JRA4 work-package
Post-processing and Visualization

Brussels

30 March 2011

Scope

- Unified visualization tools
  - Common to different complex codes → CPO, UAL
  - Integrated into the simulation platform → Kepler
- Visualization of fusion data
  - At runtime, partial results during simulation
  - Post-processing, analysis after simulation
  - Different visualization needs
    - Simulation of tokamak discharge → different kinds of physics
    - 1D, 2D, 3D and multi-dimensional datasets (>3D)
- Partners
  - UDS
  - UOL

Deliverables & milestones

Del. no.	Deliverable name	Delivery date
DJRA4.1	Post-processing visualization	M12 on time
DJRA4.2	Visualization actors in Kepler	M18, on time
DJRA4.3	Lossy compression format	M24, on time
DJRA4.4	4D-5D visualization tool	M24, on time

- Milestones are associated to deliverables (MJRA4.1, MJRA4.2, MJRA4.3, MJRA4.4) and consist in code prototype implementing each deliverable solution

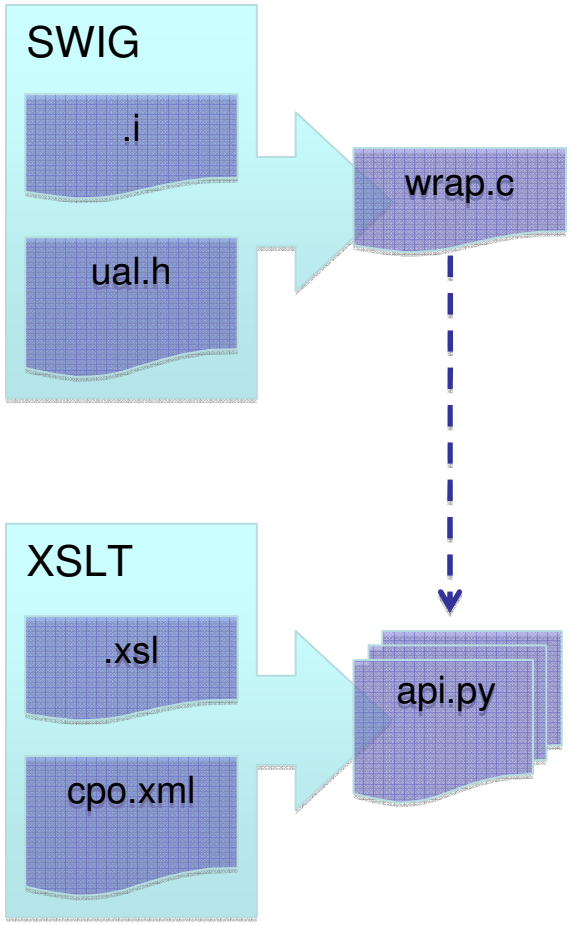
Covering visualization needs

- Python
  - Simple to learn and use
    - Script based language with interpreter
    - High level and object oriented programming
  - Complete: many scientific packages available

- VisIt
  - Based on Vtk (scientific visualization library standard)
  - Simple for both users and developers
    - Nice plots in a few clicks
    - XML helpers tools for adding new functionalities
  - Parallel capabilities

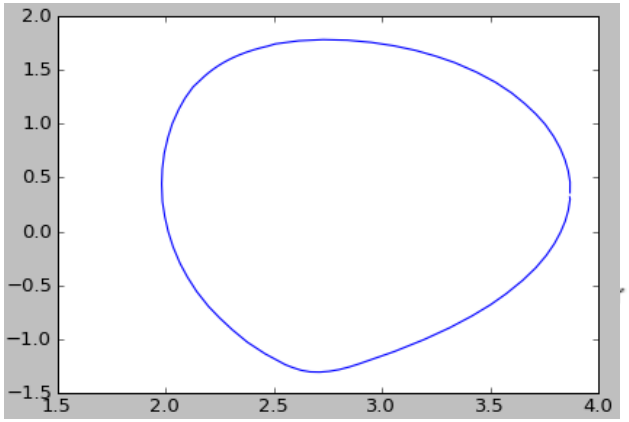
Python interface to the UAL

- Low-level interface
  - Wrapper for UAL library in C
  - Generated by SWIG
  - Interface file for advanced uses
- High-level interface
  - Hide low-level complexity
  - End-user programming API
  - Object oriented (CPO objects)
  - Generated by XSLT
    + XML CPO description



