Code optimizations in the BSC Advanced Computing Hub: Implementation of matrix compression for the coupling of JOREK to the 3D realistic conducting wall structures

F. Cipolletta, N. Schwarz, M. Hoelzl, S. Ventre, N. Isernia, G. Rubinacci, A. Soba

EFTC 2023 | 2nd Oct. 2023 - 5th Oct. 2023

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

ACH-BSC within the E-TASC vision

EUROfusion – Theory and Advanced Simulation Coordination **E-TASC**

Advanced Computing Hubs
ACH - wikipage
Advanced Computing Techniques and Expertise in computer science and software engineering

Theory, Simulation, Verification and Validation
TSVV
Fundamental Research in Nuclear Fusion

Cat. 1 – High Performance Computing:

- *Barcelona Supercomputing Center* **BSC**
- *École polytechnique fédérale de Lausanne* **EPFL**
- *Max Planck Garching* **MPG**

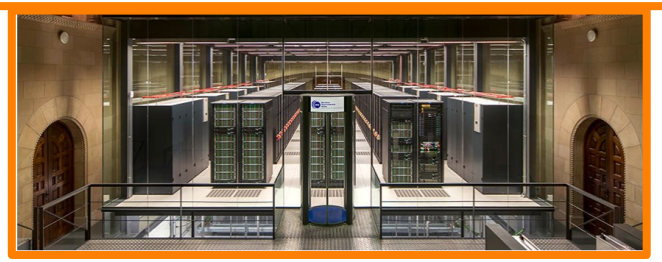
Cat. 2 – Integrated Modeling and Control:

- *Institute of Plasma Physics and Laser Microfusion* **IPPLM**

Cat. 3 – Data Management:

- *Technical Research Institute of Finland* **VTT**

1) Fusion Group
2) Operations
3) Best Practices for Performance and Programmability



BSC – ACH

Code	Description	Work Required	Start
ERO 2.0	Simulation of particle transport in fusion plasma devices via Monte-Carlo method	GPU Porting	2021
SPICE	Simulations of particles in a fixed magnetic and self-consistent electric field	Implementing parallel Poisson Solver Implementing parallel E calculation	2021
STELLA	Evolves electrostatic fluctuations in general magnetic geometry via solving Gyrokinetic-Poisson equations	Optimization	2021
KNOSOS	Simulates neoclassical transports in low-collisional plasma via orbit-averaging	Optimization	2022
BIT1	Particle-In-Cell code for plasma edge modelling	GPU Porting	2022
SOLPS	Plasma boundary code package	Check OpenMP Improve compilation Consider vectorization towards use on Cray platforms	2023
XTOR-K	MHD code including full integration of kinetic particle orbits in Particle-In-Cell	GPU Porting	2023
GENE-X	Gyrokinetic continuum code based on the flux-coordinate independent approach	Implementing support for unstructured grids in arbitrary order Testing performance and reordering	2023
JOREK	Simulates the dynamics of the high-energy magnetically confined plasma	Reduce Memory consumption Improve performance	2023

Outline

1. Introduction to JOREK and how it works
2. Description of the work of the ACH for the JOREK code
3. First glimpse about interesting applications and preliminary results
4. Summary and Outlook

Outline

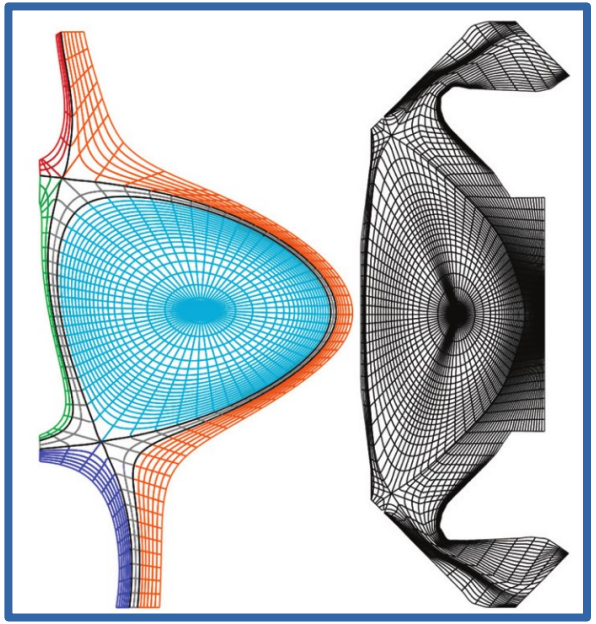
1. **Introduction to JOREK and how it works**
2. Description of the work of the ACH for the JOREK code
3. First glimpse about interesting applications and preliminary results
4. Summary and Outlook

The JOREK code

- Non-linear MHD + many extensions including kinetic & hybrid models
- Bezier finite element + toroidal Fourier expansion
- Fully implicit time-evolution
- Divertor tokamaks including X-point(s) – **(1)**
- Adopted for simulations of plasma instabilities – **(2)**
- Originally developed at CEA Cadarache:
  - Czarny and Huysmans (2008); Huysmans and Czarny (2007)
- In the present work Reduced MHD for simplicity
- Fortran 90/95
- MPI + OpenMP hybrid parallelization

