CERN

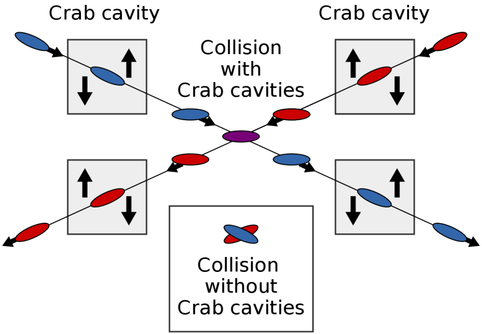
ETH zürich

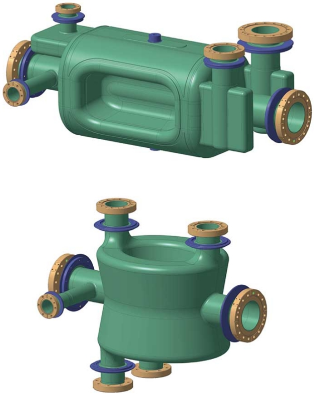
Development of Warp Simulations for 3D RF Structures

L. Giacomel, G. Iadarola, J.-L. Vay

Many thanks to: R. Calaga, M. Carlà, S. De Santis, M. Furman, D. Grote, T. Luo, E. Metral, K. Paraschou, G. Penn, L. Sabato, B. Salvant, M. Schwinzerl, C. Zannini

The Crab Cavities





Computational challenge:

- Complex 3D Geometry
- Cavity EM Mode + Proton Bunch E-field + ECloud self-fields

PyECLOUD Is Not Enough

PyECLOUD implements 2D quasi-static Particle-In-Cell (PIC). For the Crab cavities this is not enough because:

PyECLOUD Is Not Enough

PyECLOUD implements 2D quasi-static Particle-In-Cell (PIC). For the Crab cavities this is not enough because:

- 2D simulations would miss the possible **longitudinal motion** of the ECloud

PyECLOUD Is Not Enough

PyECLOUD implements 2D quasi-static Particle-In-Cell (PIC). For the Crab cavities this is not enough because:

- 2D simulations would miss the possible **longitudinal motion** of the ECloud
- the rapid motion of electrons pushed by the cavity might require an **electromagnetic solver**

PyECLOUD Is Not Enough

PyECLOUD implements 2D quasi-static Particle-In-Cell (PIC). For the Crab cavities this is not enough because:

- 2D simulations would miss the possible **longitudinal motion** of the ECloud
- the rapid motion of electrons pushed by the cavity might require an **electromagnetic solver**
- an electromagnetic solver is needed to compute the **cavity fields**

PyECLOUD Is Not Enough

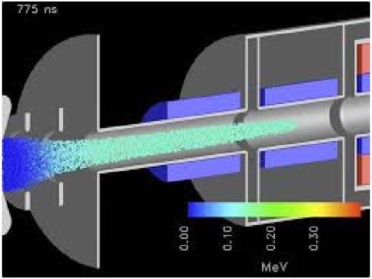
PyECLOUD implements 2D quasi-static Particle-In-Cell (PIC). For the Crab cavities this is not enough because:

- 2D simulations would miss the possible **longitudinal motion** of the ECloud
- the rapid motion of electrons pushed by the cavity might require an **electromagnetic solver**
- an electromagnetic solver is needed to compute the **cavity fields**

Warp, a 3D PIC framework developed at LBNL, offers many of the requeired features.

