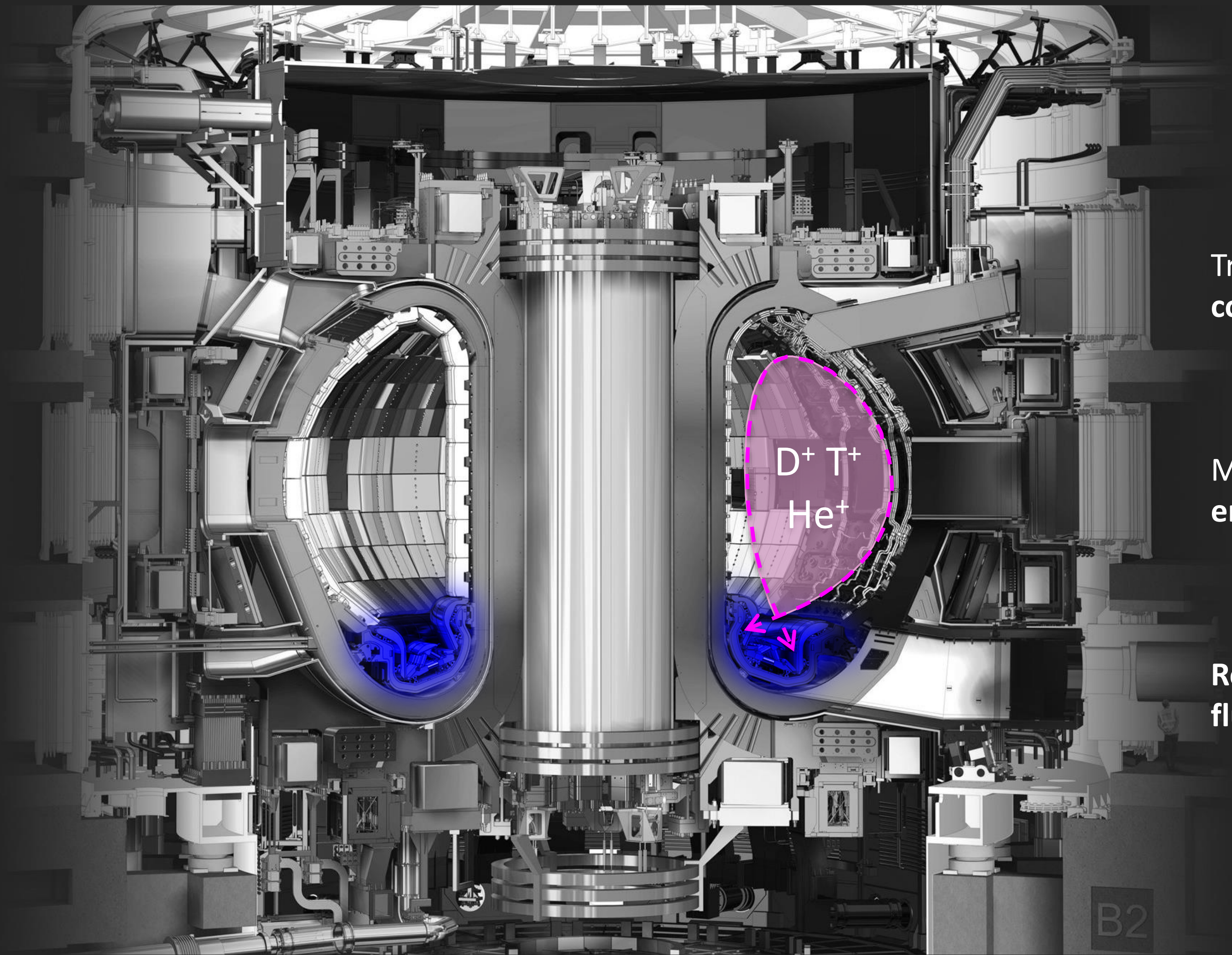
FESTIM, a modelling code for hydrogen transport in materials for nuclear fusion applications

R. Delaporte-Mathurin[1,2], E. A. Hodille[1], J. Dark[2], F. Leblond[1], J. Mougenot[2], Y. Charles[2], C. Grisolia[1]

[1] CEA, IRFM/GCFPM, F-13108 Saint-Paul-lez-Durance, France

[2] Université Sorbonne Paris Nord, Laboratoire des Sciences des Procédés et des Matériaux, LSPM, CNRS, UPR 3407, F-93430, Villetaneuse, France