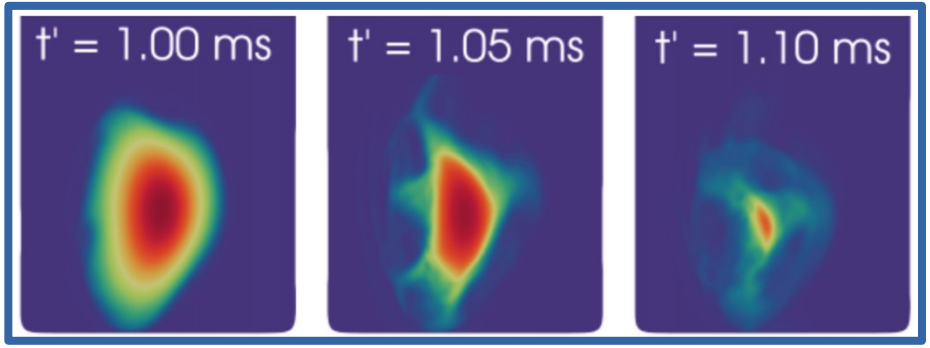
More details in the presentation M. Hoelzl (2012)

(1) Typical grids used in JOREK



See Hoelzl et al (2021)

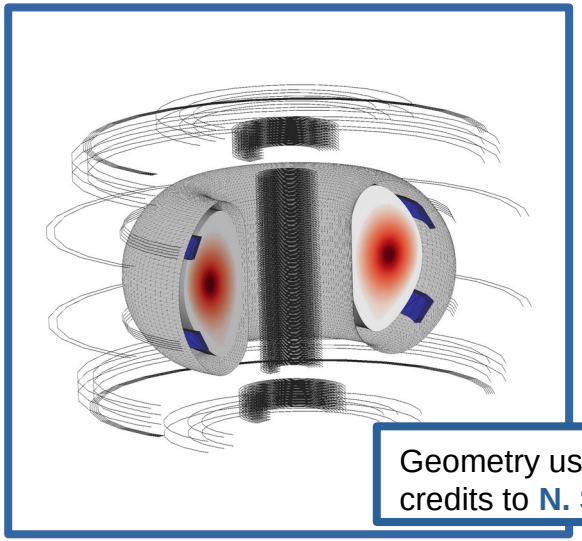
(2) Snapshots of evolution of the pressure in the $\phi = 0$ plane in a 3D Vertical Displacement Event (**VDE**)



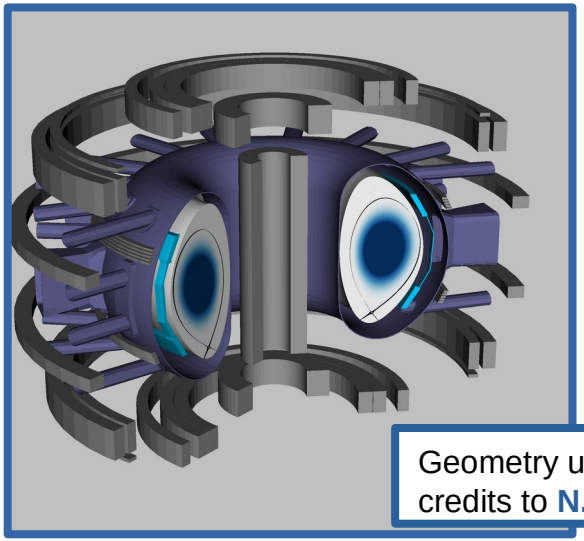
STARWALL and CARIDDI

The free-boundary and resistive wall extension of JOREK are obtained through coupling to external codes, which provides response matrices, in particular:

- **STARWALL** → 3D thin resistive walls response – see Merkel, Strumberger (2015)
- **CARIDDI** → 3D volumetric resistive walls – see Isernia et al (2022) + in preparation



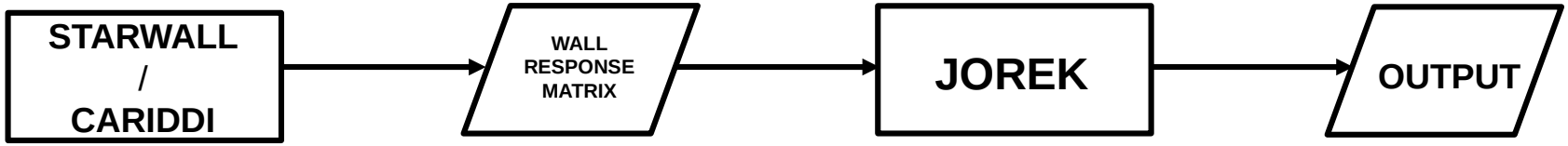
Geometry used in **STARWALL** credits to **N. Schwarz**



Geometry used in **CARIDDI** credits to **N. Schwarz**

The JOREK code – STARWALL/CARIDDI coupling

Starting point for this work



What we do

BSC-ACH contribution to JOREK

Goal

Enable modeling capabilities of realistic and accurate 3D wall structures within MHD simulations of plasma instabilities inside a Tokamak

