Basix: a runtime finite element basis evaluation library

Matthew W. Scroggs[2], **Igor A. Baratta**[2], **Chris N. Richardson**[1], and **Garth N. Wells**[2]

1 BP Institute, University of Cambridge **2** Department of Engineering, University of Cambridge

Summary

The finite element method (FEM) (Ciarlet, 1978) is a widely used numerical method for approximating the solution of partial differential equations (PDEs). Solving a problem using FEM involves discretising the problem and searching for a solution in a finite dimensional space: these finite spaces are created by defining a finite element on each cell of a mesh.

Following Ciarlet (1978), a finite element is commonly defined by a triple $(R, \mathcal{V}, \mathcal{L})$, where:

- R is the reference cell, for example a triangle with vertices at (0,0), (1,0) and (0,1);
- $\mathcal{V}$ is a finite dimensional polynomial space, for example $\mathrm{span}\{1, x, y, x^2, xy, y^2\}$;
- $\mathcal{L}$ is a basis of the dual space $\{f : \mathcal{V} \to \mathbb{R} \,|\, f \text{ is linear}\}$, for example the set of functionals that evaluate a function at the vertices of the triangle and at the midpoints of its edges.

The basis functions of the finite element are the polynomials in $\mathcal{V}$ such that one functional in $\mathcal{L}$ gives the value 1 for that function and all other functions in $\mathcal{L}$ give 0. The examples given above define a degree 2 Lagrange space on a triangle; the basis functions of this space are shown in Figure 1.
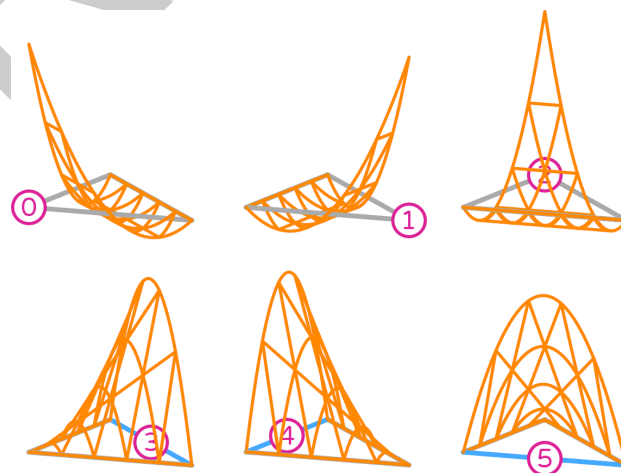
Figure 1: The six basis functions of an order 2 Lagrange space on a triangle. The upper three functions arise from point evaluations at the vertices. The lower three arise from point evaluations at the midpoints of the edges. These diagrams are taken from DefElement (The DefElement contributors, 2021).

The functionals in $\mathcal{L}$ are each associated with a degree of freedom (DOF) of the finite element space. Each functional (or DOF) is additionally associated with a sub-entity of the

²¹ reference cell. Ensuring that the same coefficients are assigned to the DOFs of neighbouring
²² cells associated with a shared sub-entity gives the finite element space the desired continuity
²³ properties.

²⁴ Basix is a C++ library that creates and tabulates a range of finite elements on triangles,
²⁵ tetrahedra, quadrilaterals, hexahedra, pyramids, and prisms. A full list of currently supported
²⁶ elements is included below.

²⁷ For many elements, the functionals in $\mathcal{L}$ are defined to be integrals on a sub-entity of the
²⁸ cell. Basix provides a range of quadrature rules to compute these integrals, including Gauss–
²⁹ Jacobi, Gauss–Lobatto–Legendre, and Xiao–Gimbutas (Xiao & Gimbutas, 2010). Internally,
³⁰ Basix uses xtensor (Mabille et al., 2021) for matrix and tensor storage and manipulation. The
³¹ majority of Basix's functionality can be used via the library's Python interface.

³² Basix forms part of FEniCSx alongside DOLFINx (Wells, Ballarin, et al., 2021), FFCx (Wells,
³³ Baratta, et al., 2021), and UFL (Alnæs et al., 2014). FEniCSx is the latest development
³⁴ version of FEniCS, a popular open source finite element project (Alnæs et al., 2015).

