ToFu

An open-source python/cython library for synthetic tomography diagnostics on tokamaks

---

Laura S. Mendoza[1], Didier Vezinet[2]

Seminaire, Strasbourg, France

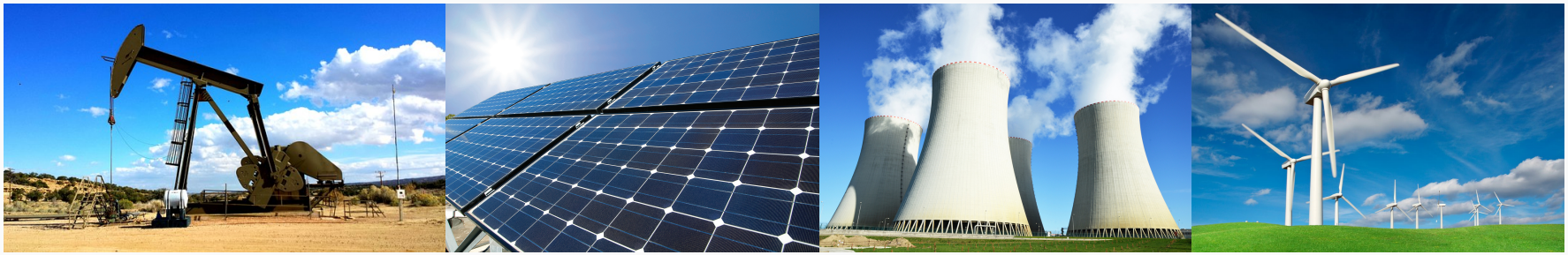
[1] INRIA Grand-Est, TONUS Team, Strasbourg, France

[2] CEA, Cadarache, France

Table of contents

Context

Context: energy needs vs resources and climate change



Worldwide growing energy needs (population, standards of living...)

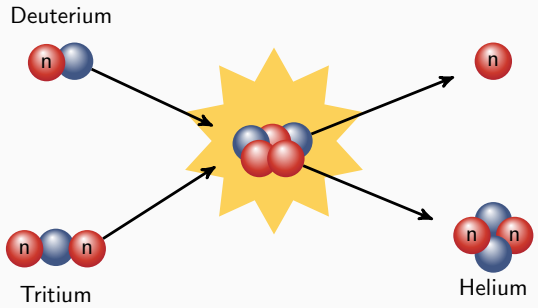
$\Rightarrow$ Production of CO_2, limited resources, radioactive waste, not efficient enough, harmful to surrounding environment

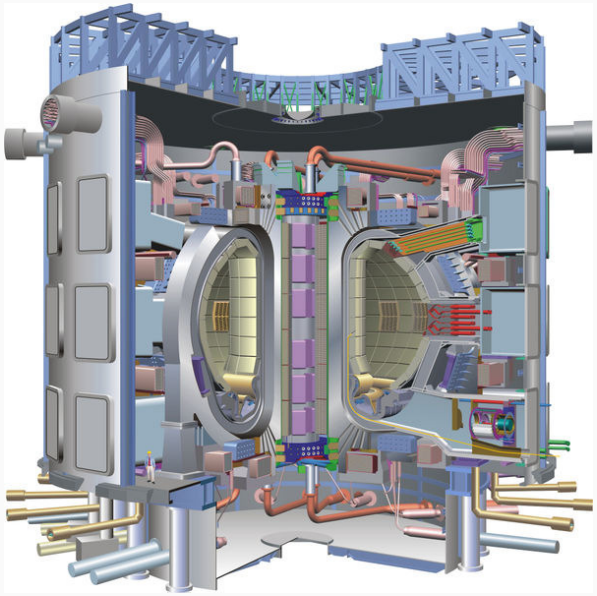
$\Rightarrow$ need to decrease consumption + alternative production means