Objective related to JOREK

Reduce Memory required by response matrices and **Improve Performance** → Apply factorization and compression techniques to matrices provided by STARWALL or CARIDDI

Challenges

TASK-RELATED – connected to matrix dimensions (could become big for realistic geometries)

CODE-RELATED – complex collection of several Fortran files with ID computed externally

PROJECT-RELATED – many parallel developments → "*code-orthogonality*" is mandatory

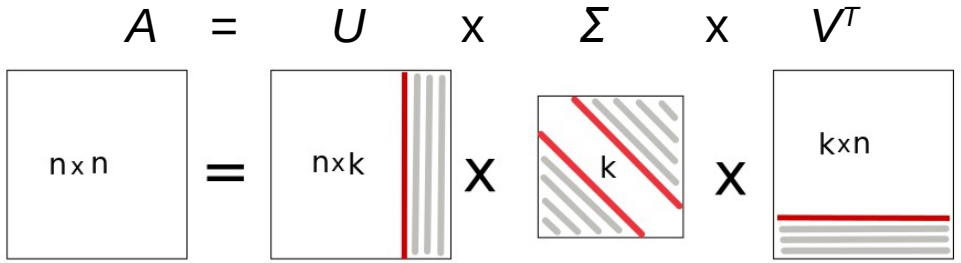
Outline

1. Introduction to JOREK and how it works
2. **Description of the work of the ACH for the JOREK code**
3. First glimpse about interesting applications and preliminary results
4. Summary and Outlook

Techniques – Singular Value Decomposition (SVD)

Given a dense matrix, with one accurate SVD, we could write

A dense
U, V orthogonal
Σ diagonal



Required Dimensions for the factorized representation:

rank(Σ) = k, storing UΣ and V^T ==> you need *(2nk)* instead of n^2 (you **save** when **k < n/2**)

Powerful Features in view of applications:

1. The SVD can be always performed
2. An SVD with singular values in descending order always exists
3. The SVD is an optimal approximation with respect to the residual

Implementation:

The Scalable Linear Algebra PACKage (ScaLAPACK) parallelized library offers the routine `pdgesvd` to compute the SVD of a given matrix (https://netlib.org/scalapack/)

How the compression code works

- Reads typical STARWALL/CARIDDI response file
- At compilation, the user selects:
  - Which matrices to compress
  - What fraction of singular values of the SVD to retain
- ScaLAPACK returns decreasing order Singular Values (SVs)
- **Compressing** = *Retaining the biggest SVs*
- Runs are typically really fast, since they rely on the optimized ScaLAPACK library
- The compression needs to be run only one time per JOREK simulation (*separate module*)
- An output file is printed with all the required information (*factorization in $U\Sigma$ and V^T*)
- Possibility to re-aggregate the SVD decomposition (*VALIDATION*)

Outline

1. Introduction to JOREK and how it works
2. Description of the work of the ACH for the JOREK code
3. **First glimpse about interesting applications and preliminary results**
4. Summary and Outlook

A first application – VDE

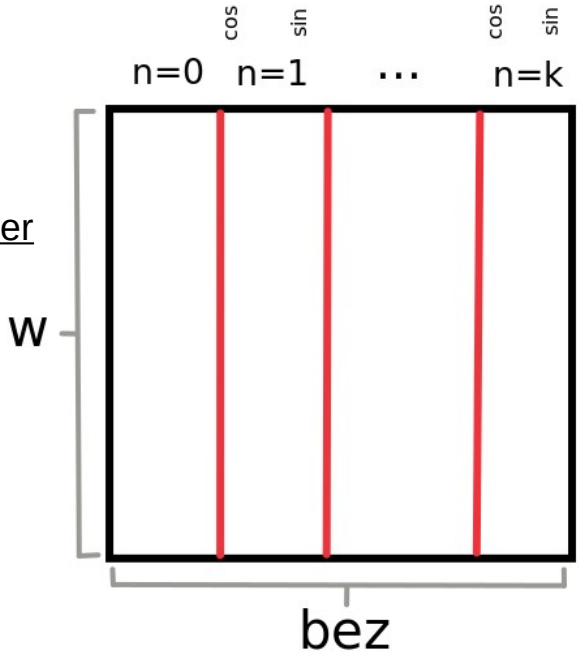
- Vertical Displacement Event (VDE) simulations are performed as follows:
  - Start in axisymmetry with only 1 toroidal harmonic, considering a perturbation in the coils currents
  - Run in axisymmetry until the magnetic axis is moved to a certain position
  - Start the full 3D simulation, up to the MHD burst leading to loss of the thermal confinement
- Consider the decomposition of 2 response matrices, adopted for algebraic operations, representing the mutual plasma-walls interactions
- Need of:
  - Produce compressed response at different rates of compression
  - Changes inside JOREK related to the algebraic operations
  - Run both the standard and modified version of JOREK for the sake of VALIDATION

