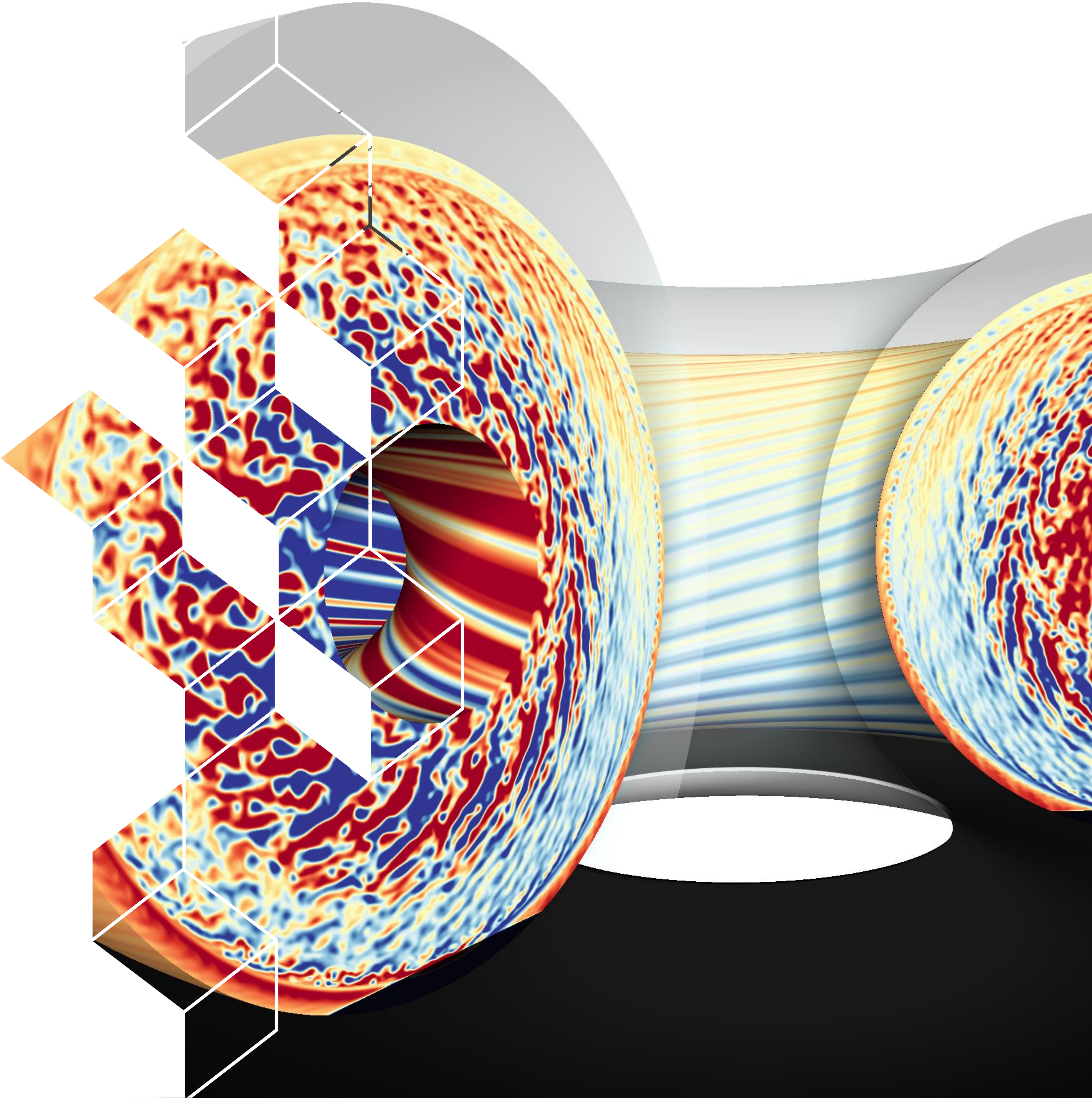
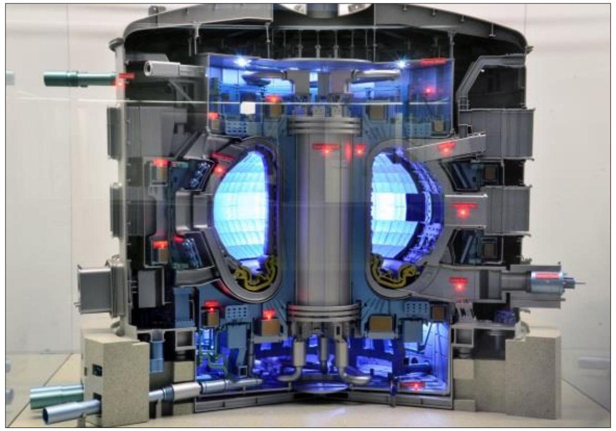
GyselaX++: Exascale Challenges for tokamak plasma turbulence simulations