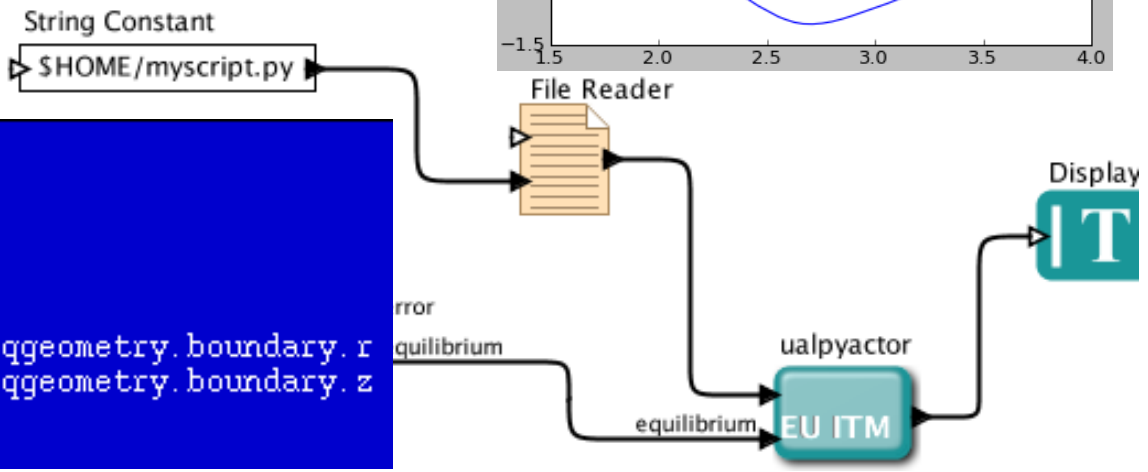
Python integration into Kepler

- Implementation
  - CPO and user script as input (or as parameter)
  - Handle a separate Python process to execute scripts saved in temporary files
  - Executed script mixes
    - automatic CPO object initialization
    - user operations on those variables



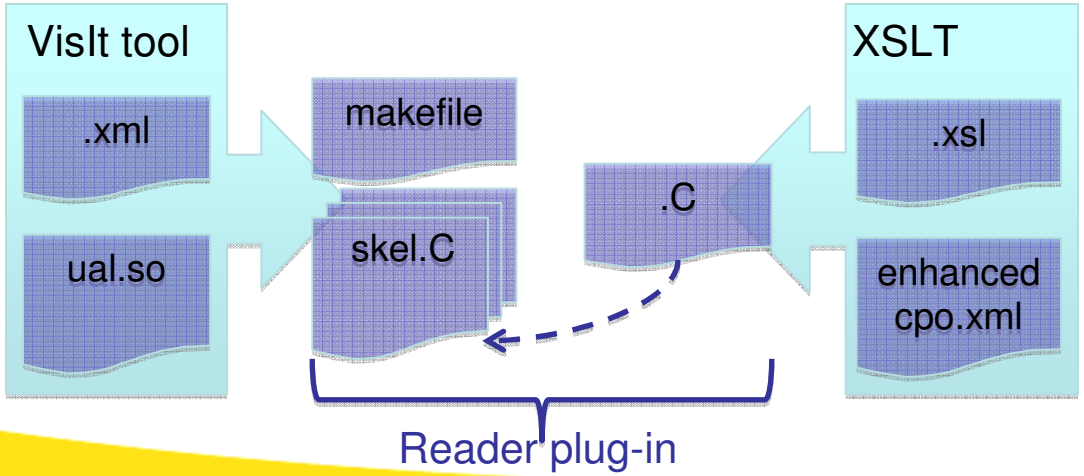

```
import ual
from pylab import *

cpo = ual.itm(51782,1)
cpo.open()
cpo.equilibriumArray.get()
r = cpo.equilibriumArray.array[0].eqgeometry.boundary.r
z = cpo.equilibriumArray.array[0].eqgeometry.boundary.z
plot(r,z)
show()
```