$\Rightarrow$ Need for cleaner, more reliable, more powerful energy source

Context: Controlled fusion and magnetic confinement

D-T Fusion reaction

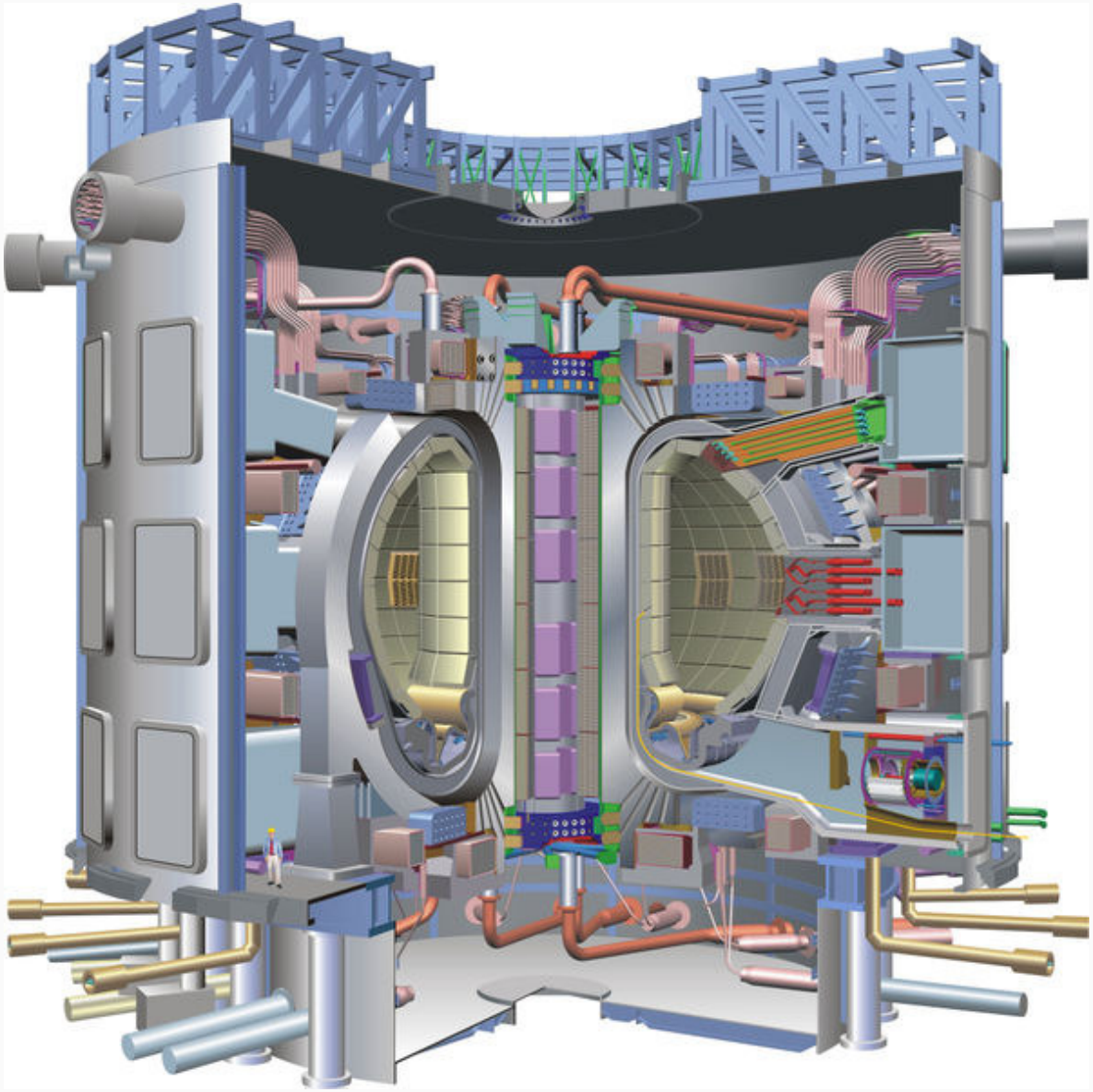




- Gas $>$ 100 Million°K composed of positive ions and negative electrons: plasma
- Confinement using electromagnetic fields
- break-even not obtained yet

In a nutshell: toroidal vacuum vessel, filled with H plasma

Tokamak diagnostics to measure plasma quantities



Tomography diagnostics

Tokamak diagnostics to measure plasma quantities

Diagnostics

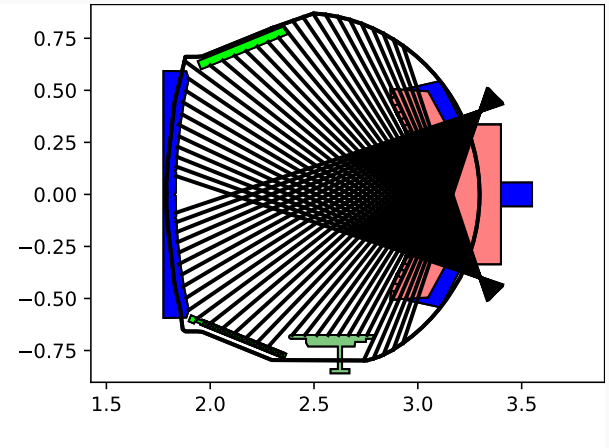
Set of instruments to measure plasma quantities, for understanding, control, optimization.

e.g: magnetic field, neutrons, **emitted light**, temperature, density...

$\Rightarrow$ cameras (1D or 2D) for measuring light in various wavelengths

Tomography diagnostics - numerical context

$$M_i(t) = \iiint_{V_i} \overrightarrow{\varepsilon(x,t)} \cdot \vec{n}\, \Omega_i \, dV$$



- **Direct problem** (synthetic diagnostic):

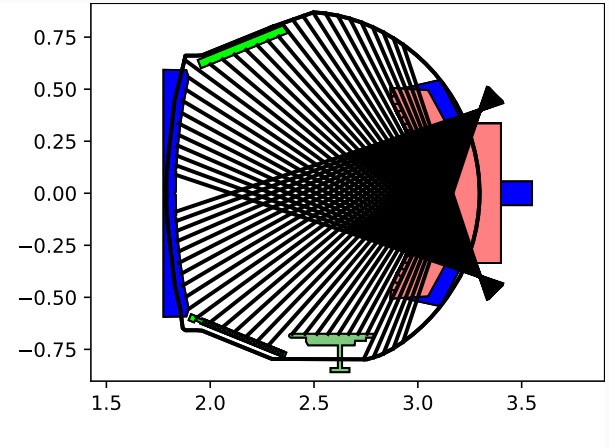
  Simulated emissivity $\longrightarrow$ integrated measurements

- **Inverse problem** (tomography):

  Integrated measurements $\longrightarrow$ Reconstructed emissivity

Tomography diagnostics - numerical context

$$M_i(t) = \iiint_{V_i} \overrightarrow{\varepsilon(x,t)} \cdot \vec{n}\, \Omega_i \, dV$$



- **Direct problem** (synthetic diagnostic):

  Simulated emissivity $\longrightarrow$ measurements

  Spatial integration

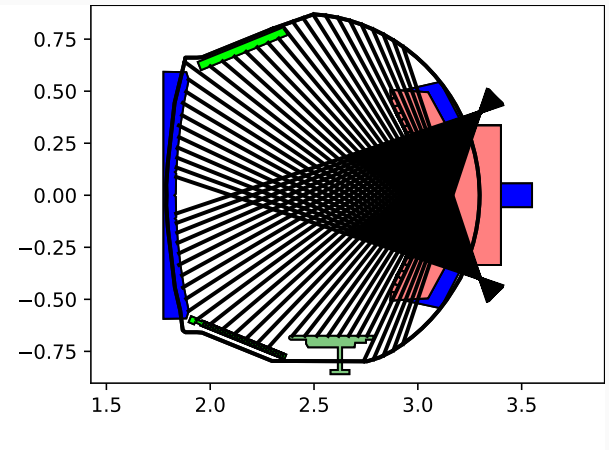
- **Inverse problem** (tomography):

Integrated measurements $\longrightarrow$ Reconstructed emissivity

Mesh and basis functions construction, spatial integration, data filtering, inversion routines, etc.

Tomography diagnostics - numerical context

$$M_i(t) = \iiint_{V_i} \overrightarrow{\varepsilon(x,t)} \cdot \vec{n}\, \Omega_i \, dV$$



- **Direct problem** (synthetic diagnostic):

  Simulated emissivity $\longrightarrow$ measurements

  Spatial integration

- **Inverse problem** (tomography):

Integrated measurements $\longrightarrow$ Reconstructed emissivity

Mesh and basis functions construction, spatial integration, data filtering, inversion routines, etc.

Tomography is **ill-posed**, very sensitive to errors, noise and bias

$\longrightarrow$ Reputation for low reproducibility / reliability

The ToFu package

Motivation: “current” state

In the fusion community, codes for tomography diagnostic are often:

- developed by physicists (with little programming experience)
- in Matlab (or IDL)
- written from scratch, re-done by new students
- not distributed (few users), rarely documented

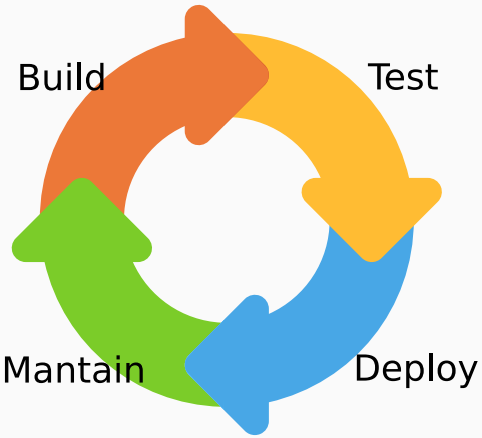
... which means

- waste of resources: time, man-power
- low traceability, reproducibility
- low standardization, unclear assumptions / methods