Matrices Involved – Dimensionality

The chosen matrices have dimesions (*w*, *bez*) or (bez, *w*)

w = wall dof → kept constant for every order *k* of toroidal harmonics

bez = Bezier dof times Fourier harmonics → **changes** with different order *k* of toroidal harmonics



NOTE: Considering k toroidal harmonics modes number, means considering the sin and cos components for each n > 0 mode (see sketch in the case of *w* x *bez* with only 1 Bezier dof)

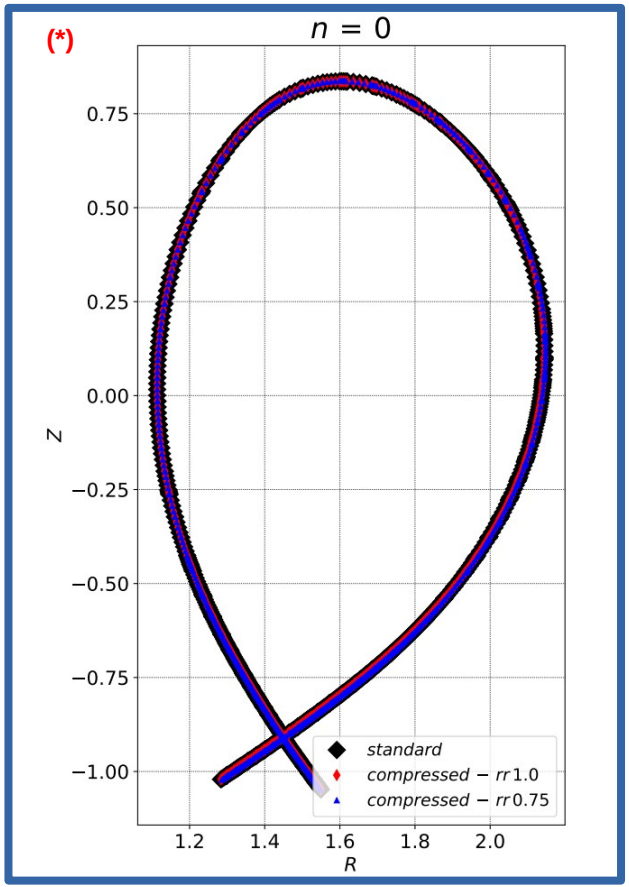
Selected test cases

Harmonics used	Bezier DoF	bez	Wall DoF	Size [GB]
n=0	180	180	20.322	0,027
n=0...4	180	1.620	20.322	0,245
n=0...9	180	3.420	20.322	0,518

- Selected retain rates of the singular values (**rr**):
  - 1.0 (uncompressed)
  - 0.75 (¾ of the singular values are retained)
  - 0.5 (½ of singular values are retained)
  - ...
- Produce Initial Data via CARIDDI
- Run tests with standard and compressed versions

Preliminary tests with n=0

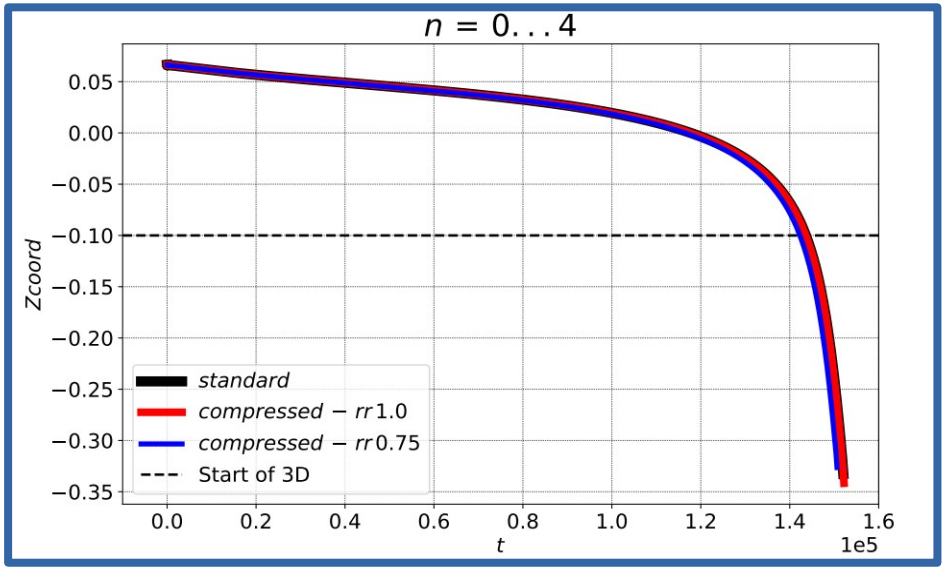
- Performance issues (rr=1.0 required much more time with respect to standard)
  - ✓ *Factorized matrix distribution across MPI tasks + OpenMP parallelization of loops*
- rr = 1.0 results matching the standard code
  - ✓ *Implementation is validated*
- The n=0 part of the matrix is very sensitive to the compression (*)
  - ✓ *increasing n is expected to improve "compressibility"*

