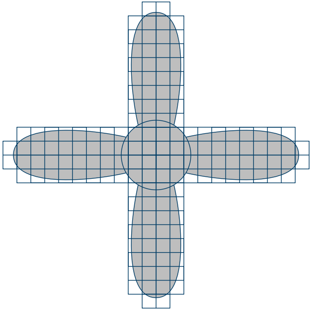
CutFEM-style methods in FEniCS-X: CAD and level sets

August Johansson, Vibeke Skytt
Mathematics and Cybernetics, SINTEF Digital

FEniCS 2021

2021-03-24

Finite element methods on non-matching meshes

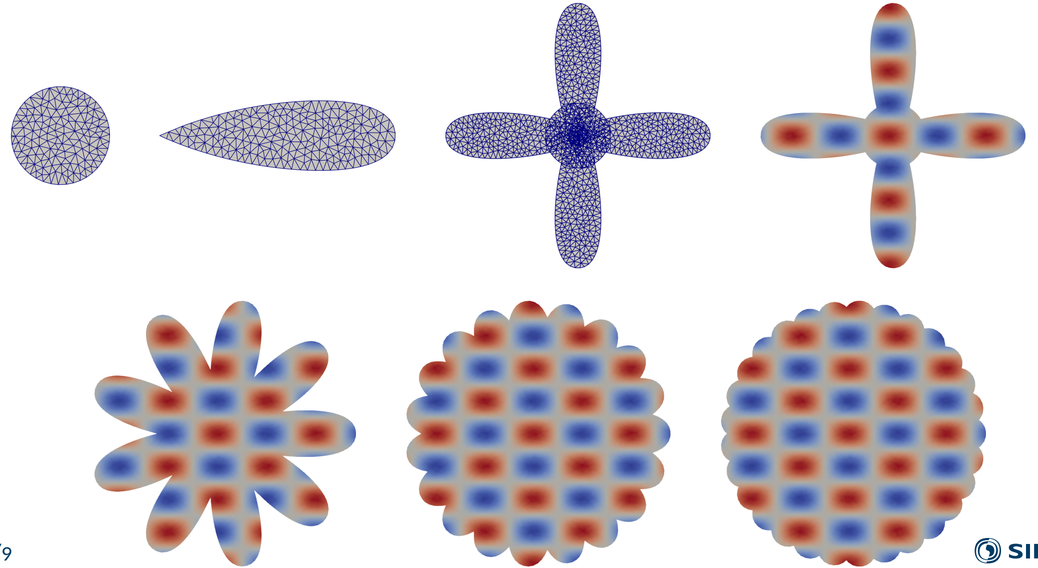


- Non-matching mesh construction is easy.
- The mesh is used for the finite element approximation.
- The mesh is **not** used for geometry approximation.
- Several methods exist: CutFEM, ϕ-FEM, TraceFEM, etc.
- Beneficial for problems with dynamic domains.
- **Cut elements** are $K : K \cap \partial\Omega \neq \emptyset$.
- Custom quadrature on the cut elements for

$$\int_{K\cap\Omega} dx \quad \text{and} \quad \int_{K\cap\partial\Omega} ds$$

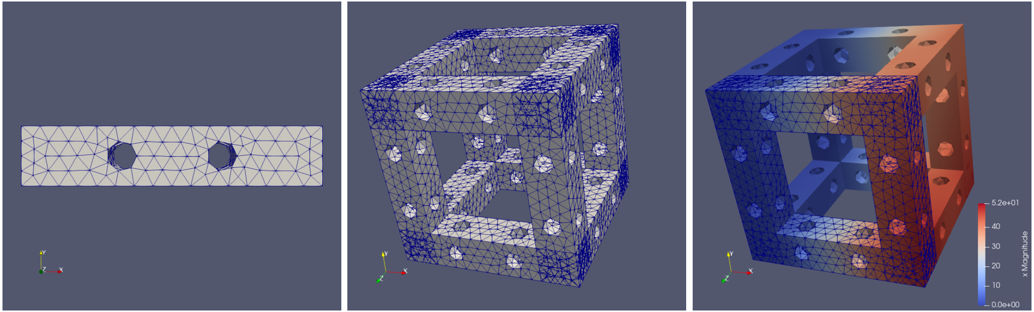
- **This talk**: a "library" with custom quadrature based on FEniCS-X.

Custom quadrature in FEniCS: MultiMesh 2D



SINTEF

Custom quadrature in FEniCS: MultiMesh 3D



SINTEF

Extensions to FFC-X & DOLFIN-X

The **FEniCS** pipeline has functionality for custom integrals:

```
void tabulate_tensor_custom(A, w, c, coord_dofs,
                            num_qr, qr_pts, qr_w, normals)
```

Extend **FFC-X**:

- Include metadata as

  ```
  ufl.dx(metadata={'quadrature_rule': 'runtime'}, domain=mesh)
  ```
- Make such integrals generate code for the custom integral type.
- Generate code for
  - calling `evaluate_basis_derivatives()`.
  - using the provided normals.