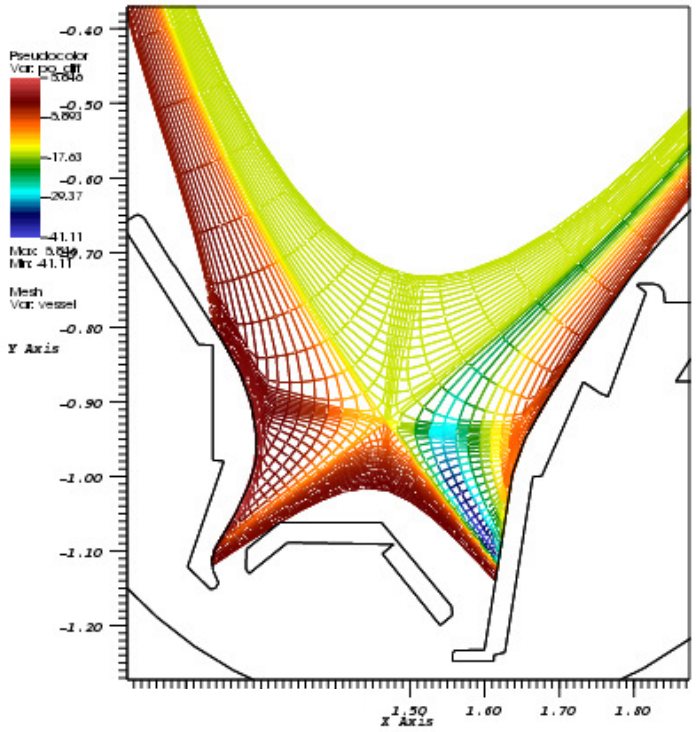
VisIt interface to the UAL

- Reader plug-in
  - Mostly generated by VisIt XML helper tools
  - Specific part generated by XSLT using C++ UAL interface
- Enhancement of CPO description
  - Ontology proposal for mesh types
  - Expert knowledge for associating data to mesh type

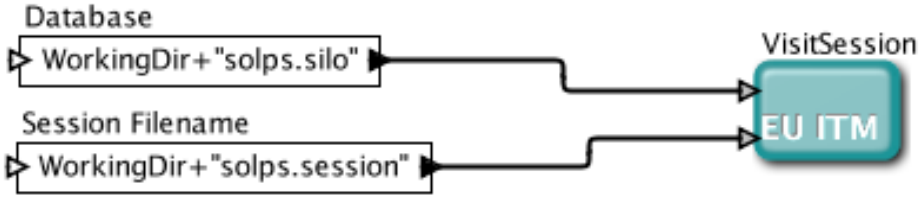


VisIt integration into Kepler

- VisIt generic actor
  - VisitSession build above jvisit interface
  - Input port:
    - Data file
    - Session file (restore mechanism)
  - Can also open GUI in Kepler

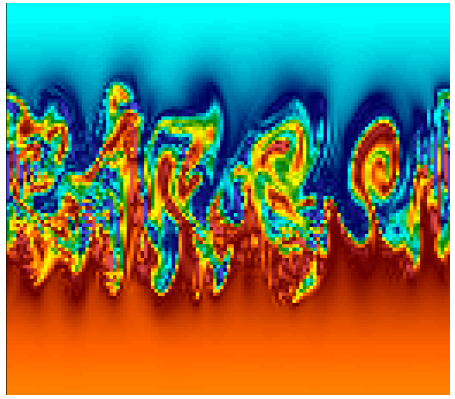


- UAL dedicated actor
  - UALVisit as a composite actor
  - CPO as input (for fusion users)



High dimensional visualization

- Where lies >3D data?
  - Kynetic simulation of plasma turbulence
  - Phase space: space+velocity