Tritium
contamination

Material
embrittlement

Recycling fluxes

Images: ITER Organization

$$\partial_t c_\mathrm{m} = \nabla(D(T) \cdot \nabla c_\mathrm{m}) - \sum_i \partial_t c_{\mathrm{t},i} \quad \text{on } \Omega$$

$$\partial_t c_{\mathrm{t},i} = k(T) \cdot c_\mathrm{m}\left(n_i - c_{\mathrm{t},i}\right) - p(T) \cdot c_{\mathrm{t},i} \quad \text{on } \Omega$$

Hydrogen transport
McNabb & Foster - Trans. Metall. Soc. (1963)

$$\frac{c_\mathrm{m}^-}{S(T)^-} = \frac{c_\mathrm{m}^+}{S(T)^+} \quad \text{on } \Omega_i \cap \Omega_j$$

Conservation of chemical potential at interfaces

$$\rho C_p \, \partial_t T = \nabla(\lambda \cdot \nabla T) + Q \quad \text{on } \Omega$$

Energy equation

- $c_\mathrm{m}, c_{\mathrm{t},i}, T$ H concentrations and temperature
- i corresponds to a type of sink

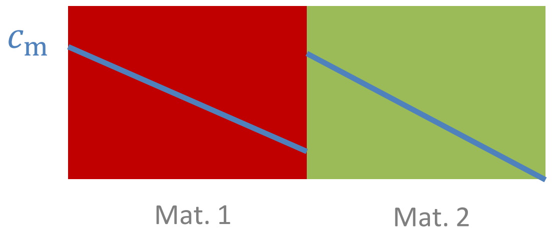
FESTIM

- Finite Element Simulation of Tritium In Materials
- Based on FEniCS
- 1/2/3D
- Multi-materials

For more info:

Delaporte-Mathurin *et al*, NME (2019)

$$\frac{c_\mathrm{m}^-}{S^-} = \frac{c_\mathrm{m}^+}{S^+} \quad \text{at interfaces}$$



- Modelling discontinuities in FEniCS

$$\partial_t c_\mathrm{m} = \nabla(D \cdot \nabla c_\mathrm{m}) - \sum_i \partial_t c_{\mathrm{t},i} \quad \text{on } \Omega$$

$$\theta = c_\mathrm{m}/S$$

1. Solve : $\partial_t(\theta S) = \nabla(D \cdot \nabla(\theta S)) - \sum_i \partial_t c_{\mathrm{t},i} \quad \text{on } \Omega$
2. Post-processing:

   $c_\mathrm{m} = \theta \cdot S$

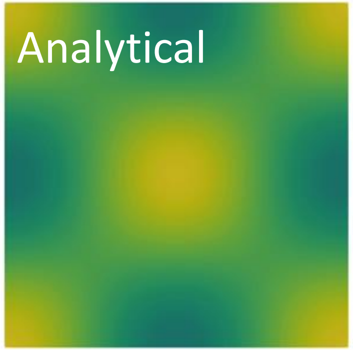
   project on DG1 space

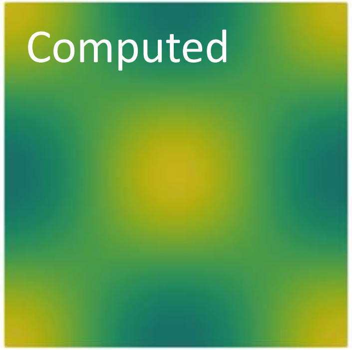
```
V_DG1 = FunctionSpace(mesh, ‘DG’, 1)
c_m = project(theta*S, V_DG1)
```

Delaporte-Mathurin *et al,* Nucl. Fusion (2021)

