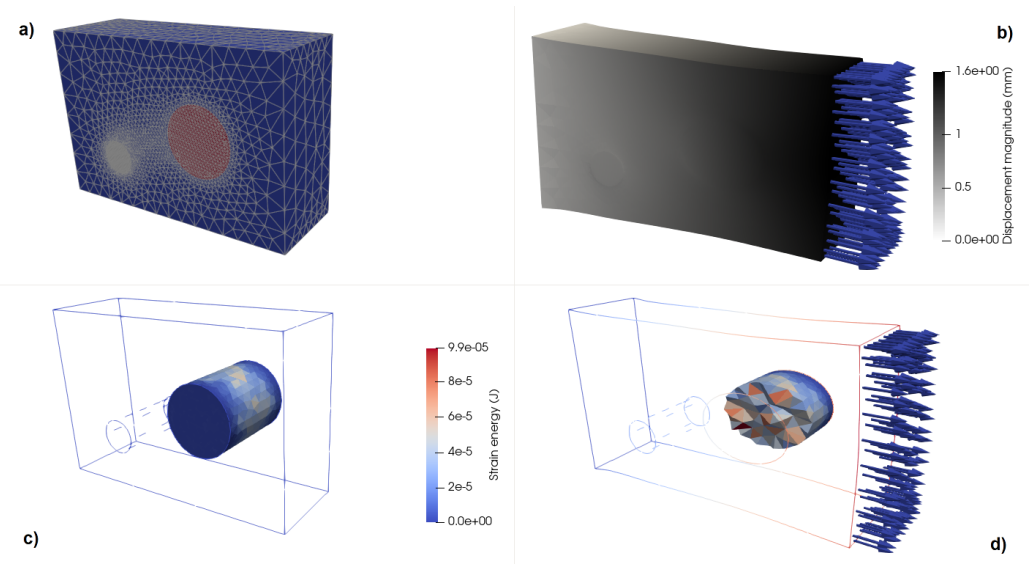

Figure 2: Analysis of a mechanical thick plate with two inclusions a) illustration of the mesh with highlighting of the two inclusions, b) magnitude of the displacement solution on the deformed mesh (with applied traction illustrated), c) strain energy in the large inclusion, d) cutaway view of the strain energy in the large inclusion (with applied traction illustrated).

References

Ainsworth, M. (2001). Essential boundary conditions and multi-point constraints in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, *190*(48), 6323–6339. https://doi.org/10.1016/S0045-7825(01)00236-5

Akkari, N., Casenave, F., Daniel, T., & Ryckelynck, D. (2021). Data-targeted prior distribution for variational AutoEncoder. *Fluids*, *6*(10). https://doi.org/10.3390/fluids6100343

Alnæs, M., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C., Ring, J., Rognes, M. E., & Wells, G. N. (2015). The FEniCS project version 1.5. *Archive of Numerical Software*, *3*(100). https://doi.org/10.11588/ans.2015.100.20553

CGNS contributors. (1994-present). *CFD general notation system*. http://cgns.github.io/

Daniel, T., Casenave, F., Akkari, N., & Ryckelynck, D. (2020). Model order reduction assisted by deep neural networks (ROM-net). *Adv. Model. And Simul. In Eng. Sci.*, *7*(16). https://doi.org/10.1186/s40323-020-00153-6

Daniel, T., Casenave, F., Akkari, N., & Ryckelynck, D. (2021). Data augmentation and feature selection for automatic model recommendation in computational physics. *Math. Comput. Appl.*, *26*(1). https://doi.org/10.3390/mca26010017

Daniel, T., Casenave, F., Akkari, N., Ryckelynck, D., & Rey, C. (2022). Uncertainty quantification for industrial numerical simulation using dictionaries of reduced order models. *Mechanics & Industry*, *23*, 3. https://doi.org/10.1051/meca/2022001

Geuzaine, C., & Remacle, J.-F. (2009). Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, *79*(11), 1309–1331. https://doi.org/10.1002/nme.2579

Hecht, F. (2012). New development in FreeFem++. *Journal of Numerical Mathematics*, *20*(3-4), 251–266. https://doi.org/10.1515/jnum-2012-0013

Mines ParisTech, & ONERA the French aerospace lab. (1981-present). *Zset: Nonlinear material & structure analysis suite*. http://www.zset-software.com

Nardoni, C., Danan, D., Mang, C., Bordeu, F., & Cortial, J. (2022). A R&D software platform for shape and topology optimization using body-fitted meshes. In R. Sevilla, S. Perotto, & K. Morgan (Eds.), *Mesh generation and adaptation: Cutting-edge techniques* (pp. 23–39). Springer International Publishing. https://doi.org/10.1007/978-3-030-92540-6_2

Proudhon, H. (2013-present). *Pymicro*. https://github.com/heprom/pymicro

Schlömer, N. (2015-present). *meshio: Tools for mesh files*. https://github.com/nschloe/meshio

Schroeder, W., Martin, K., & Lorensen, B. (2006). *Visualization Toolkit: An object-oriented approach to 3D graphics* (Fourth). Kitware, Inc. https://vtk.org/

Sullivan, C. B., & Kaszynski, A. (2019). PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software*, *4*(37), 1450. https://doi.org/10.21105/joss.01450