Outline

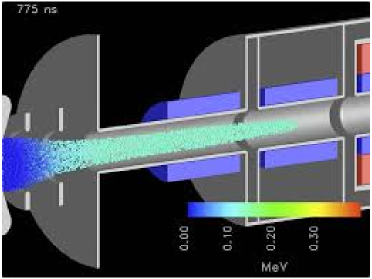
Warp



Warp

Numerical tools:

- FD 2D/3D electrostatic solver
- Several electromagnetic FDTD/PS solvers
- Lorentz-boosted frame
- Boris tracking (classic and relativistic)



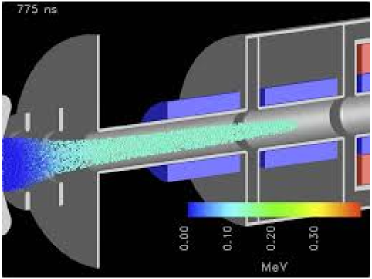
Warp

Numerical tools:

- FD 2D/3D electrostatic solver
- Several electromagnetic FDTD/PS solvers
- Lorentz-boosted frame
- Boris tracking (classic and relativistic)

Geometry: modelling of (almost) any sort of conductor.

Multi-CPU parallel simulations (MPI)



Warp

Numerical tools:

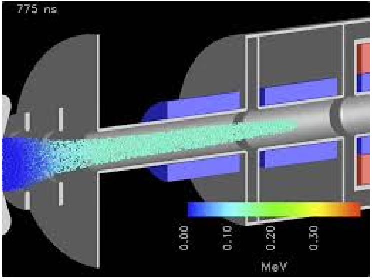
- FD 2D/3D electrostatic solver
- Several electromagnetic FDTD/PS solvers
- Lorentz-boosted frame
- Boris tracking (classic and relativistic)

Typical applications:

- Beam transport
- Laser-plasma acceleration
- ECloud in static 3d structures (coupled with POSINST)

Geometry: modelling of (almost) any sort of conductor.

Multi-CPU parallel simulations (MPI)



Warp

Numerical tools:

- FD 2D/3D electrostatic solver
- Several electromagnetic

Typical applications:

- Beam transport
- Laser-plasma acceleration
- ECloud in static 3d structures

ST)

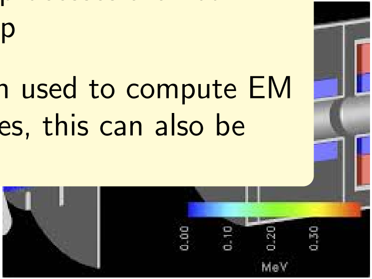
But:

- Secondary emission processes are not implemented in Warp
- Warp has never been used to compute EM fields in RF structures, this can also be handled externally

Geo

any

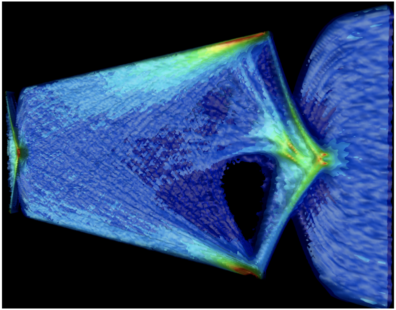
Multi-CPU parallel simulations (MPI)



Warp-POSINST

In the past Warp has been used to simulate the electron cloud in **static structures** jointly with POSINST.

Courtesy of J-L. Vay

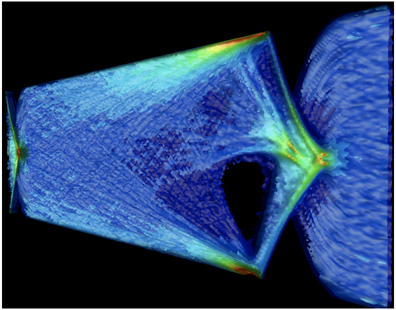


Warp-POSINST

In the past Warp has been used to simulate the electron cloud in **static structures** jointly with POSINST.

POSINST is a Fortran package, developed at LBNL by M. Furman, which has the same functionalities as PyECLOUD. The two codes have been benchmarked against each other[1].

