DE LA RECHERCHE À L'INDUSTRIE

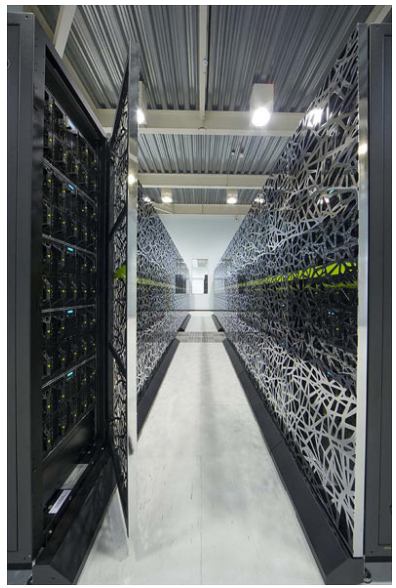
www.cea.fr

Jorek, a parallel code for modelling non linear MHD in Tokamaks

M. Becoulet, G. Dif-Pradalier, A. Fil, V. Grandgirard, G. Latu, E. Nardon, F. Orain, C. Passeron, A. Ratnani, C. Reux

Collaborations with: INRIA, IPP Garching, ITER Org., several french Universities

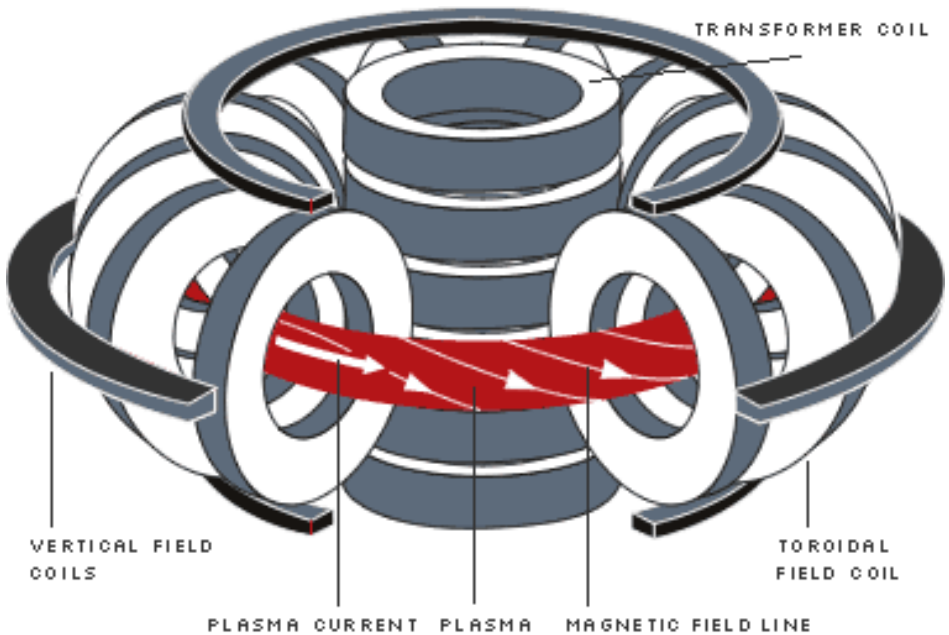
May 16, 2013



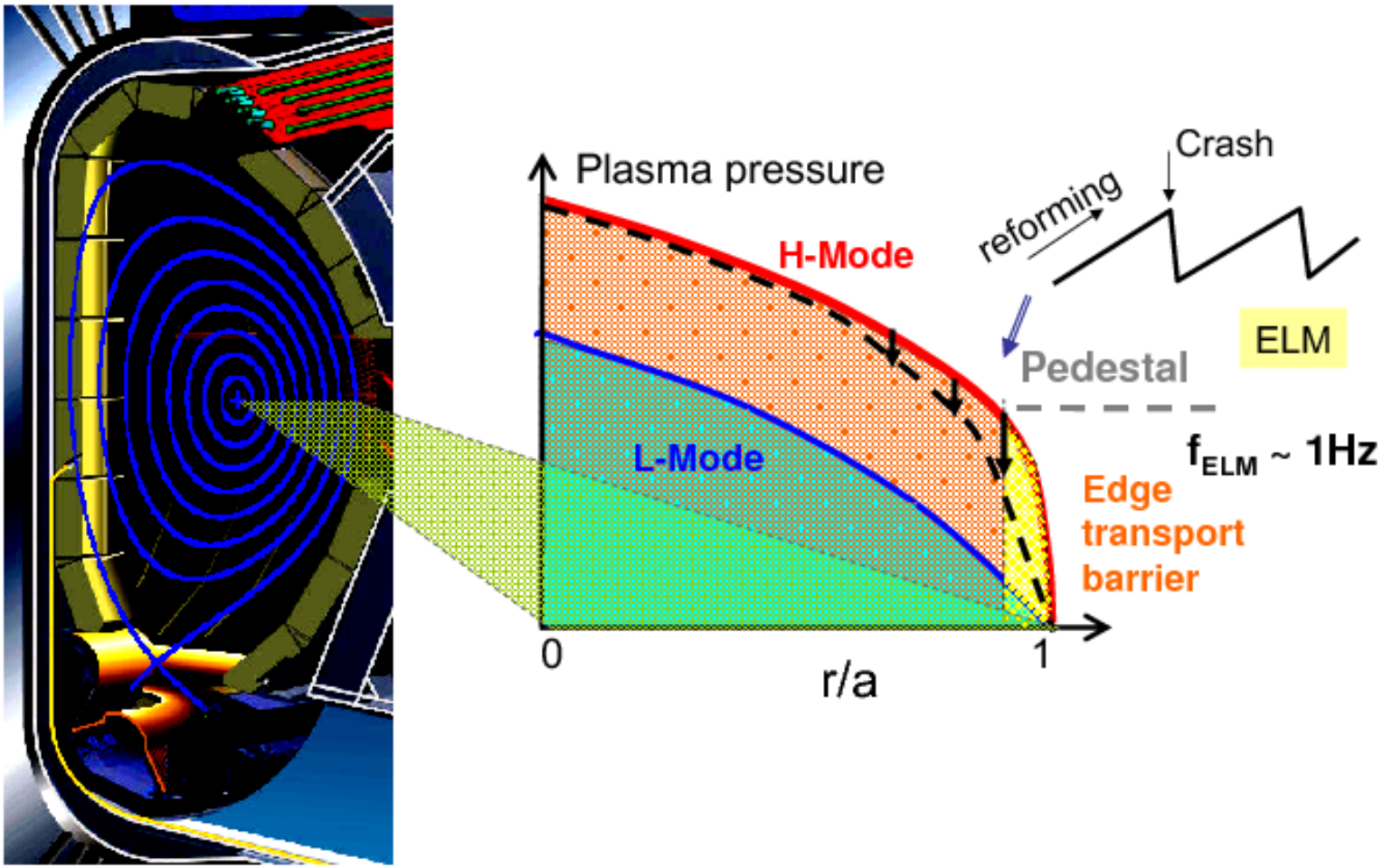
Helios, Japan

Outline

- JOREK code
  - Context: Physics, Bottlenecks, Collaborations
  - Non Regression Testing
  - Parallel performance
  - Perspectives



JOREK motivation: ELMs



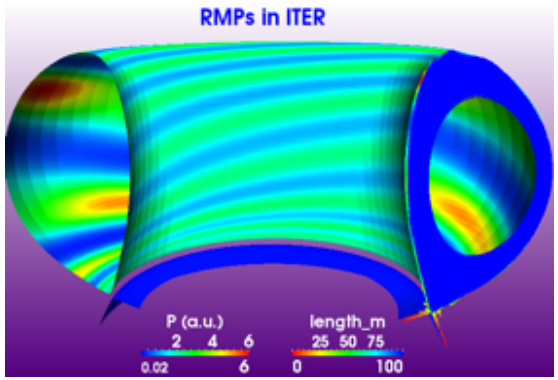
Extracted from [Liang Yunfeng, 2010]