A code for Tomography for Fusion

Develop a common tool:

- Generic (geometry independent)
- Portable (Python)
- Optimized / parallelized
- Documented online
- Continuous integration



ToFu[1][2][3] = **To**mography for **Fu**sion

[1] repository: `https://github.com/ToFuProject/tofu`

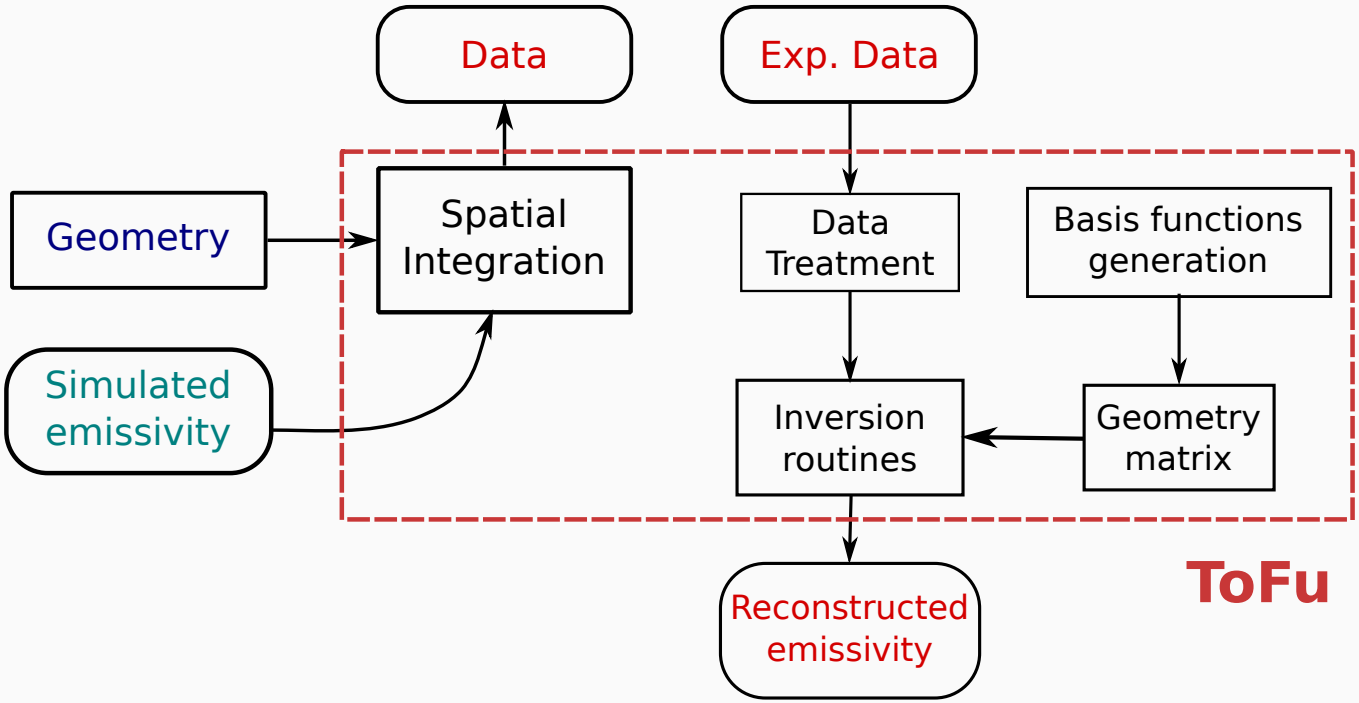
[2] documentation: `https://tofuproject.github.io/tofu/index.html`

[3] D Vezinet et al. "Non-monotonic growth rates of sawtooth precursors evidenced with a new method on ASDEX Upgrade". In: *Nuclear Fusion* 8 (2016).

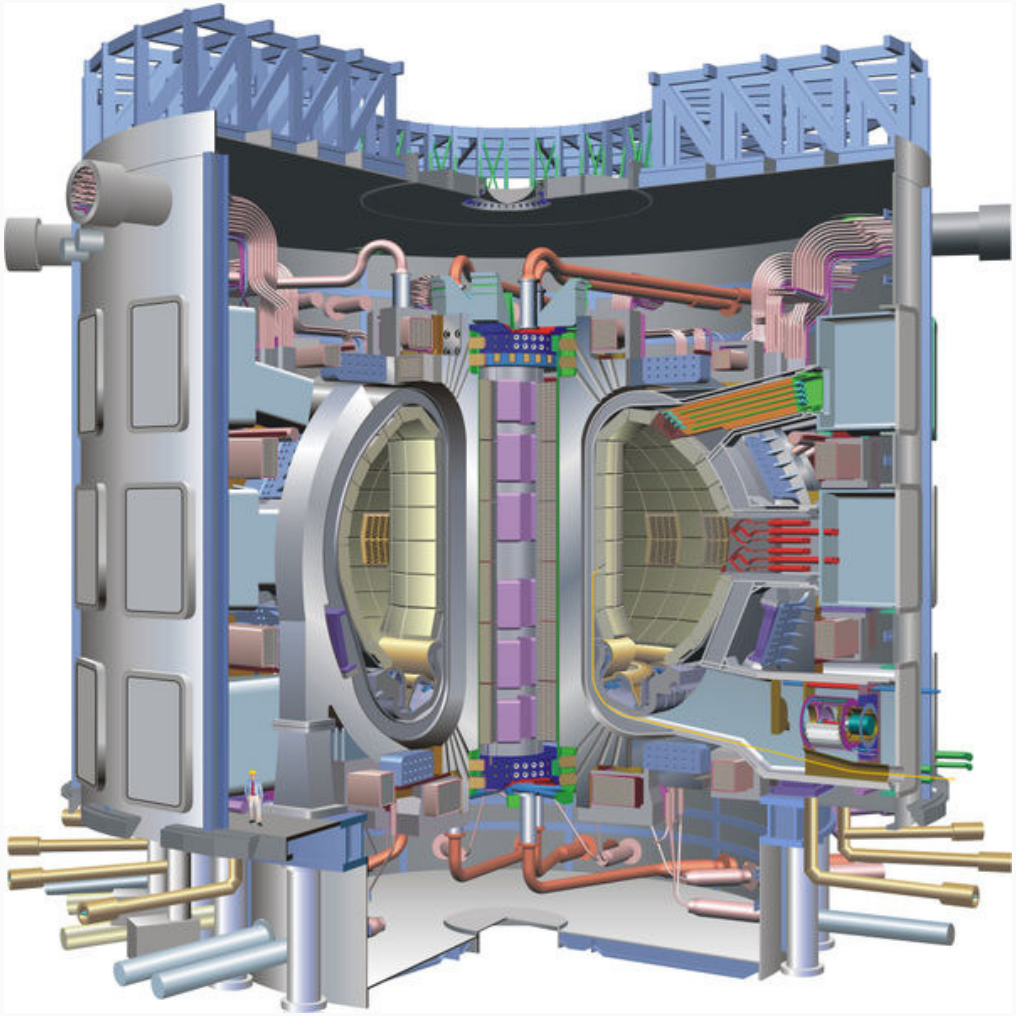
More about Tofu

- Created in 2014
- Open Source: **MIT license**
- **Python** 3 + **Cython**
- Continuous integration: **Travis** CI
- **conda**, **pip**
- Two (main) developers:
  - Didier Vezinet (creator, Physics)
  - Laura S. Mendoza (since 06.2018, Applied Maths)
- Contributors:
  - Jorge Morales
  - Florian Le Bourdais
  - Arpan Khandelwal
  - Koyo Munechika