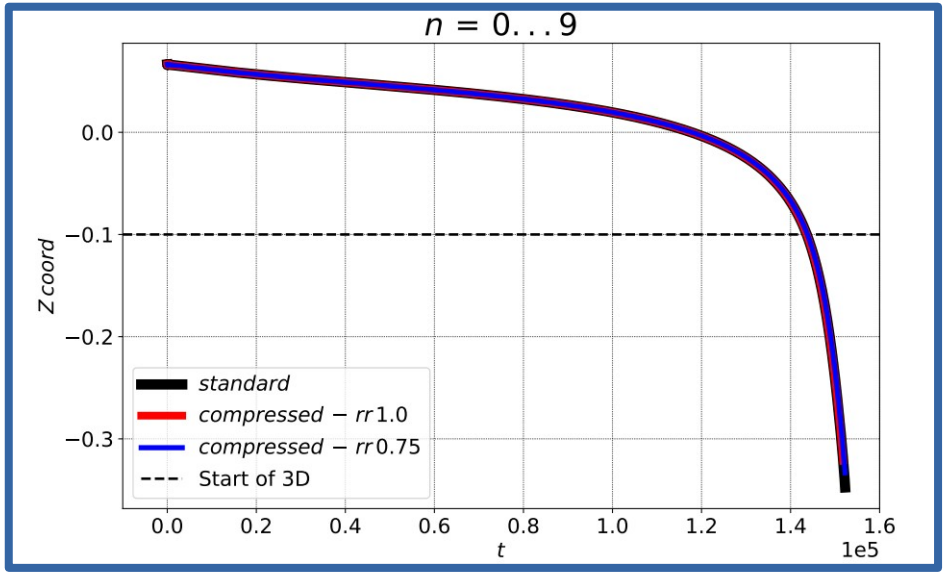


(*) Separatrix of the equilibrium configuration at different compression rates

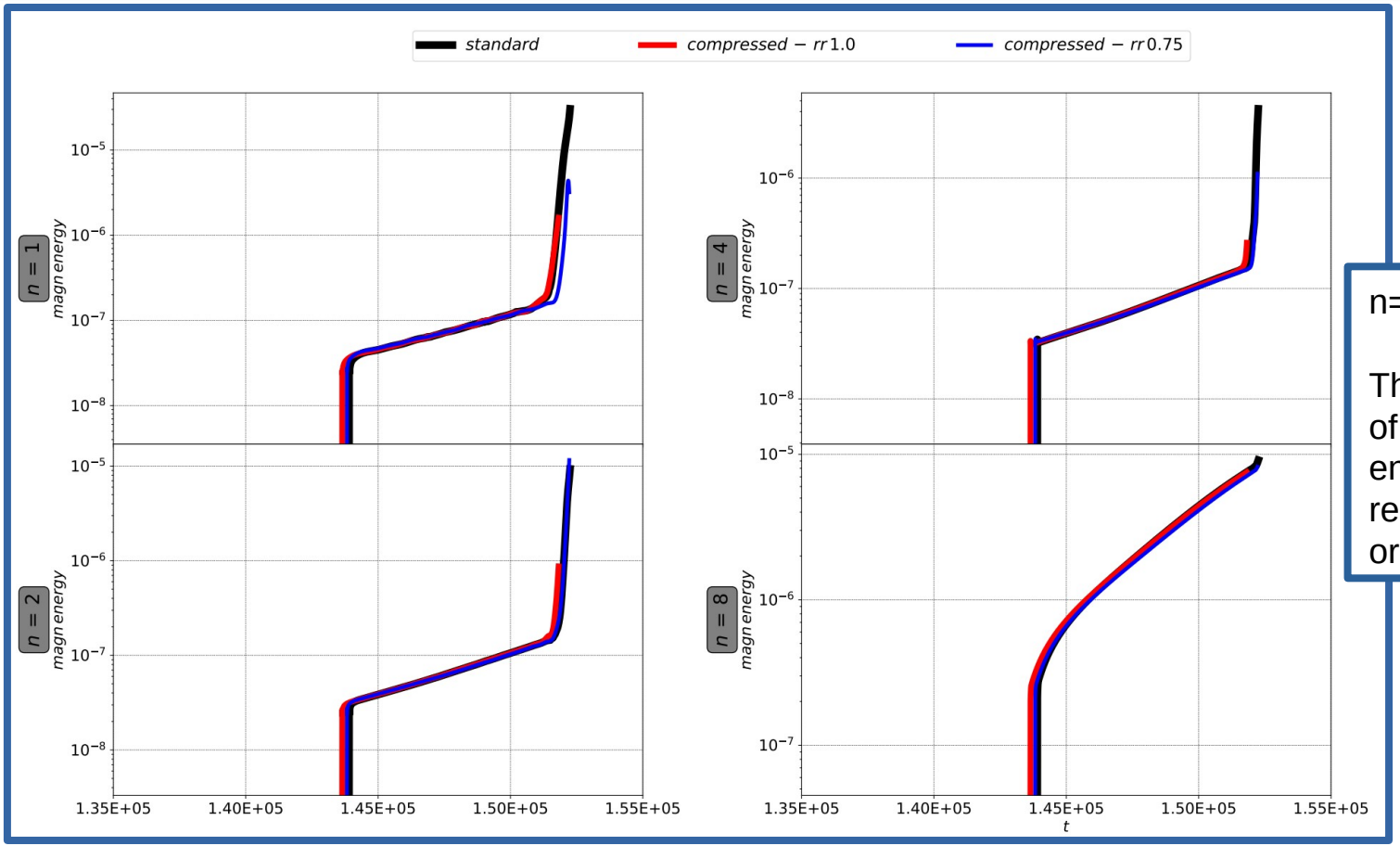
Even at low compression rate (blue color for rr 0.75) the equilibrium starts to be slightly shifted with respect to the standard and uncompressed cases