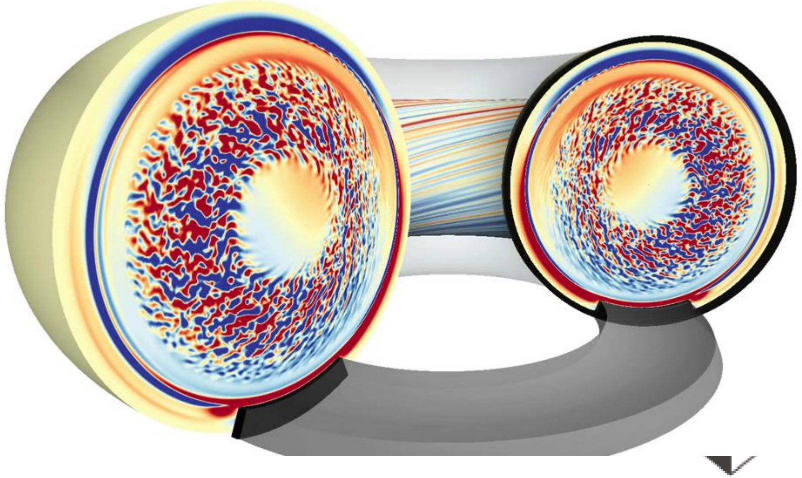
Virginie Grandgirard



First principle simulations required for ITER
→ Gyrokinetic plasma turbulence simulations



ITER project



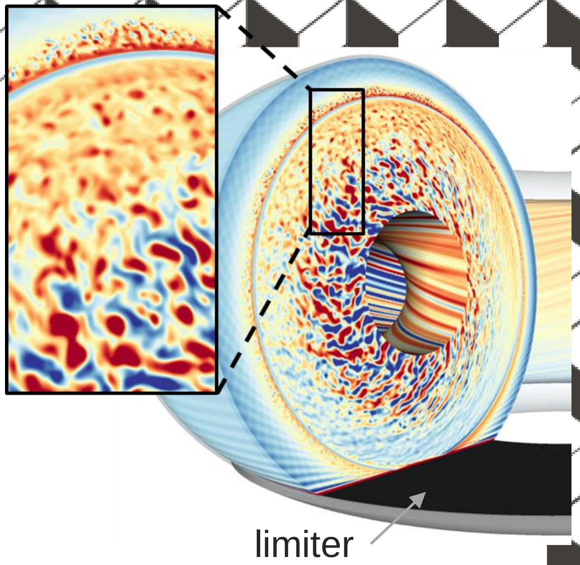
GYSELA simulation

- 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

GYSELA: a highly parallelised code running at petascale

- Gyrokinetic codes require state-of-the-art HPC techniques and must run efficiently on several thousand processors
  - Non-linear 5D simulations (3D in space + 2D in velocity)