Verification using MMS

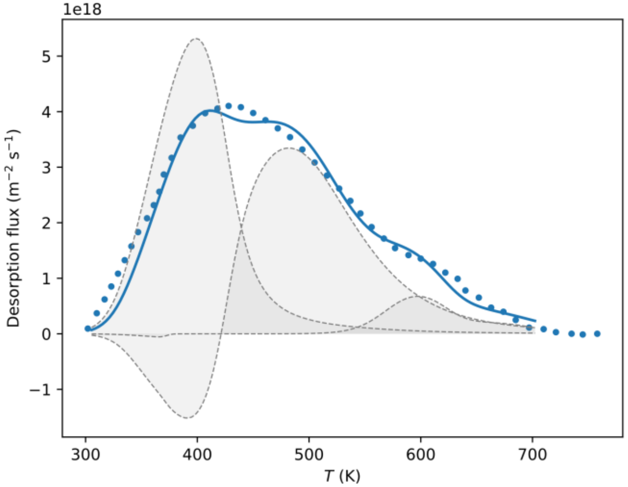
- $c_{\mathrm{m,D}} = 1 + \cos(2\pi x)\cos(2\pi y) + \cos(2\pi t)$
- $T = 500 + 30\cos(2\pi x)$
- Multi-material





Experimental validation

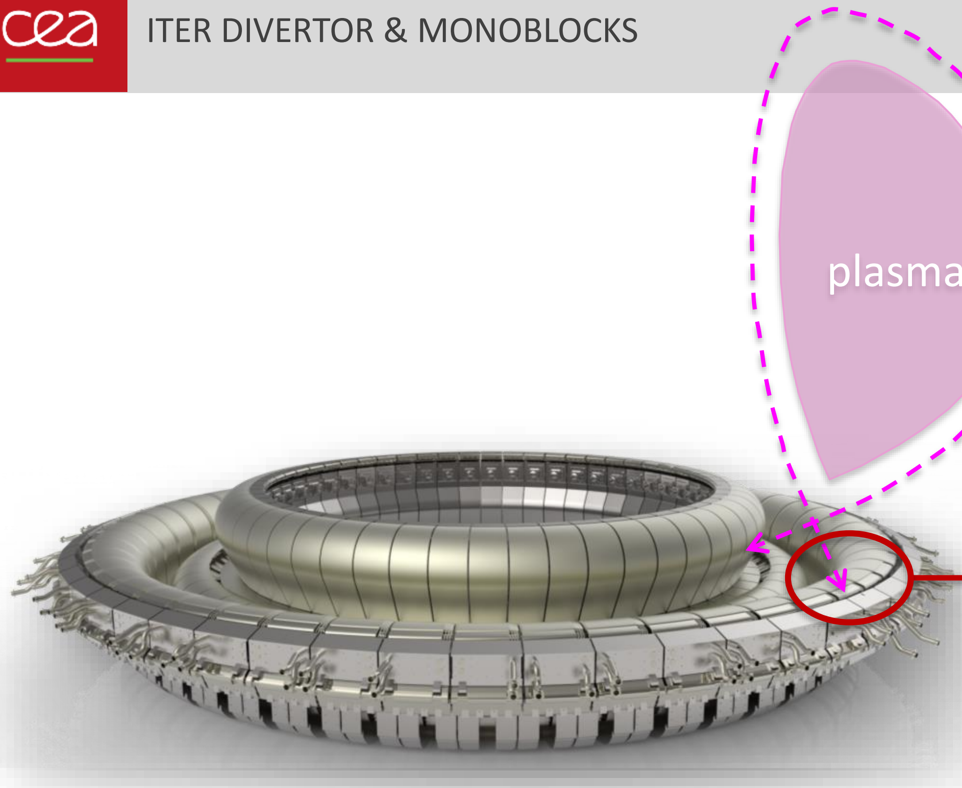
- Thermo-desorption experiments
- Parametric optimisation

