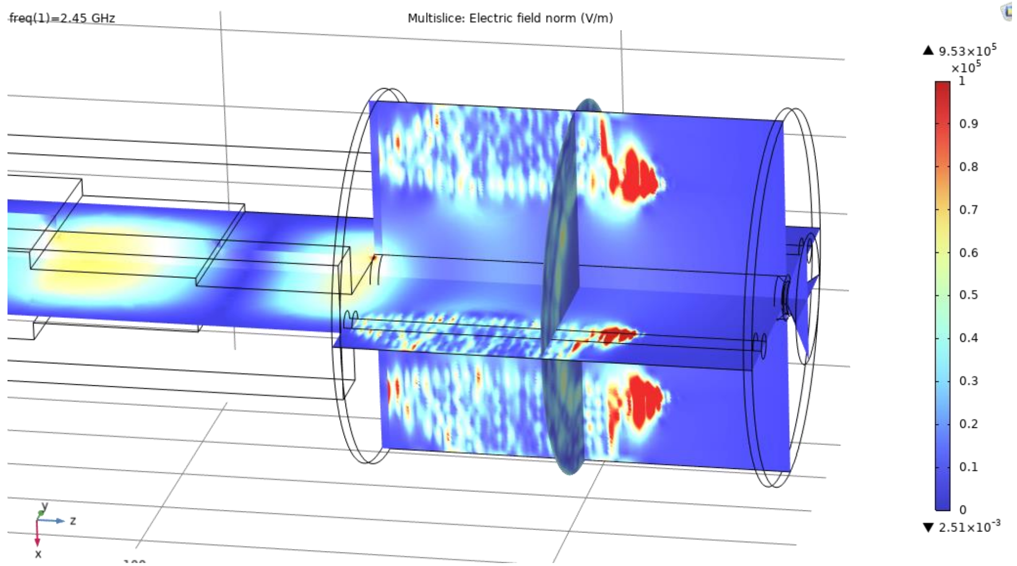
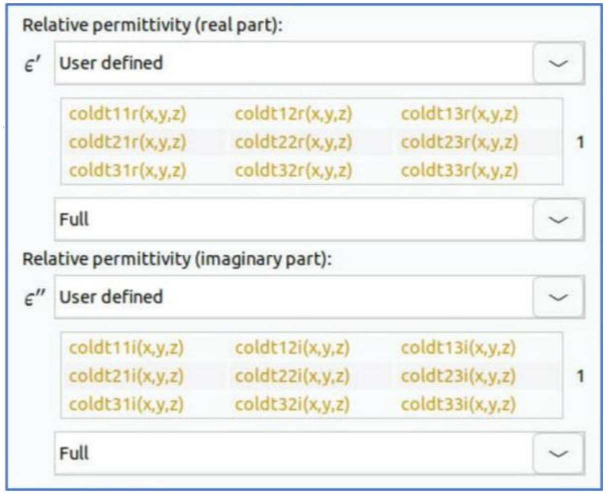
PIC simulation of a 2.45 GHz ECR ion source using COMSOL tensorial permittivity RF capability



Lorenzo Neri

INFN-LNS



COMSOL CONFERENCE 2024 FLORENCE

October 22–24

FLORENCE TEATRO DEL MAGGIO MUSICALE FIORENTINO

Outline

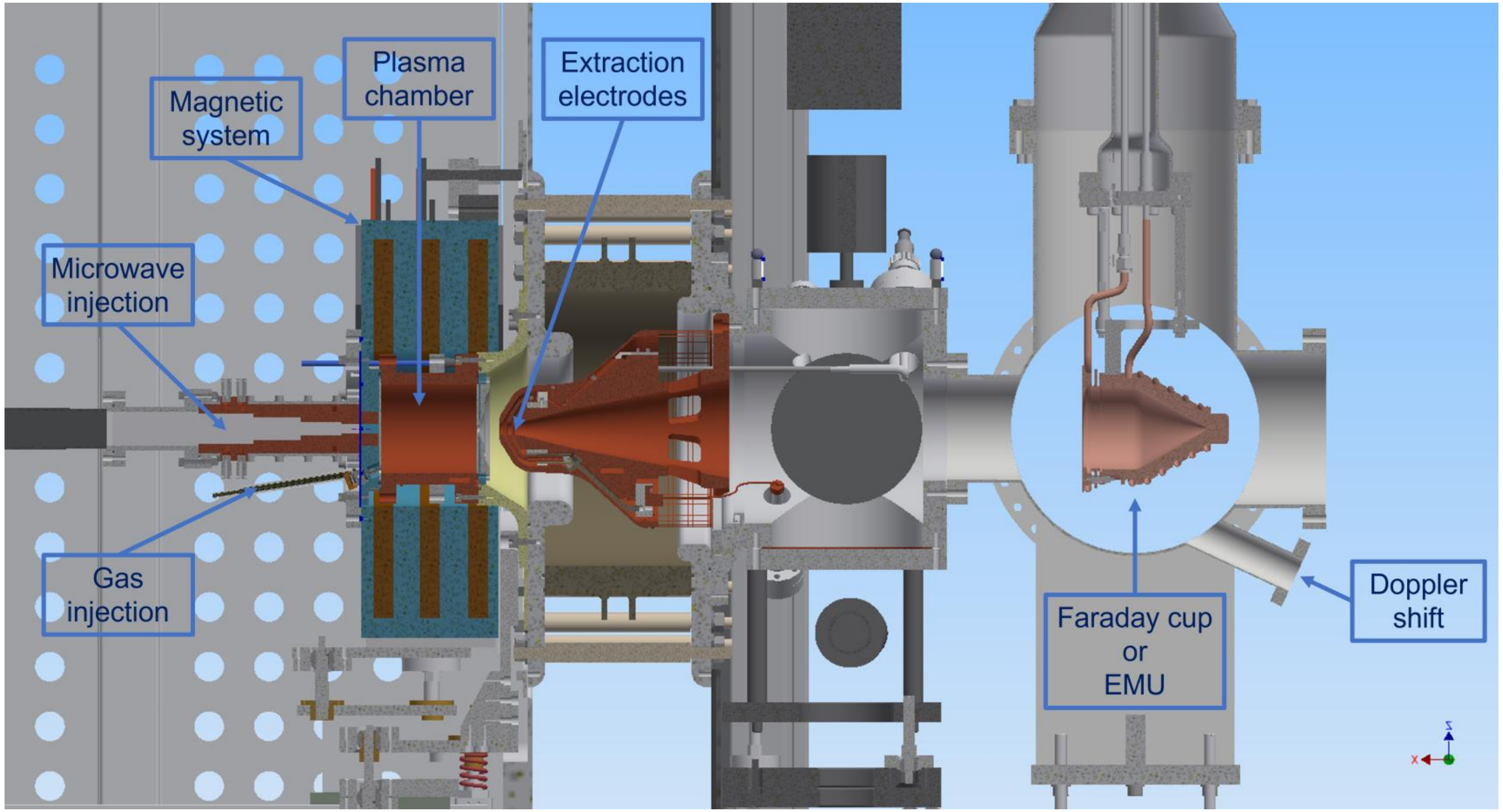
Standard Outline

- How
- Results
- Perspectives

Outline for COMSOL users

- Results (brief scientific report)
- How COMSOL was used
- Perspectives about COMSOL use

