EUROfusion

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

4th Fusion HPC Workshop | 30 Nov. 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 within the E-TASC

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* **CIEMAT-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

Outline

1. Brief introduction to JOREK and the response matrices
2. Matrix Compression
   a. Chosen method
   b. How it is implemented
3. How the method performs within applications
   a. Vertical Displacement Event (VDE)
   b. Tearing Mode (TM) instability
   c. Issues and Ideas
4. Next steps

Outline

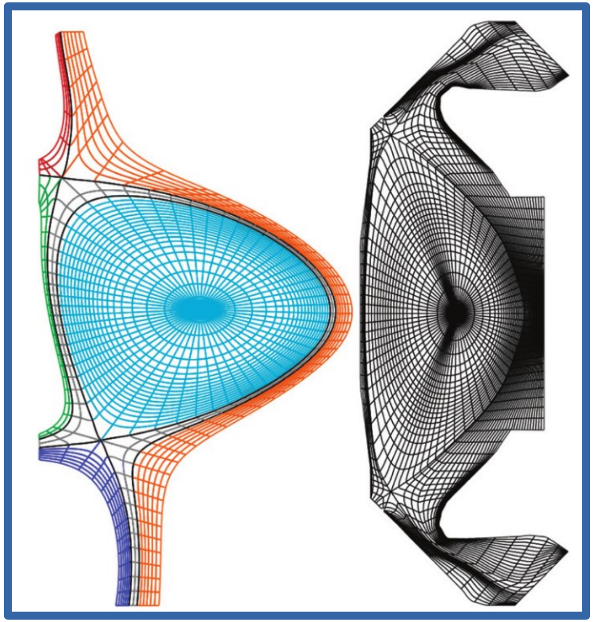
1. **Brief introduction to JOREK and the response matrices**
2. Matrix Compression
    a. Chosen method
    b. How it is implemented
3. How the method performs within applications
    a. Vertical Displacement Event (VDE)
    b. Tearing Mode (TM) instability
    c. Issues and Ideas
4. Next steps

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

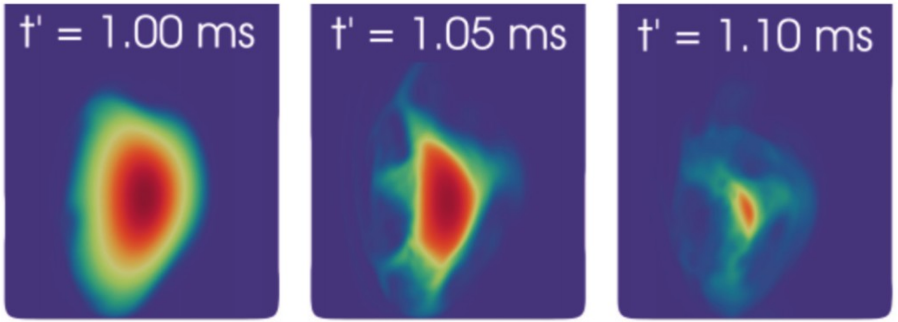
More details in the presentation M. Hoelzl (2012)

(1) Typical grids used in JOREK



See Hoelzl et al (2021)

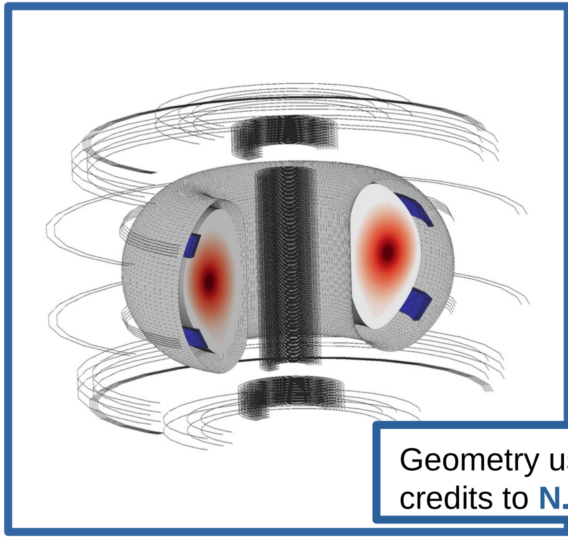
(2) Snapshots of evolution of the pressure in the $\phi = 0$ plane in a 3D Vertical Displacement Event (**VDE**)