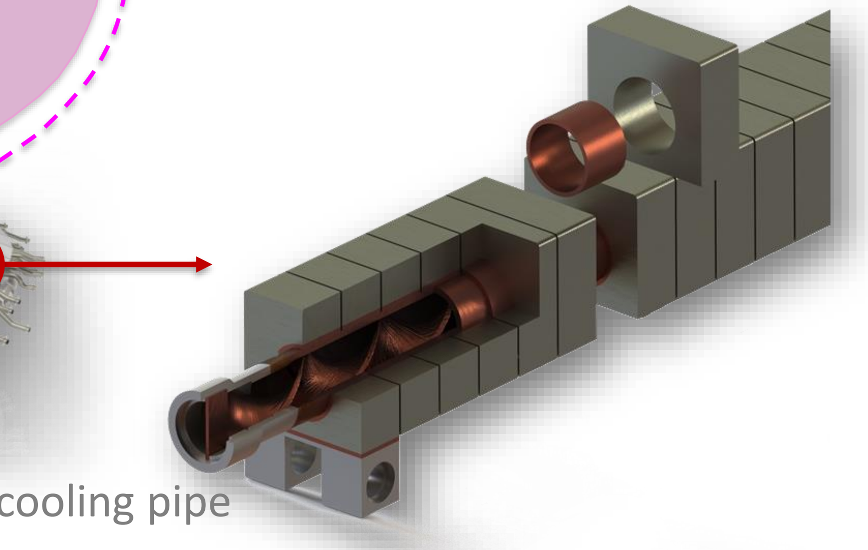


Delaporte-Mathurin *et al,* NME (2021)

Application:

Tokamak components

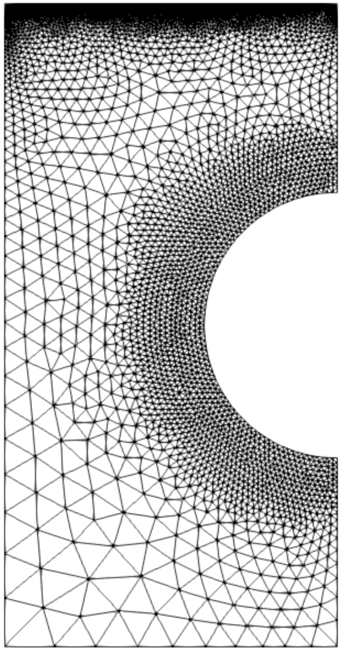




How much hydrogen is retained ?

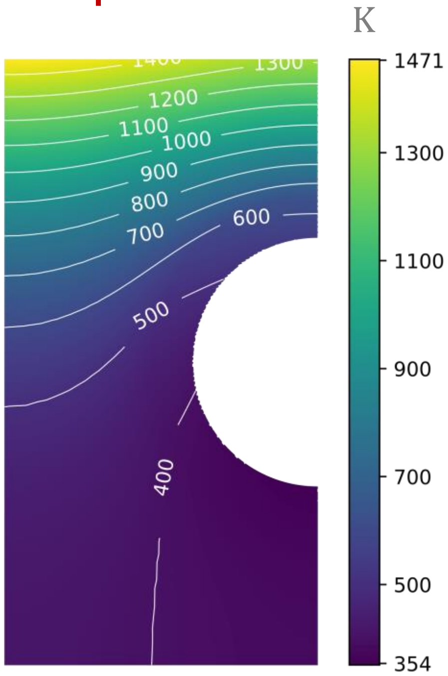
Images: ITER Organization

- Meshed with **SALOME** (open-source)
- Converted from .med to .xdmf with **meshio** [1]
- High refinement:
  - on the top surface
  - at interfaces
- Planned: using Adaptive Mesh Refinement

