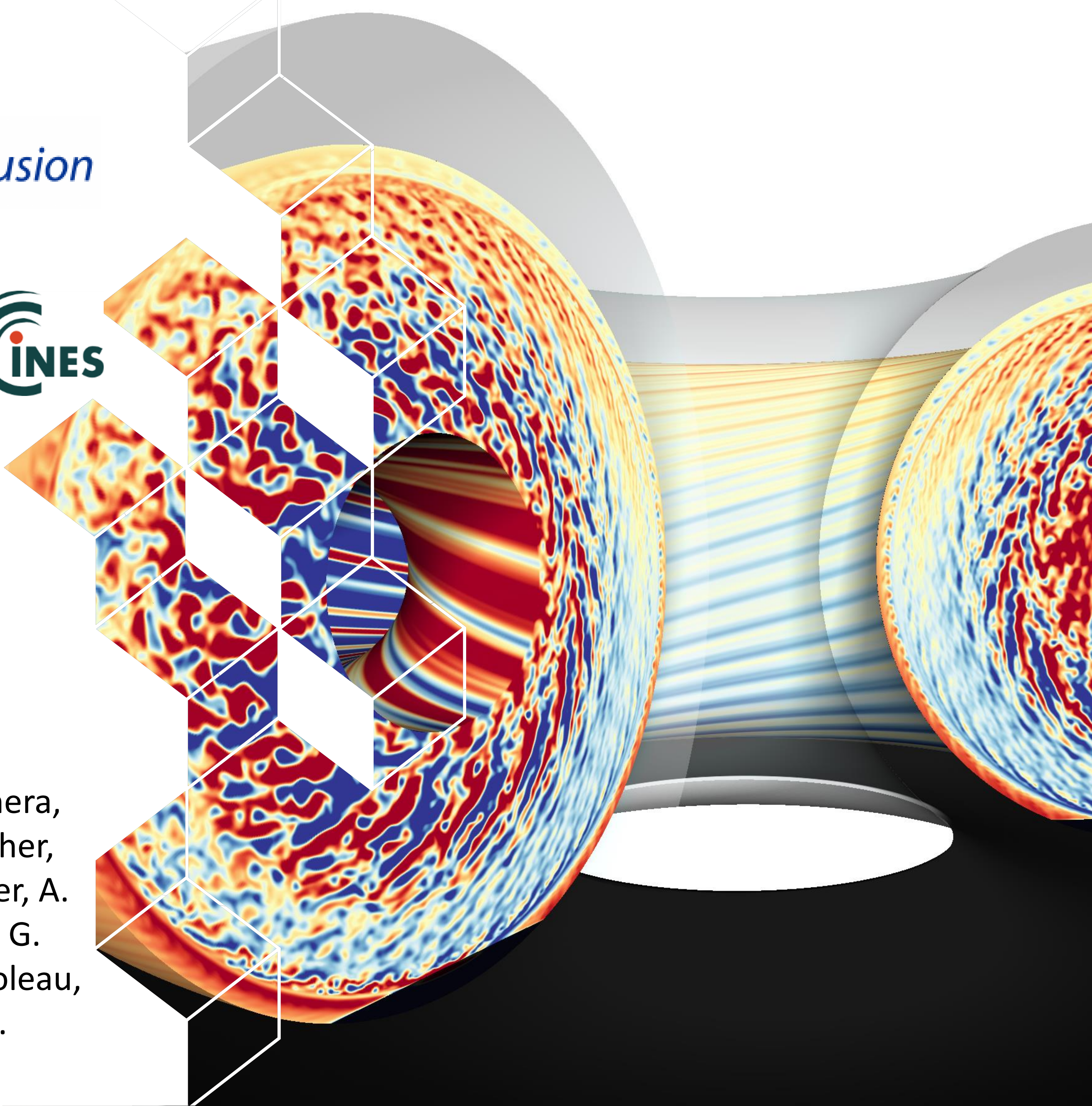
Exascale challenges for tokamak plasma turbulence simulations

Y. Asahi, R. Bigué, J. Bigot, E. Bourne, N. Bouzat, E. Caschera, Y. Cho, L. De Gianni, G. Dif-Pradalier, P. Donnel, C. Ehrlacher, X. Garbet, P. Ghendrih, C. Gillot, V. Grandgirard, E. Gravier, A. Hoffmann, A. Kara, G. Latu, B. Legouix, M. Lesur, K. Lim, G. Lo-Cascio, E. Malabeouf, Y. Munschy, K. Obrejan, T. Padioleau, C. Passeron, M. Peybernes, Z. Qu, T. Rouyer, Y. Sarazin, E. Sonnendrücker, R. Varennes, P. Vidal



Outline

1. General context

2. The GYSELA code: 10 years of upgrades

3. Roadmap for the future: upgrades to target Exascale simulations

4. Conclusion

Two strategies to obtain controlled fusion reactions in a lab: inertial fusion vs magnetic fusion

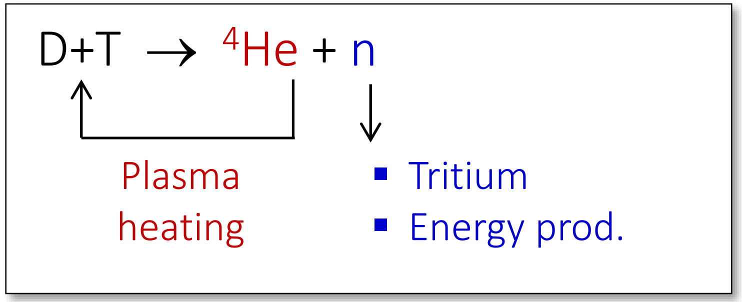
To sustain a plasma without external heating ($P_{fus} = P_{loss}$), one needs to fullfil the Lawson criterion