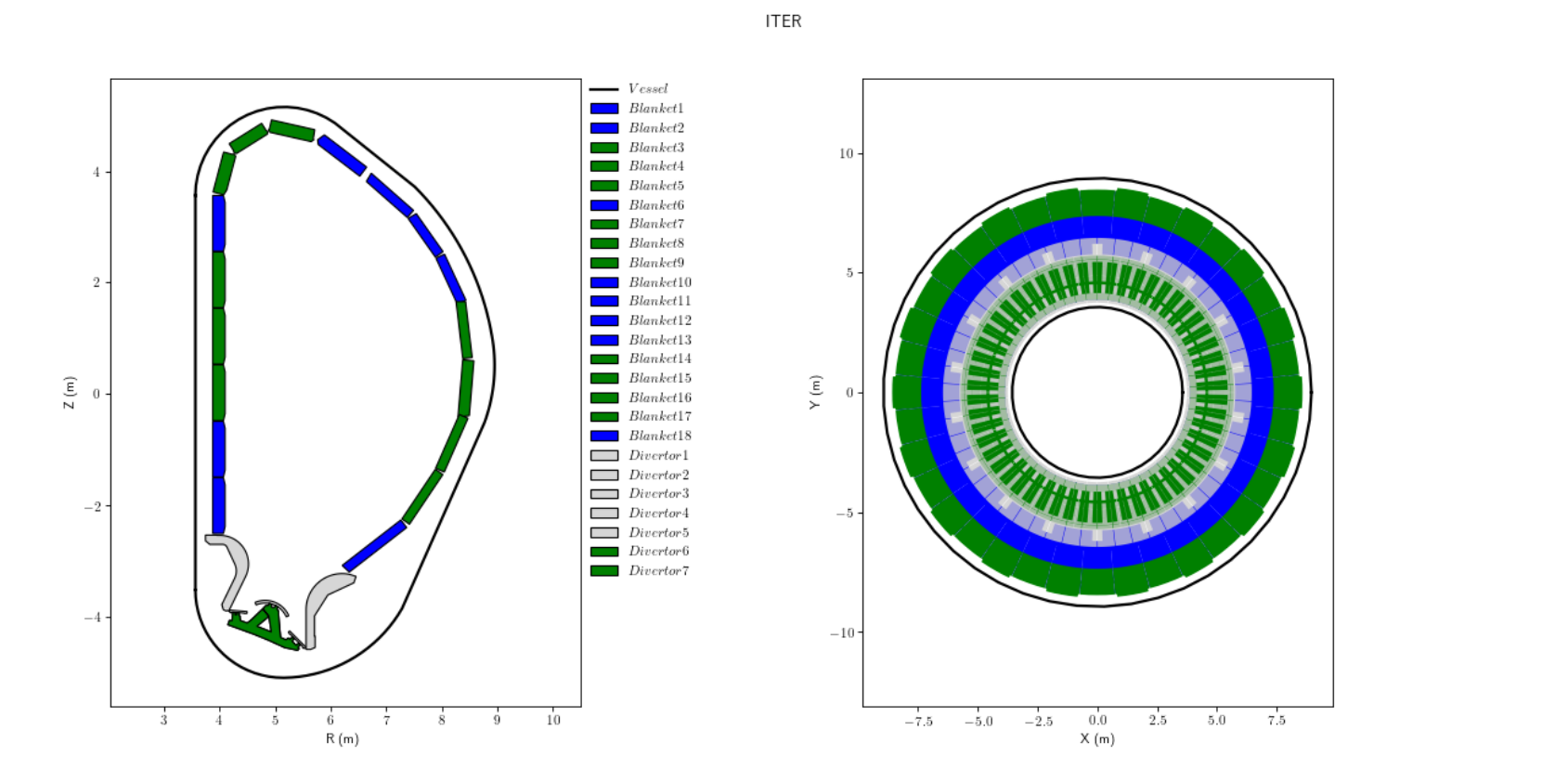
Tofu's structure



Tofu's geometry representation



Tofu's geometry representation

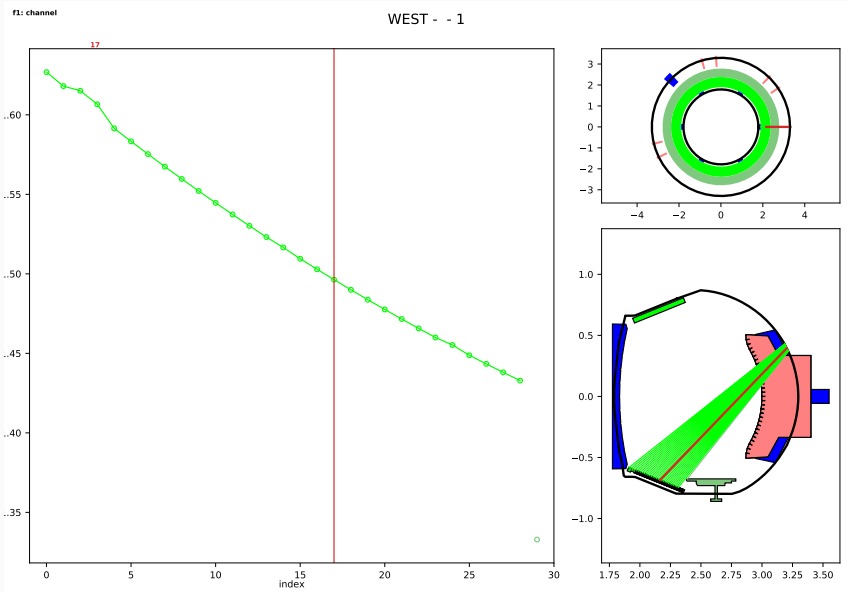


Demo

Demo

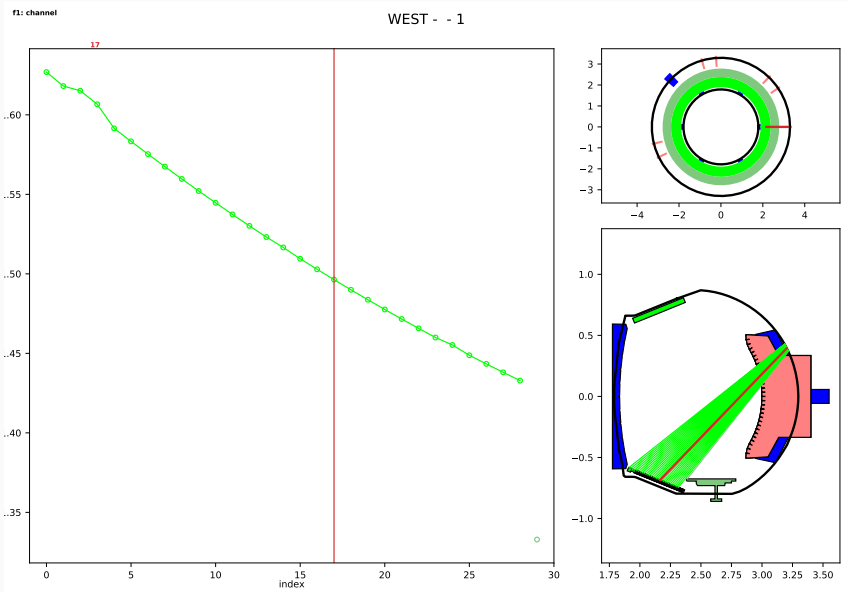
tofu.geom: modeling of simplified geometry

1D Camera (tofu.geom.CamLOS1D)