    + multi-scale problem in space and time

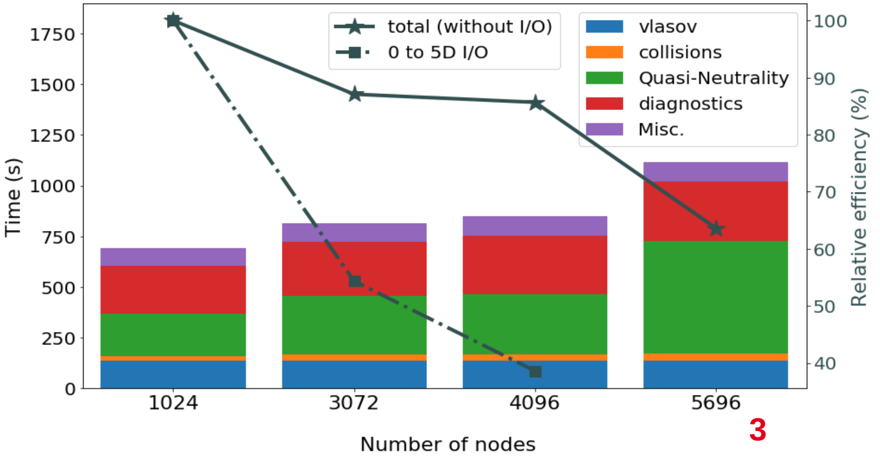


- Even more resources required when modelling both core & edge plasmas like GYSELA

- GYSELA = Fortran 90 code with hybrid MPI/OpenMP parallelisation optimized up to 730,000 cores
  - **Relative efficiency of 85% on more than 500k cores** and 63% on 730k cores on CEA-HF (AMD EPYC 7763)

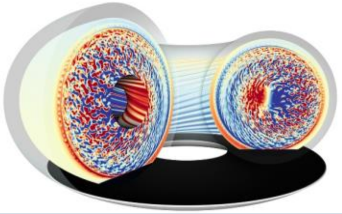
- **Intensive use of petascale resources**: ~ 150 millions of hours / year
  - (GENCI + PRACE + HPC Fusion resources)

Weak scaling of GYSELA on CEA-HF



GYSELA: Exascale needs for ITER plasma turbulence simulations

- **Petascale simulations** nowadays → Electrostatic simulations on ion-scale turbulence for WEST-like plasmas

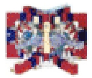


	Degree of Freedom 5D mesh x #species	#iterations	#CPUs x days	Cost of 1 simulation
- Standard simulation	(512x512x64x128x64) x 2 ~ 275 billion	~ 10000 it.	~ 65k x 3.6	~ 7 Mh CPU
- Extreme simulation performed on CEA-HF → Core-edge-limiter with trapped kinetic electrons	(1024x512x128x128x64) x 2 ~ 1100 billion	~ 6500 it.	~ 262k x 2.5	~16 Mh CPU

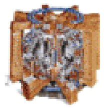
- Huge amount of data per simu. : Handles tens Pbytes data → Only few Tbytes saved

Ongoing work:
AI to optimize I/O ?

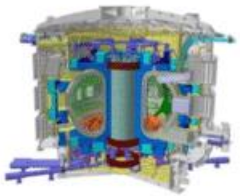
WEST
IRFM/CEA
France



JET



ITER



DEMO

V = 25 m³ V = 80 m³ V = 800 m³ V = ≈ 1000-3500 m³

mesh size x 32 required for ITER

Exascale needs for ITER plasma turbulence simulation with electromagnetic effects

Joint efforts
HPC+HPDA+IA

How to prepare GYSELA to HPC exascale architectures ?
→ Huge efforts of optimization and porting during EoCoE-II

- Target architectures:
  - 3 different architectures in the top 20

- Porting in 2021-2022 via CEA-RIKEN collaboration and GENCI support with ATOS

- Porting in 2022-2023 with HPE and EOLEN in the frame of ADASTRA Contrat de Progrès at CINES and with SCITAS-EPFL in the frame of EUROfusion Advanced Computing Hub