Ignition = $P_{\text{self-heating}} > P_{\text{loss}}$ ⇔ $nT\tau_E \geq 3 \cdot 10^{21} \mathrm{keV\ s/m^3}$

density

Temperature
(Optimal ~$1.5*10^6$K)

Energy confinement time



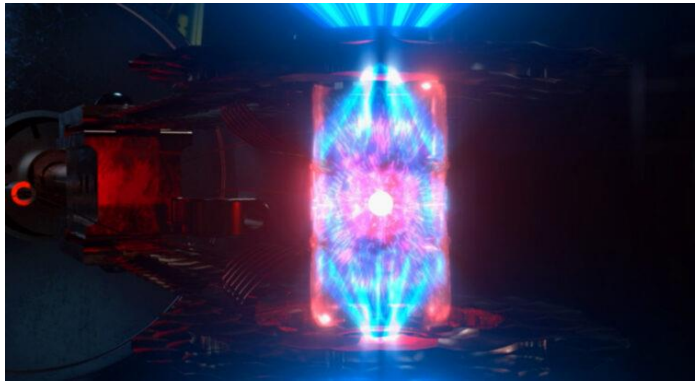
D+T → ^{4}He + n

Plasma heating

- Tritium
- Energy prod.

inertial fusion

NIF

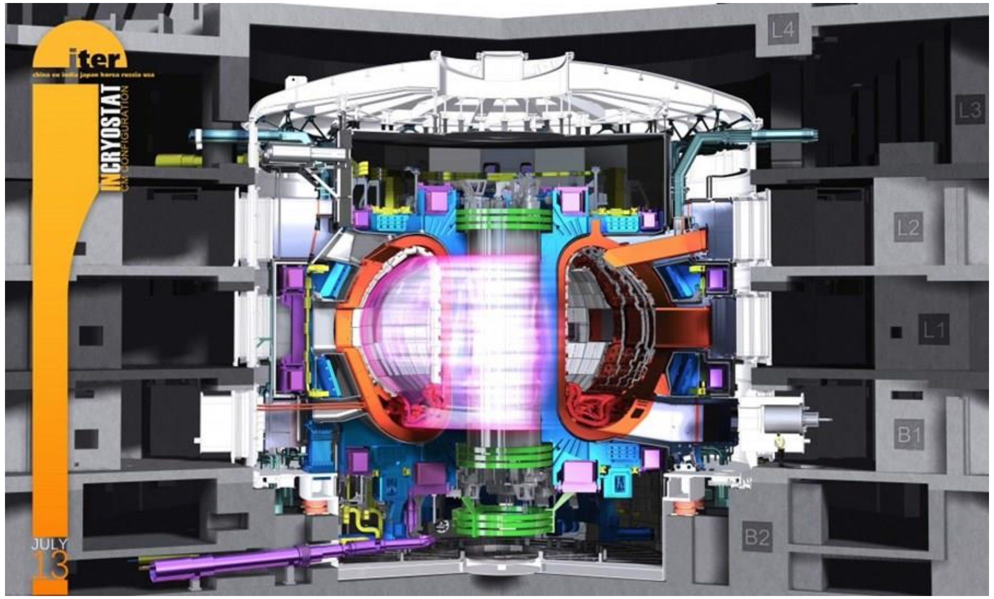


Ignition obtained for the first time! $P_{X-rays} < P_{fus}$
[H. Abu-Shawareb, PRL 2022]