Increasing n - Evolution of the Magnetic Axis position





Evolution of the Magnetic energies – the 3D phase



n=0..9

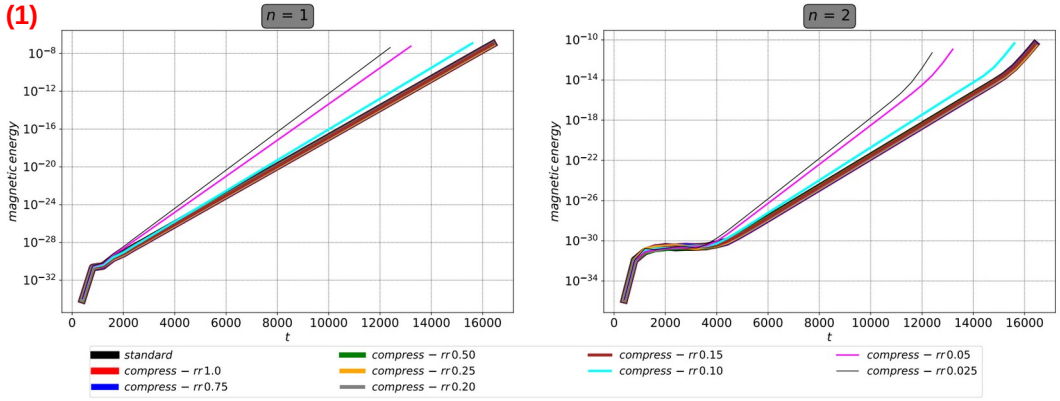
The biggest errors of the magnetic energy seems to reside in the lowest order mode

Tearing Mode Instability

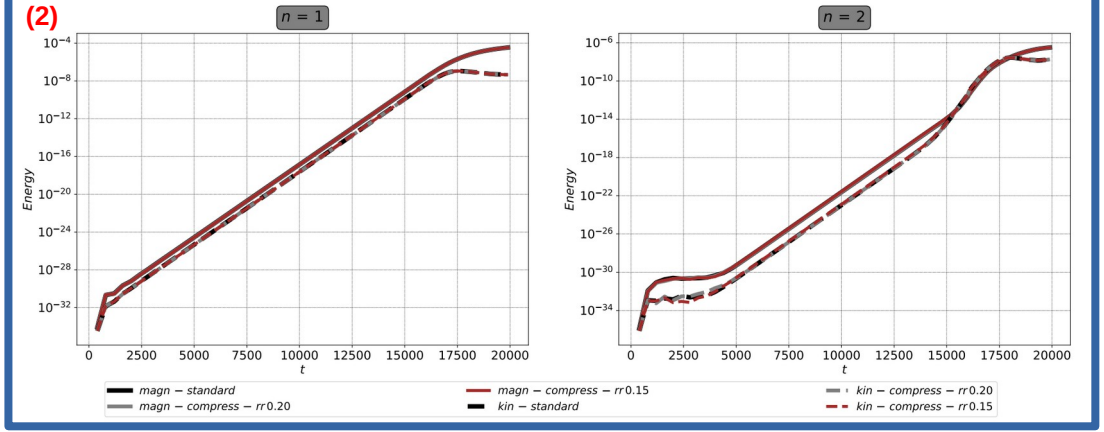
- Test running directly with n=0...2 toroidal harmonics
- The n=0 part is being kept fixed
- Several compression level from rr 1.0 down to rr 0.025

- **(1)** shows matching results for the magnetic energy down to rr 0.15

(1)



(2)



- **(2)** shows close evolution of magnetic and kinetic energy up to saturation phase