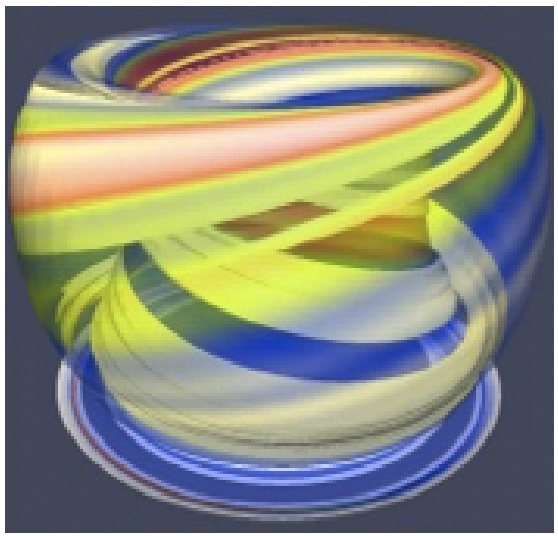
JOREK: global view

- ELM cycle & control, Disruptions
  - ELMs *[G. Dif-Pradalier, M. Bécoulet, S. Pamela]*
  - Resonant Magnetic Perturbations (RMPs) *[M. Bécoulet, F. Orain]*
  - pellets injection, vertical kicks *[G. Huijsmans, S. Futatani]*
  - VDE, β limit disruptions, density limit *[C. Reux, E. Nardon, A. Fil]*
- X-point geometry
  - cubic finite elements, flux aligned poloidal grid
  - Fourier series in toroidal direction
- Challenges to improve handled Physics
  - exact geometry** & boundary conditions**
  - non-linear MHD equ. in toroidal geometry over long time scales* ($\mu s \rightarrow s$)
  - realistic physical variables*** [resistivity, parallel conductivity, collisionality]
  - **open problems** how many $n-$modes***, background turbulence$^{?}$
- Steps of a typical computation:
  i. build the mesh
  ii. solve the Grad Shafranov equation
  iii. obtain a converged $n = 0$ flow equilibrium
  iv. add unstable $n \neq 0$ modes on top of the equilibrium

JOREK: context, issues

- Physics studies:
  Production code for *non-linear MHD*
- Mathematical issues:
  - → Mesh, robustness, convergence
  - → large cases cost (memory, computation)
- Parallel computing issues:
  - → Depend on linear solver perf. (PASTIX)
  - → Save memory space (larger cases)
- Collaborative issues:
  - → Need to modify a single code, to check results





JOREK: current bottlenecks ... and collaborations to solve them

- Physics: ITER, FOM (Netherlands), IPP (Germany), JET(UK)
- Mathematical bottlenecks: **convergence**, **large cases**
  - → INRIA CASTOR (ANR ANEMOS 2012-2015): other finite elem. (robustness/accuracy)
  - → IPP Garching: preconditioner, time scheme (convergence)
  - → IRFM (ANR ANEMOS - Postdoc A. Ratnani): isogeometric analysis (reduced costs)
- Parallel computing bottlenecks: **large cases**
  - → INRIA HIEPACS + IRFM (ANR ANEMOS): coupling with PASTIX , save memory space
- Collaboration/operational issues: **healthy code**
  - → Healthy code on SVN repository, maintenability

Interdisciplinary Context: Production & development

- Jorek activities:
  1. production runs on supercomputers to investigate physics
  2. physical features are added gradually
  3. numerics & parallelism are improved gradually
  4. a SVN repository hosts code sources, **shared by all actors**
- **Need for a set of reference cases** (ANR ANEMOS):
  1. share between collaborators a set of well-known cases
  2. obtain reproductibility of numerical experiments
  3. have a set of cases in order to test numerical improvements
  4. check code modifications before SVN commit

NRT: comparing execution trace
Reproducibility problem

- A JOREK scenario is not exactly **repeatable**
  - Starting point of linear phase is dominated by noise
    - close to equilibrium, noise is amplified by MHD instabilities
    - linear solver is limited in term of accuracy
  - Threads scheduling (OpenMP) alters less significant digits
  - Global summations with MPI (addition is not commutative)
  - GMRES solver (iterative + threshold $\rightarrow$ numerical noise)
- Goal: discriminate **acceptable differences** versus **bugs**
  - The metric excludes the beginning of simulation
  - Compare growth rates of Fourier modes during linear phase
  - The metric excludes non-linear saturation (noise amplified)

NRT: comparing execution trace
Metric and comparison

- Method (given one simulation, one reference directory)
  - Extract growth rates of Fourier modes of the two simulations
  - Select only a given time steps interval (*in the linear phase*)
  - Compare kinetic and magnetic energies of these modes

    (percentage difference should be lower than a threshold)
- Tools
  - Metric is included in a script `nrt_compare.sh` (SVN repository)
  - **Reference cases are stored** on SVN repository

NRT: Benchmarking using reference cases
MPI library

- As we **store reference cases** for NRT
  → **benchmarks**, exec. time comparisons easy to perform
  → A **tool** has been developed to extract exec. time
- Example: compare MPI libraries on a given machine

```
$ ./timing_bench.sh facto out_loop5 grep1 model302/helios_?
== model302/helios_a ( bullxmpi + FUNNELED )
0 ## Elapsed time, facto : 36.4806480
== model302/helios_b ( intelmpi + FUNNELED )
0 ## Elapsed time, facto : 143.5415650
```

- **Timer comparisons** with reference cases help to better understand performance while **porting** on new systems