- Historical physics result!
- But still far from Electricity production

$P_{fus} \sim P_{lasers} \ll P_{grid}$ + low repetition rate

magnetic confinement



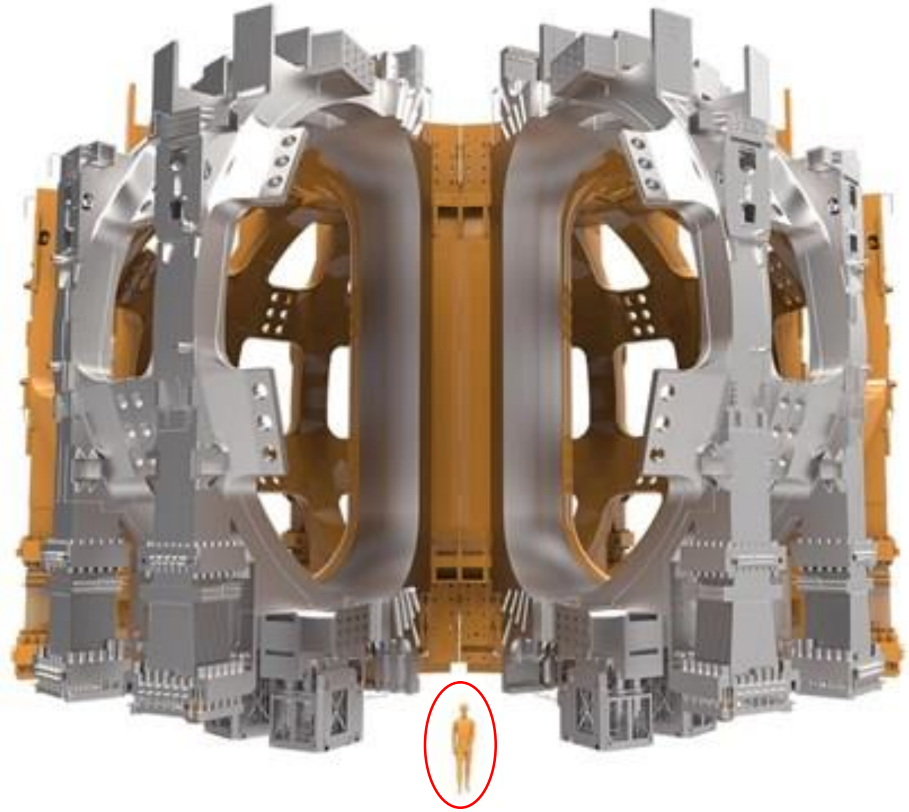
ITER

ITER project

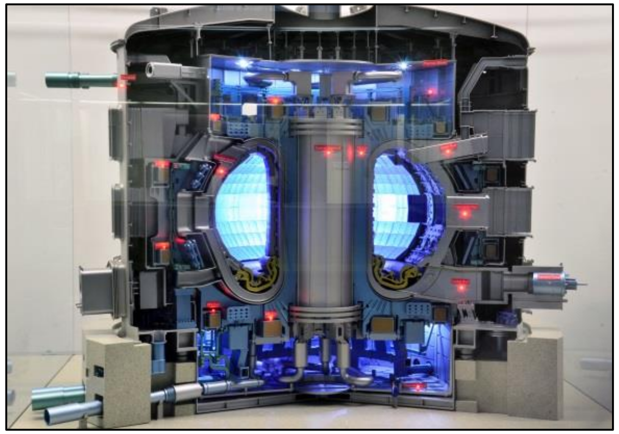
One of the largest scientific project of mankind history

International project: China, India, Japan, Korea, Russia, UE, USA

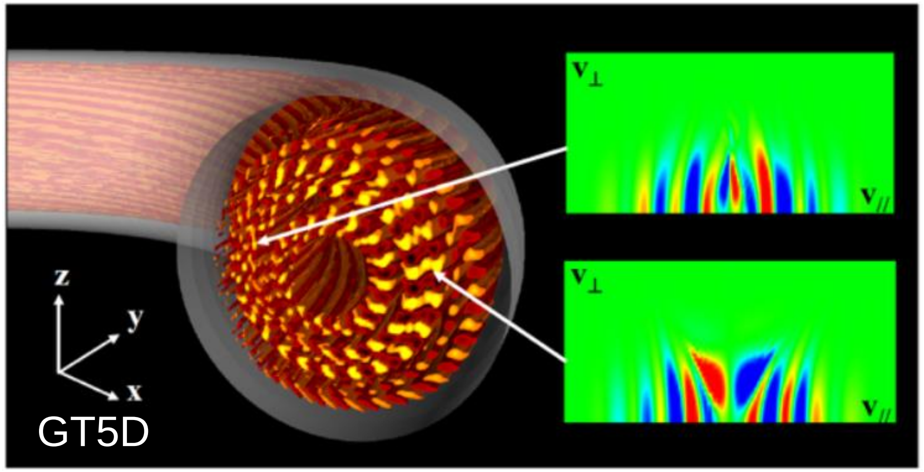
Under construction at Cadarache



Success of fusion requires understanding and prediction of transport in tokamaks



ITER project



- To optimize performance and minimize risks, each ITER scenario will have to be numerically validated.
- A complete chain of numerical tools will be required, ranging from scale models, which can be used in real time, to first-principles simulations, which are more costly but more reliable.
- Turbulent transport mainly governs confinement in Tokamaks
- Tokamak plasmas weakly collisional → Kinetic approach mandatory
    - Fusion plasma turbulence is low frequency → fast gyro-motion is averaged out
    - Gyrokinetic approach: phase space reduction from 6D to 5D

Outline

1. General context

2. The GYSELA code: 10 years of upgrades

3. Roadmap for the future: upgrades to target Exascale simulations

4. Conclusion

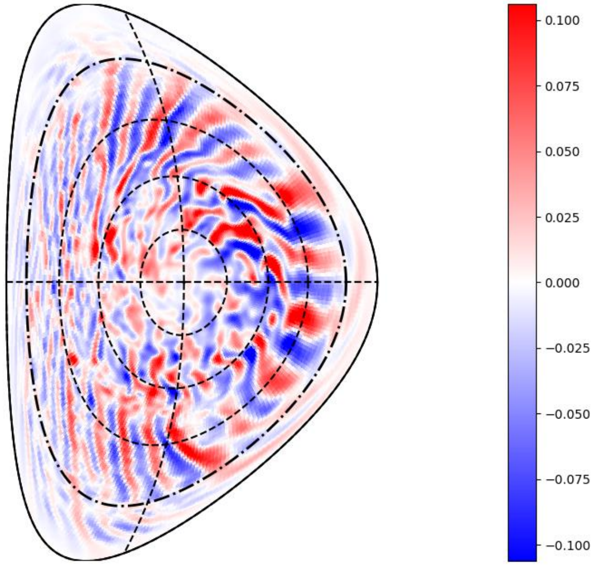
The GYSELA code

- GYSELA is the unique gyrokinetic code based on a **semi-Lagrangian** scheme
  - Fortran 90 + few C modules with hybrid MPI/OpenMP parallelisation optimized up for up to 750,000 cores
- Development in strong collaboration between physicists, mathematicians and computer scientists
  - Ex: **SELALIB** software library = **joint effort INRIA** Project Lab FRATRES - **CEA**/IRFM – MINGUS (INRIA/Rennes) and TONUS (INRIA/Strasbourg) + Max-Planck-Institut für Plasmaphysic (**IPP**/Garching) started in 2011.