Courtesy of J-L. Vay



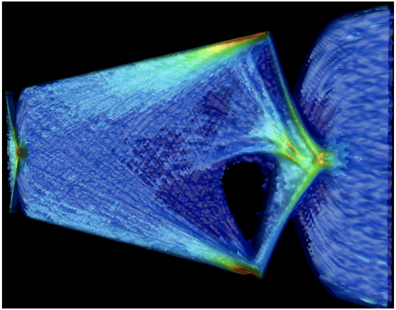
Warp-POSINST

[1] E. Wulff, G. Iadarola
https://cds.cern.ch/record/2683285

In the past Warp has been used to simulate the electron cloud in **static structures** jointly with POSINST.

POSINST is a Fortran package, developed at LBNL by M. Furman, which has the same functionalities as PyECLOUD. The two codes have been benchmarked against each other[1].

Courtesy of J-L. Vay

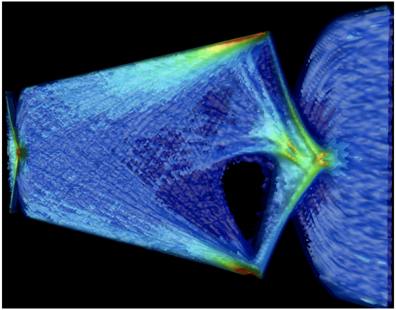


Warp-POSINST

[1]E. Wulff, G. Iadarola
https://cds.cern.ch/record/2683285

In the past Warp has been used to simulate the electron cloud in **static structures** jointly with POSINST.

Courtesy of J-L. Vay



POSINST is a Fortran package, developed at LBNL by M. Furman, which has the same functionalities as PyECLOUD. The two codes have been benchmarked against each other[1].

Posinst is an old code:

- the future of POSINST is unclear
- it is not very readable
- Warp-POSINST coupling is difficult to maintain

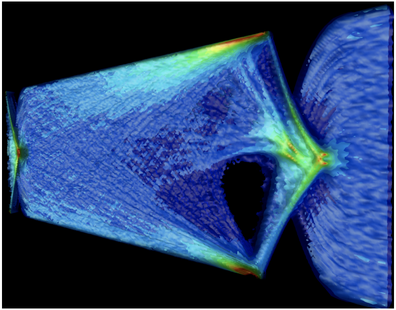
Warp-POSINST

[1] E. Wulff, G. Iadarola
https://cds.cern.ch/record/2683285

In the past Warp has been used to simulate the electron cloud in **static structures** jointly with POSINST.

POSINST is a Fortran package, developed at LBNL by M. Furman, which has the same functionalities as PyECLOUD. The two codes have been benchmarked against each other[1].

Courtesy of J-L. Vay



Posinst is an old code:

- the future of POSINST is unclear
- it is not very readable
- Warp-POSINST coupling is difficult to maintain

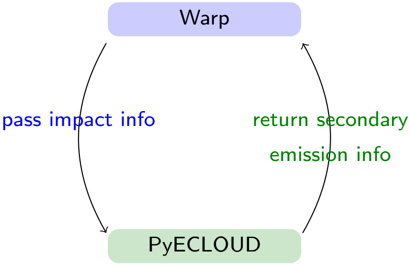
We can use PyECLOUD instead!

Outline

Warp-PyECLOUD

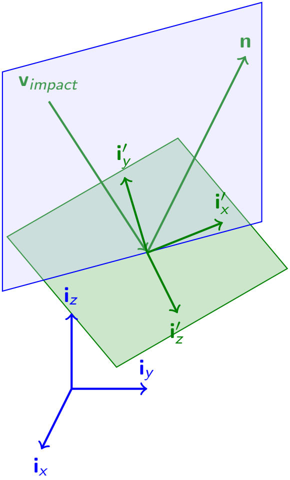
Warp has a Python wrapper from which we can call routines belonging to other Python packages.

We decided to exploit this feature to interface Warp with PyECLOUD.



The interface has been developed by L. Giacomel, G. Iadarola, J-L Vay during Gianni's visit at LBNL in October.

