These are the supplementary materials for the poster [7] on fenicstools [6] presented by the authors on FEniCS'16 workshop.

CLÉMENT INTERPOLATION

The implementation of Clément interpolation in fenictools follows the exposition of the topic presented in [2, ch. 2.6]. Therein the interpolant is constructed by finding the best patch-wise constant approximation. However, this is only a special case in the theory of Clément interpolants [3]. We note that the treatment of H_0^1 interpolantion is discussed in [8]. Finally, we remark that the notion of finite element spaces on patches (more general polygonal domains) would allow for Virtual Finite Element Method [1] to be used within FEniCS.

Foundation for the averaging matrix A is the summation matrix, which is a $(0,1)$-matrix such that in j-th row the unit values are in columns that represent the cells that make up a patch surrounding the j-th degree of freedom. The matrix is constructed by using properties of the vertex quadrature. Let K a cell and $f \in P_1(K)$ then

$$\int_K f \,\mathrm{d}x = \frac{|K|}{n_v} \sum_{j=1}^{n_v} f(x_j),$$

where n_v is the number of vertices of the simplex. Further let ϕ_i^K the i-th basis function of linear Lagrange element over K. Then integrating $\frac{n_v}{|K|}\phi_i^K$ by the vertex quadrature gives the value one as desired. The averaging matrix is obtained by diagonally scaling the summation matrix by a vector of cell volumes. The total cost of constructing A is close to assembling the mass matrix for scalar CG_1 space.

In fenictools, the averaging matrix is constructed only for scalars. In order to interpolate tensor fields, each component is averaged and then assigned appropriately. The mapping from components to tensor is provided by DOLFIN's *FunctionAssigner* class and is built only once. Its construction is the final step of setting up the Clément interpolator.

Setting up *FunctionAssigner* makes up for a significant part of the total construction time of Clément infrastructure, cf. Table 1. Overall, the new method is about four times, see Table 2, more expensive to setup than L^2 interpolation (assembly of mass matrix + setting up conjugate gradient solver with multigrid preconditioning). However, with the action of Clément interpolator being more than eight times faster, the initial investment pays off during repetitive evaluation.

Table 1: MPI-averaged setup cost of Clément interpolant infrastructure. The assembly of the averaging operator (in seconds) as a unit time for computing the mappings for *FunctionAssigner*.

CPUs	Degrees of freedom				
	132098	526338	2101250	8396802	14612418
1	(0.42, 0.84)	(1.46, 0.65)	(3.65, 1.17)	(16.16, 1.93)	(31.05, 1.75)
2	(0.11, 0.79)	(0.26, 1.06)	(1.04, 1.40)	(4.37, 1.63)	(8.02, 1.30)
4	(0.04, 0.70)	(0.12, 0.84)	(0.28, 0.98)	(1.20, 1.06)	(2.26, 0.96)
8	(0.01, 0.61)	(0.04, 0.60)	(0.12, 0.76)	(0.53, 0.98)	(0.95, 0.94)
16	(0.00, 0.57)	(0.01, 0.70)	(0.04, 0.70)	(0.13, 0.95)	(0.23, 0.82)

Table 2: MPI-averaged setup cost (in seconds) of infrastructure for Clément interpolation and L^2 projection.

CPUs	Degrees of freedom				
	132098	526338	2101250	8396802	14612418
1	(0.87, 0.31)	(2.81, 0.76)	(8.78, 3.04)	(51.01, 14.98)	(91.94, 26.94)
2	(0.47, 0.17)	(1.23, 0.46)	(5.30, 1.75)	(24.03, 8.24)	(43.68, 16.20)
4	(0.32, 0.09)	(1.06, 0.36)	(2.73, 0.99)	(12.18, 4.43)	(21.94, 11.69)
8	(0.24, 0.06)	(0.67, 0.22)	(2.19, 0.77)	(10.09, 3.34)	(18.27, 5.74)
16	(0.16, 0.03)	(0.36, 0.10)	(1.32, 0.39)	(5.26, 1.65)	(9.29, 2.99)

LAGRANGIAN TRACKING

LagrangianParticles class is organized as a dictionary, in which a cell containing some of the tracked particles is a key and a list of the particles contained in the said cell is a value. Based on the motion of particles, new cells are added to or removed from the dictionary.

The data structure is very efficient in time integration where it eliminates the expensive search problem of locating for each particle a parent cell in which the velocity field is to be evaluated. In fenicstools the required cell is always available. Further advantage of the dictionary is a fact that for cell K with multiple particles a restriction of the velocity field, $\boldsymbol{u}|_K$, is performed only once.

The figures in the poster are part of the master thesis of Per Thomas Haga [4] who used *LagrangianParticles* to study drug delivery in spinal chord. The thesis developed into a recently submitted paper [5].

REFERENCES

[1] L. Beirão Da Veiga, F. Brezzi, A. Cangiani, G. Manzini, L. D. Marini, and A. Russo. Basic principles of virtual element methods. *Mathematical Models and Methods in Applied Sciences*, 23(01):199–214, 2013.

[2] D. Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 2001.

[3] P. Clément. Approximation by finite element functions using local regularization. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 9(R2):77–84, 1975.

[4] P. T. Haga. Numeriske simuleringer av adveksjons-dominert skalar-blanding anvendt på CSF-strømning og medikamenttransport. Master's thesis, University of Oslo, Norway, 2015.

[5] P.T. Haga, G. Pizzichelli, M. Mortensen, M. Kuchta, S. H. Pahlavian, E. Sinibaldi, B. Martin, and K.-A. Mardal. A numerical investigation of intrathecal drug and gene vector dispersion within the cervical subarachnoid space. *Plos One*, 2016. submitted.

[6] M. Mortensen and M. Kuchta. fenicstools. `https://github.com/mikaem/fenicstools`, 2016.

[7] M. Mortensen and M. Kuchta. fenicstools poster for fenics'16 workshop. `https://github.com/MiroK/fenics16-poster/blob/master/poster.pdf`, 2016.

[8] L. R. Scott and S. Zhang. Finite element interpolation of nonsmooth functions satisfying boundary conditions. *Mathematics of Computation*, 54(190):483–493, 1990.