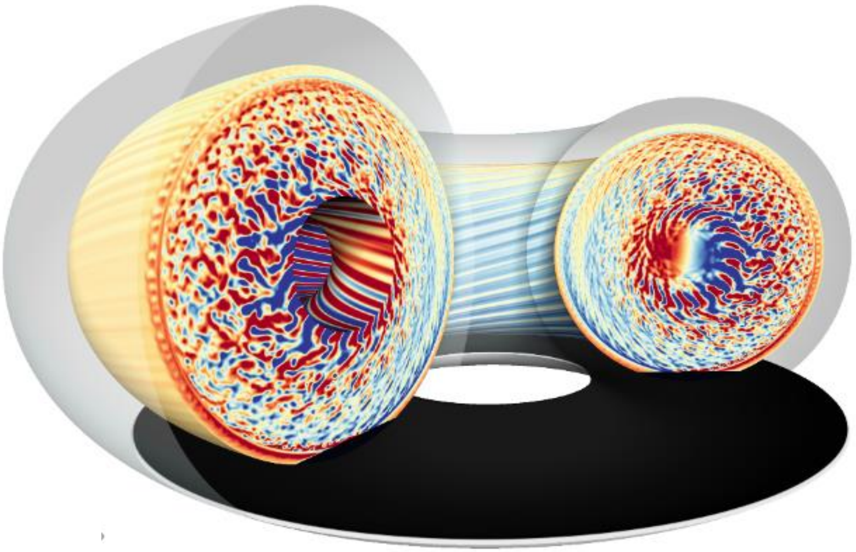
→ SELALIB modules recently successfully coupled to GYSELA *[Emily Bourne, PhD 2022]*

International collaboration France + Switzerland+ Germany + Singapour

Improvements of the physics model in the last 10 years



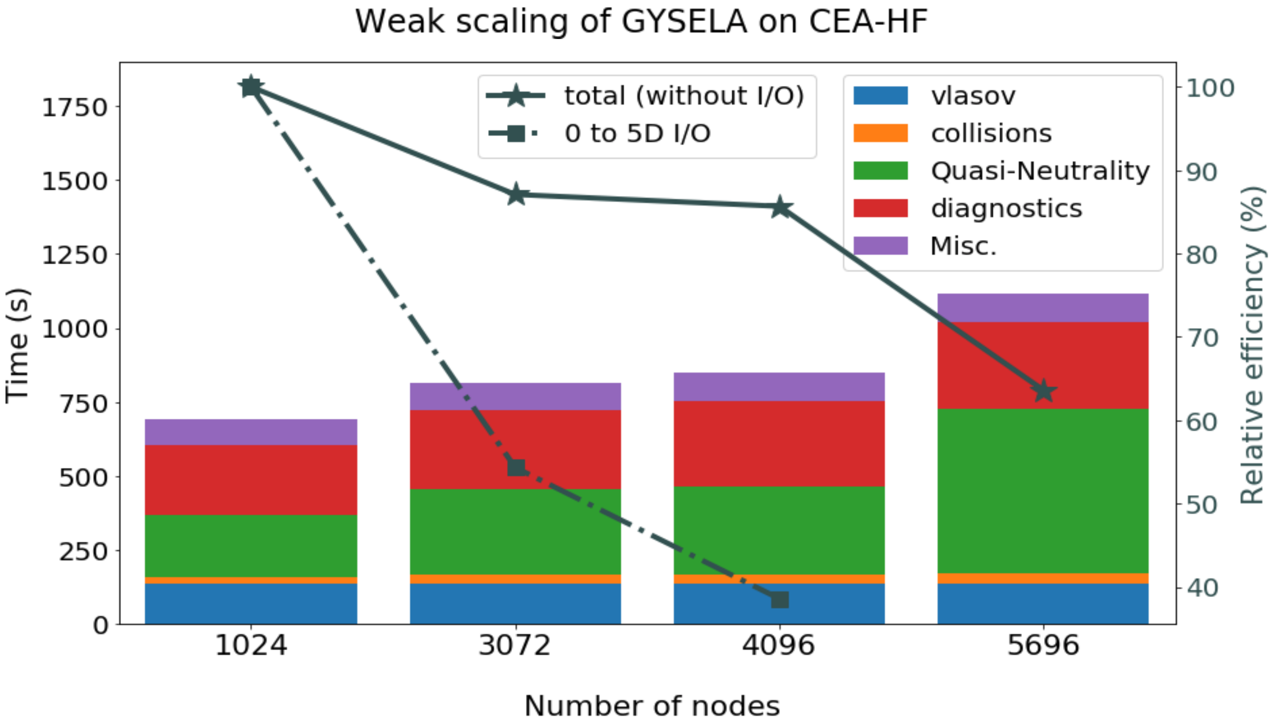
- Electrons: from adiabatic response (not simulated) to kinetic
  [C. Ehrlacher 2018, V. Grandgirard 2019]
- Simulation of impurities (= minority ion species)
  [D. Estève 2018, P. Donnel 2019, K. Lim 2023]
- Geometry: from circular to shaped plasmas + ripple
  [Bourne 2023, R. Varennes 2023, PhD L. De Gianni 2023-2026]
- Boundary conditions: from simple buffer to more realistic immersed boundaries
  [E. Caschera 2018, Dif-Pradalier 2022, PhD Y. Munschy 2021-2024]
- From electrostatic (constant magnetic field) to electromagnetic
  [PhD C. Gillot 2017-2020, PhD R. Bigué 2023-2026]
- More complex operators: collisions, sources…
  [P. Donnel 2018, G. Lo-Cascio 2022]

