**AC-ACH.01.CIEMAT-T001-D001**

**Actualization Report: SPICE2**

1. **Introduction**
   1. *SPICE code.*

The **SPICE** (SheathParticle In CEll) package includes two codes: SPICE2 (2D3V) and SPICE3 (3D3V) [1]. These codes are dedicated to performing simulations of particles in a fixed magnetic and self-consistent electric field and have been successfully used for the study of plasma deposition near castellated plasma facing components (PFCs).

The PIC grid brings certain limitations:

1. The size of grid cell should not be larger than Debye length.

2. There should be at least 50 particles in each cell for statistical reference that leads to high memory requirements.

3. Time step requirements: particles should not cross more than 1 cell during 1 time step and there should be at least 10 time points per 1 Larmor period.

SPICE is written in Fortran 90 and provides its output in the Matlab MAT binary format. SPICE parallelization is implemented using domain decomposition principles and message passing interface (MPI). All internal routines are parallel except for the Poisson solver. The Poisson solver is serial (taking 3% of the overall calculation time). It operates with global matrices of potential and charge density.

Currently implemented Direct Poisson solver is based on LU decomposition and employs the UMFPACK and BLAS libraries. The previous attempts to implement parallel solver using various libraries were slow and/or unstable. In order to perform detailed simulations, the sequential Poisson solver has to be parallelized.

The target for the ACH [2,3]:

* Implementation of 2D parallel Poisson solver with good scaling and speed so that the number of cores in simulations can be increased to at least 128 (current practical limit is around 32) and the grid size can be increased (UMFPACK has a limit of ~4000 cells in one dimension);
* Implementation of parallel routine for E-field calculation (later task).
  1. *Summary of the BSC work and optimization results.*

We included in this section a summary of the work into ACH project do it by BSC team in SPICE2

* The code was subjected to an exhaustive performance analysis and general optimizations were made.
* Three Poisson solvers were included in the code: psolver=7 (a serial Jacobi Preconditioned Conjugated gradient Solver), Psolver=8 (a MPI parallelized version of psolver 7), Psolver=10, a solver based in PETSC library [2].
* A subroutine for a parallel calculation of electric field was implemented.

As a result of this work, the code was tested up to 512 processors over a domain of 10000x2000. The scalability in MARCONI cluster was acceptable up to 128. In Mn4 also were runner with acceptable scalability.

The accuracies of the results of both parallel solvers are good and were compared with the direct solver when this was possible, in small and medium domains.

The detailed analysis of this work is included in section 2, 3, 4 and 5 of this text. Some recommendation to the developers is included in the last section in order to gain more performance and portability for SPICE2.

* 1. *Brief description of the hardware used*

We install, compile and run SPICE2 in two environments.

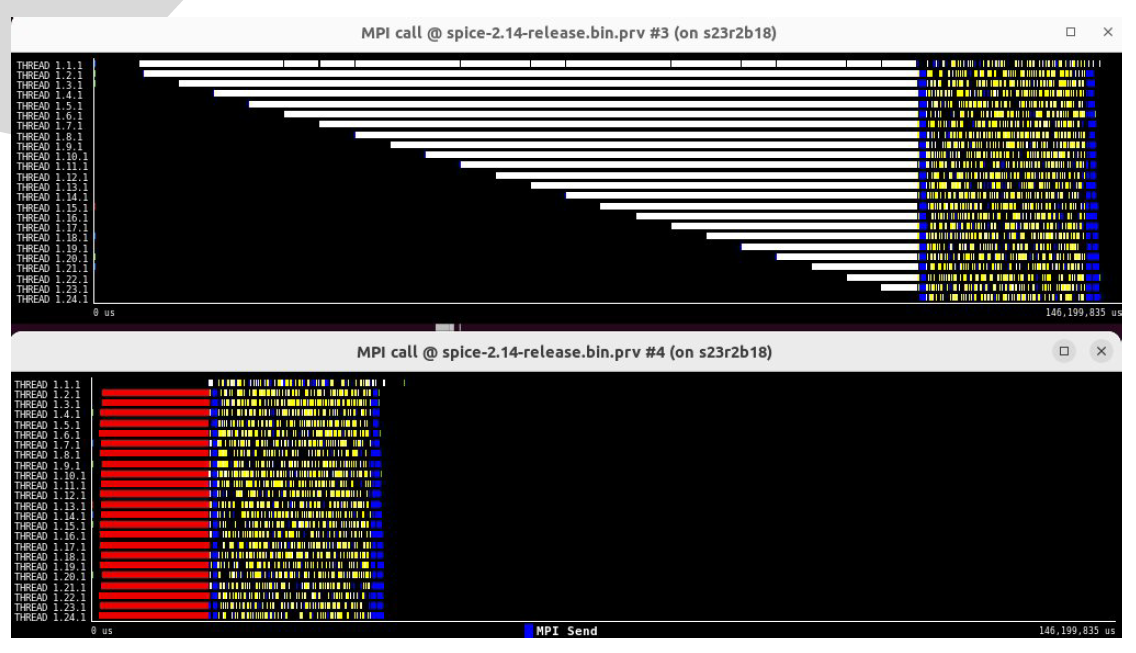
MARCONI: We use the partition on CINECA centre in the context of EUROfusion community (account FUSIO\_HLST\_1). The main queue used was **MARCONI\_Fusion** (Skylake) SKL [4]. This consists in nodes with two processors of 24 cores each. (2 x 24-cores Intel Xeon 8160 CPU (Skylake) at 2.10 GHz) with 192 GB/node [4]. The partition used was A3, and the main queue, skl\_fua\_prod. If a job used more than 128 nodes, a different queue need to be used (skl\_qos\_fuabprod).