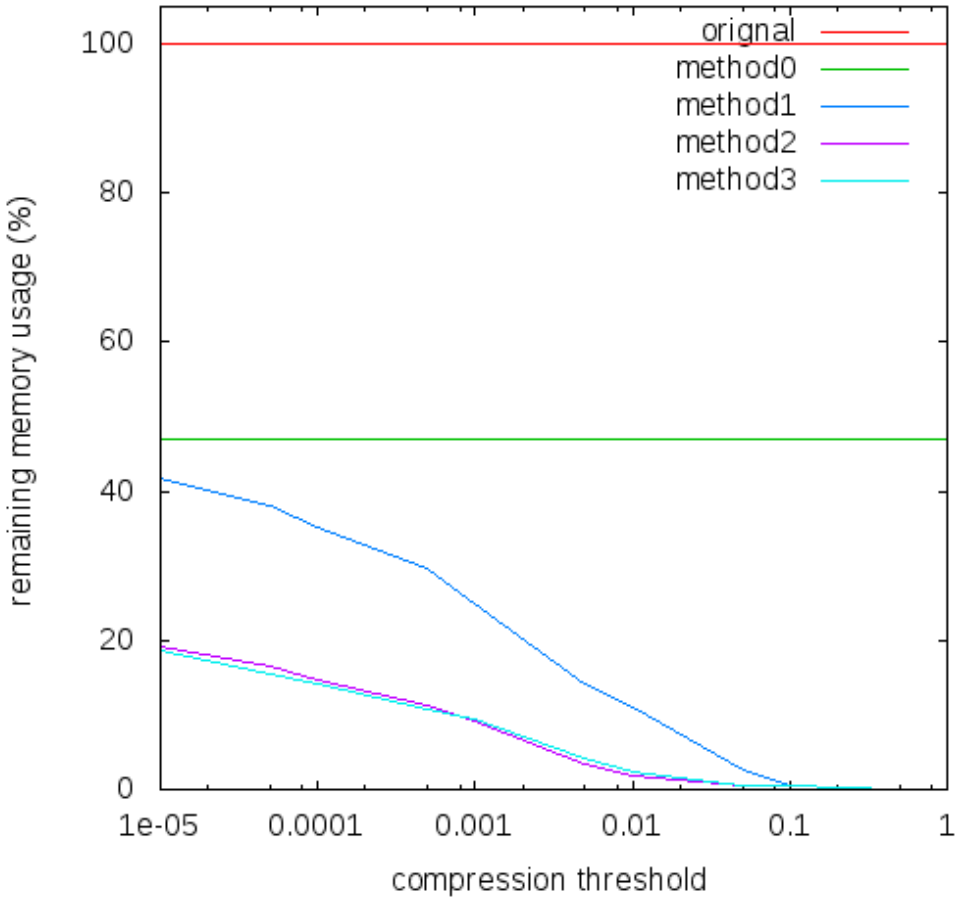


- Challenge
  - Data size, storage and memory access
  - Visualization, interpretation of information

- Proposal
  - Fast lossy compression library
  - Interactive navigation through a set of 2D slices

Compression tool

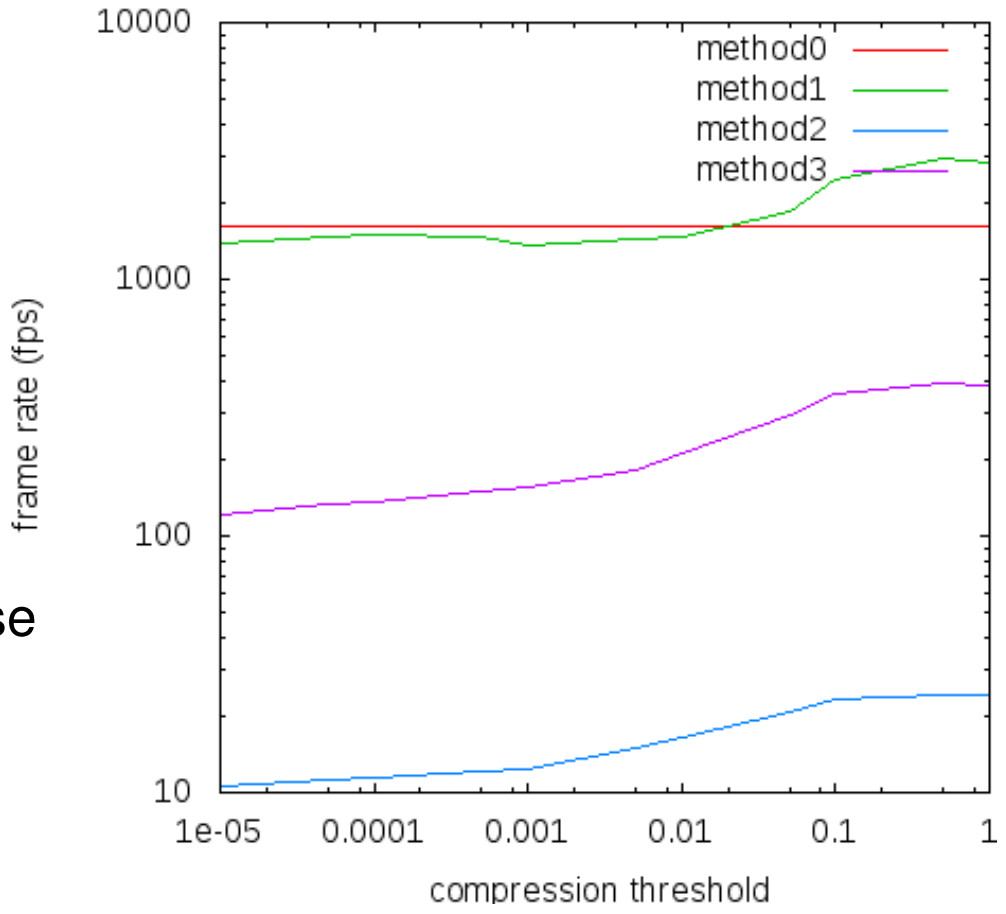
- Scheme
  - Hierarchical basis of FE
  - Bloc based algorithm
  - Parameterized threshold
- Implementation
  - Export library in C/Fortran
  - Description of parallel data decomposition
  - 2-level sparse data structure
  - Disc export through distributed HDF5 files