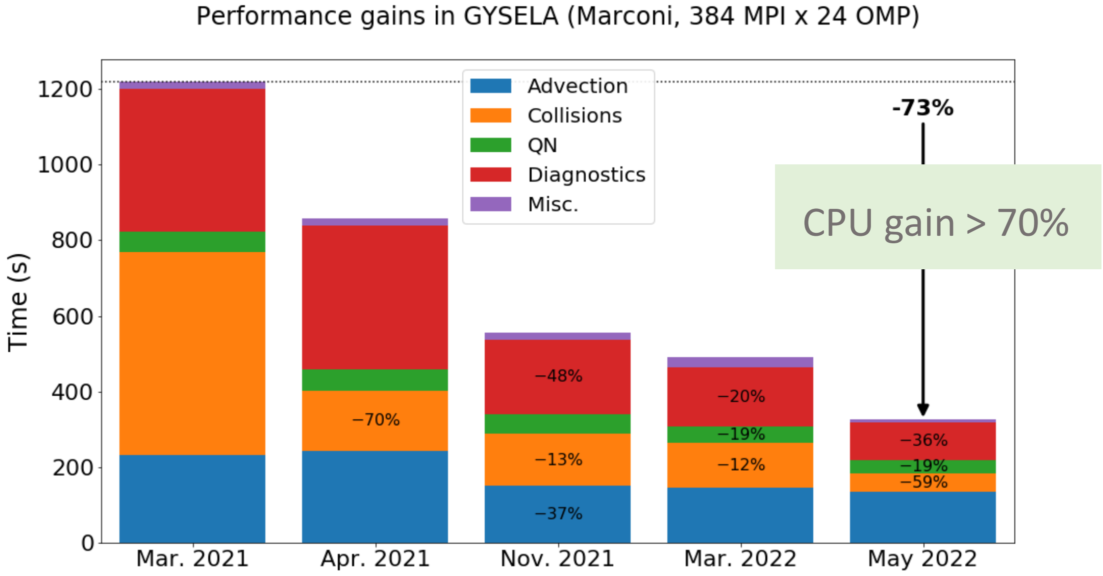


Need for constant code optimization to partly compensate for the increased complexity of the physics + adapation to new architectures

Numerical improvements in 2021-2022

- Vectorisation
- Blocking → Cache optimisation
- Asynchronous MPI communications





CPU gain > 70%

Relative efficiency of 85% on more than 500k cores and 63% on 729 088 cores

[V. Grandgirard et al., PASC 2022]

Outline

1. General context

2. The GYSELA code: 10 years of upgrades

3. Roadmap for the future: upgrades to target Exascale simulations

4. Conclusion

Need for Exascale computing to simulate ITER

- The dominant transport in tokamaks is related to electromagnetic turbulence. A first principle code is intrinsically multi-scale as it needs to solve simultaneously small scale turbulence and large scale evolution of the plasma

- **Spatial scale**
  - Turbulent structures proportionnal to the ion Larmor radius $\rho_i = \frac{\sqrt{mk_bT}}{eB}$
  - Size of plasma 'a'
  - The grid resolution scales as $\rho_*^{-2}$where $\rho_* = \frac{\rho_i}{a}$. WEST: $\rho_*^{-1} = 250$, ITER: $\rho_*^{-1} = 1000$

➢ Grid resolution needs to be increase by a factor ∼ 16 from WEST ($\sim 10^{11}$points) to ITER ($\sim 1.6 * 10^{12}$points)

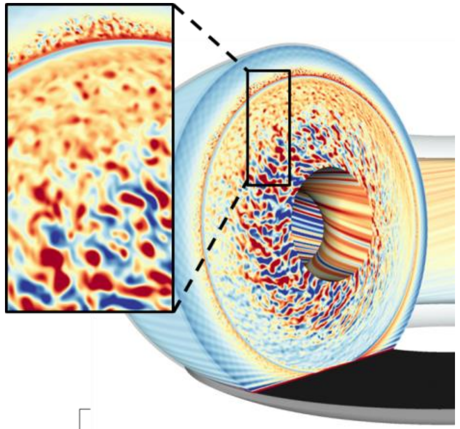
- **Temporal resolution**
  - Correlation time of turbulence proportionnal to $\Omega_{ci} = \frac{eB}{m}$
  - Confinement time of the plasma scales as number of iterations = $\tau_E \Omega_{ci} \propto \rho_*^{-2}$(pessimistic) or ρ_*^{-3}(optimistic)

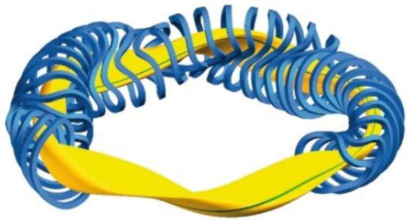
➢ Number of iterations increased by a factor 16 to 64 from WEST to ITER

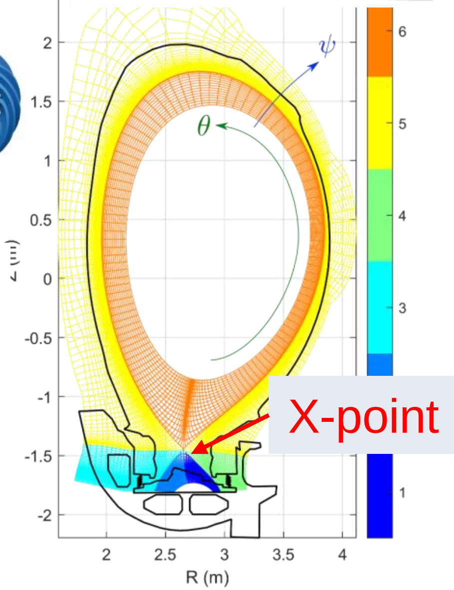
- Well resolved simulations of WEST ∼ 10 millions of CPU hours → impossible to simulate ITER with current code& HPC

Roadmap for Gysela-X++ towards exascale
→ Why do we choose to rewrite GYSELA ?