Warp-PyECLOUD interface



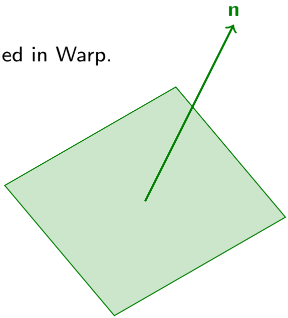
The interface which we wrote implements the transformation between the two reference systems

$$\{\mathbf{i}_x, \mathbf{i}_y, \mathbf{i}_z\} \rightarrow \text{Warp}$$
$$\{\mathbf{i}'_x, \mathbf{i}'_y, \mathbf{i}'_z\} \rightarrow \text{PyECLOUD}$$

We report the transformation in the next slides (also for documentation purposes)

Warp-PyECLOUD Interface

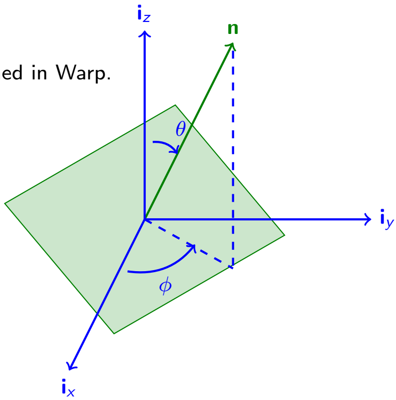
We want to find a normal plane which is easily defined in Warp.



Warp-PyECLOUD Interface

We want to find a normal plane which is easily defined in Warp.

For every impact Warp returns (ϕ,θ)

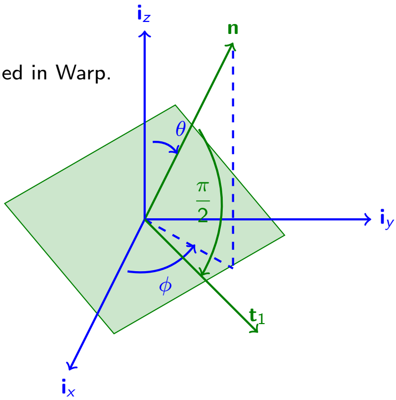


Warp-PyECLOUD Interface

We want to find a normal plane which is easily defined in Warp.

For every impact Warp returns (ϕ,θ), in terms of which the normal vector and a tangential vector are defined respectively as

$$\mathbf{n} = \begin{bmatrix} \cos(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) \\ \cos(\theta) \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos\theta) \\ -\sin(\theta) \end{bmatrix}.$$

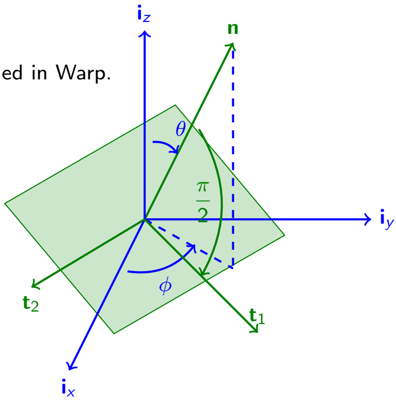


Warp-PyECLOUD Interface

We want to find a normal plane which is easily defined in Warp.

For every impact Warp returns (ϕ,θ), in terms of which the normal vector and a tangential vector are defined respectively as

$$\mathbf{n} = \begin{bmatrix} \cos(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) \\ \cos(\theta) \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos\theta) \\ -\sin(\theta) \end{bmatrix}.$$



The tangential vector perpendicular to both is

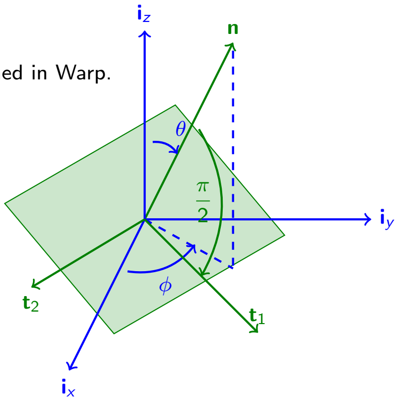
$$\mathbf{t}_2 = \mathbf{t}_1 \times \mathbf{n}.$$

Warp-PyECLOUD Interface

We want to find a normal plane which is easily defined in Warp.