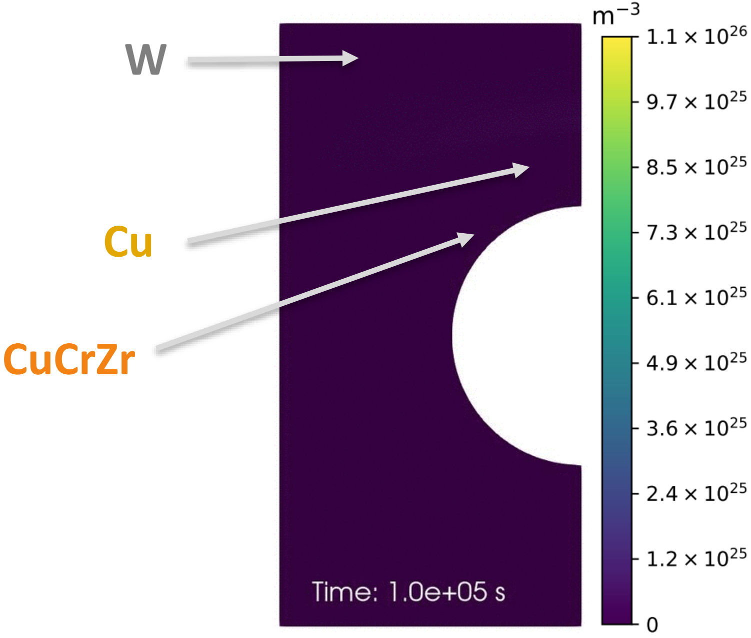


[1] Meshio: 10.5281/zenodo.4590119

Temperature field



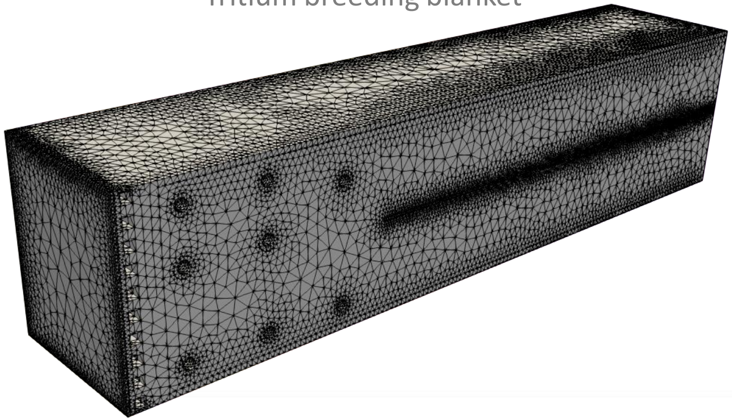
Hydrogen concentration

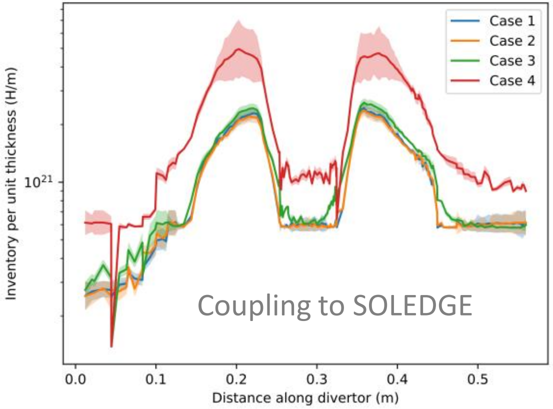


- Total inventory of H : $\int (c_m + c_t)\, dx$
- Coolant contamination : $\int D(T) \nabla c_m \cdot \boldsymbol{n}\, dS$

- Applications to more complex tokamak components
- Coupling with other physics
  - CFD
  - MHD,
  - co-deposition models
  - He transport…
- Coupling with external plasma codes

Tritium breeding blanket





Coupling to SOLEDGE

Thank you for your attention

Plots were made with Matplotlib and Paraview

- FESTIM is a FEniCS-based simulation interface
- Hydrogen transport (including diffusion and trapping) is modelled and coupled to heat transfer.
- FESTIM applications:
  - Simulating fusion reactors components
  - Identifying materials properties
- Perspectives:
  - Applications to more complex tokamak components (tritium breeding blankets)
  - Coupling with other physics (CFD, MHD, co-deposition models, He transport…)
  - Coupling with external plasma codes (SOLEDGE, SOLPS)

```python
import FESTIM

parameters = {
    "materials": [
        {
          "E_D": 0.1,
          "D_0": 1,
          "id": 1
         }
      ],
    "traps": [],
    "mesh_parameters": {
            "initial_number_of_cells": 200,
            "size": 1,
            "refinements": [
            ],
        },
    "boundary_conditions": [
         ],
    "temperature": {
          "type": "expression",
          "value": 300
        },
    "solving_parameters": {
        "final_time": 100,
        "initial_stepsize": 0.1,
        "newton_solver": {
            "absolute_tolerance": 1e-10,
            "relative_tolerance": 1e-9,
            "maximum_iterations": 50,
         }
        },
}

output = FESTIM.run(parameters)
```

```
class UserCoeff(UserExpression):
    def __init__(self, mesh, vm, T, **kwargs):
        super().__init__(kwargs)
        self._mesh = mesh
        self._vm = vm  # MeshFunction for volume markers
        self._T = T

    def eval_cell(self, value, x, ufc_cell):
        cell = Cell(self._mesh, ufc_cell.index)
        subdomain_id = self._vm[cell]
        if subdomain_id == 1:
            value[0] = self._T(x)
        else:
            value[0] = 2

    def value_shape(self):
        return ()

S = UserCoeff(mesh, vm, T)

V_DG1 = FunctionSpace(mesh, ‘DG’, 1)
c_m = project(theta*S, V_DG1)
```