³⁵ Statement of need

³⁶ Basix allows users to:

- ³⁷ evaluate finite element basis functions and their derivatives at a set of points;
- ³⁸ access geometric and topological information about reference cells;
- ³⁹ apply push forward and pull back operations to map data between a reference cell and
  ⁴⁰ a physical cell;
- ⁴¹ permute and transform DOFs to allow higher-order elements to be used on arbitrary
  ⁴² meshes; and
- ⁴³ interpolate into a finite element space and between finite element spaces.

⁴⁴ In many FEM libraries, the definitions of elements are included within the code of the library
⁴⁵ rather than separating the element definition and tabulation into a standalone library as we do.
⁴⁶ Following the latter approach allows us to adjust how elements are implemented and add new
⁴⁷ elements to Basix without needing to make changes to the rest of the library. This also allows
⁴⁸ users who want to create custom integration kernels to get information about elements from
⁴⁹ Basix without having to extract information from the core of the full finite element library.

⁵⁰ The Python library FIAT (Kirby, 2004) (which is part of the legacy FEniCS library alongside
⁵¹ UFL, FFC (Logg et al., 2012) and DOLFIN (Logg & Wells, 2010)) serves a similar purpose as
⁵² Basix and can perform many of the same operations (with the exception of permutations and
⁵³ transformations) on triangles, tetrahedra, quadrilaterals, and hexahedra. As FIAT is written
⁵⁴ in Python, the FFC library would use the information from FIAT to generate code that could
⁵⁵ be used by the C++ finite element library DOLFIN.

⁵⁶ An advantage of using Basix is the ability to call functions from C++ at runtime. This has
⁵⁷ allowed us to greatly reduce the amount of code generated in FFCx compared to FFC, as well
⁵⁸ as simplifying much of the implementation, while still allowing FFCx to access the information
⁵⁹ it needs using Basix's Python interface.

⁶⁰ Another key advantage of Basix is its support for permuting and transforming DOFs for higher-
⁶¹ order elements. As described in Scroggs et al. (2021), these operations are necessary when
⁶² solving problems on arbitrary meshes, as differences in how neighbouring cells orient their
⁶³ sub-entities can otherwise cause issues.

Supported elements

Interval

In Basix, the sub-entities of the reference interval are numbered as shown in Figure 2. The following elements are supported on an interval:

- Lagrange
- bubble
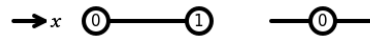- serendipity (Arnold & Awanou, 2011)

Figure 2: The numbering of a reference interval.

Triangle

In Basix, the sub-entities of the reference triangle are numbered as shown in Figure 3. The following elements are supported on a triangle:

- Lagrange
- Nédélec first kind (Nédélec, 1980)
- Raviart–Thomas (Raviart & Thomas, 1977)
- Nédélec second kind (Nédélec, 1986)
- Brezzi–Douglas–Marini (Brezzi et al., 1985)
- Regge (Christiansen, 2011; Regge, 1961)
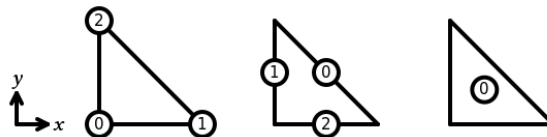- Crouzeix–Raviart (Crouzeix & Raviart, 1973)
- bubble

Figure 3: The numbering of a reference triangle.

Quadrilateral

In Basix, the sub-entities of the reference quadrilateral are numbered as shown in Figure 4. The following elements are supported on a quadrilateral:

- Lagrange
- Nédélec first kind
- Raviart–Thomas
- Nédélec second kind (Arnold & Awanou, 2014)
- Brezzi–Douglas–Marini (Arnold & Awanou, 2014)
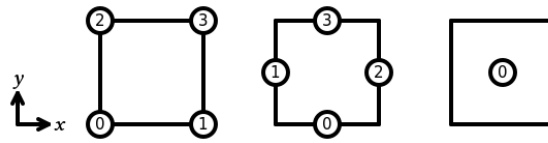- bubble
- DPC
- serendipity

Figure 4: The numbering of a reference quadrilateral.