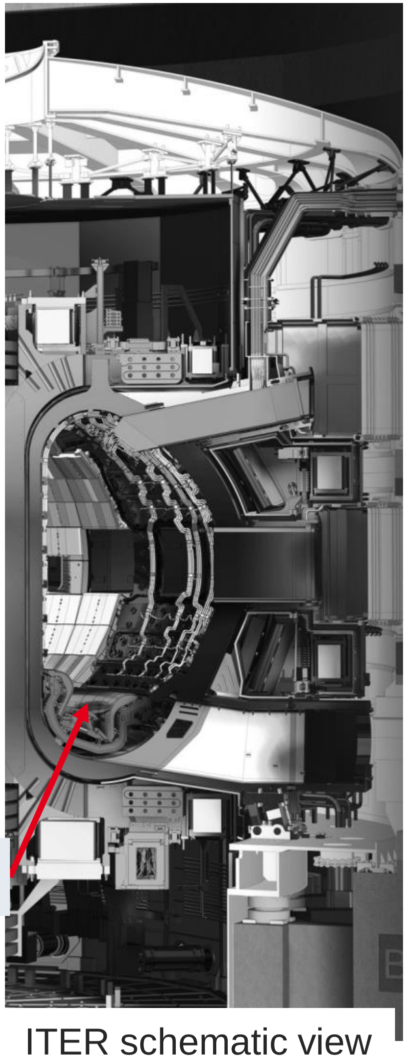
- **Gysela-X++** = GYSELA in modern C++ with X-point for exascale ITER core-edge turbulence simulations (+ 3D via BPI / Renaissance Fusion)
  - **Rewritting of the code in modern C++ with MPI + Kokkos**
    - Portable code on new exascale architectures
  - **Non-uniform meshes**
    - relevant density & temperature gradients at edge-SOL
  - Semi-Lagrangian scheme for multi-patches
    - **X-point** geometry
  - Implementation of a **3D scalable Poisson solver**
    - X-point configuration + stellarator configuration
  - **Scalable I/O and in-situ diagnostics**
- Development of the new code supported by multiple projects/collab.
  - National projects: Moonshot CExA (2023-2025), PTC Dose (2023-2024), PTC Assist (2023-2024) + PEPR NUMPEX (2023-2028) + ANR AIM4EP (2022-2025)
  - EoCoE-III
  - BPI funding in kink with Startup Renaissance Fusion (2024-2027)
  - European projects: EUROfusion TSVV (2022-2026)
  - Joint research center (SAFE) Singapore Alliance with France for Fusion Energy







SOLEDGE-3X
X-point geometry



ITER schematic view

Gysela-X++ towards exascale
→ Complete rewriting of the code in modern C++ (1/2)

- Proof of Concept: 2D prototype VOICE++ in modern C++ to address plasma-wall interaction problem

Input YAML + Paraconf (developed at MDLS)

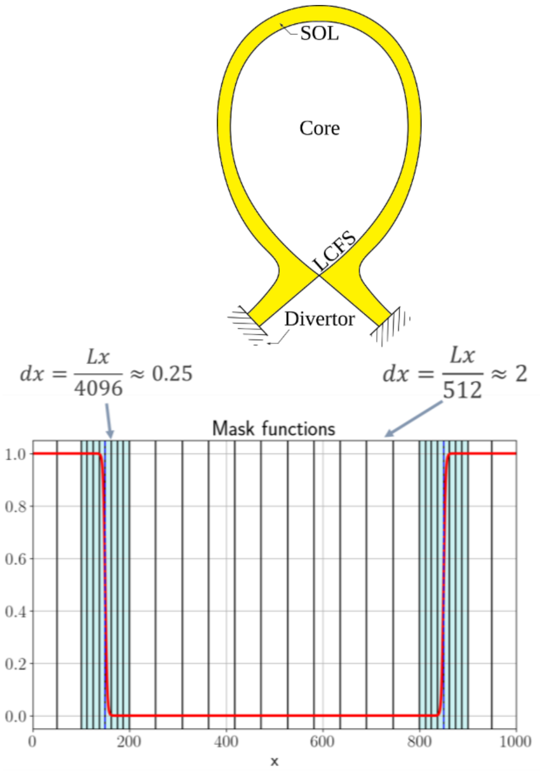
Geometry C++ template meta-programming domain decomposition library (developed at MDLS)

Vlasov (1D,1V) Semi-lagrangian scheme with non-equidistant splines

Poisson 1D solver based on spline finite elements

I/O HDF5 via PDI

+ GITLAB + google tests + GITLAB-CI + SPACK installation



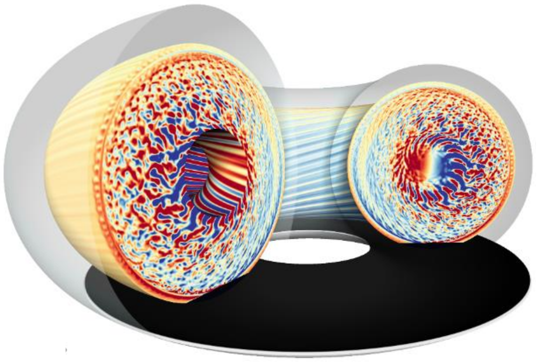
[E. Bourne et al., JCP 2023]

[Y. Munschy et al., NF 2024]

https://github.com/gyselax/gyselalibxx

Gysela-X++ towards exascale

→ Complete rewriting of the code in modern C++ (2/2)



Input YAML + Paraconf (developed at MDLS)

+

High level of parallelism MPI + KOKKOS

Geometry DDC a domain decomposition library — C++ template meta-programming domain decomposition library (developed at MDLS)

