# Dxtr: A Python package for Discrete Exterior Calculus.

Olivier Ali ⓘ [1]

1 Laboratoire Reproduction et Développement des Plantes, Université de Lyon, ENS de Lyon, UCB Lyon 1, CNRS, INRAE, INRIA, F-69342 Lyon, France.

Summary

Dxtr is an open-source Python library designed to implement *Discrete Exterior Calculus (DEC)*. The library has two primary objectives: *(i)* to provide a precise and reliable implementation of simplicial complexes, cochains, and differential operators in arbitrary dimensions; and *(ii)* to offer an intuitive and user-friendly interface, enabling non-specialists to easily explore and experiment with DEC.

Statement of need

Nearly two decades ago, Desbrun and collaborators introduced the theory of Discrete Exterior Calculus (Desbrun et al., 2006), a discretized counterpart to *Exterior Calculus* (Cartan, 1971). Initially developed in the context of computer graphics, interest in *DEC* quickly spread to other scientific fields requiring the discretization of field theories for numerical simulations, such as electromagnetic theory (Chen & Chew, 2017) and elasticity theory (Kanso et al., 2007; Rashad et al., 2023). The strength of *DEC* lies in its algebraic organization of differential geometry, which can be viewed as a *"exotic"* form of finite element method (Arnold et al., 2006), where degrees of freedom are not only assigned to mesh nodes but also to higher-order topological elements (e.g., edges, triangles, tetrahedra).

Despite its significant potential and utility, interest *DEC* has yet to reach a broad audience and remains primarily the domain of applied mathematicians and scientific computing experts. This limited adoption is partly due to the steep learning curve required to understand both its theoretical foundations as well as its numerical implementation. The **Dxtr** library addresses this challenge by offering an intuitive implementation of *DEC* in Python, a widely-used and accessible programming language. To further ease the learning process, the library is accompanied by comprehensive documentation.

In 2012, Bell and Hirani introduced **PyDEC** (Bell & Hirani, 2012), the first open-source Python library for *DEC*. Since then, only a few Python-based alternatives have been developed (Eftimiades, 2014; Sellán et al., 2023), making **PyDEC** the primary reference for those willing to explore *DEC*. The **Dxtr** library builds upon **PyDEC**'s foundational work, particularly in its methods for instantiating basic data structures (e.g., `AbstractSimplicialComplex`). However, it extends functionality by including features such as implementations of the *wedge product* and *musical isomorphisms*. Furthermore, **Dxtr** adopts a design philosophy rooted more closely in the mathematical principles underlying *DEC*, making it both more intuitive and user-friendly.

Library general description

The cornerstone of *DEC* is the (discrete) Stokes-Cartan theorem:

$$\int_{\partial(\sigma)} \omega = \int_{\sigma} d(\omega), \forall \sigma \subset \mathcal{K}, \tag{1}$$

where, in the discrete version, ω and σ respectively depict a *k*-cochain, accounting for a discrete *k*-form, and a *(k+1)*-chain, a subset of the nD simplicial complex $\mathcal{K}$ ($k < n$); while the $d(\cdot)$ and $\delta(\cdot)$ operators stand for the discrete exterior derivative and boundary operators. The **Dxtr** library has been crafted around three main modules encapsulating these core mathematical concepts, see Figure 1:

1. The `complexes` module, where the `AbstractSimplicialComplex` and `SimplicialComplex` classes correspond to combinatorial objects grasping respectively the topological and geometrical properties of simplicial complexes
2. The `cochains` module that includes the `Cochain` class, implementation of the discrete versions of differential forms defined on simplicial complexes, as well as the `WhitneyMap` class to linearly interpolate discrete *k*-forms on top-level simplices
3. Finally, the `operators` module where methods defining operators acting on `Cochain` instances can be found; this module is organized into thematic submodules dedicated to specific types of operators, *e.g.* the `operators.differential` submodule regrouping implementations of the various differential operators such as the `exterior_derivative()` or the `codifferential()` ones

To support these core modules, the library also encompasses a `math` module aggregating math-related functions called by the various classes and methods of the library. Finally, three "housekeeping" modules add on functionalities: `io` to read/write files on disk, `visu` to generate visual representations of the various objects and `utils` where some useful data structures, decorators and data ubiquitously used within the library are stored.
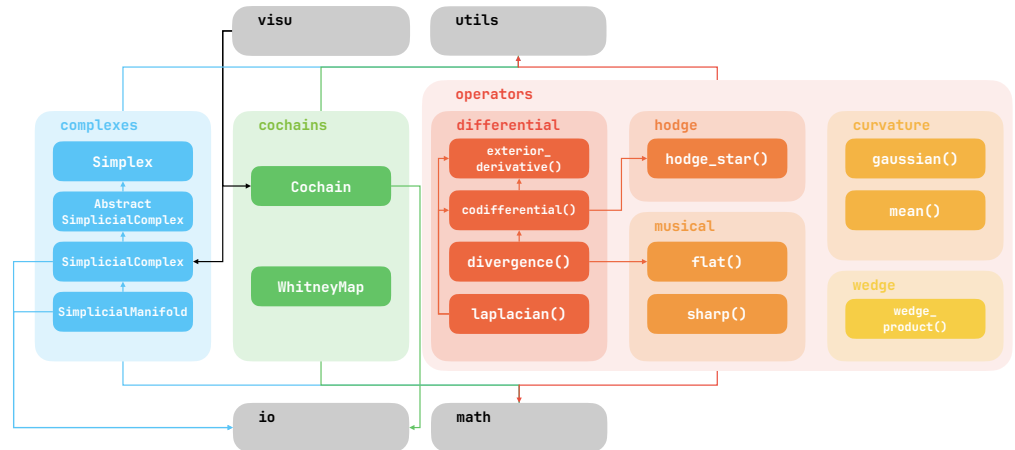