- May 2022: Opportunity to run during « Grand Challenge » campaign

TOP 500 The List. NOVEMBER 2022

Rank	System	Cores	Rpeak (Pflop/s)
2	**Supercomputer Fugaku** – A64FX 48C, Fujitsu - RIKEN Center for Computational Science – **Japan**	7,630,848	537.21
11	**Adastra** – HPE Cray, AMD Instinct MI250X GENCI-CINES – **France**	319,072	46.10
20	**CEA-HF** – BullSequana XH2000, AMD EPYC 7763, Atos – CEA – **France**	810,240	23.24

- Operator refactoring (collisions, sources) + Performance optimization at node level (vectorization, blocking, asynchronous MPI communications) → Gain > 70%

Impossible without HPC experts

⚠ Good performance on the 3 architectures with same Fortran code via OpenMP directives
→ Not feasible without rewritting, duplication of most of the kernels

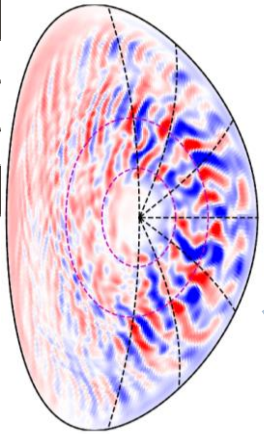
Roadmap for GyselaX++ towards exascale
→ Why do we choose to rewrite GYSELA ?

- 20 years-old code written in Fortran with hybrid MPI/OpenMP parallelism
- Unique code for both CPU (AMD milan or ARM-A64FX) and GPU with OpenMP directives is NOT optimal → extremely difficult to optimize on all architectures.
- Non-equidistant mesh mandatory for core-edge-SOL turbulence simulations

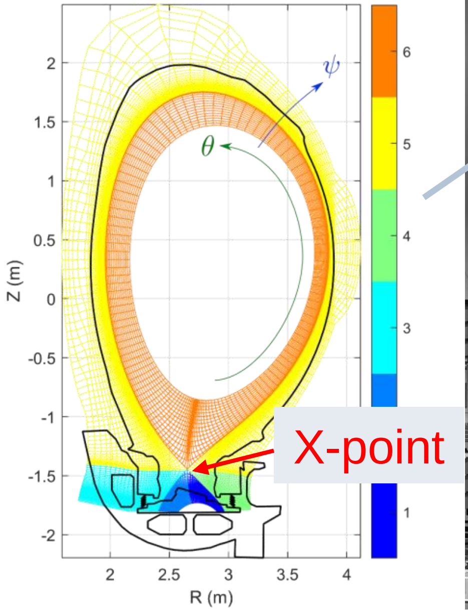
  → Modifying splines in GYSELA = rewrite most of the kernels
- X-point geometry