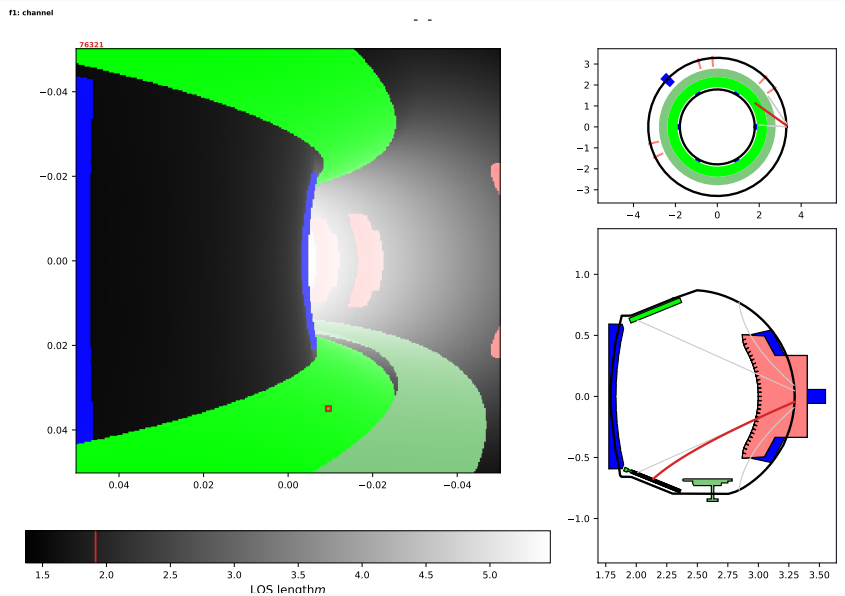


tofu.geom: modeling of simplified geometry

1D Camera (tofu.geom.CamLOS1D)

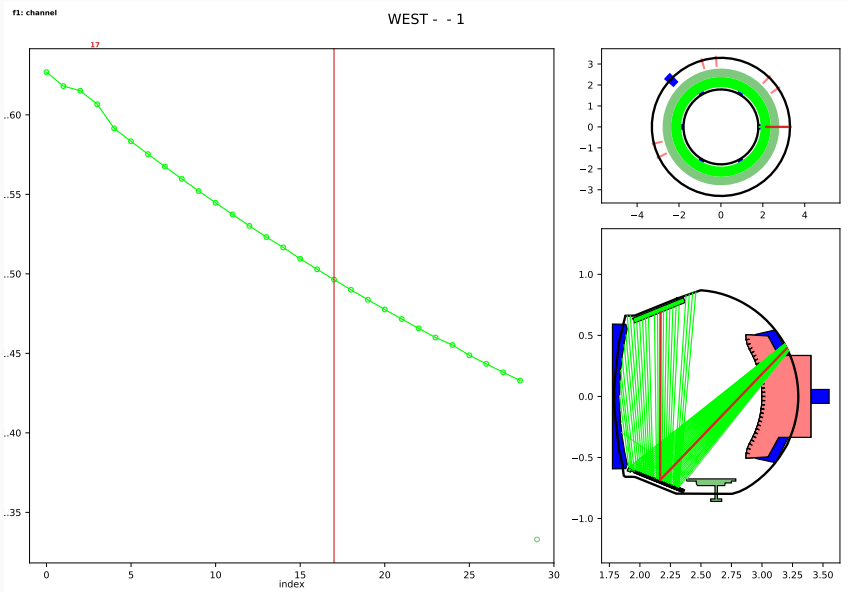


2D Camera (tofu.geom.CamLOS2D)

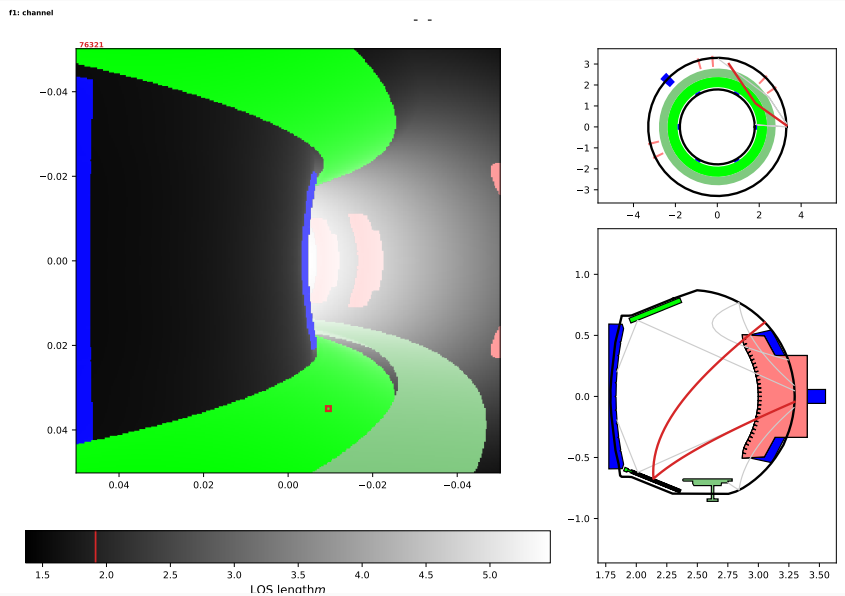


tofu.geom: handle basic reflexions

1D Camera with reflexions (tofu.geom.CamLOS1D)