For every impact Warp returns (ϕ,θ), in terms of which the normal vector and a tangential vector are defined respectively as

$$\mathbf{n} = \begin{bmatrix} \cos(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) \\ \cos(\theta) \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos\theta) \\ -\sin(\theta) \end{bmatrix}.$$



The tangential vector perpendicular to both is

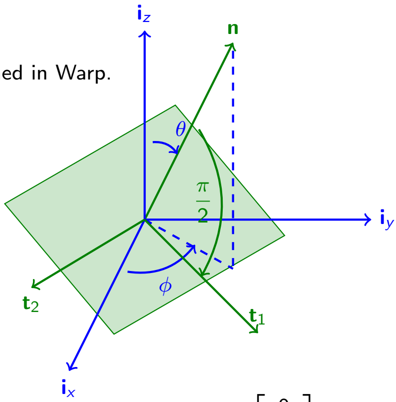
$$\mathbf{t}_2 = \mathbf{t}_1 \times \mathbf{n}.$$

Then in PyECLOUD we use the frame $\{\mathbf{i}'_x, \mathbf{i}'_y, \mathbf{i}'_z\} = \{\mathbf{t}_1, \mathbf{n}, \mathbf{t}_2\}$

Warp-PyECLOUD Interface

We want to find a normal plane which is easily defined in Warp.

For every impact Warp returns (ϕ,θ), in terms of which the normal vector and a tangential vector are defined respectively as



$$\mathbf{n} = \begin{bmatrix} \cos(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) \\ \cos(\theta) \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos\theta) \\ -\sin(\theta) \end{bmatrix}.$$

The tangential vector perpendicular to both is

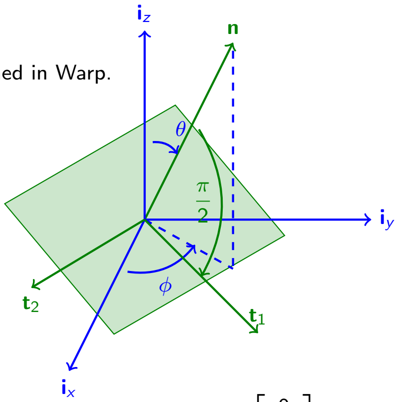
$$\mathbf{t}_2 = \mathbf{t}_1 \times \mathbf{n}.$$

Then in PyECLOUD we use the frame $\{\mathbf{i}'_x, \mathbf{i}'_y, \mathbf{i}'_z\} = \{\mathbf{t}_1, \mathbf{n}, \mathbf{t}_2\}$ and we have $\mathbf{n}' = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

Warp-PyECLOUD Interface

We want to find a normal plane which is easily defined in Warp.

For every impact Warp returns (ϕ,θ), in terms of which the normal vector and a tangential vector are defined respectively as



$$\mathbf{n} = \begin{bmatrix} \cos(\phi)\sin(\theta) \\ \sin(\phi)\sin(\theta) \\ \cos(\theta) \end{bmatrix}, \mathbf{t}_1 = \begin{bmatrix} \cos(\phi)\cos(\theta) \\ \sin(\phi)\cos\theta) \\ -\sin(\theta) \end{bmatrix}.$$

The tangential vector perpendicular to both is

$$\mathbf{t}_2 = \mathbf{t}_1 \times \mathbf{n}.$$

Then in PyECLOUD we use the frame $\{\mathbf{i}'_x, \mathbf{i}'_y, \mathbf{i}'_z\} = \{\mathbf{t}_1, \mathbf{n}, \mathbf{t}_2\}$ and we have $\mathbf{n}' = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