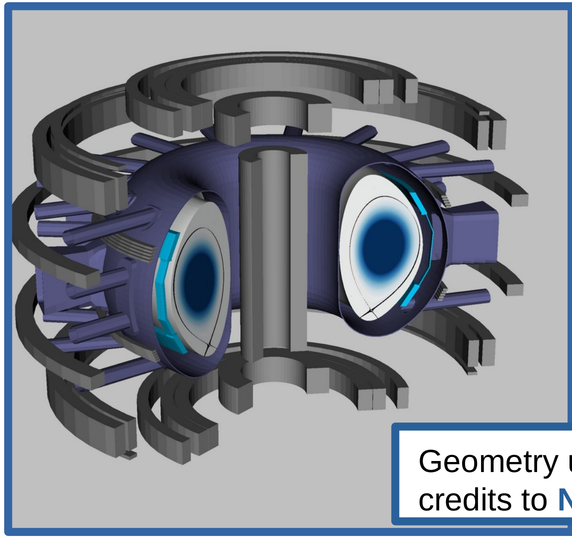
STARWALL and CARIDDI

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**

Work requested to BSC-ACH

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. Brief introduction to JOREK and the response matrices
2. **Matrix Compression**
   1. **Chosen method**
   2. How it is implemented
3. How the method performs within applications
   1. Vertical Displacement Event (VDE)
   2. Tearing Mode (TM) instability
   3. Issues and Ideas
4. Next steps

Singular Value Decomposition (SVD)

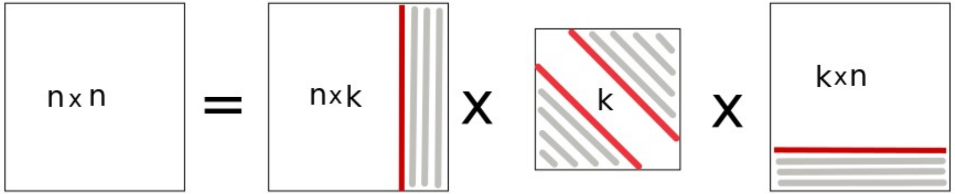
Given a dense matrix, with one accurate SVD, we could write

$$A \;=\; (\,U \quad \times \quad \Sigma\,) \quad \times \quad V^T$$

A dense

U, V orthogonal

Σ diagonal



Required Dimensions for the factorized representation:

$rank(\Sigma) = k$, storing $(U\Sigma)$ and V^T => $(2nk)$ elements instead of n^2

A rectangular (m,n) => $(mk+nk)$ elements instead of (mn) (size scales linearly with k)

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 computed via Frobenius norm

Implementation:

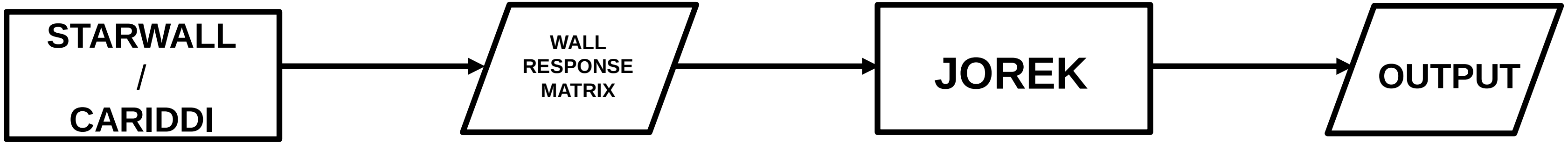
The Scalable Linear Algebra PACKage (ScaLAPACK) parallelized library offers the routine `pdgesvd` to compute the SVD of a given matrix (https://netlib.org/scalapack/)