Marenostrun 4 [5]. MareNostrum4 is a supercomputer based on Intel Xeon Platinum processors from the Skylake generation. It is a Lenovo system composed of SD530. Compute nodes are equipped with 2 sockets Intel Xeon Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of 48 cores per node and 96 GB of main memory. Exist 216 nodes with high memory that have 383Gb.

1. **General optimizations using Extrae and Paraver**

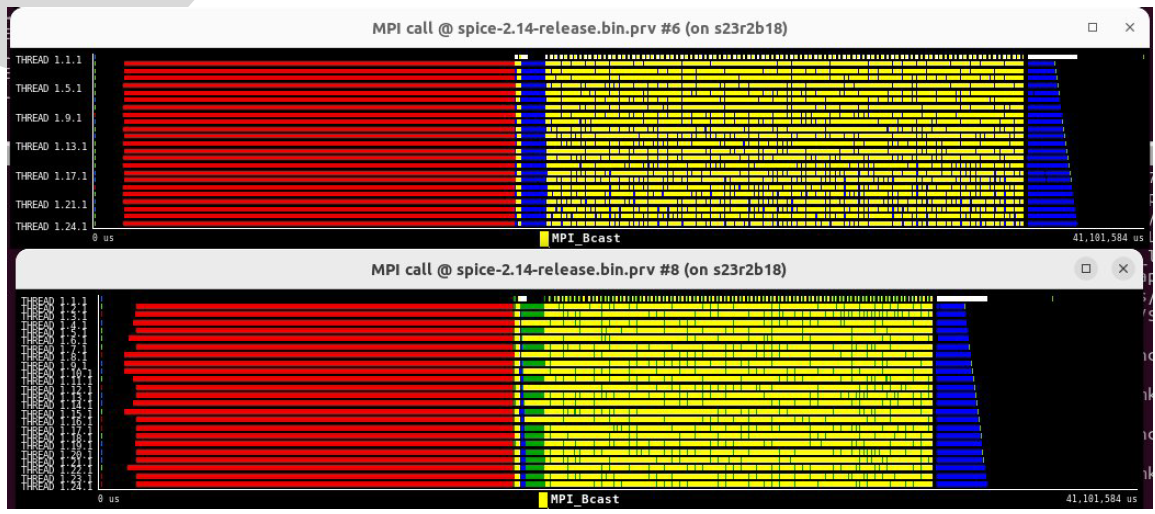
The BSC tools Paraver [8] and Extrae [9] were used to analyze memory consumption and communications environments. Some issues were identifies and changes in order to generate a more efficient use of computer time.

All processors write the same file at the beginning of the simulation. Each process waits 6\**their\_rank* second to stars writing. This accumulates unnecessary waiting time. The code was updated to only do one writing on the process with rank 0 and the other processes wait a barrier (red color on the trace, figure 1)



*Figure 1: Paraver traces that show the barrier and the waiting time saved.*

The communications Point to Point to send/received the potential and density matrix to all the processors were changed to a collective MPI\_broadcast/MPI\_reduce communicators.