Vlasov **5D semi-lagrangian** scheme with non-equidistant splines

+ Adding of all operators (gyroaverage + sources + collisions)

Poisson **3D solver** based on 2D spline FEM + 1D

+ **Ampere** **(2D+1D) solver**

I/O **IMAS** via PDI

+ In-Situ diagnostics

+ GITLAB + google tests + GITLAB-CI + SPACK installation

- Main idea: Mutualize all modules independent on the 3D space geometry between Fortran code and C++ code

Extract F90 modules: rewrite them in C++/ GPU then plug them to F90 old code + C++ new code

Conclusions

- The GYSELA code at the era of pre-exascale for ion-scale turbulence simulations for current tokamaks
  - Optimized up to more than 500k cores on standard CPU architecture (ex: AMD milan)
  - Resource needs: more than 150 millions of CPU hours / year
  - Petabytes of data manipulated per simulation with huge reduction to limit the storage to few Terabytes

→ Lot of physics still to be explored with this version during the development of Gysela-X++ .

- Gysela-X++ : Rewritting in modern C++, more modular and scalable on different accelerated architectures
  - More realistic temperature gradients at the edge: Non-equidistant mesh
  - More realistic geometry: X-point and stellarators
  - More physics: neutrals, fusion reactions…
  - Based on DDC library + Kokkos
  - In situ diagnostics foreseen