Outline

1. Brief introduction to JOREK and the response matrices
2. **Matrix Compression**
   a. Chosen method
   b. **How it is implemented**
3. How the method performs within applications
   a. Vertical Displacement Event (VDE)
   b. Tearing Mode (TM) instability
   c. Issues and Ideas
4. Next steps

The JOREK code – STARWALL/CARIDDI coupling

Starting point for this work



STARWALL / CARIDDI

WALL RESPONSE MATRIX

JOREK

OUTPUT

What we do

STARWALL / CARIDDI

WALL RESPONSE MATRIX

COMPRESSION TOOL

JOREK

OUTPUT

How the COMPRESSION TOOL works

- At compilation, the user selects:
  - Which matrices to compress
  - What fraction of singular values of the SVD to retain
- Reads STARWALL/CARIDDI response file
- ScaLAPACK performs SVD and returns decreasing order Singular Values (SVs)
- **Compressing** = *Retaining the biggest SVs*
- 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*)
- Runs are typically fast (minutes), since they rely on the optimized ScaLAPACK library
- The compression needs to be run only one time per JOREK simulation (*separate module*)

Notes on the current implementation in JOREK

ADAPTATION of JOREK

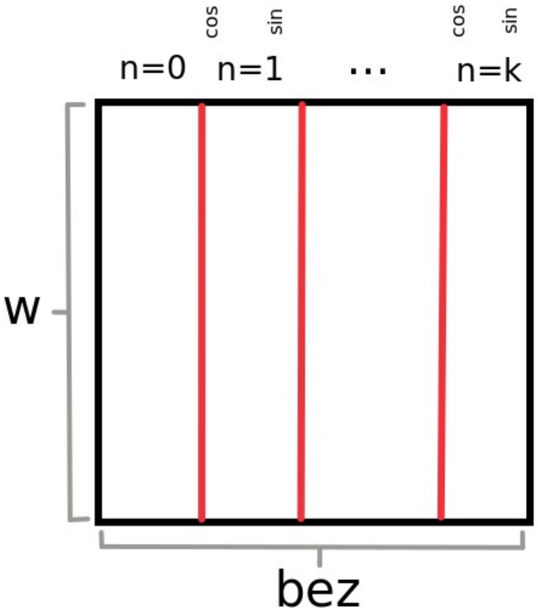
- Need to choose one or two matrices as a starting point
- Need to adapt JOREK to treat the compressed and factorized matrices
- Already implemented hybrid MPI/OpenMP parallelization for additional operation

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

w = wall dof → <u>kept constant for every order *k* of toroidal harmonics</u>

bez = Bezier dof times Fourier harmonics → **changes** <u>with different *k*</u>

NOTE: the rank of the matrix is always ≤ *min(w,bez)*



Outline

1. Brief introduction to JOREK and the response matrices
2. Matrix Compression
   a. Chosen method
   b. How it is implemented
3. **How the method performs within applications**
   a. **Vertical Displacement Event (VDE)**
   b. Tearing Mode (TM) instability
   c. Issues and Ideas
4. Next steps