→ 15% to 10% remaining memory for reasonable threshold

Reconstruction tool

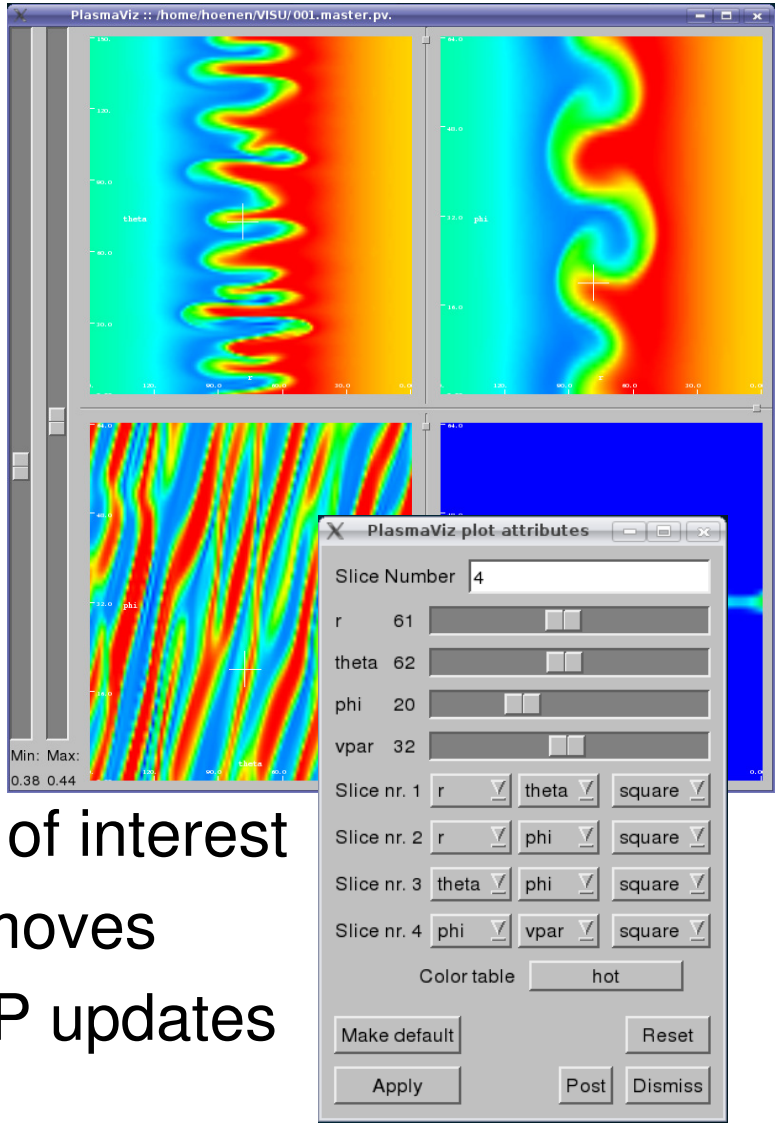
- Scheme
  - Bloc based reconstruction
  - Not yet parallelized
- Implementation
  - Import library in C++
  - Intensive use of templates
  - Fast reconstruction of dense 2D slices



→ worst case of 10 fps = ready for interactive visualization

4D visualization tool

- Tool principle
  - Linked with import library
  - Build as VisIt plug-ins
    - Reader: load compressed files
    - Plot: own Qt rendering window
    - GUI & install procedure