*Figure 2: Paraver traces that show the time saved when collective communicators are used.*

1. **Parallel Solvers and local electric field calculation**

The solver currently used in SPICE2 is a direct solver based in UMFPACK and BLAS libraries, using a sparse way to manage the matrix information. The solver, predictably, is very fast although it contains the limitations of any direct solver, when reaching the memory limit of the system. Other parallel implementations of the solver into the code present limitations in the scalability.

* 1. *Conjugated gradient solver (in house)*

A serial and a parallel Poisson solver based on the Preconditioned Conjugate Gradient Method (PGC) were implemented in the corresponding subroutines psolver\_CG and psolver\_CG\_MPI. The used parallelization scheme of the PCG is described in detail in [6]. The PCG algorithm was previously proved by the HLST to be the effective in the type of tasks with which SPICE is operating [1].

Nevertheless the scalability of the solver is not so good as the expected, although the results are as accuracy as the obtained with direct solver.

The solver works splitting the domain in z directions, that’s mean taking domains of (Lz/nprocessors)\*Ly, i.e: the variables in $domaindecomp need to be choised as no\_slices\_z= nprocessors and no\_slices\_y=1.

* 1. *Petsc solver*

The second parallel solver used is based in Petsc library. We use the KSP linear solver context [7] and we explore several options for solve the Poisson equation from SPICE2. From all the solvers tested, we identify 4 of them that’s have the better CPU time and scalability, and for all this the option KSP=1 is the best. The option can be introduced by input file in the namelist *$bscinput*, using the variable ***kspsolver***. In the same namelist, ***crit\_conv*** allow to choose the convergence tolerance of the solver.

In table I we list and describe the KSP tested.

*Tabla I: KSP solvers tested in Petsc implantation into Spice2. The stars (\*) indicate the faster solvers.*

|  |  |  |
| --- | --- | --- |
| kspsolver | Name | Description |
| 1\*\*\* | KSPCG | (preconditioned) conjugate method |
| 2 | KSPBICG | (preconditioned) biconjugate residuals method |
| 3 | KSPMINRES | (preconditioned) Minimum residual method |
| 4 | KSPCR | (preconditioned) conjugate residuals method |
| 5\*\* | KSPBCGS, | (preconditioned) biconjugate stabilized method |
| 6\* | KSPPIPEBCGS | pipelined BiCGStab method |
| 7\* | KSPFBCGSR | mathematically equivalent variant of flexible bi-CG-stab |
| 8 | KSPPIPECG | pipelined CGS method |
| 9 | KSPRICHARDSON | preconditioned Richardson iterative method |
| 10 | KSPSYMMLQ | SYMMLQ method |
| 11 | KSPIBCGS | Improved Stabilized version of BiConjugate Gradient |
| 12 | KSPDGMRES | deflated GMRES as defined |
| default | KSPGMRES | GMRES |

Is necessary to explain that the solver from Petsc need to be used choosing the number of nodes and processors in a way to avoid the saturating of the memory access. Then, in two socket architecture, like MARCONI and MN4 architecture, with two processors with 24 cores each, the better performance of Petsc is reached using a configuration of certain number of processors by node. That’s mean that only certain configuration are acceptable. And this is architecture depending, then, in each machine in where SPICE2 want to be run, will be a different configuration that need to be explored before. In the case of MARCONI, two processors, one per socket, shows the better performance of the solver. In the Marenostrum 4 case the wall time starts to growth with respect to cpu time, when we choose 4 processors by socket.

Another point to consider is that the solver works splitting the domain in z directions, that’s mean taking domains of (Lz/nprocessors)\*Ly, i.e: the variables in $domaindecomp need to be choice as no\_slices\_z= nprocessors and no\_slices\_y=1.

Finally, the solver Petsc consider a homogenous mesh with dz=dy=1.0. Different relation of these values is under development at this moment.

* 1. *Parallel electric field subroutine*

The subroutine ***calc\_E\_field\_local*** was builder following the original global subroutine that starts from the potential matrix in order to make derivatives establishing different condition in the boundaries. In each local domain we create an extended potential matrix that has *(local\_dimz+2; local\_dimy+2)***,** in *z* direction, received the last and first column from the respective neighbors; in *y* direction, the rod *2* and *local\_dimy-1* are copy in the rod *local\_dimy+2* and *1* respectively. Using this extended potential, the calculus of the electric field are developed over all the nodes without worry about when are the node positioned.