Tetrahedron

In Basix, the sub-entities of the reference tetrahedron are numbered as shown in Figure 5. The following elements are supported on a tetrahedron:

- Lagrange
- Nédélec first kind
- Raviart–Thomas
- Nédélec second kind
- Brezzi–Douglas–Marini
- Regge
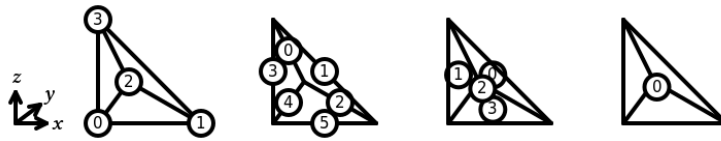- Crouzeix–Raviart
- bubble

Figure 5: The numbering of a reference tetrahedron.

Hexahedron

In Basix, the sub-entities of the reference hexahedron are numbered as shown in Figure 6. The following elements are supported on a hexahedron:

- Lagrange
- Nédélec first kind
- Raviart–Thomas
- Nédélec second kind
- Brezzi–Douglas–Marini
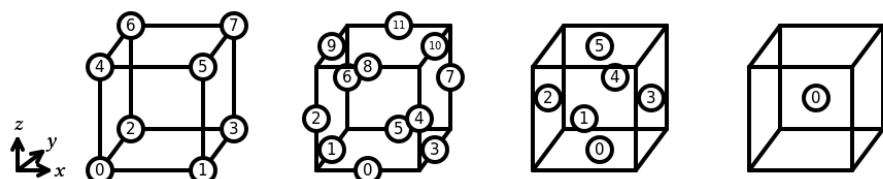- bubble
- DPC
- serendipity

Figure 6: The numbering of a reference hexahedron.

Prism

In Basix, the sub-entities of the reference prism are numbered as shown in Figure 7. The following elements are supported on a prism:

- Lagrange

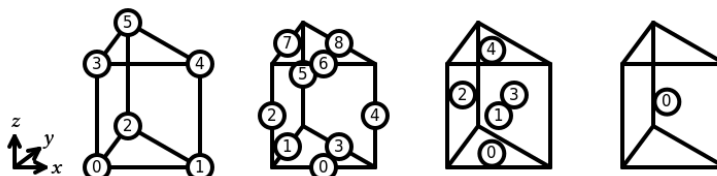**Figure 7:** The numbering of a reference prism.

Pyramid

In Basix, the sub-entities of the reference pyramid are numbered as shown in Figure 8. The following elements are supported on a pyramid:

- Lagrange

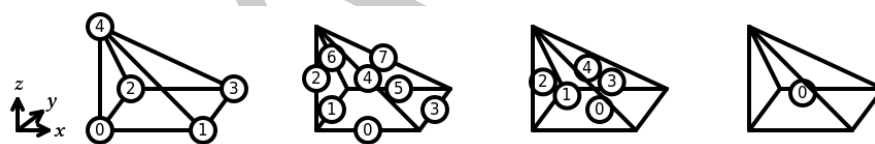**Figure 8:** The numbering of a reference pyramid.

References

Alnæs, M. S., Blechta, J., Hake, J., Johansson, A., Kehlet, B., Logg, A., Richardson, C. N., Ring, J., Rognes, M. E., & Wells, G. N. (2015). The FEniCS project version 1.5. *Archive of Numerical Software*, *3*(100), 9–23. https://doi.org/10.11588/ans.2015.100.20553

Alnæs, M. S., Logg, A., Ølgaard, K. B., Rognes, M. E., & Wells, G. N. (2014). Unified Form Language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software*. https://doi.org/10.1145/2566630

Arnold, D. N., & Awanou, G. (2011). The serendipity family of finite elements. *Foundations of Computational Mathematics*, *11*(3), 337–344. https://doi.org/10.1007/s10208-011-9087-3