Performance: Code structure - implicit scheme - 3 main parts

Input : *Physics parameters*, *equilibrium*
Output : *Diagnostics*

for time step $n \geq 0$ **do**
  // Fill large sparse matrix A (resistive MHD equations)
  parallel loop on cubic Beziers elements ;
  // Build preconditioner for A
  if needed **then**
    in P MPI_COMMUNICATORS (P linear systems):
      factorization of a A-submatrix (PASTIX);
  // Preconditioned iterative GMRES:
  while not converged **do**
    in P MPI_COMMUNICATORS:
      apply preconditioner (direct solve - PASTIX) on a vector;
    multiply A by a vector;

Performance: Hybrid MPI + Threads

Typically one **MPI process** per node (with **threads** inside)

1. Fill the sparse matrix A - Assembly step
   → MPI + OpenMP: parallel loop over the elements
2. Build the preconditioner (*once a while*):
   Factorisation of P submatrices of A:
   → MPI: parallel loop over P communicators
   → MPI+ Posix threads: parallelization inside PASTIX lib.
3. Preconditioned iterative GMRES ($A\,x = b$)
   Direct solve on P submatrices of A
   → MPI: parallel loop over P communicators
   → MPI+ Posix threads: parallelization inside PASTIX lib.
   Multiply distributed matrix A by a vector (MPI + OpenMP)

Performance: Strong scaling with MPI_THREAD_MULTIPLE - Marseille cluster

- Reference simulation - small case (model 302) using MPI_THREAD_MULTIPLE mode, Intel Westmere-EP nodes:

Nb cores	**24**	**48**	**96**	**192**
Nb nodes	2	4	8	16
Steps (time in sec.)				
construct_matrix	6.9	3.8	2.0	1.3
factorisation	33	22	16	12
gmres/solve	1.9	1.6	0.8	0.7
iteration time	48	32	22	18
rel. efficiency	100%	75%	55%	33%

Table : one iteration **with Facto.**

Nb cores	**24**	**48**	**96**	**192**
Nb nodes	2	4	8	16
Steps (time in sec.)				
construct_matrix	6.9	3.8	2.0	1.3
factorisation	0.	0.	0.	0.
gmres/solve	5.6	3.9	2.4	1.3
iteration time	12	7	4.5	2.6
rel. efficiency	100%	86%	67%	58%

Table : one it. - **no Facto.**

- Globally the JOREK code scales from 24 to 96 cores
- Relative efficiency (whole code) $\approx$ 60% at 96 cores

Performance: Strong scaling with MPI_THREAD_FUNNELED - Marseille cluster

- Reference simulation (model 302) - small case using MPI_THREAD_FUNNELED mode:

Nb cores	**24**	**48**	**96**	**192**
Nb nodes	2	4	8	16
Steps (time in sec.)				
construct_matrix	6.9	3.8	2.2	1.2
factorisation	35	22	18	16
gmres/solve	2.4	1.8	2.4	2.1
iteration time	49	32	27	24
rel. efficiency	100%	77%	38%	26%

Table : one iteration **with Facto.**

Nb cores	**24**	**48**	**96**	**192**
Nb nodes	2	4	8	16
Steps (time in sec.)				
construct_matrix	6.9	3.8	2.2	1.2
factorisation	0.	0.	0.	0.
gmres/solve	5.8	**4.1**	**5.5**	**5.5**
iteration time	12	**8.5**	**8.5**	**8**
rel. efficiency	100%	71%	35%	19%

Table : one it. - **no Facto.**

- Relative efficiency$\approx$ 36% at 96 cores (vs 60% previously)
- MPI_THREAD_FUNNELED in direct solver $\Rightarrow$ scalability issues

Performance: Strong scaling with MPI_THREAD_FUNNELED - Helios

- Reference simulation (model 302) - big case using MPI_THREAD_FUNNELED mode:

Nb cores	**128**	**256**	**512**	**1024**
Nb nodes	8	16	32	64
Steps				
construct_matrix	48	29	15	8.5
factorisation	0.	0.	0.	0.
gmres/solve	17	11	**12**	**13**
iteration time	65	41	**29**	**25**
rel. efficiency	100%	80%	**56%**	**32%**

Table : one iteration - no Facto.

- Good result: 60% rel. efficiency (whole code) at 512 cores
- But MPI_THREAD_MULTIPLE may help → other MPI librairies ...