2.45 GHz ECR ion sources



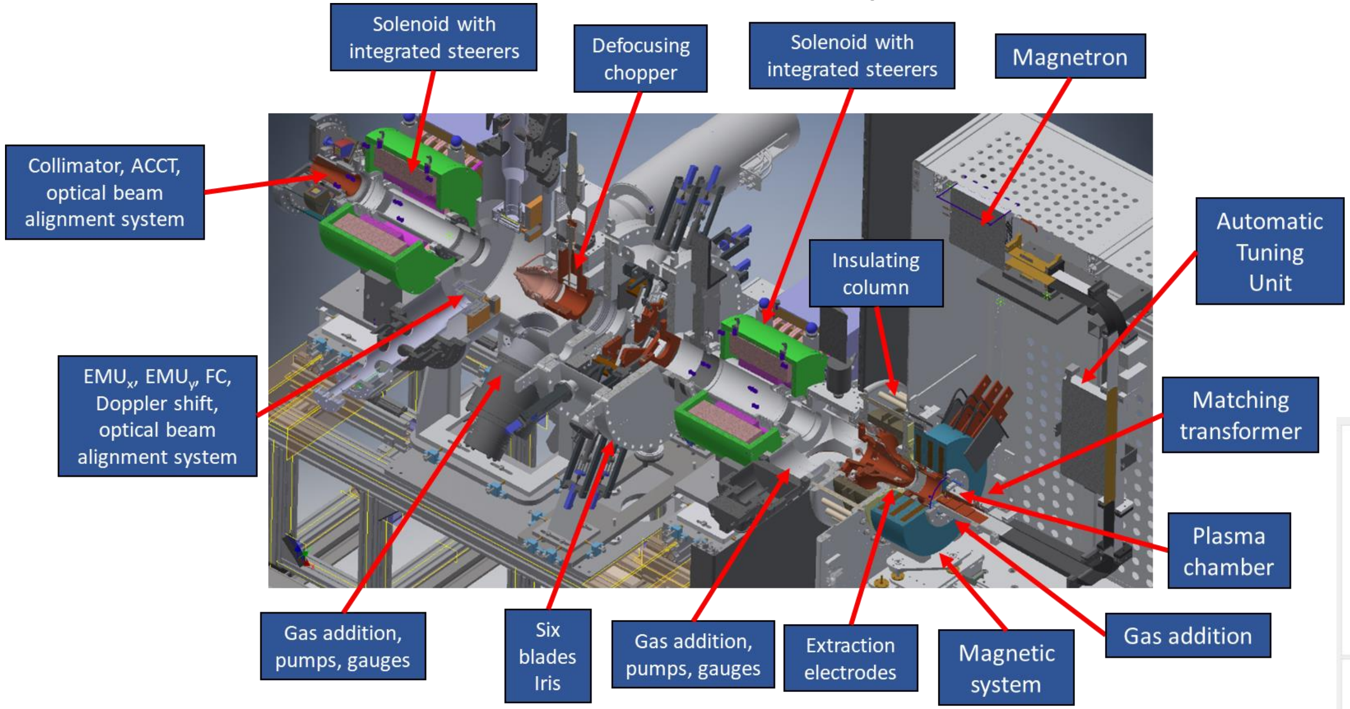
Proton Source for the European Spallation Source (PS-ESS), developed and commissioned at INFN-LNS and sent to ESS in Sweden 01/02/2018

ECR resonance: $\omega_{RF} = \frac{q_e B}{m_e}$

Gas (4 SCCM H_2)
+
Magnetic field (≈875 Gauss)
+
Microwave power (2.45 GHz)
=
Plasma (1E17 m^{-3})

+
High voltage (75 kV)
=
Beam (100 mA)

Proton Source for European Spallation Source (PS-ESS) and the new optimum ion source magnetic configuration

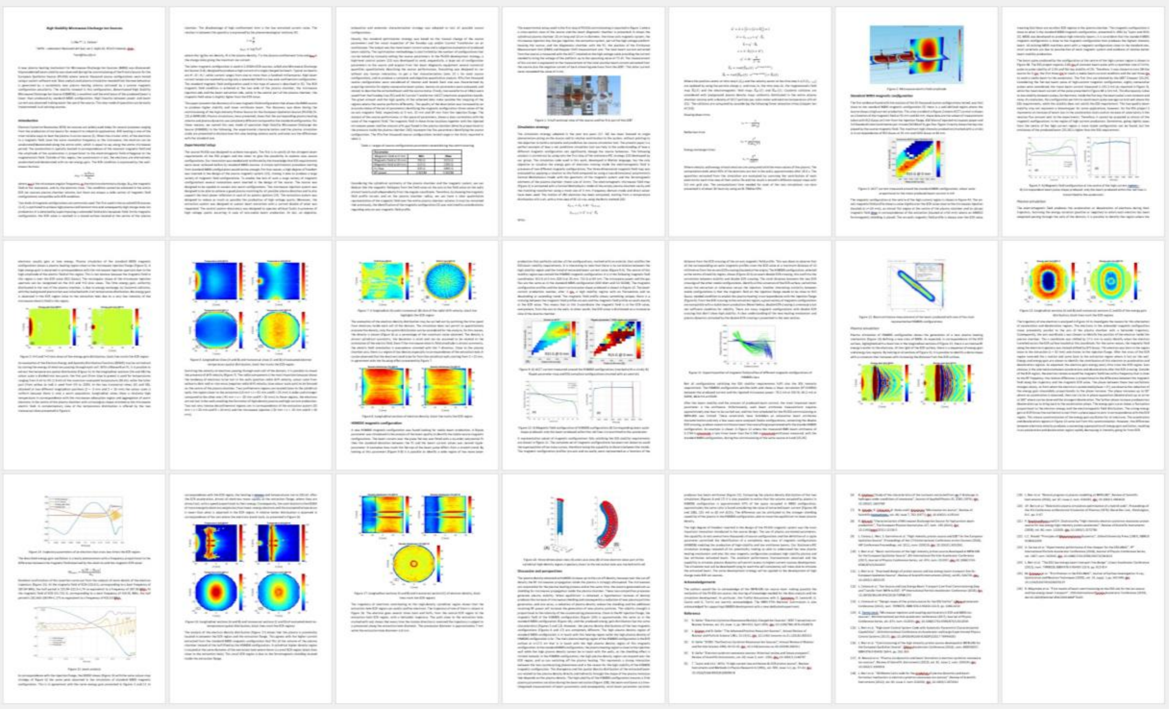


Title: High Stability Microwave Discharge Ion Sources

Authors: L. Neri, L. Celona

Journal: Nature Scientific Reports

12, 3064 (2022) https://doi.org/10.1038/s41598-022-06937-7



EUROPEAN SPALLATION SOURCE

The team of the HSMDIS project (INFN-CSN5)