Arnold, D. N., & Awanou, G. (2014). Finite element differential forms on cubical meshes. *Mathematics of Computation*, *83*, 1551–5170. https://doi.org/10.1090/S0025-5718-2013-02783-4

Brezzi, F., Douglas, J., & Marini, L. D. (1985). Two families of mixed finite elements for second order elliptic problems. *Numerische Mathematik*, *47*, 217–235. https://doi.org/10.1007/BF01389710

139 Christiansen, S. H. (2011). On the linearization of Regge calculus. *Numerische Mathematik*,
140   *119*(4), 613–640. https://doi.org/10.1007/s00211-011-0394-z

141 Ciarlet, P. G. (1978). *The finite element method for elliptic problems*. North-Holland.

142 Crouzeix, M., & Raviart, P.-A. (1973). Conforming and nonconforming finite element methods
143   for solving the stationary Stokes equations. *Revue Française d'Automatique, Informatique*
144   *Et Recherche Opérationnelle*, *3*, 33–75. https://doi.org/10.1051/m2an/197307R300331

145 Kirby, R. C. (2004). Algorithm 839: FIAT, a new paradigm for computing finite element
146   basis functions. *ACM Transactions on Mathematical Software*, *30*(4), 502–516. https:
147   //doi.org/10.1145/1039813.1039820

148 Logg, A., & Wells, G. N. (2010). DOLFIN: Automated finite element computing. *ACM*
149   *Transactions on Mathematical Software*, *37*. https://doi.org/10.1145/1731022.1731030

150 Logg, A., Ølgaard, K. B., Rognes, M. E., & Wells, G. N. (2012). *FFC: The FEniCS Form*
151   *Compiler*. 283–302. https://doi.org/10.1007/978-3-642-23099-8_11

152 Mabille, J., Beier, T., Brochart, D., Corlay, S., de Geus, T., Delsalle, A., Koethe, U., GitHub
153   user kolibri91, Prouvost, A., Renou, M., GitHub user SoundDev, Vollprecht, W., & Zhu,
154   J. (2021). *xtensor: C++ tensors with broadcasting and lazy computing*. https://github.
155   com/xtensor-stack/xtensor

156 Nédélec, J.-C. (1980). Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, *35*(3), 315–341.
157   https://doi.org/10.1007/BF01396415

158 Nédélec, J.-C. (1986). A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*,
159   *50*(1), 57–81. https://doi.org/10.1007/BF01389668

160 Raviart, P.-A., & Thomas, J.-M. (1977). A mixed finite element method for 2nd order elliptic
161   problems. In I. Galligani & E. Magenes (Eds.), *Mathematical aspects of finite element*
162   *methods* (Vol. 606, pp. 292–315).

163 Regge, T. (1961). General relativity without coordinates. *Il Nuovo Cimento*, *19*(3), 558–571.
164   https://doi.org/10.1007/BF02733251

165 Scroggs, M. W., Dokken, J. S., Richardson, C. N., & Wells, G. N. (2021). *Construction of*
166   *arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell*
167   *meshes*. http://arxiv.org/abs/2102.11901

168 The DefElement contributors. (2021). *DefElement: An encyclopedia of finite element defini-*
169   *tions*. https://defelement.com.

170 Wells, G. N., Ballarin, F., Baratta, I. A., Dean, J. P., Dokken, J. S., Hale, J. S., Habera,
171   M., Richardson, C. N., Scroggs, M. W., & Sime, N. (2021). *DOLFINx: Next generation*
172   *FEniCS problem solving environment*. https://github.com/FEniCS/dolfinx

173 Wells, G. N., Baratta, I. A., Habera, M., Hale, J. S., Richardson, C. N., & Scroggs, M. W.
174   (2021). *FFCx: Next generation FEniCS form compiler*. https://github.com/FEniCS/ffcx

175 Xiao, H., & Gimbutas, Z. (2010). A numerical algorithm for the construction of efficient
176   quadrature rules in two and higher dimensions. *Computers & Mathematics with Applica-*
177   *tions*, *59*(2), 663–676. https://doi.org/10.1016/j.camwa.2009.10.027