Performance: **Memory** and **communication** bottlenecks

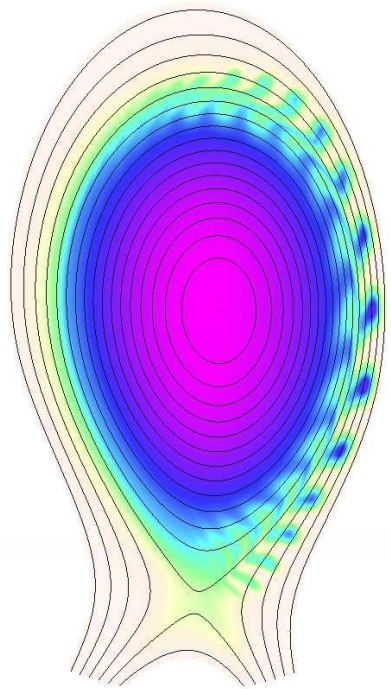
- **Goal 1:** accessing finer resolution (memory exhausted)
  → JOREK Memory tracing tool
  A **module** has been made to trace each MPI process
- **Goal 2:** accessing finer resolution (memory exhausted)
  → Improve **memory** consumption
  - reduce memory overheads in **matrix build-up** (JOREK)
  - better **distribute** memory costs among nodes (PASTIX)
- **Goal 3:** reduce comm./memory overheads
  → Use **distributed** interface of PASTIX (named MURGE)
  - reduce several **comm. overheads** (matrix redistribution)
  - avoid several **memory overheads** (matrix centralization)
    → collaboration with PASTIX people
    Xavier Lacoste (ANEMOS Phd student), Pierre Ramet

Perspectives: JOREK HPC developments

- Keep going with PASTIX people (INRIA)
  & current collaborations (INRIA, IPP)

- Challenging project with HLST
  (*High Level Support Team: HPC support to scientists from all EFDA Associates*)
  → Adapt JOREK for new arch.: Intel MIC

- Large scale initiative HPC - C2S@Exa
  → possible targets in JOREK: Mesh build, Preconditioner, Software engineering



Single fluid RMHD equations $[\rho, T, \mathbf{v}_\perp, v_\parallel, \psi]$ & boundary conditions, as solved in JOREK

1—density: $\frac{\partial}{\partial t}\rho = -\nabla\cdot(\rho\mathbf{v}) + \nabla\cdot(D_\perp\nabla_\perp\rho) + S_\rho \quad ; \quad \mathbf{v} = -R\nabla\phi(t)\times\mathbf{e}_\varphi + v_\parallel(t)\mathbf{B} + \mathbf{v}_\star$

2—temperature: $\rho\frac{\partial}{\partial t}T = -\rho\mathbf{v}\cdot\nabla T - (\gamma-1)\rho T\nabla\cdot\mathbf{v} + \nabla\cdot(\kappa_\perp\nabla_\perp T + \kappa_\parallel\nabla_\parallel T) + S_T$

3—perp. and parallel momentum:

$$\mathbf{e}_\varphi\cdot\nabla\times\left(\rho\frac{\partial}{\partial t}\mathbf{v} = -\rho(\mathbf{v}\cdot\nabla)\mathbf{v} - \nabla(\rho T) + \mathbf{J}\times\mathbf{B} + \mu\Delta\mathbf{v} - \nabla\cdot\Pi^{neo} + S_v\right)$$

$$\mathbf{B}\cdot\left(\rho\frac{\partial}{\partial t}\mathbf{v} = -\rho(\mathbf{v}\cdot\nabla)\mathbf{v} - \nabla(\rho T) + \mathbf{J}\times\mathbf{B} + \mu\Delta\mathbf{v} - \nabla\cdot\Pi^{neo} + S_v\right)$$

5—induction: $\frac{\partial}{\partial t}\mathbf{A} = -\eta(T)\mathbf{J} - \frac{m}{\rho e}\nabla_\parallel(\rho T) + \mathbf{v}\times\mathbf{B} - F_0\nabla\phi$

6—B field & closure:

$$\mathbf{B} = \frac{F_0}{R}\mathbf{e}_\varphi + \frac{\nabla\psi(t)}{R}\times\mathbf{e}_\varphi \quad ; \quad \kappa_\parallel(T) = \kappa_{\parallel,0}\left(T/T_0\right)^{-5/2} \quad ; \quad \eta(T) = \eta_0\left(T/T_0\right)^{-3/2}$$

7—boundary conditions:

- Zero perturbations on wall aligned with last flux surface
- Bohm boundary conditions on the target: $v_\parallel = c_s \quad ; \quad \kappa_\parallel\mathbf{b}\cdot\nabla T = (\gamma-1)nTc_s$