INFN-LNS in Catania:
Lorenzo Neri (developper), **Giuseppe Castro, Ornella Leonardi, Andrea Miraglia, Lugi Celona, Santo Gammino**

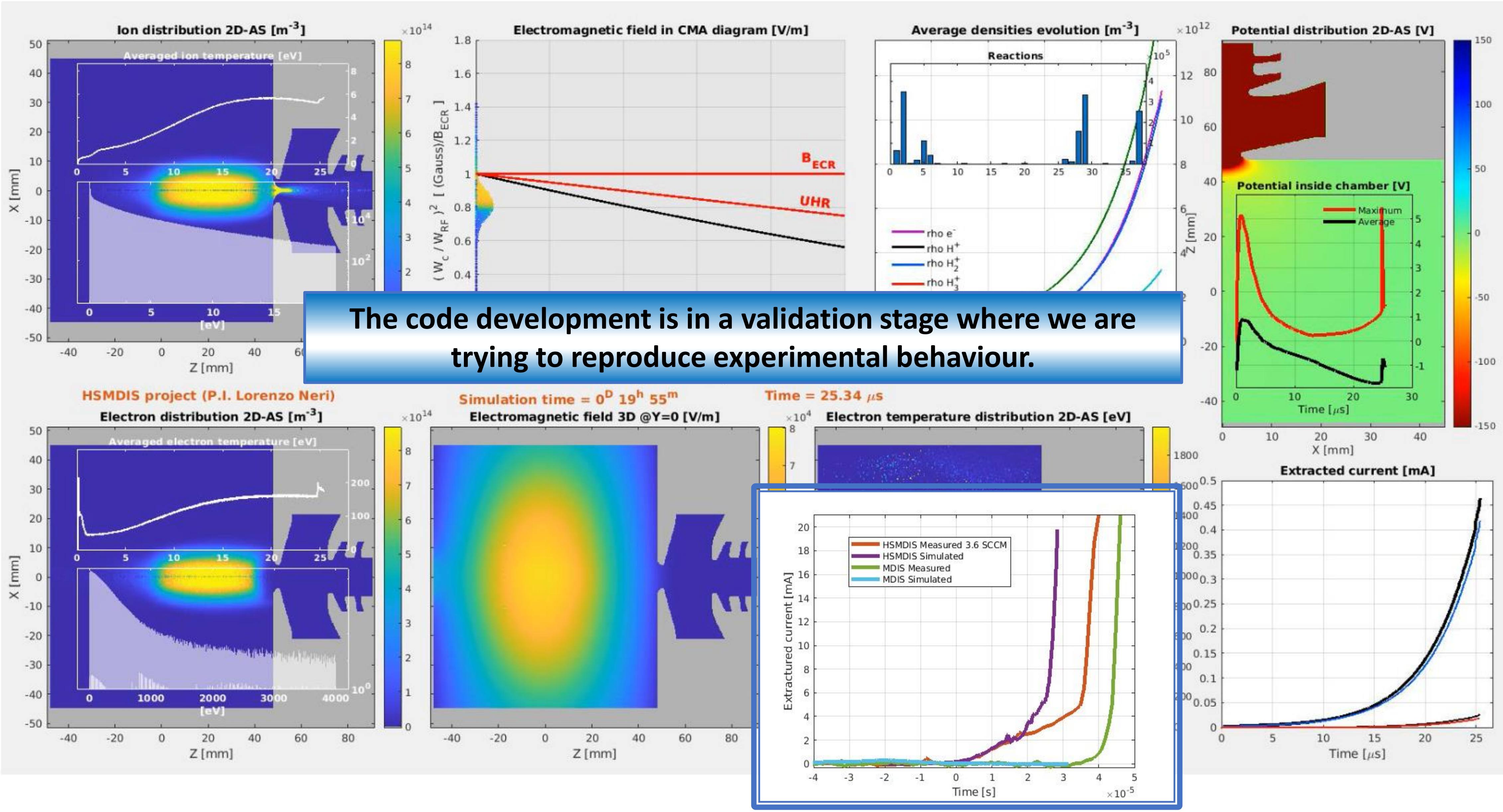
INFN-LNL in Legnaro:
Francesco Grespan, Michele Comunian

UNICT-DMI in Catania:
Giovanni Russo, Sebastiano Boscarino, Armando Coco
(mathematicians working on custom Poisson solver)

INGV in Catania:
Giuseppe Bilotta
(mathematician working on code optimization)

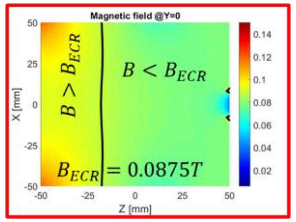
CNR-ISTP in Bari:
Gianpiero Colonna, Annarita Laricchiuta, Francesco Taccogna
(working on plasma chemistry)

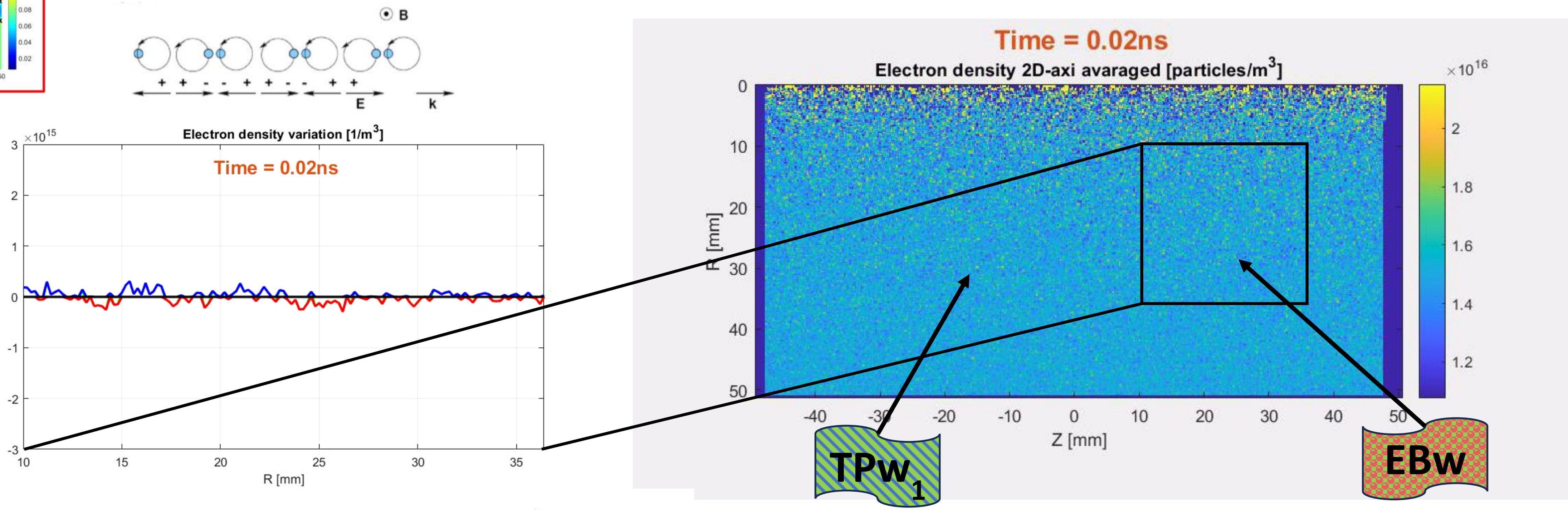
PIC simulation of the HSMDIS magnetic configuration