Figure 1: Schematic representation of the Dxtr library Architecture: Dimmed color boxes figure modules and submodules, plain color ones the main classes or functions they contain. Colored lines depict functional dependencies between classes, functions or modules.

Scope and range

In terms of geometrical structures: In its current version, the **Dxtr** library enables the generation of simplicial complexes of arbitrary dimensions. These complexes can either be embedded in a

Euclidean space of arbitrary dimension or remain abstract[1].

In terms of cochain algebraic manipulations: Once a n-dimensional simplicial complex is defined (abstract or not), k-cochains ($k \leq n$) can be defined on it. A cochain can be defined as scalar-valued or vector-valued. On `SimplicialManifold` instances, dual cochains can be defined and manipulated as well. Discrete vector fields can be defined as vector-valued dual 0-cochains. Basic algebraic manipulations are readily available for all these objects as overloading of basic binary operators (+, -, *[2]).

Visual representations[3] of simplicial complexes, cochains, and vector fields can be performed, assuming that these structures are embedded within the usual 3D Euclidean space.

In terms of operators: Several differential operators are directly available: the exterior derivative on any `SimplicialComplex` instance; the codifferential and the Laplacian on `SimplicialManifold` instances. The Hodge star operator is also available on `SimplicialManifold` instances as well as musical isomorphisms to convert vector fields into cochains and vice versa. A wedge product implementation is also available but limited to primal `Cochain` instances. Finally, some basic curvature operators (Gaussian curvature and Mean curvature) have also been implemented.

Combined together, this corpus enables **Dxtr** users to address some discrete differential geometry problems, such as numerical estimation of Boundary Value Problems or Spectral analysis of differential operators on simplicial manifolds, c.f., the "Hands on Dxtr" section in the online documentation.

Future developments

It is worth noticing that the **Dxtr** remains in an active state of development. Features currently in development include: (i) extension of the wedge product to dual cochain, which will not only expand the algebraic abilities of the library but also enable the definition of a discrete version of the Lie derivative for Cochains, and (ii) extension of the curvature module to encompass more general curvature-related operators.

Acknowledgements

References

Arnold, D. N., Falk, R. S., & Winther, R. (2006). Finite element exterior calculus, homological techniques, and applications. *Acta Numerica*, *15*, 1–155. https://doi.org/10.1017/s0962492906210018

Bell, N., & Hirani, A. N. (2012). PyDEC. *ACM Transactions on Mathematical Software*, *39*(1), 1–41. https://doi.org/10.1145/2382585.2382588

Cartan, E. (1971). *Les systèmes différentiels extérieurs et leurs applications géométriques* (p. 210). Hermann & cie.

---

[1] i.e., without any geometrical embedding.
[2] Here * refers to the scalar multiplication, not the multiplication between cochain objects.
[3] These visual representations rely on two external libraries: `plotly` and `pyvista`.

Chen, S. C., & Chew, W. C. (2017). Electromagnetic theory with discrete exterior calculus. *Progress In Electromagnetics Research*, *159*, 59–78. https://doi.org/10.2528/pier17051501

Desbrun, M., Hirani, A. N., & Leok, M. (2006). *Discrete exterior calculus, preprint (2005)*. https://doi.org/10.48550/arXiv.math/0508341

Eftimiades, A. (2014). Kahler: An implementation of discrete exterior calculus on Hermitian manifolds. *arXiv*. https://doi.org/10.48550/arxiv.1405.7879

Kanso, E., Arroyo, M., Tong, Y., Yavari, A., Marsden, J. G., & Desbrun, M. (2007). On the geometric character of stress in continuum mechanics. *Zeitschrift Für Angewandte Mathematik Und Physik*, *58*(5), 843–856. https://doi.org/10.1007/s00033-007-6141-8

Rashad, R., Brugnoli, A., Califano, F., Luesink, E., & Stramigioli, S. (2023). Intrinsic nonlinear elasticity: An exterior calculus formulation. *Journal of Nonlinear Science*, *33*(5), 84. https://doi.org/10.1007/s00332-023-09945-7

Sellán, S., Stein, O., & others. (2023). *gpytoolbox: A Python geometry processing toolbox*. https://gpytoolbox.org/