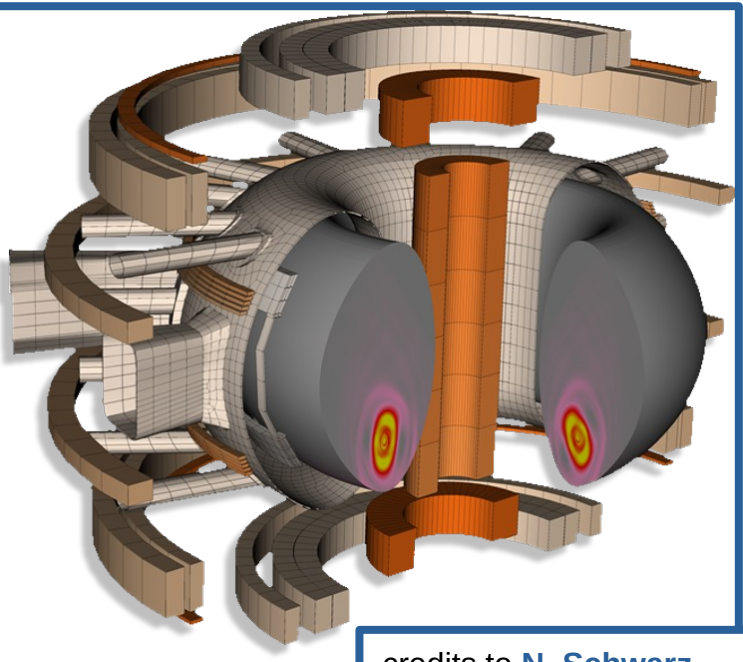
Test case provided by **N. Schwarz**

Outline

1. Introduction to JOREK and how it works
2. Description of the work of the ACH for the JOREK code
3. First glimpse about interesting applications and preliminary results
4. **Summary and Outlook**

Summary and Outlook

- JOREK solves non-linear 3D MHD in fusion devices
- Compressing the large matrices representing the electromagnetic plasma-wall interaction
- SVD based method implemented and verified
- First tests show good compressibility and results for particular production cases (Tearing Mode) where the n=0 part of matrices is kept fixed
- For other cases where n=0 part is not fixed (VDE), we will treat the n=0 and n>0 parts separately



credits to **N. Schwarz**

Thank you for your attention

Questions?

Backup Slides

How to perform SVD - The ScaLAPACK library

- The requirement is to supply **2D block cyclicly distributed** matrices:

  i) Definition of a **blacs grid of processes**

  ii) Definition of a proper **block size**

  iii) Definition of **matrices' descriptors** to distribute them across processes

- Actual implementation:

  a) All in a separate module called by a separate program contained in the main repo

  b) Produce SVD, with internally defined distributed matrices

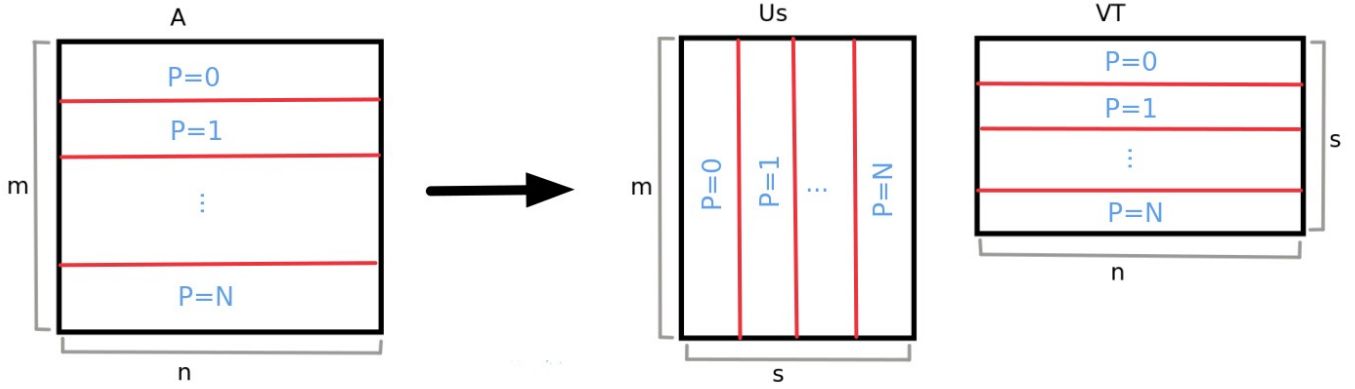
Matrix distribution inside JOREK

Given *N* MPI tasks:

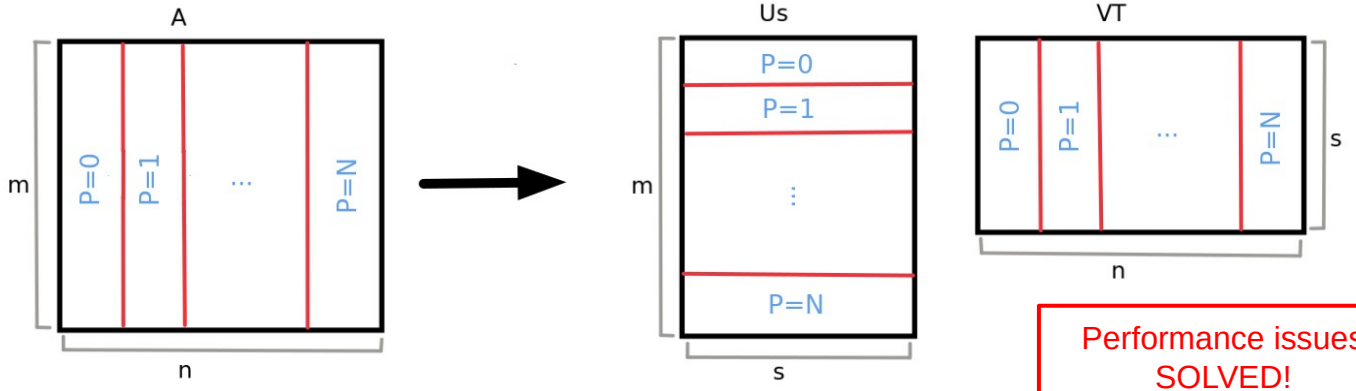
Standard

Factorized

Row-wise



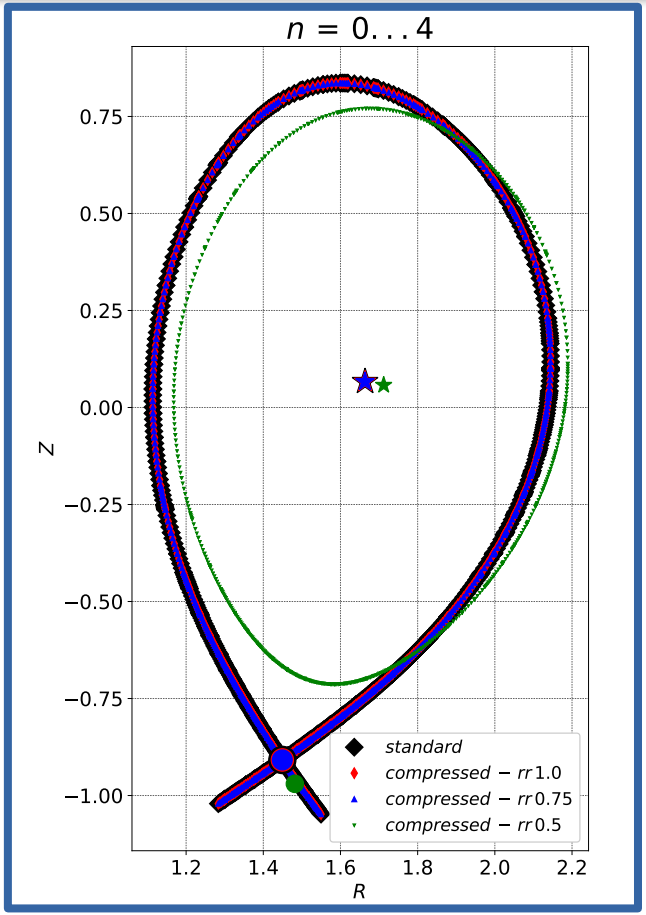
Column-wise

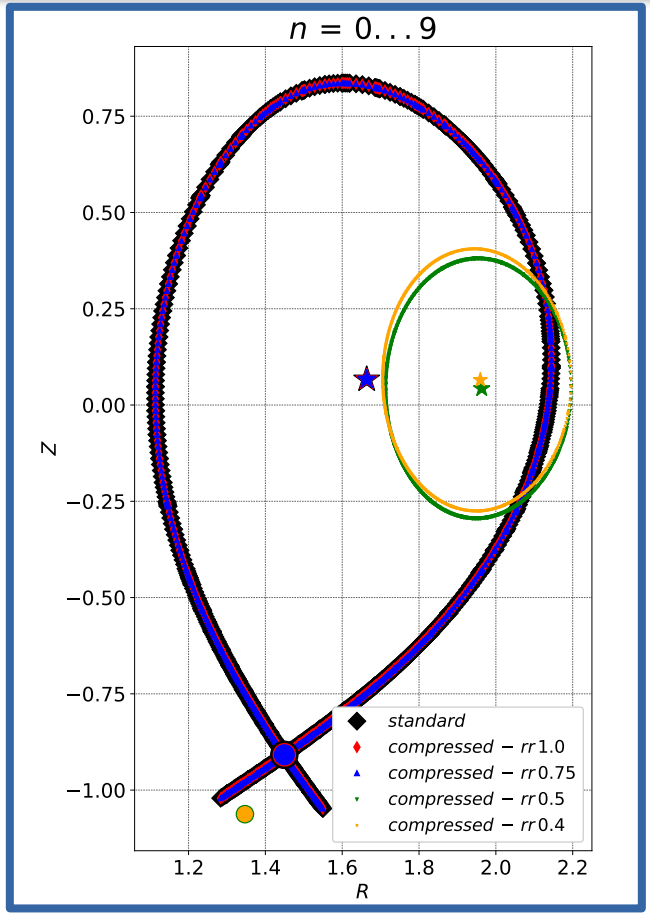


Each MPI task then, splits the work across the available OpenMP threads

Performance issues SOLVED!

The Equilibrium of VDE when applying compression





Plots showing equilibrium configurations in the meridional plane in axis symmetry

Stars locate the magnetic axis

Filled circles locate the X-point

Different symbols show the separatrix

Takeaway

- We can not go to rr 0.5 or below
- We need some other improvements

Toroidal harmonics representation and compression

Given one order k and a matrix A, the toroidal harmonics representation of A is a block matrix as on the side

Since we noticed that the n=0 part is not really compressible, we may want to keep the gray block unaltered and compress only the other part of the matrix