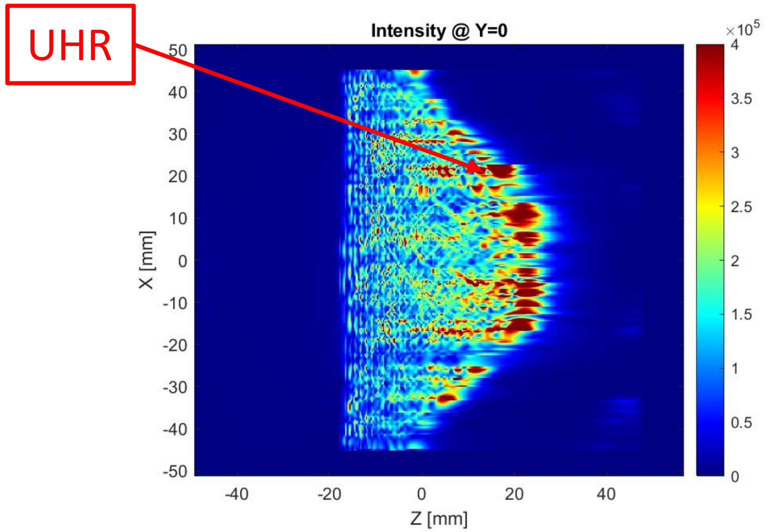


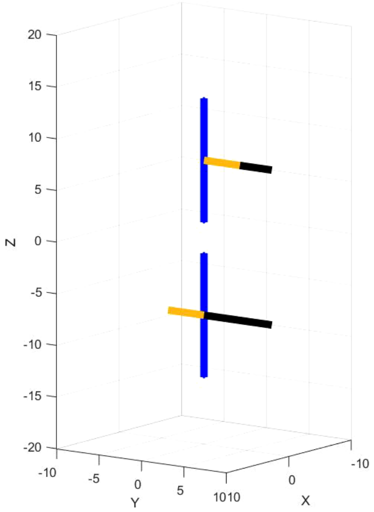
The code development is in a validation stage where we are trying to reproduce experimental behaviour.

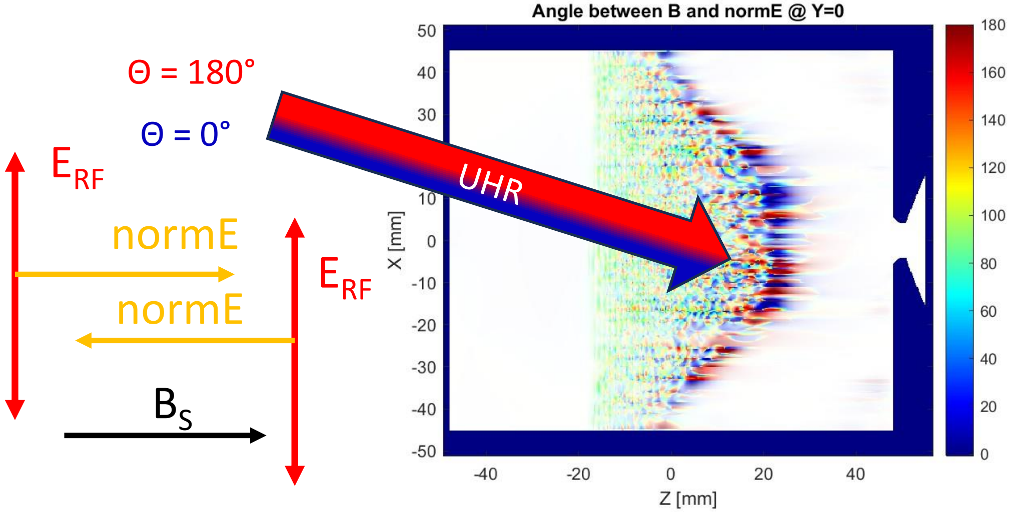
Generation of Electron Bernstein waves





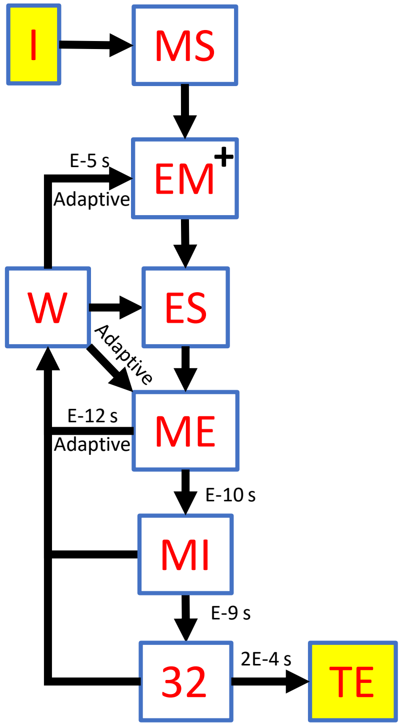






PIC simulation tool

From plasma formation to beam extraction



- **3D Initialization** of 1E7 particles
- **3D MagnetoStatic** simulation
- **3D** (2.45GHz) **ElectroMagnetic** simulation with tensorial complex permittivity

First electrostatic computations in COMSOL last **10 seconds**
➔
Now custom Poisson solver needs only **0.064 seconds**

e1: $H_2 + e \Rightarrow H + H + e$
e2: $H_2 + e \Rightarrow H_2^+ + e + e$
e3: $H_2 + e \Rightarrow H^+ + H + e + e$
e4: $H_2 + e \Rightarrow H^+ + H + e + e$
e5: $H_2 + e \Rightarrow H_2 + e$
e6: $H_2 + e \Rightarrow H_2$*(singlets) $+ e \Rightarrow H_2^v + e + h\nu$
e7: $H_2 + e \Rightarrow H_2^- + e + e \Rightarrow H_2^v + e$
e8: $H_2^v + e \Rightarrow H_2^+ + e + e$
e9: $H_2^v + e \Rightarrow H + H + e$