1. **Main BSC modifications into the code.**

* Three Poisson solvers are included in the code: psolver= 7 (a serial Preconditioned using Jacobi method Conjugated gradient Solver), Psolver=8 (a MPI parallelized version of psolver 7), Psolver=10, a based in PETSC library solution.
* A subroutine for a parallel calculation of electric field was implemented.
* Two files were included in the code: BSC2spice\_sbrt.f90, with the new subroutines added to the code and mod\_PETSC\_Poisson2DparAIJ.F90, the Petsc solution subroutines.
* The file MATFILE.f90, in subroutine save\_temp\_proc\_file() needed to be changed in order to allow the code write in a number of files greater than 99.
* Makefile was changed in order to use PETSC library. A demo executable is also compiled (if the option is turn on in the compile.sh script) in order to test the solvers independently. The file can be deleted and the makefile changed without loose anything in SPICE2 code.
* A new inputs namelist (namelist /bscinput/ kspsolver,crit\_conv,n\_saca\_bsc) with the kind of KPS solver for PETSC library option, the convergence criteria for both parallel solvers and a integer in order to select output by screen message when the debug option of the code are not used.

1. **Scalability studies and solution comparisons.** 
   1. *Scalability of the parallel solvers*

The new solvers were analyzed in separate instances, in order to take measures and performance without any interference. In figure 3 and 4 those measures are presented, showing the best performance in the Petsc case. In this last case, is could be noted that for certain KSP choice, super scalability are obtained, but when the Wall time are compared, the better performance always is reached with KPSPCG.

These tests were do it for a medium matrix size, generated with a LzxLy=1612x1612 domain.

Chart, line chart

Description automatically generated

*Figure 3: Scalability of the parallel Conjugated Gradient solver.*

Chart

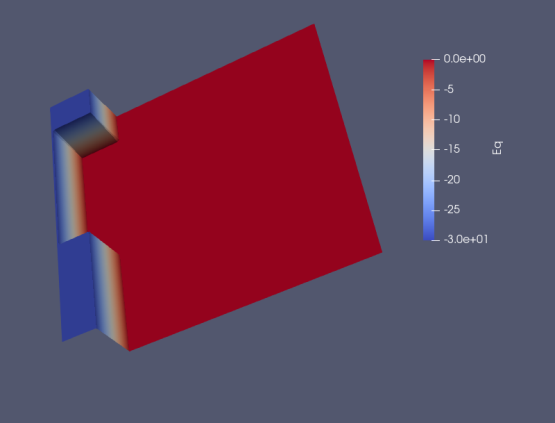
Description automatically generatedChart

Description automatically generated

*Figure 4: Scalability of selected KSP Petsc solver analized (left); Wall time consumed by each KSP solver (right).*

* 1. *Spice2: small case*

To compare results, we create a small case of LzxLy=160x160. Figure 5 shows the domain with the equipotential chosen. Figure 6, 7 and 8 shows respectively the solution across the z axis in four y positions; the density of charge and the potential over the domain obtained after 1000 iteration using the three solvers (psolver=2, 8 and 10).

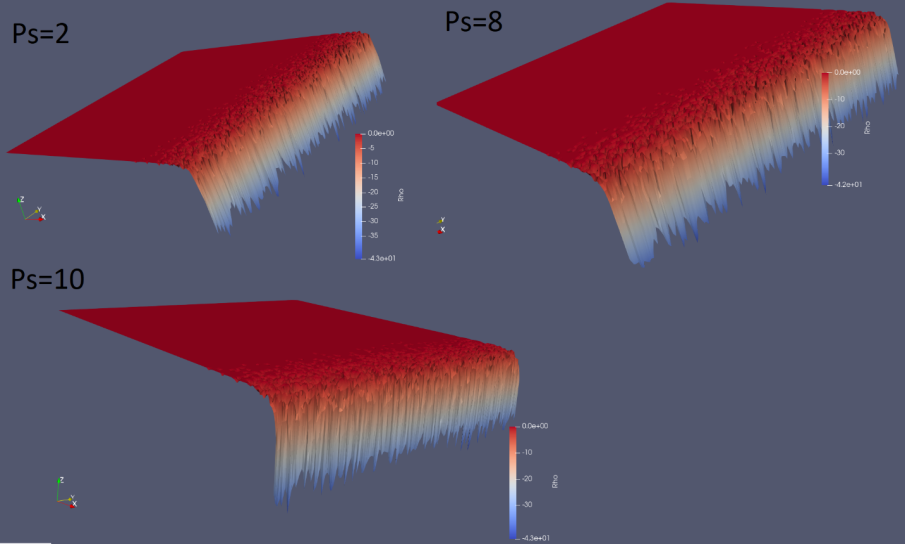


*Figure 5: domain used to test the accuracy of the solvers into spice. We use a LzxLy=161x161.*