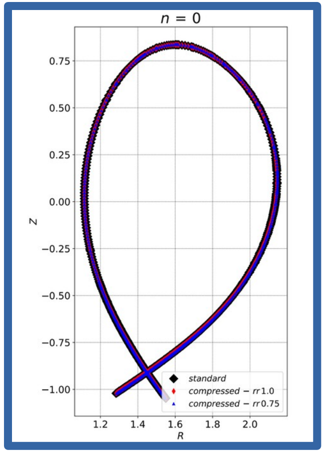
Memory compression for VDE test case

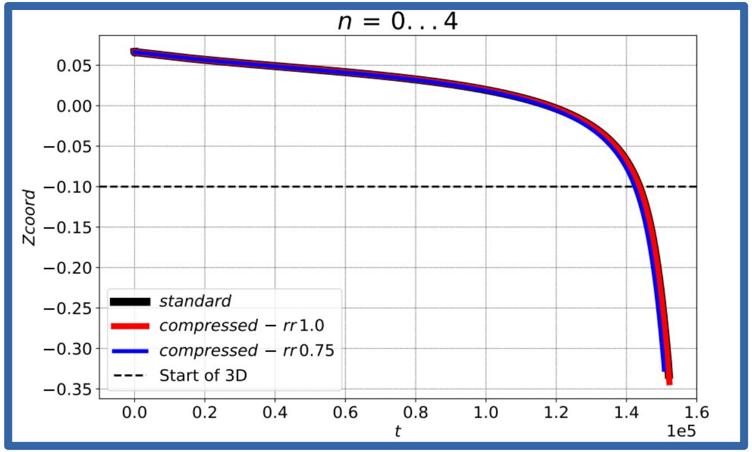
$\mathbf{r_r}$ = rate of retained singular values → smaller = more compression

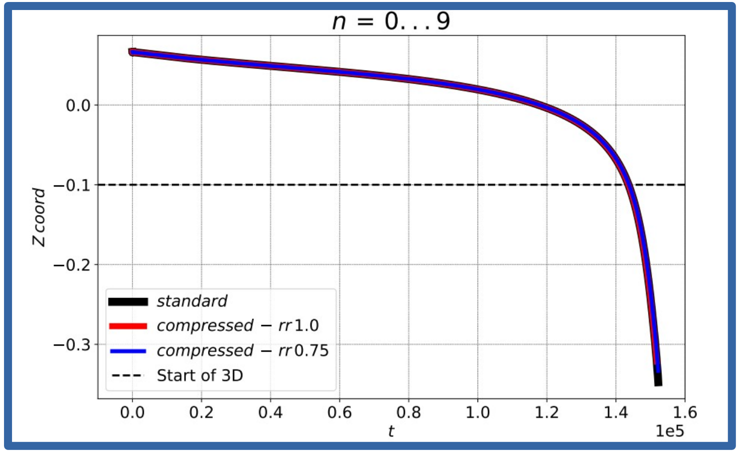
Convergence + Accurate Results
Inaccurate Results

- Selected rates of $\mathbf{r_r}$:
  - 1.0 (factorized but not compressed)
  - 0.75 (¾ of the singular values are retained)
  - 0.5 (½ of singular values are retained)
  - ...
- Produce Initial Data via CARIDDI
- Tests with standard and compressed matrices

$\mathbf{r_r}$	**Size n=0 [GiB]**	**Size n=9 [GiB]**	**Size n=19 [GiB]**
-	**0.0273**	**0.245**	**0.518**
1.0	**0.0275**	**0.265**	**0.605**
0.75	**0.0206**	**0.199**	**0.454**
0.5	**0.0137**	**0.132**	**0.302**
0.4	**0.0110**	**0.106**	**0.242**
...			







VDE - Recap

- Performance issues (r_r=1.0 required much more time with respect to standard)
  - ✓ *Factorized matrix distribution across MPI tasks + OpenMP parallelization of loops*
- r_r = 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 slightly improves “compressibility”*