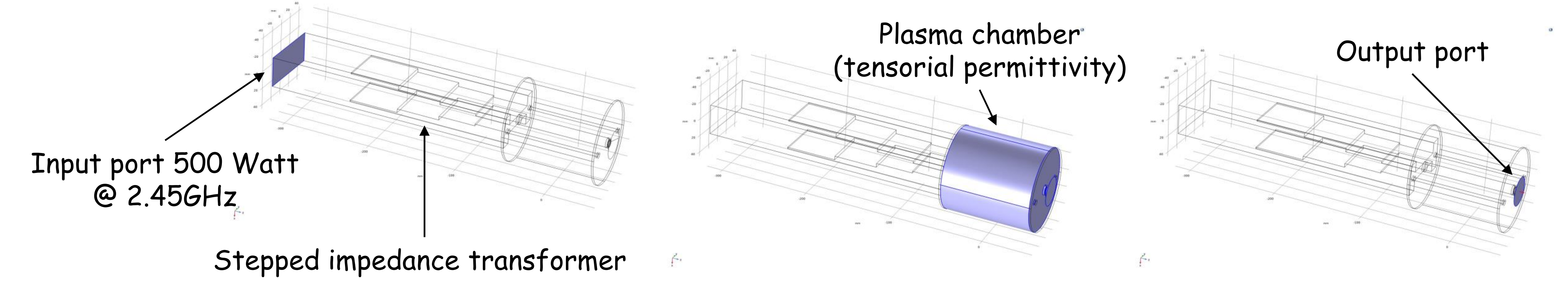
e10: $H_2^v + e \Rightarrow H_2 + e$
e11: $H + e \Rightarrow H^+ + e + e$
e12: $H + e \Rightarrow H + e$
e13: $H + e \Rightarrow H^n + e$
e14: $H^n + e \Rightarrow H + e$
e15: $H^n + e \Rightarrow H^+ + e + e$
e16: $H^+ + e \Rightarrow H^+ + e$
e17: $H_2^+ + e \Rightarrow H^+ + H + e$
e18: $H_2^+ + e \Rightarrow H^+ + H^+ + e + e$

e19: $H_2^+ + e \Rightarrow H + H$
e20: $H_2^+ + e \Rightarrow H_2^+ + e$
e21: $H_3^+ + e \Rightarrow H^+ + H + H + e$
e22: $H_3^+ + e \Rightarrow H + H + H$
e23: $H_3^+ + e \Rightarrow H_3^+ + e$

time: $H^n \Rightarrow H + h\nu$

i1: $H_2^+ + H_2 \Rightarrow H_3^+ + H$
i2: $H_2^+ + H_2 \Rightarrow H_2 + H_2^+$
i3: $H_2^+ + H \Rightarrow H_2^v + H^+$
i4: $H^+ + H \Rightarrow H + H^+$
i5: $H^+ + H_2 \Rightarrow H + H_2^+$
i6: $H^+ + H_2^v \Rightarrow H + H_2^+$
i7: $H_3^+ + H \Rightarrow H_2 + H_2^+$
i8: $H_3^+ + H_2 \Rightarrow H^+ + H_2 + H_2$

Electromagnetic simulation



Cold tensor approximation

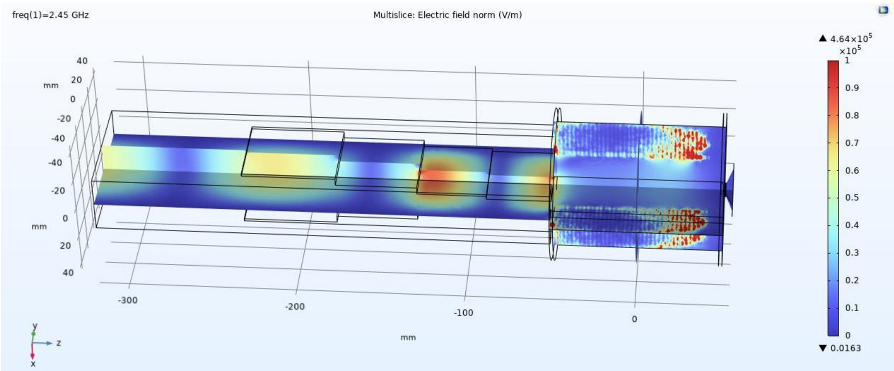
$$\bar{\bar{\varepsilon}}_r = \bar{\bar{I}} - \frac{n_e q_e}{iw\varepsilon_0 (A^3 + AB^2)} \begin{bmatrix} A^2 + B_x^2 & B_xB_y + AB_z & B_xB_z - AB_y \\ B_xB_y - AB_z & A^2 + B_y^2 & B_yB_z + AB_x \\ B_xB_z + AB_y & B_yB_z - AB_x & A^2 + B_z^2 \end{bmatrix} = \bar{\bar{\varepsilon}}_r \left(n_e(\vec{r}), T_e(\vec{r}), \vec{B}(\vec{r}) \right)$$

where:

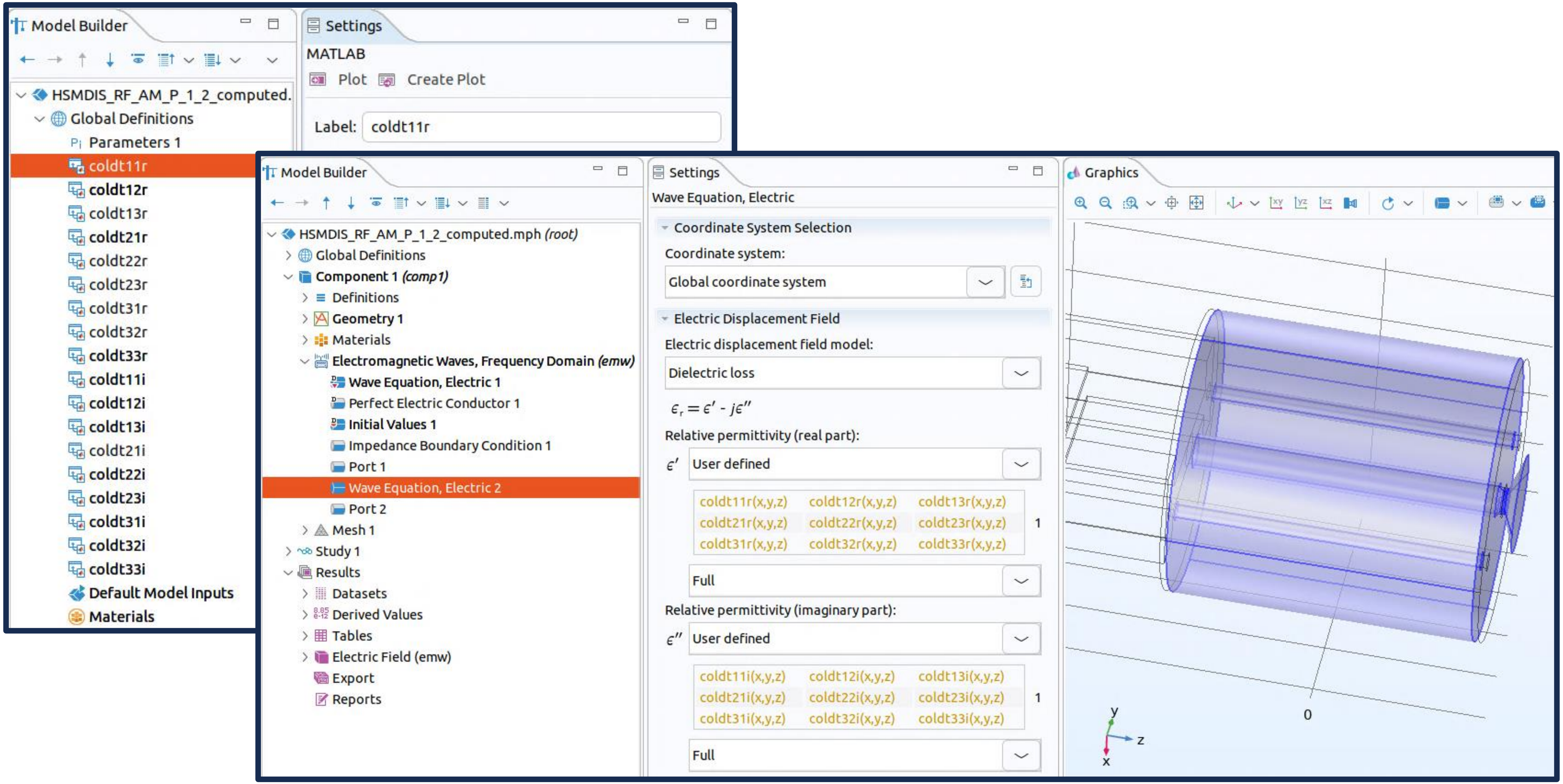
$$A = \frac{-iw - w_{eff}(T_e)}{e/m_e}$$