Extend **DOLFIN-X**:

- Add support for the custom integral type.
- Mimic setup for cell integrals.

SINTEF

Important high-level functionality

- **Custom assembler** mimicking examples from test_custom_assembler.py:

```
custom_assemble_matrix(form, [(cells, qr_pts, qr_w, normals)])
```

- Works as standard FE assembly:
  - Set up sparsity pattern (same as cell integral type).
  - Loop over cells (typically the cut cells).
  - Call the custom integral kernel.
  - Assemble into the global matrix (e.g. calling `MatSetValues`).
- Utilities for setting **mesh tags** from the geometry representation.
- Function for **locking inactive dofs** similar to `DirichletBC`.

SINTEF

Python example for testing bulk and surface quadrature

```
geom_kernel = Geometry('circle.iges')
cut_cells, uncut_cells = geom_kernel.cells()
qr_pts, qr_w = geom_kernel.bulk_qr()
qr_pts_surf, qr_w_surf, normals = geom_kernel.surf_qr()
bulk_data = (cut_cells, qr_pts, qr_w)
surf_data = (cut_cells, qr_pts_surf, qr_w_surf, normals)

cell_tags = get_cell_tags(mesh, uncut_cells, uncut_cell_tag=1)

dx_cut = ufl.dx(metadata={'quadrature_rule': 'runtime'}, domain=mesh)
dx_uncut = ufl.dx(subdomain_data=cell_tags, domain=mesh)

area = custom_assemble_scalar(1.0*dx_cut, [surf_data])
vol = custom_assemble_scalar(1.0*dx_cut, [bulk_data])
      + dolfinx.assemble_scalar(1.0*dx_uncut(uncut_cell_tag))
```

Obtain quadrature from external libraries

CAD: GoTools library

- `https://github.com/SINTEF-Geometry/GoTools`
- Spline geometry libraries by V. Skytt, T. Dokken et al (SINTEF).
- Quadrature created from tensor product-type constructions (work in progress).

Level set: Algoim library

- `https://algoim.github.io`
- Level set library by R. Saye (LBL).
- Provides accurate quadrature.

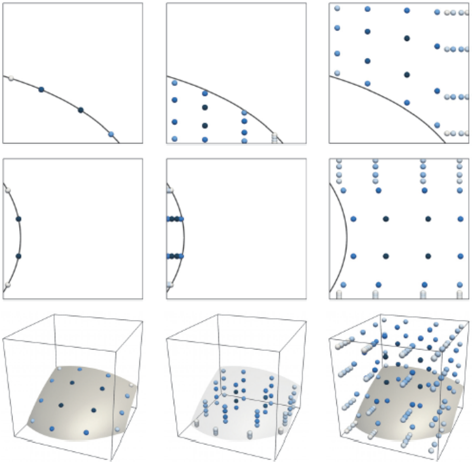


Figure: From `https://algoim.github.io` (with permission).

SINTEF

Example: CutFEM Poisson on a circle, **spline** geometry

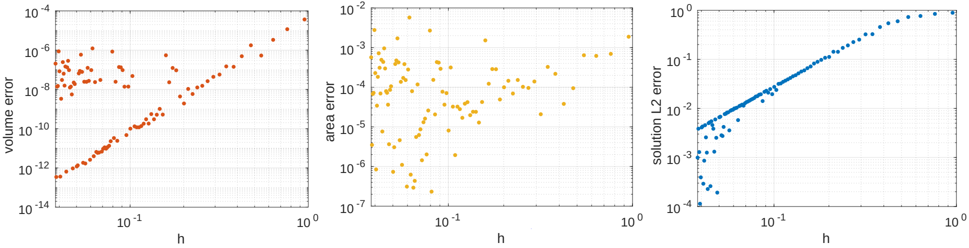


Figure: Relative volume error, relative area error and L^2 error.

Conclusions:

- Quadrature not correct.
- Do not see all errors in a weak norm.

Example: CutFEM Poisson on a circle, **level set** geometry

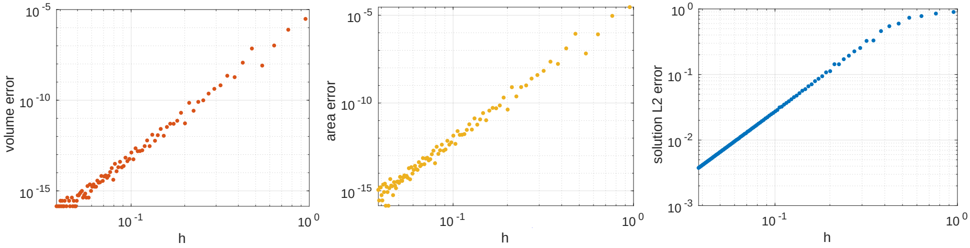


Figure: Relative volume error, relative area error and L^2 error.

Conclusions:

- Quadrature correct.
- Perfect convergence.

SINTEF

Technology for a better society