Graphical user interface, chart, line chart

Description automatically generated

*Figure 6: Potential obtained after 1001 iterations in small domain for different Ly positions.*



*Figure 7: Density distribution after 1001 iterations in small domain using the solvers 2, 8 and 10 and the parallel subroutine for electric field.*



*Figure 8: Potential after 1001 iterations in small domain using the solvers 2, 8 and 10 and the parallel subroutine for electric field.*

* 1. *Spice2: medium case*

A medium size domain (LzxLy=1612x1612) where used to take performance measures and accuracy of the code, being able to compare the three solvers development. Figure 9 shows the time consumed of Spice2 using the three options, and the reduction of time in the case of Petsc option. Up to 64 processors, the psolver=2 still being the faster option, but we can check the good trend of psolver 10.

Figure 10 shows the good accuracy of the three solutions in different Ly lines. Figure 11 shows the results of electric field prediction using parallel and serial subroutines.

Chart

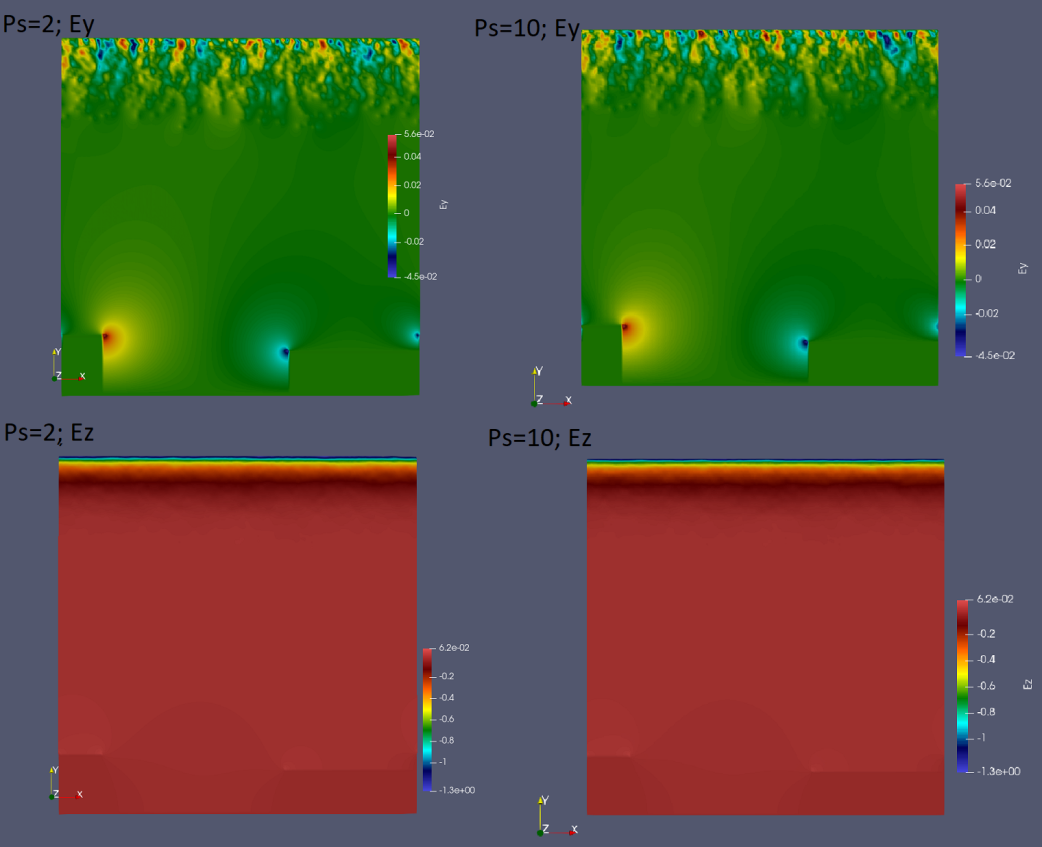
Description automatically generated

*Figure 9: Performance comparison of Spice 2 using a medium size domain.*

Graphical user interface, chart, line chart

Description automatically generated

*Figure 10: Potential obtained after 1001 iterations in medium domain for different Ly positions and for psolver 2, 8 and 10.*



*Figure 11: Electric field obtained after 1001 iterations in medium domain for psolver 2 and 10. The parallel electric field subroutine is used in the last case.*

* 1. *Spice2: Big case*

We increase the domain size up to a matrix of LzxLy=10000x2000, the direct solver reaches the *“Undefined Case!!”* output, and is not possible to be used. We run this case using psolver=10 with ***kspsolver***=1 and ***crit\_conv***=1e-5.