Tensorial permittivity is needed because the plasma is not uniformly magnetized

Matlab and COMSOL meshes are different, and two interpolation steps are needed



Electromagnetic simulation

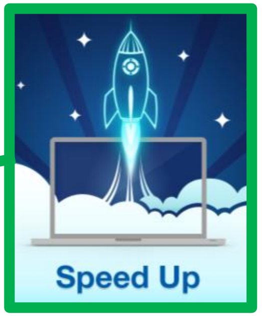


Electromagnetic simulation

1) Compute the complex permittivity and save 18 griddedInterpolant functions

```
Er11I=griddedInterpolant(X,Y,Z,real(Le11)); save([DataDir,'Er11I'],'Er11I','-v7.3','-nocompression')
Er12I=griddedInterpolant(X,Y,Z,real(Le12)); save([DataDir,'Er12I'],'Er12I','-v7.3','-nocompression')
Er13I=griddedInterpolant(X,Y,Z,real(Le13)); save([DataDir,'Er13I'],'Er13I','-v7.3','-nocompression')
Er21I=griddedInterpolant(X,Y,Z,real(Le21)); save([DataDir,'Er21I'],'Er21I','-v7.3','-nocompression')
Er22I=griddedInterpolant(X,Y,Z,real(Le22)); save([DataDir,'Er22I'],'Er22I','-v7.3','-nocompression')
Er23I=griddedInterpolant(X,Y,Z,real(Le23)); save([DataDir,'Er23I'],'Er23I','-v7.3','-nocompression')
Er31I=griddedInterpolant(X,Y,Z,real(Le31)); save([DataDir,'Er31I'],'Er31I','-v7.3','-nocompression')
Er32I=griddedInterpolant(X,Y,Z,real(Le32)); save([DataDir,'Er32I'],'Er32I','-v7.3','-nocompression')
Er33I=griddedInterpolant(X,Y,Z,real(Le33)); save([DataDir,'Er33I'],'Er33I','-v7.3','-nocompression')

Ei11I=griddedInterpolant(X,Y,Z,imag(Le11)); save([DataDir,'Ei11I'],'Ei11I','-v7.3','-nocompression')
Ei12I=griddedInterpolant(X,Y,Z,imag(Le12)); save([DataDir,'Ei12I'],'Ei12I','-v7.3','-nocompression')
Ei13I=griddedInterpolant(X,Y,Z,imag(Le13)); save([DataDir,'Ei13I'],'Ei13I','-v7.3','-nocompression')
Ei21I=griddedInterpolant(X,Y,Z,imag(Le21)); save([DataDir,'Ei21I'],'Ei21I','-v7.3','-nocompression')
Ei22I=griddedInterpolant(X,Y,Z,imag(Le22)); save([DataDir,'Ei22I'],'Ei22I','-v7.3','-nocompression')
Ei23I=griddedInterpolant(X,Y,Z,imag(Le23)); save([DataDir,'Ei23I'],'Ei23I','-v7.3','-nocompression')
Ei31I=griddedInterpolant(X,Y,Z,imag(Le31)); save([DataDir,'Ei31I'],'Ei31I','-v7.3','-nocompression')
```




```
tic;
modelRF.sol('sol1').runAll;
disp([num2str(toc),' sec per calcolare risolvere il modello RF'])
```

2) Compute model using tensorial permittivity provided by Matlab functions

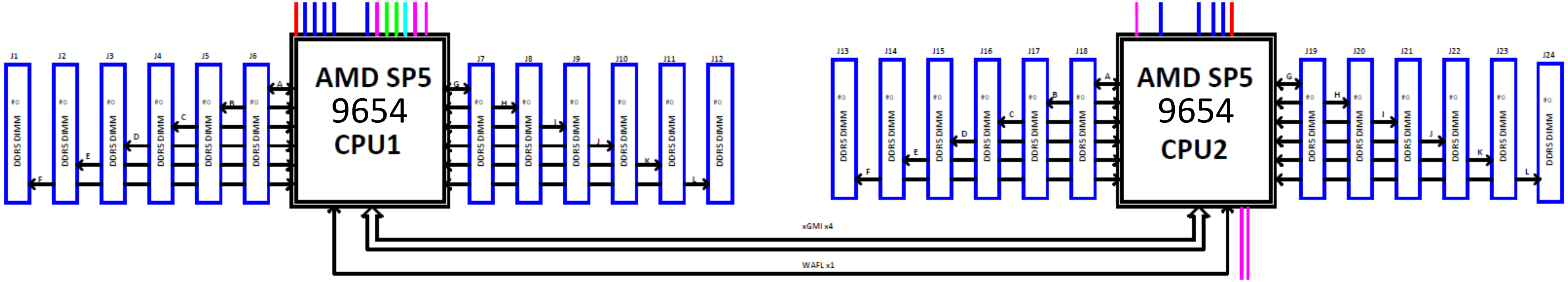
MATLAB

Plot Create Plot

3) Retrieve solution from COMSOL

```
for i=1:numel(CooI)-1
    [Erfx(maskRFf(CooI(i):CooI(i+1))),Erfy(maskRFf(CooI(i):CooI(i+1))),Erfz(maskRFf(CooI(i):CooI(i+1))),...
     Brfx(maskRFf(CooI(i):CooI(i+1))),Brfy(maskRFf(CooI(i):CooI(i+1))),Brfz(maskRFf(CooI(i):CooI(i+1)))]=mphinterp(modelRF,{...
     'emw.Ex','emw.Ey','emw.Ez','emw.Bx','emw.By','emw.Bz'},'coord',Coo(:,maskRFf(CooI(i):CooI(i+1))),...
     'Complexout','on','Unit',{'V/m','V/m','V/m','T','T','T'});
end
```