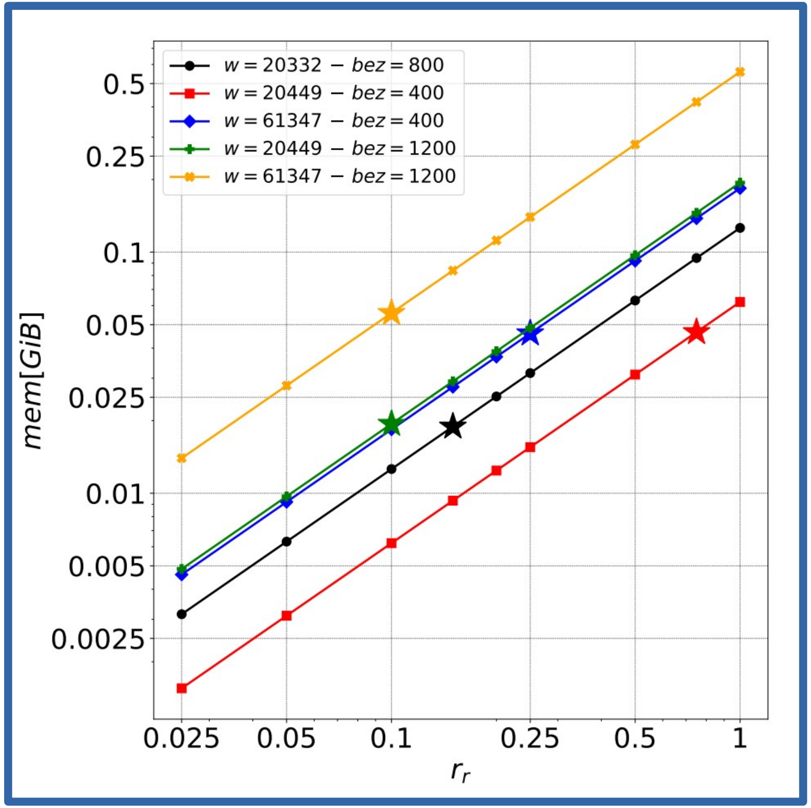
TAKEAWAY:
Implementation **optimized** and **validated**, efficiency is still **improvable**

Outline

1. Brief introduction to JOREK and the response matrices
2. Matrix Compression
    a. Chosen method
    b. How it is implemented
3. **How the method performs within applications**
    a. Vertical Displacement Event (VDE)
    b. **Tearing Mode (TM) instability**
    c. Issues and Ideas
4. Next steps

Memory compression for TM test cases

r_r = rate of retained singular values → smaller = more compression



Here, <u>the matrices do not contain the n=0 part</u>

NOTE: ★ indicates the smallest r_r providing accurate results, i.e. <u>the more it is towards the left the more the compression is efficient</u>

- First case if the **black** curve (shows <u>good compression</u>)
- Started a sistematic study on resolution:
- **Red** curve taken as basis
  - Tripling *w* in the **blue** curve
  - Tripling *bez* in the **green** curve
  - Tripling both *w* and *bez* in the **orange** curve

TAKEAWAY:
Compression is more efficient on large matrices, in particular for <u>larger resolutions on the plasma side</u>

Outline

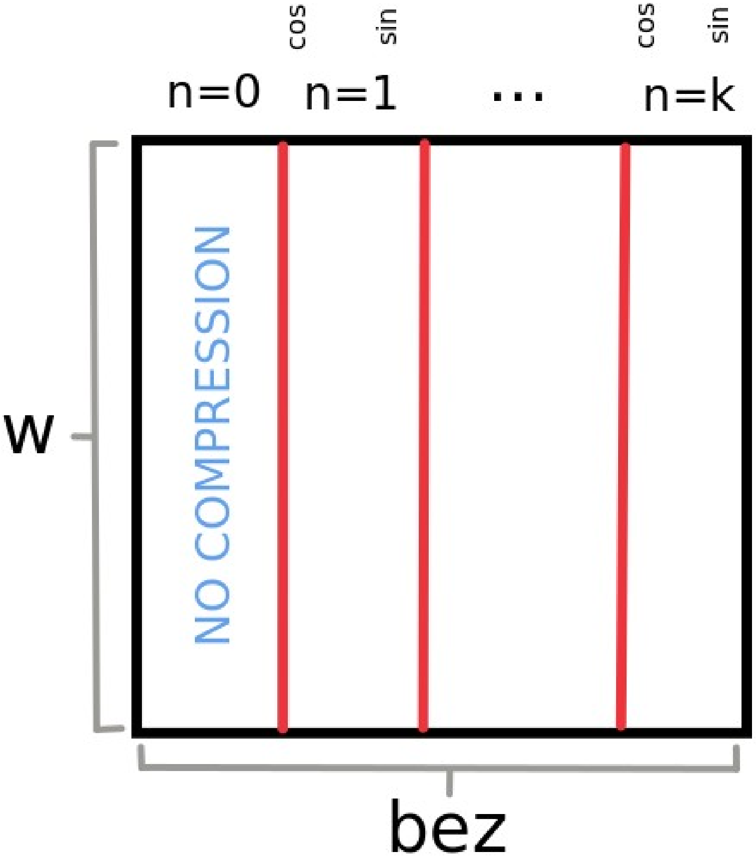
1. Brief introduction to JOREK and the response matrices
2. Matrix Compression
   a. Chosen method
   b. How it is implemented
3. **How the method performs within applications**
   a. Vertical Displacement Event (VDE)
   b. Tearing Mode (TM) instability
   c. **Issues and Ideas**
4. Next steps