2D Camera with reflexions (tofu.geom.CamLOS2D)



What ToFu can do

- Model simplified 3D geometry
- 3D modeling of a 1D and 2D LOS camera
- Handle basic reflections
- Computing synthetic signals
- Native support for IMAS interfacing
- Data easy interactive visualization and basic treatment

What ToFu can do

- Model simplified 3D geometry
- 3D modeling of a 1D and 2D LOS camera
- Handle basic reflections
- Computing synthetic signals
- Native support for IMAS interfacing
- Data easy interactive visualization and basic treatment
- ...and soon (being re-written / developed):
  - finite beam width (VOS, in 1.4.3, mid 2020)
  - meshing and basis functions (mid 2020)
  - tomographic inversion (late 2020 - 2021)
  - dust particle trajectory tracking (new, Arpan)
  - faster Matplotlib + PyQtGraph visualization
  - magnetic field line tracing (new)
  - statistical data analysis (pandas) integrated

Code Optimization

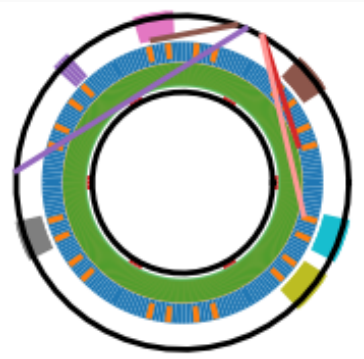
Geometry reconstruction: ray-tracing techniques

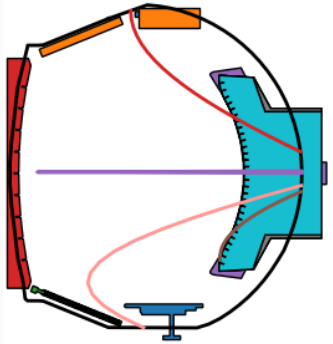
To reconstruct emissivity we need to take account:

- Up to hundreds of structural elements in vessel
- Scale of the vessel: 10^4 bigger than smaller structural detail

$\Rightarrow$ Geometry defined with minimal data polygon (R, Z) extruded along φ

$\Rightarrow$ Symmetry of vessel along φ



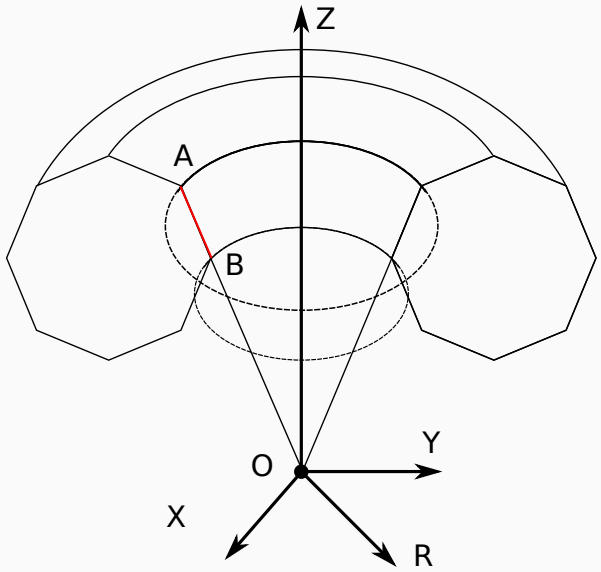


Optimization of ray-tracing algorithm

- Description of geometry:
  - Vessel and structures: set of 2D polygon $\mathcal{P}_j = \cup_{i=1}^{n} \overline{\mathrm{A}_i\mathrm{B}_i}$
  - Extruded along $[\varphi_{min}, \varphi_{max}]$
  - Detectors defined as set of rays (of origin D and direction u)

  $\Rightarrow$ Light memory-wise

$\Rightarrow$ Equivalent to: set of truncated cones (frustums) of generatrix A_iB_i



Ray-tracing algorithm on fusion device $\longrightarrow$ Computation of cone-Ray intersection

$$\exists (q,k) \in [0;1] \times [0;\infty[, \begin{cases} R - R_A = q(R_B - R_A) \\ Z - Z_A = q(Z_B - Z_A) \\ DM = ku \end{cases}$$

Optimization of ray-tracing algorithm

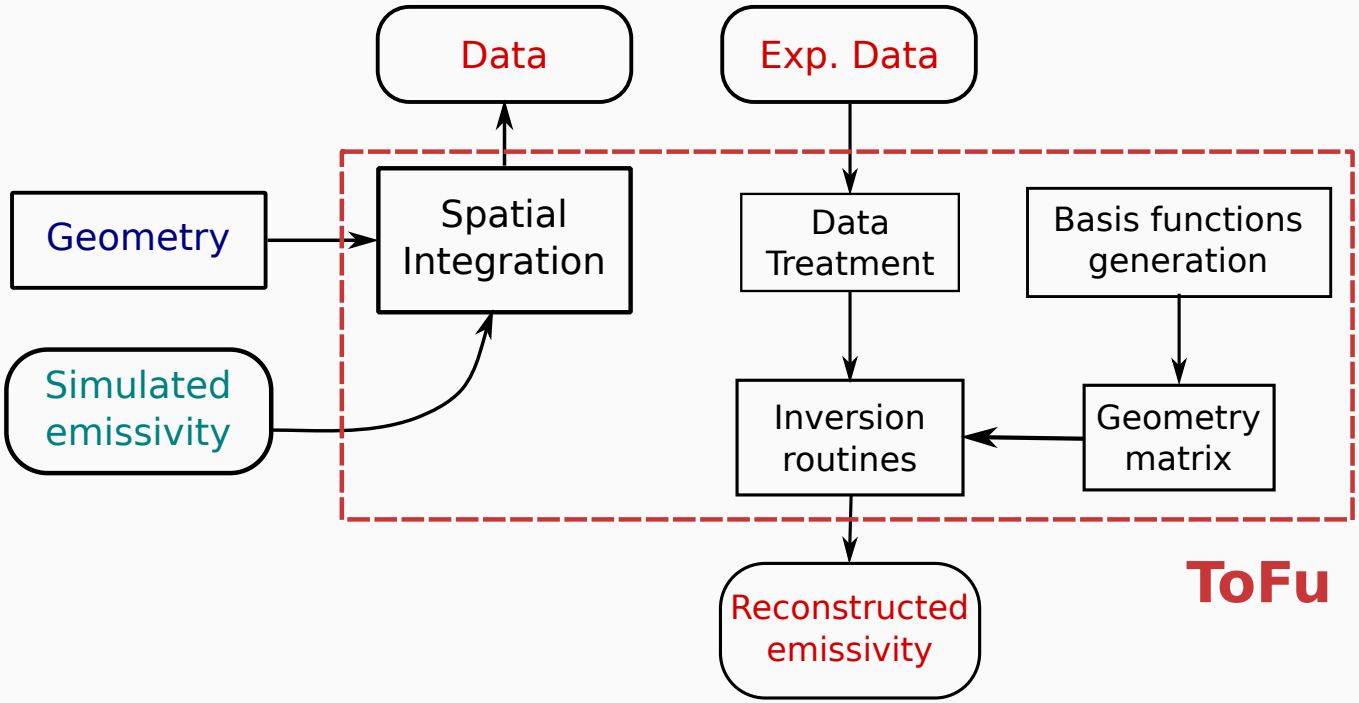
- Algorithmic optimizations:
  - Test intersection **bounding-box**
  - **special cases** (ray direction, segment, etc.)
  - Cone-Ray intersection algorithm: solution of a **quadratic equation**
- Pre-computation of variables
- Core functions in **Cython**
- OpenMP parallelization (Cython **prange** loops)