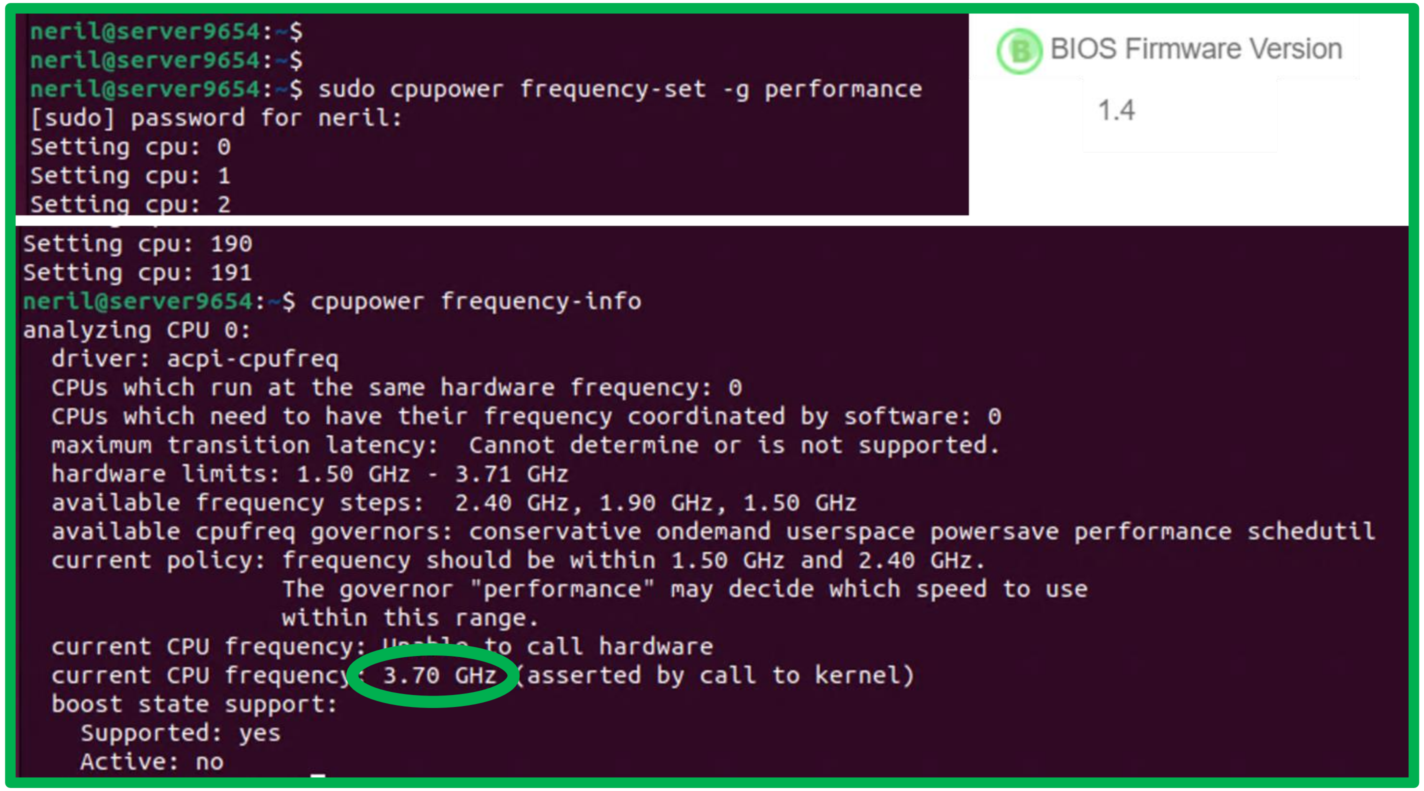
Hardware



30 k€

- Supermicro AS-2025HS-TNR
- 2x AMD 9654, 96 cores each, 2.4GHz / 3.7GHz
- 1.6TB RAM DDR5 4800MT/s
- 6.4TB SSD NVME
- 16TB Hard Disk SATA3

Dummy VGA plug ➔ Enable
1920x1080 remote desktop instead of 800x600

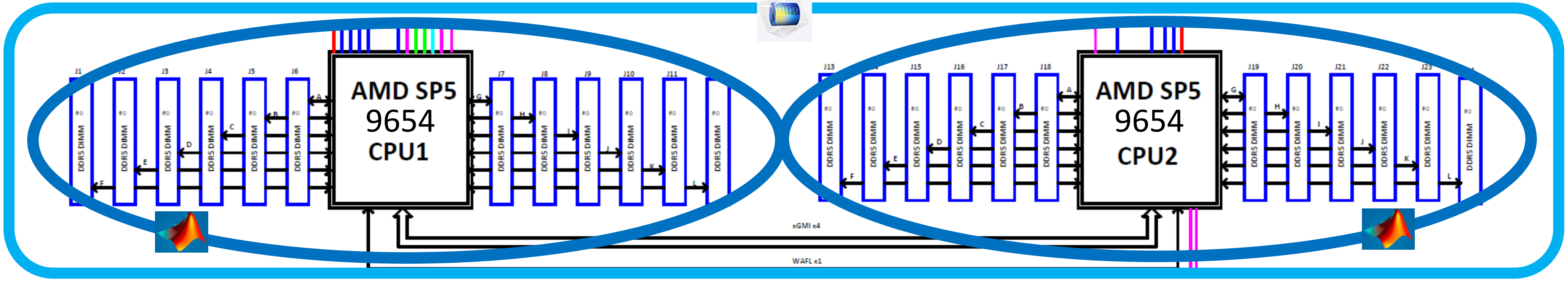


```
neril@server9654:~$
neril@server9654:~$
neril@server9654:~$ sudo cpupower frequency-set -g performance
[sudo] password for neril:
Setting cpu: 0
Setting cpu: 1
Setting cpu: 2
```

BIOS Firmware Version
1.4

```
Setting cpu: 190
Setting cpu: 191
neril@server9654:~$ cpupower frequency-info
analyzing CPU 0:
  driver: acpi-cpufreq
  CPUs which run at the same hardware frequency: 0
  CPUs which need to have their frequency coordinated by software: 0
  maximum transition latency:  Cannot determine or is not supported.
  hardware limits: 1.50 GHz - 3.71 GHz
  available frequency steps:  2.40 GHz, 1.90 GHz, 1.50 GHz
  available cpufreq governors: conservative ondemand userspace powersave performance schedutil
  current policy: frequency should be within 1.50 GHz and 2.40 GHz.
                  The governor "performance" may decide which speed to use
                  within this range.
  current CPU frequency: Unable to call hardware
  current CPU frequency: 3.70 GHz (asserted by call to kernel)
  boost state support:
    Supported: yes
    Active: no
```

One COMSOL server for two Matlab instances



`taskset -c 0-95 matlab`

PIC execution optimized for single CPU

```
%% Connection to Comsol server
disp('Waiting for connection')
BusyComsol=1;
while BusyComsol>0
    try
        load([DataDir,'BusyComsol.mat'],'BusyComsol')
    catch
        BusyComsol=1;
    end
    if BusyComsol==0
        BusyComsol=ID;
        save([DataDir,'BusyComsol.mat'],'BusyComsol')
        BusyComsol=0;
    else
        pause(60)
    end
end

mphstart(2037)
import com.comsol.model.*
import com.comsol.model.util.*
disp('Connected')

BusyComsol=0;
save([DataDir,'BusyComsol.mat'],'BusyComsol')
```