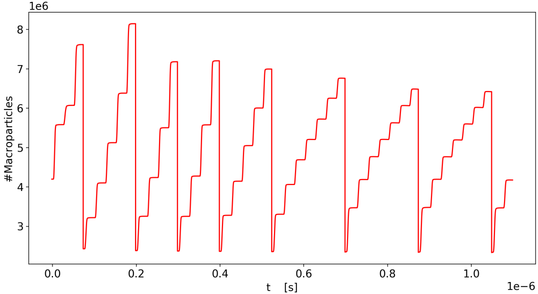
Transformation of velocities:
$$v'_x = \mathbf{v} \cdot \mathbf{i}'_\mathbf{x}$$
$$v'_y = \mathbf{v} \cdot \mathbf{i}'_\mathbf{y}$$
$$v'_z = \mathbf{v} \cdot \mathbf{i}'_\mathbf{z}$$

Additionally Implemented Features

- **Checkpointing**: the user can decide to dump the state of the simulation after the bunch passage. The simulation can then be restarted from the checkpoint;

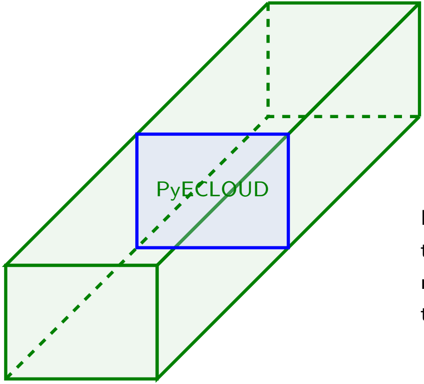
Additionally Implemented Features

- **Checkpointing**: the user can decide to dump the state of the simulation after the bunch passage. The simulation can then be restarted from the checkpoint;
- **Regenerations**: during the buildup the number of MPs increases exponentially. When the number of MPs becomes too high, a fraction of the MPs is killed and the weight of the others is increased accordingly;



Warp-PyECLOUD benchmark

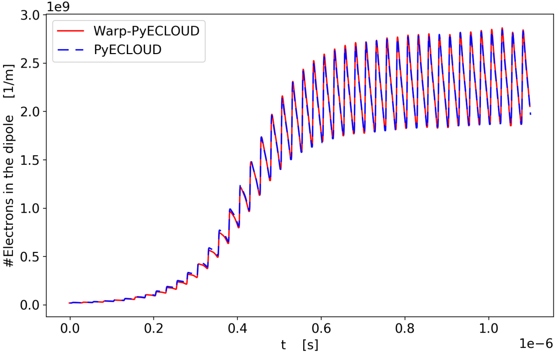
We use as a benchmark case a one-meter-long dipole with rectangular beam pipe.



In this case, we can reasonably compare the PyECLOUD 2D simulation with the middle section of the Warp 3D simulation.

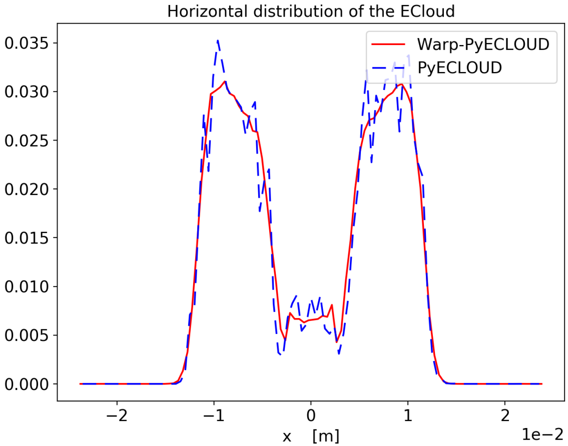
Warp-PyECLOUD benchmark

We compare the number of electrons per meter of dipole.