Toroidal harmonics representation and compression

Issues shown for the previous tests of **VDE** are probably due to the fact that the n=0 part is sensitive to compression

Given a response matrix A, it is composed of blocks as the sketch on the side, for each node element at the boundary of the plasma

The current development is testing the possibility to leave the n=0 parts of the matrices uncompressed



Current Limitations

Limitations of efficiency of compression in the **TM** tests with respect to *w* is given by the definition of SVD and the depicted geometry

From the dimensions point of view:

$$A(m,n) = U(m,k) \times \Sigma(k,k) \times V^T(k,n)$$

With

$$k \leq \min(m,n)$$

1. *min(m,n)* “rules” the amount of compression
2. In JOREK cases studied *bez* < *w*

Need to develop of way of selecting “worthy” regions of the wall to be compressed

Outline

1. Brief introduction to JOREK and the response matrices
2. Matrix Compression
   1. Chosen method
   2. How it is implemented
3. How the method performs within applications
   1. Vertical Displacement Event (VDE)
   2. Tearing Mode (TM) instability
   3. Issues and Ideas
4. **Next steps**

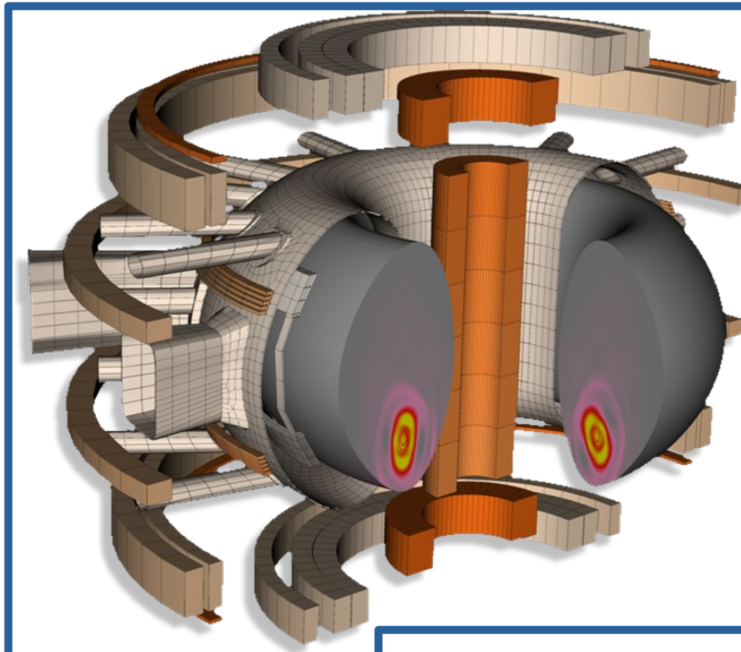
Future Plans

Summary

- First implementation of a tool for the compression of response matrices have been provided, along with its validation and assessment

Future

- Treat the n=0 and n>0 parts separately (possibly treat each mode separately)
- Assess effectivity of the new implementation via *VDE tests*
- Improve efficency in compressing in *w* “direction” (e.g. *AMG*, *H-matrices* methods or similar)
- Implement and test compression for other matrices (including **full MHD** and **halo current** couplings)



credits to **N. Schwarz**