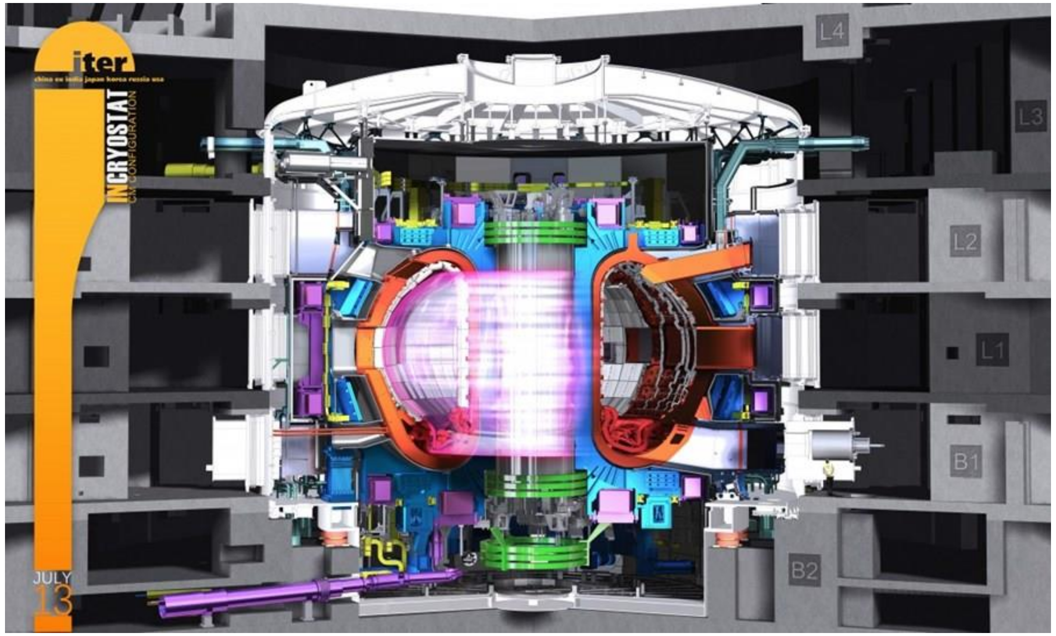
Backup slides

Two main magnetic topologies to confine a plasma

Tokamak

☺ Confinement

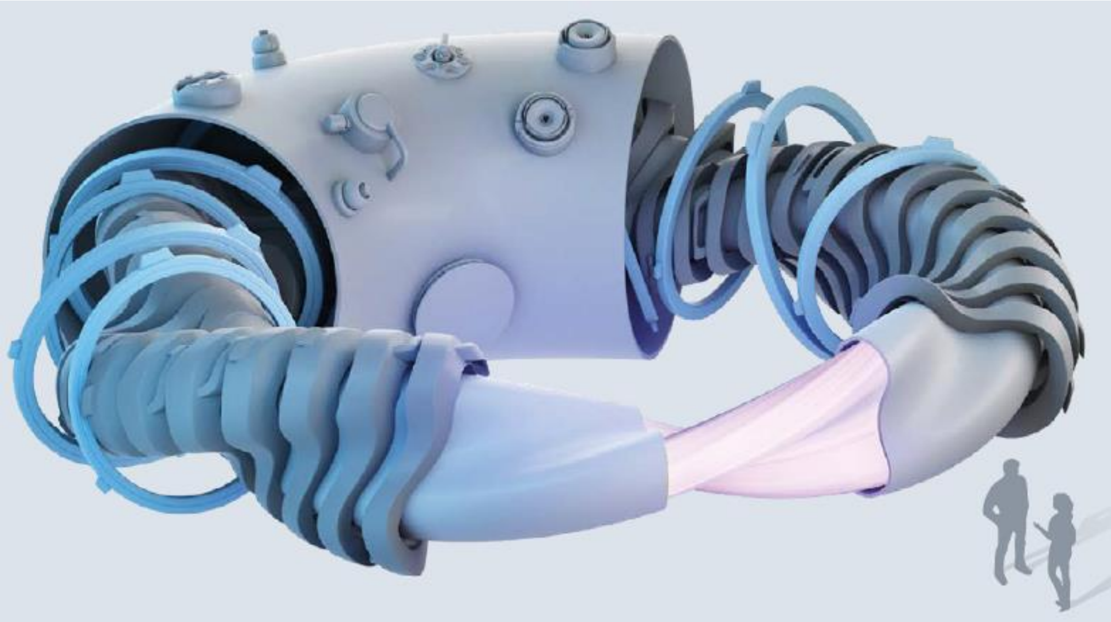
☹ Plasma current, Disruptions



Stellarator

☺ Stability

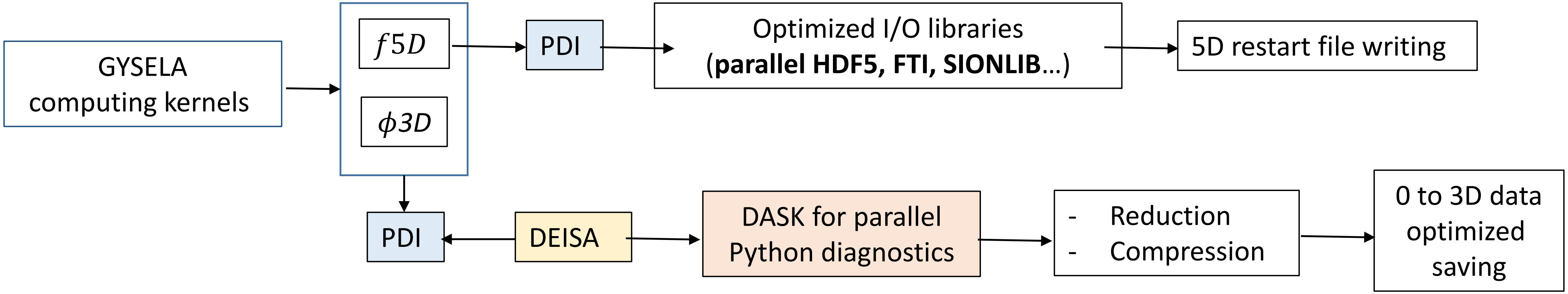
☹ α-confinement, Technology



Work in progress : Optimized Data workflow

- Main idea : **Decouple I/O from computing kernels**
  - HPC/IA coupling not trivial : CPU or GPU for computing kernels (Fortran or C++) + GPU for diagnostics +AI (python)

- **Development of in-situ diagnostics framework based on PDI + DEISA + DASK**
  - PDI Data Interface for handling I/O (developed at MDLS) https://pdi.julien-bigot.fr/master/
  - DEISA (dask-enabled in situ analytics) library (developed at MDLS+INRIA) *[A. Gueroudji et al., 2023]*
  - DASK a flexible library for parallel computing in Python https://docs.dask.org/



GYSELA computing kernels

$f5D$

$\phi 3D$

PDI

Optimized I/O libraries
(**parallel HDF5, FTI, SIONLIB**…)

5D restart file writing

PDI

DEISA

DASK for parallel Python diagnostics

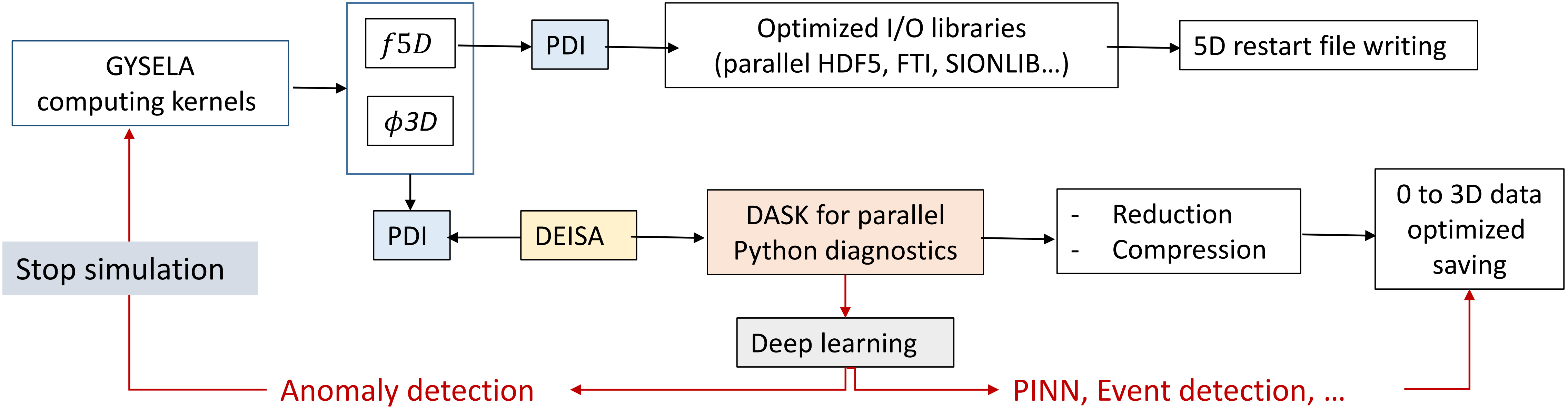
- Reduction
- Compression

0 to 3D data optimized saving

Within national PEPR NUMPEX (2023-2027)

2028 objective: In-situ AI diagnostics

- Development of **in-situ AI diagnostics to optimize exascale simulations**:
  - Data compression
  - Automatic anomaly detection: Automatic stop of simulation → CPU consumption optimization
  - Automatic rare event detection: Optimisation of diagnostic saving → Memory storage reduction



GYSELA computing kernels

$f5D$

$\phi 3D$

PDI

Optimized I/O libraries (parallel HDF5, FTI, SIONLIB…)

5D restart file writing

PDI

DEISA

DASK for parallel Python diagnostics

- Reduction
- Compression

0 to 3D data optimized saving

Deep learning

Stop simulation

Anomaly detection

PINN, Event detection, …