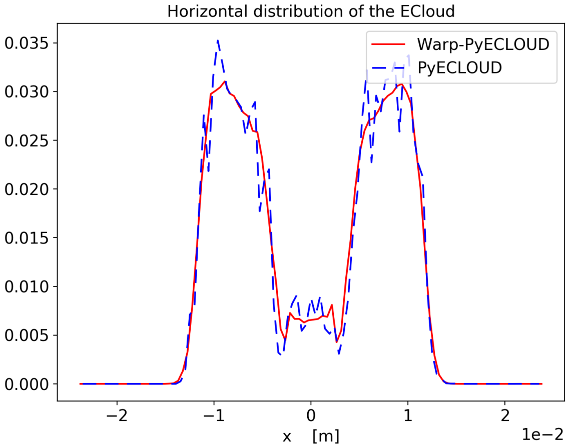
Warp-PyECLOUD benchmark (cont'd)

Comparing a 2D PyECLOUD simulation with a 3D simulation in a one-meter-long rectangular dipole magnet.



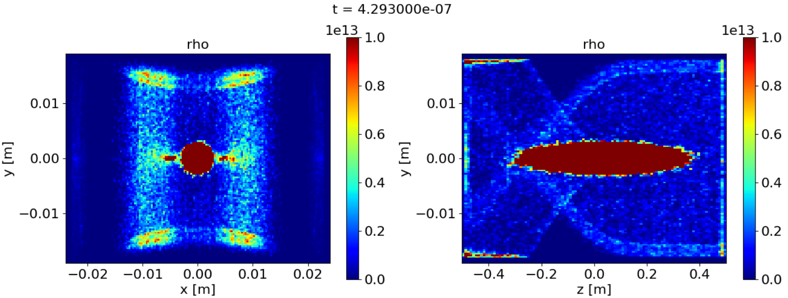
Warp-PyECLOUD benchmark (cont'd)

Comparing a 2D PyECLOUD simulation with a 3D simulation in a one-meter-long rectangular dipole magnet.



Warp results are less noisy because they consist of the averaging of many sections of the dipole.

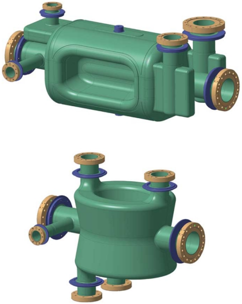
3D Pinch

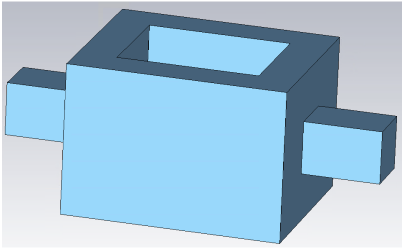

t = 4.293000e-07
rho
1e13
y [m]
x [m]
0.01
0.00
−0.01
−0.02
−0.01
0.00
0.01
0.02
1.0
0.8
0.6
0.4
0.2
0.0
rho
1e13
y [m]
z [m]
0.01
0.00
−0.01
−0.4
−0.2
0.0
0.2
0.4
1.0
0.8
0.6
0.4
0.2
0.0

Outline

Simplification of the Crab Cavities

The structure of the Crab Cavities has been heavily simplified to carry out some preliminary simulations (the electromagnetic solver of Warp has troubles with curved boundaries).



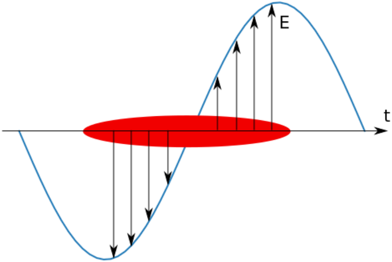


RF Fields

In an RF cavity, such as the Crab Cavities, the particles are pushed by a high frequency electromagnetic field. In the crab cavities the frequency is 400MHz.