- Viewer implementation
  - Set of 2D slices for dimension of interest
  - Focus point following mouse moves
  - Interactive slice refresh with FP updates

Impact on community

- Python tools widely used by ITM ☺
  - Different users from all IMPs
  - Simple to learn, to use and to maintain
- Few usage of VisIt tools ☹
  - Lack of CPO XML description enhancement
  - Might change with grid structure in Edge CPO ☺?
- Compression tools and 4D visualization ☺
  - GYNVIZ project of HLST: post-processing data coming from EU gyrokinetic simulation codes

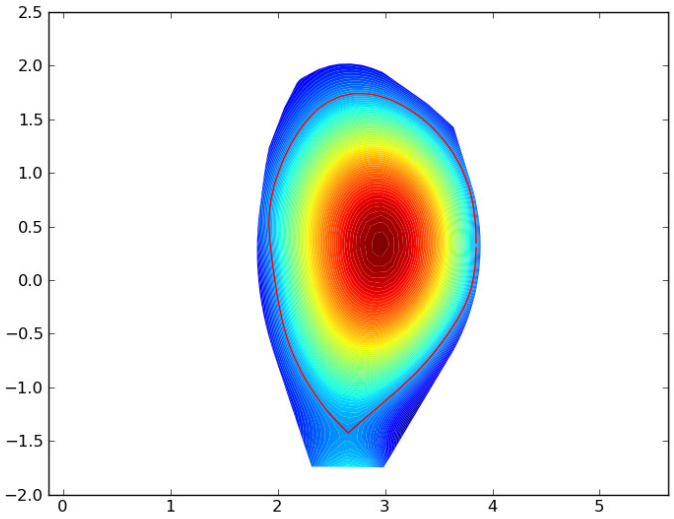
Sustainability path

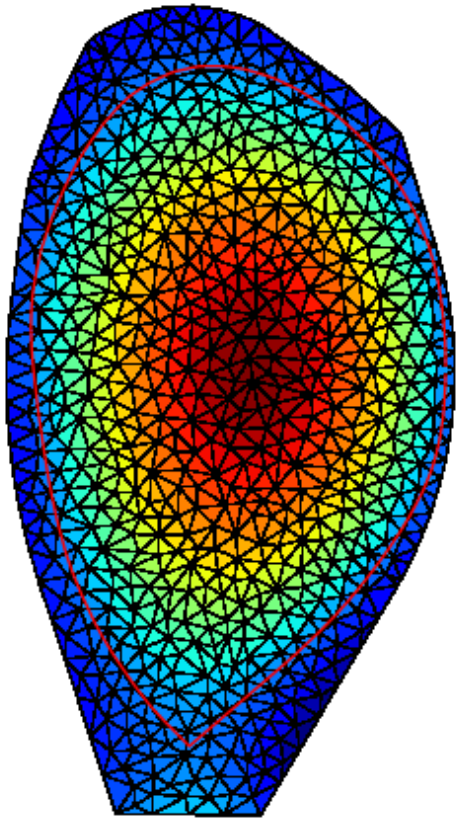
- External open source software
  - Python and VisIt supported by their community
- UAL interfaces and Kepler actors
  - Maintained by ITM for fusion community: small costs thanks to XML automatic generation! ☺
  - Generic version of actors could be taken by Kepler community
- Compression tools and 4D visualization
  - Open source (CECILL-B license)
  - Maintained by UDS

Thank you!

Python user experience

- Happy fusion user: B. Faugeras (ITM)
  - In a few hours (3-4 max)
  - Add physics code (Equinox) into Kepler
  - Interface Equinox output with Python actor
  - First visualization with 1D plots
  - Install Matplotlib last version (0.99.3 → 1.0.1)
  - Generates advance plots thanks to Matplotlib examples (here for triplot)





Relevance of 4D distribution functions

- Full kinetic model
  - 6D = 3D in space + 3D in velocity
- Gyrokinetic model
  - Considering particles motion along fields lines
  - 5D = 3D in space + 2D ($v_{//}$,$v_{\perp}$)
- Reduced gyrokinetic model
  - Introducing the adiabatic invariant μ as modulus of $v_{\perp}$
  - μ appears as a parameter in the equation
  - 4D = 3D in space + 1D ($v_{//}$) for each value of μ