Optimization of ray-tracing algorithm

- Algorithmic optimizations:
  - Test intersection **bounding-box**
  - **special cases** (ray direction, segment, etc.)
  - Cone-Ray intersection algorithm: solution of a **quadratic equation**
- Pre-computation of variables
- Core functions in **Cython**
- OpenMP parallelization (Cython **prange** loops)

Nb LOS	10^3	10^4	10^5	10^6	
original	$3.26 \cdot 10^1$	$3.10 \cdot 10^2$	$3.20 \cdot 10^3$	$3.17 \cdot 10^4$	$(8\text{h}48)$
optimized	$2.58 \cdot 10^{-2}$	$2.72 \cdot 10^{-1}$	2.74	$2.66 \cdot 10^1$	$(< 30\text{s})$
32 threads	$1.36 \cdot 10^{-2}$	$4.66 \cdot 10^{-2}$	$3.64 \cdot 10^{-1}$	2.92	

Tofu's structure



Optimization of spatial integration routines

$$M_i(t) = \iiint_{V_i} \overrightarrow{\varepsilon(x,t)} \cdot \vec{n}\, \Omega_i \, dV$$

Integration of **user-defined function** ε along a LOS:

- Integration of a python function **func** defined by user by:
  - **numpy**.sum (quad: **midpoint**)
  - Cython based sum (quad: **midpoint**)
  - **Scipy**.integrate.simps
  - **Scipy**.integrate.romb
- Optional optimizations:
  - calls to **func**: avoid Cython-Python conversion, user-defined
  - memory: fine resolutions, high number of LOS
  - hybrid: compromise

Optimization of spatial integration routines

LOS	10	10^2	10^3	10^4
original	0.46	2.24	18.1	x
calls	0.207	0.53	4.2	x
hybrid	0.08	0.44	4.32	40.3 (32Gb)
memory	0.9	8.9	96	945 (6Gb)

- Space resolution: 10^{-3}
- Number of time steps: 10^3
- Integration method: **sum** (Cython or numpy) on midpoint

What we've learned on code optimization

Importance of **cross disciplinary** expert collaborations:

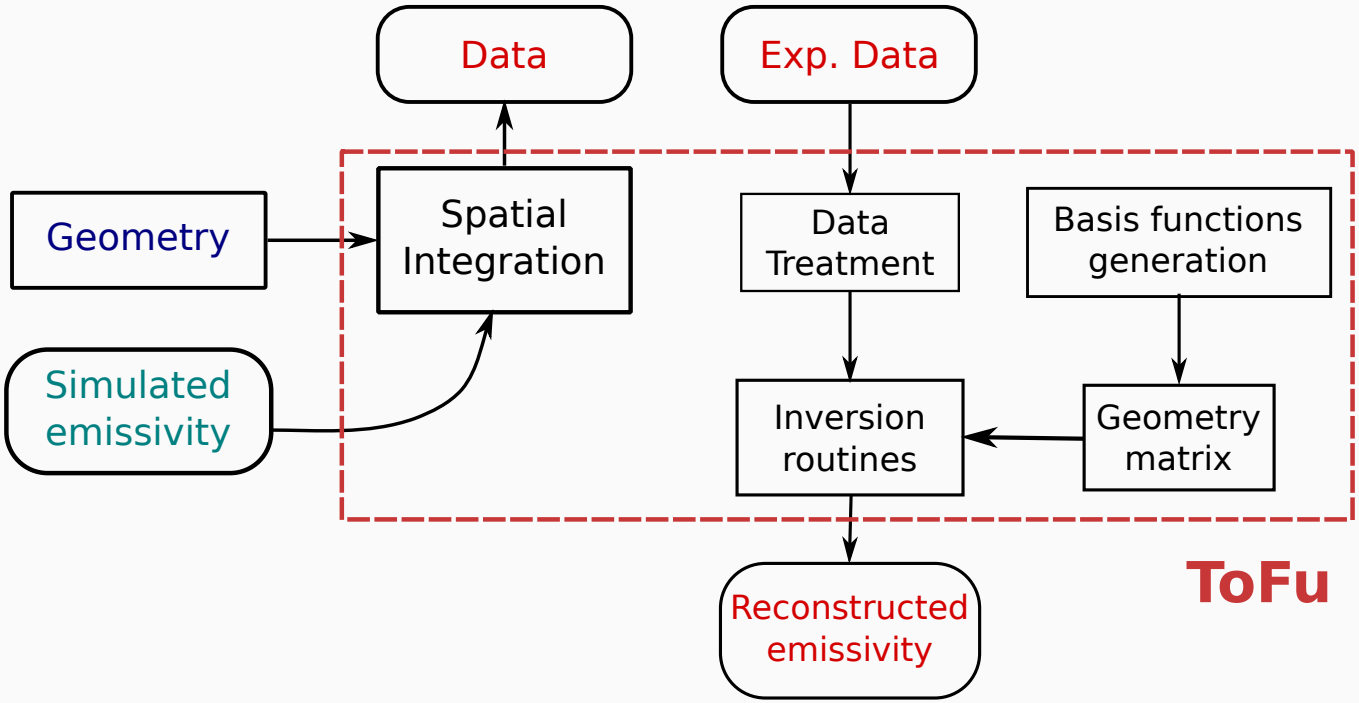
- Understand and define the problem (physics)
- Create optimal algorithm (mathematics)
- Write efficient code (computer science)

...and from there:

- **Profile** your code: *"premature optimization is the root of all evil."* - Sir Tony Hoare
- Core functions: **Cython** (or Numba or Transonic or ...)
- Cython specific:
  - **cython –annotate**
  - **type** your variables and **inline** your functions!
  - use cython decorators: **@cython.boundscheck(False)**, **@cython.wraparound(False)**
  - release the gil (when possible)!
  - see more: my notes, Jeremie du Boisberranger's tutorial