Chart, line chart

Description automatically generatedChart

Description automatically generated*Figure 12: Scalabilities measures in MARCONI.*

Gráfico, Gráfico de líneas

Descripción generada automáticamente

*Figure 13: Scalabilities measures in MN4.*

Up to 128 processors, both systems have a good scalability. Figure 12 left shows the Petsc evaluation, standalone. We can note that after 128 processors a loose of performance is obtained. Figure 13 right plots results for the whole code using psolver=10. In figure 13 both results, Petsc alone and Spice2 are showed up to 128.

In both systems the same loose of scalability is obtained for 256 and 512 processors, but since the local domain is so small, the number of communications makes the problem inefficient. However, it was possible to run it on that number of processors, making us think that if we increase the domain, that efficiency will improve.

Another point to take into account in Spice2 is the memory management. There are some recommendations in the section 6 about it, but basically, the number of particles used and the huge amount of points that big domains implies saturated the memory of the nodes in MARCONI and MN4 clusters.

1. **General recommendations far away of this report.**

Faraway of this report, some ideas to improve the management of memory and performance of SPICE2 could be:

* Avoid the use of arrays with dimension equal to the number of time steps. This is an unreasonable use of memory. Any information allocated in this way could be changed writing it in a file in disk. Probably the information is used in post processor instance and could be reader from this file. We found (6+nprocessors) vectors real\*8 and (1+nprocessors) vector integer with dimensions historie\_ntimes (in the example used 5000000x2). Also (16+6\*number of objects) real\*8 vectors with dimension Np, (the number of time steps to be made). This amount of memory that growths with the number of processors avoid the use of big domains in the code. (In MARCONI, a 20000x4000 are forbidden, even using two cores by node).
* Although there is a solver for the position equation distributed between processes, a local charge density vector, and electric field vectors also distributed, the code, at the end of each calculation, builds global matrices in each processor, and this means a huge amount of information that must be sent to all processors. If only local vectors could be used and communications avoided, the code would increase its efficiency considerably.
* The solver, as is used in SPICE2, is builder over the same matrix that covers all the nodes of the domain. But due to some portion of the domain is equipotential, the results in this portion are known before the solution, adding also that some domain portions have small variation of charge density during many time steps. For both reasons, could be a good improvement to solves the Poisson equation over a different mesh, probably based in a discrete method as finite elements, in where the limits of the domain could be reduced leafing out the equipotential portion and with a mesh less dense than the general domain used. Of course an interpolation subroutine need to be created in order to map the potential from one mesh to other, but this could be less demanding from the point of view of computer time, than solve a matrix of several million of equations in each time step. Thinking in a domain of 10000x10000 imply 1e9 x 1e9 matrix, even with the faster solver that exist, that's meaning a lot of computing time. A good selection of the Poisson grid can reduce considerably the size of the matrix and the computing time.

1. **Future work**

Some points are still in progress in BSC team:

* The performance of the code is still under analysis in different mesh sizes and conditions. Besides, the analysis of the different behavior of the Petsc Solver in MARCONI and MN4 is under researching.
* The modification of the Petsc solver to be used in cases in where dz is not equal to dy are under development.
* The BSC team is ready to test, analyze and improve any modification introduced into the code following the suggestion realized in section 6 (or any other introduced by the developers, of course).

**References**

[1] Mochalskyy, S, HLST annual report 2018.

[2] R Hatzky et al. EUROFUSION WPISA-REP(18) 23083. HLST Core Team Report 2018

[3] https://www.bsc.es/supportkc/docs/MareNostrum4/intro

[4] <https://www.hpc.cineca.it/content/documentation>

[5] https://wiki.ugov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide

[6] Shapira, Y., Matrix-Based Multigrid: Theory and Applications(Second Edition). Springer, New York, 2008.

[7] <https://petsc4py.readthedocs.io/en/stable/manual/>

[8] <https://tools.bsc.es/paraver>

[9] <https://tools.bsc.es/extrae>