`matlab`

```
>> system('/usr/local/comsol61/multiphysics/bin/comsol
-numasets 2 -blas aocl mphserver -port 2037 &');
```

COMSOL model solver works better on all CPUs

`taskset -c 96-191 matlab`

PIC execution optimized for single CPU

```
disp('Waiting for Comsol server')
BusyComsol=1;
while BusyComsol>0
    try
        load('BusyComsol.mat','BusyComsol')
    catch
        BusyComsol=1;
    end
    if BusyComsol==0
        BusyComsol=ID;
        save('BusyComsol.mat','BusyComsol')
        BusyComsol=0;
    else
        pause(60)
    end
end
tic;
modelRF.sol('sol1').runAll;
disp([num2str(toc),' sec'])
```

```
load('BusyComsol.mat','BusyComsol')
```

To wait until the COMSOL server is free

Perspectives about COMSOL use (1/2)

If you request too many values with "mphinterp"

```
Error using mphinterp
mphinterp (0): Java exception occurred:
Exception:
        com.comsol.util.exceptions.UnexpectedServerException: Out of memory on server.
java.lang.OutOfMemoryError: Out of memory.
        (rethrown as com.comsol.util.exceptions.FlException)
Messages:
        Out of memory on server.
java.lang.OutOfMemoryError: Out of memory.
```

Solution 1: increase swap memory

Solution 2: split into groups the values to be requested

```
for i=1:numel(CooI)-1
    [Erfx(maskRFf(CooI(i):CooI(i+1))),Erfy(maskRFf(CooI(i):CooI(i+1))),Erfz(maskRFf(CooI(i):CooI(i+1))),...
     Brfx(maskRFf(CooI(i):CooI(i+1))),Brfy(maskRFf(CooI(i):CooI(i+1))),Brfz(maskRFf(CooI(i):CooI(i+1)))]=mphinterp(modelRF,{...
     'emw.Ex','emw.Ey','emw.Ez','emw.Bx','emw.By','emw.Bz'},'coord',Coo(:,maskRFf(CooI(i):CooI(i+1))),...
     'Complexout','on','Unit',{'V/m','V/m','V/m','T','T','T'});
end
```

With grouped data we lose the capability to store points of interest that are requested many times, 'keep' 'on' and "getData" don't work

```
if modelnew
    [Esr(Mask),Esz(Mask),Vv(Mask)]=mphinterp(modelEs,{'es.Er','es.Ez','V'},'coord',Coo2, 'keep', 'on');
    modelnew=false;
else
    int=modelEs.result.numerical('interp_internal');
    Esr(Mask)=int.getData(0);
    Esz(Mask)=int.getData(1);
    Vv(Mask)=int.getData(2);
end
```

Is it possible to use "getData" with grouped data?

Perspectives about COMSOL use (2/2)

Custom Poisson solver was reduced to the solution of a linear system using Matlab “mldivide, \”

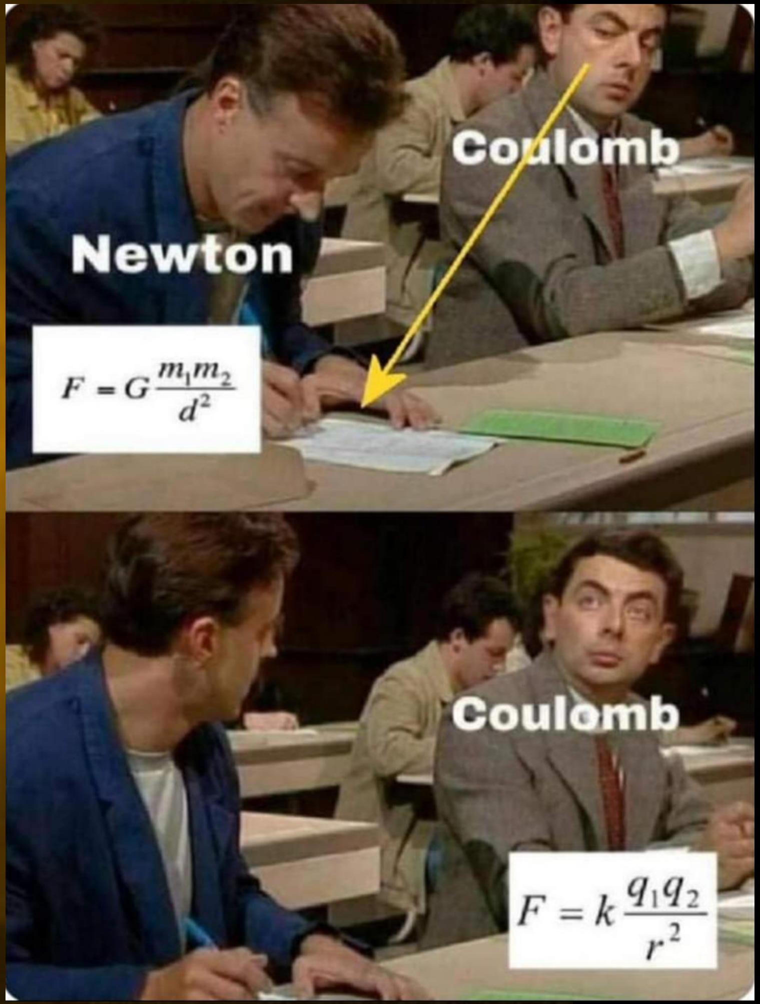
```
    %% Modello elettrostatico
if  kPsi
    % disp('Poisson')
    rhoDa=sum(rhoD(1:2,7)); rhoD(1:2,7)=rhoDa;
    rhoD(rhoD>1E16)=1E16; rhoD(rhoD<-1E16)=-1E16;
    rho=(FrhoV*(rhoNH2s-rhoNE2)+rhoD)*qe6;
    RhoPS(6:end-5,1:end-5)=rho';
    Fas(Insidei)=RhoPS(Insidei);
    Fas(Insided715231)=RhoPS(Insided);
    maxNumCompThreads(1);
    Vall=M\Fas;    ←
    maxNumCompThreads(NNN);
    Vi=reshape(Vall(1:715*231),715,231);
    Vd=reshape(Vall(715*231+(1:715*231)),715,231);
    VasALL(Insidei)=Vi(Insidei);
    VasALL(Insided)=Vd(Insided);
    Vvv=VasALL(6:end-4,1:end-4);  Vvv=Vvv';  Vvv(1,:)=Vvv(2,:);
    Esz=(Vvv(1:226,1:end-1)-Vvv(1:226,2:end))/dx;
    Esr=(Vvv(1:end-1,1:705)-Vvv(2:end,1:705))/dx;
    Esr(1,:)=Esr(2,:);
    KesL=k;
end
```

Decomposition is 9/10 of the computational cost.
In Matlab, it is possible to do it once and use it every time.

```
M=decomposition(M);
```

Is it possible to save decomposition in COMSOL?
Is it possible to use parts of the previous solver iteration?

Thank you for the attention



Lorenzo Neri