  → Development of new semi-Lagrangian scheme required to treat multipatches

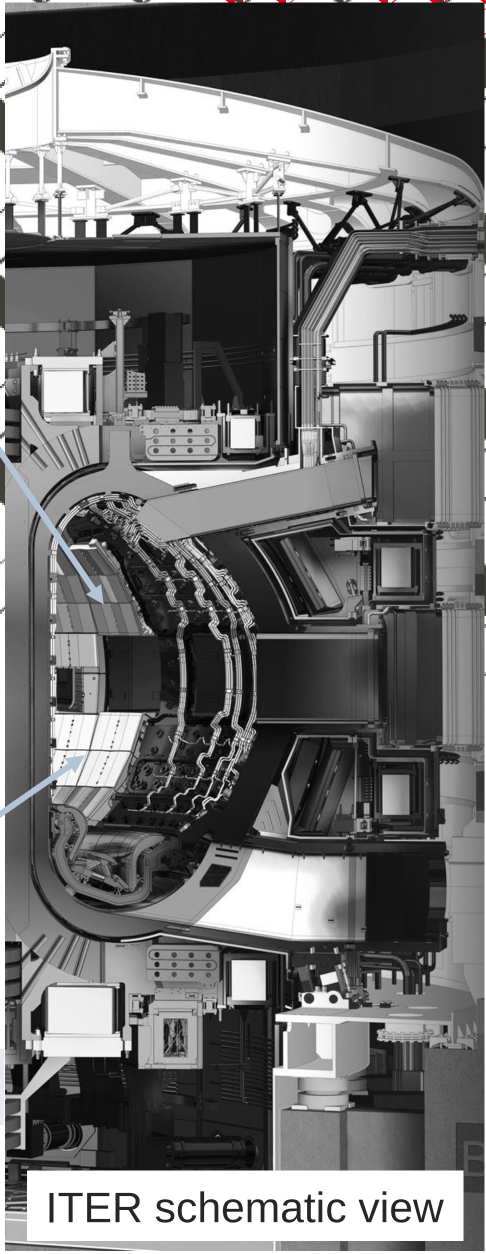
Simpler to rewrite main kernels in modern C++ from scratch → GyselaX++ code



GYSELA
D-Shape geometry



SOLEDGE-3X
X-point geometry



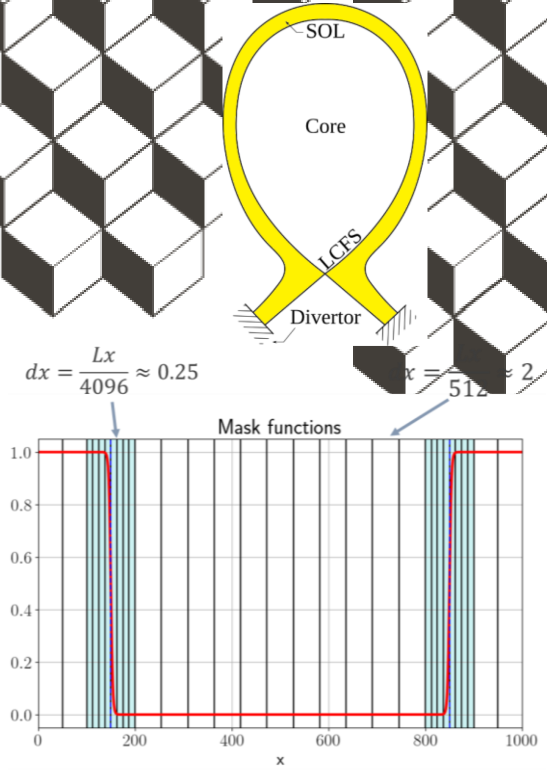
ITER schematic view

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	DDC – a Discrete Domain Computation library: 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



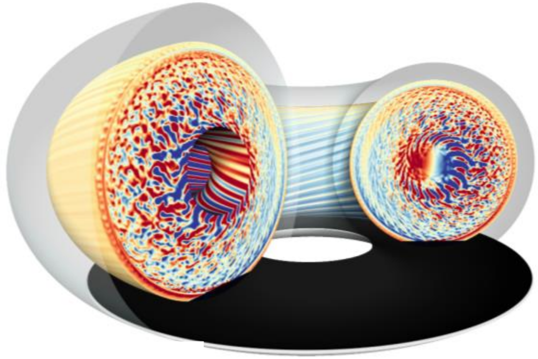
[E. Bourne et al., accepted JCP 2023]

[Y. Munschy et al., submitted to PoP]

https://github.com/gyselax/gyselalibxx

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

- 5D code in modern C++ scalable on exascale architectures



Input — YAML + Paraconf (developed at MDLS)

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

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

I/O — **IMAS** via PDI

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

+ High level of parallelism MPI + KOKKOS

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

+ Ampere (2D+1D) solver

+ In-Situ diagnostics

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 of the code for the next five years.

- GyselaX++ : 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
  - Based on DDC library + Kokkos