What's next

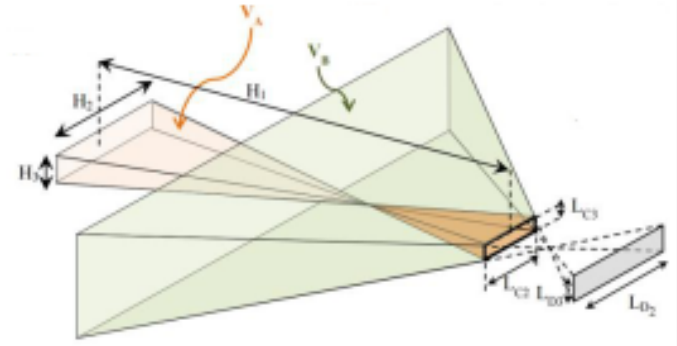
Tofu's structure



Tofu's main algorithms

Geometry:

- Finite beam width (LOS ⇒ VOS)
- More advanced reflections (specular, diffusive, curved-cubic)
- Thermal heat load computation



Meshing and Inversions:

- Meshing and Basis functions (local and global) with visualization
- Geometry matrix (fast) computation

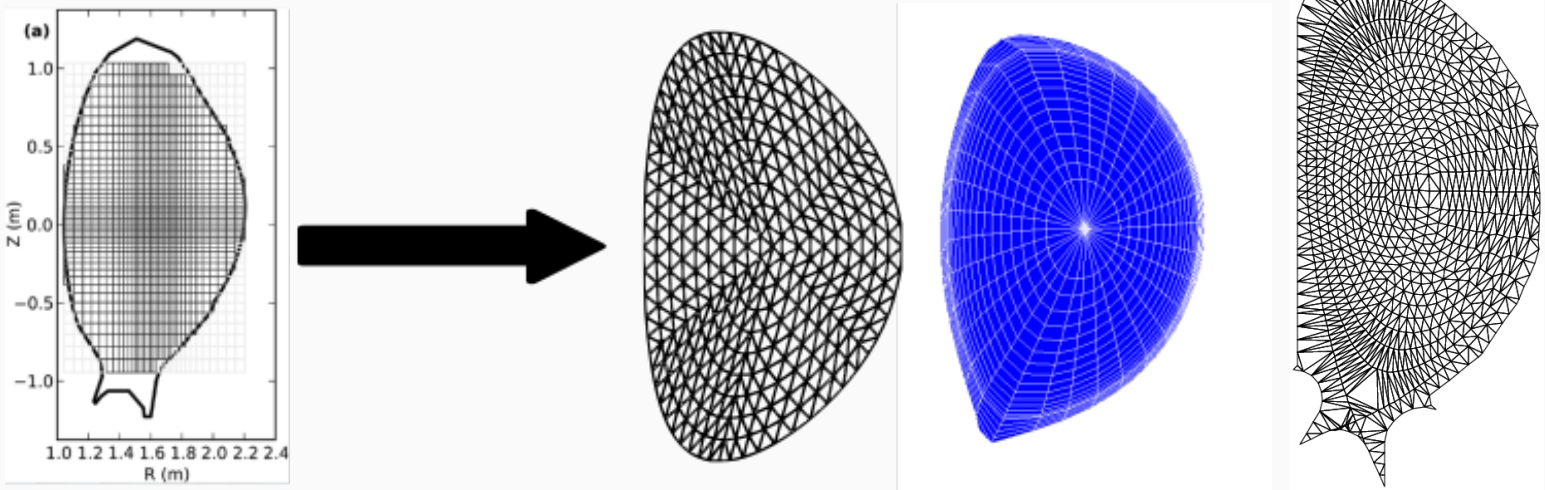
On geometry discretization: meshing

Several options for poloidal cut meshing:

- Cartesian mesh
- (Adaptive) polar mesh
- Hexagonal mesh
- Triangular mesh

For basis functions:

- B-splines
- NURBS
- Box-splines

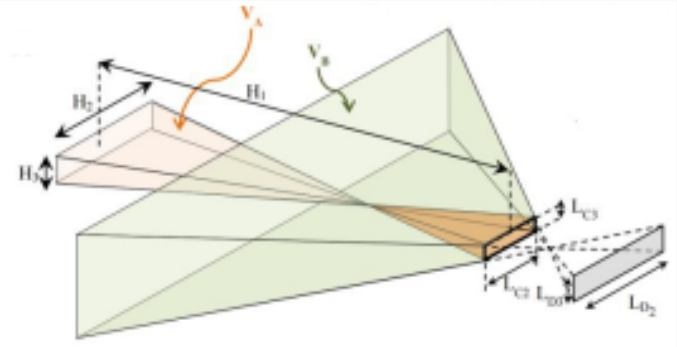


$\longrightarrow$ **Collaboration with IPP**

Tofu's main algorithms

Geometry:

- Finite beam width (LOS $\Rightarrow$ VOS)
- More advanced reflections
- Thermal heat load computation



Meshing and Inversions:

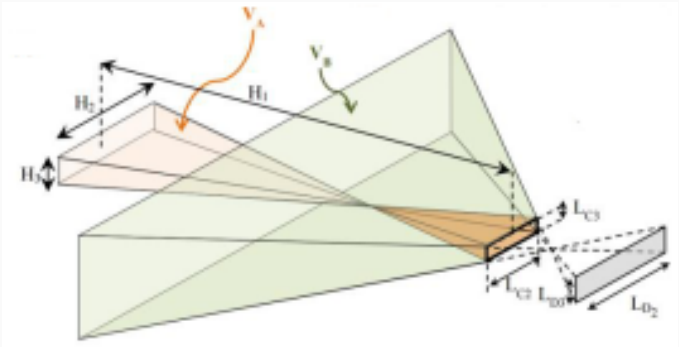
- Meshing and Basis functions (local and global) with visualization
- Geometry matrix (fast) computation and introspection plots
- Multiple Inversion-Regularization (linear and non-linear) and visualization
- pre- and post-inversion tools

$\longrightarrow$ **CEMRACS 2020 project (Machine Learning?)**

Tofu's main algorithms

Geometry:

- Finite beam width (LOS $\Rightarrow$ VOS)
- More advanced reflections
- Thermal heat load computation



Meshing and Inversions:

- Meshing and Basis functions (local and global) with visualization
- Geometry matrix (fast) computation and introspection plots
- Multiple Inversion-Regularization (linear and non-linear) and visualization
- post-inversion analysis tools

On the side:

- Statistical data analysis (tofu / pandas interface)
- Basic magnetic field line and particle trajectory tracing
- continued IMAS support

Merci !

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.