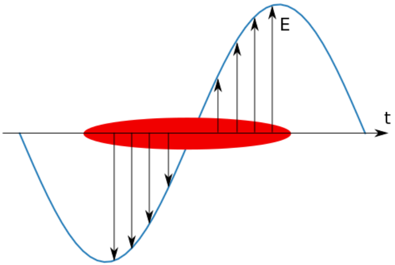
RF Fields

In an RF cavity, such as the Crab Cavities, the particles are pushed by a high frequency electromagnetic field. In the crab cavities the frequency is 400MHz.



RF Fields

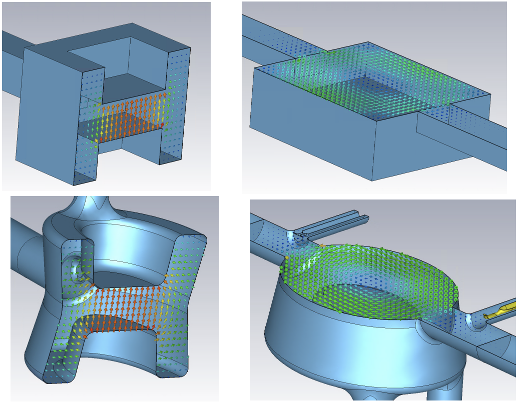
In an RF cavity, such as the Crab Cavities, the particles are pushed by a high frequency electromagnetic field. In the crab cavities the frequency is 400MHz.



The simulation of these fields require an electromagnetic solver. For now the fields can be computed externally using CST and imported into Warp. In the future the fields will be computed directly in Warp.

Fundamental Mode

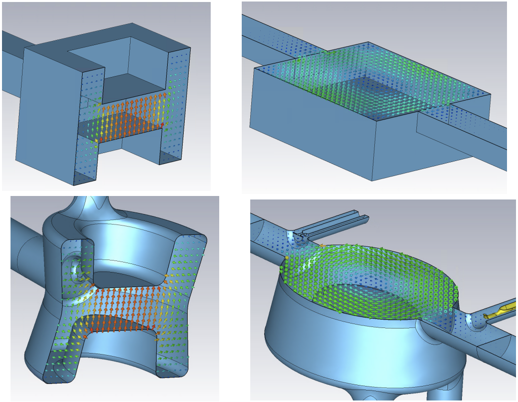
We can carry out eigenmode simulations to qualitatively check that fields of the squared cavity is similar to that of the DQW cavity and to tune the frequency.



Many thanks
B. Salvant and
C. Zannini

Fundamental Mode

We can carry out eigenmode simulations to qualitatively check that fields of the squared cavity is similar to that of the DQW cavity and to tune the frequency.



Many thanks B. Salvant and C. Zannini

Moreover, the the fields computed in the frequency domain can be converted to time domain and imported into Warp.

Import the Fields into Warp

We need to pass from frequency domain to time domain

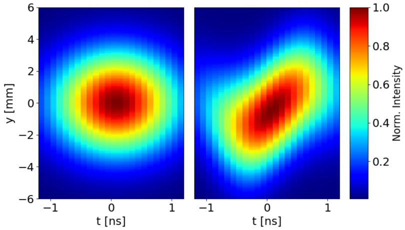
$$\tilde{\mathbf{F}}(\mathbf{x}, t) = \mathrm{Re}[\mathbf{F}(\mathbf{x}) e^{i\omega_{RF} t}] = \mathrm{Re}[\mathbf{F}(\mathbf{x})] \cos(\omega_{RF} t) - \mathrm{Im}[\mathbf{F}(\mathbf{x})] \sin(\omega_{RF} t)$$

```
### Import the fields as vectors
[x,y,z,ReEx,ReEy,ReEz] = getdatafromtextfile("ReE_field.txt",dims=[6,None])
[_,_,_,ImEx,ImEy,ImEz] = getdatafromtextfile("ImE_field.txt",dims=[6,None])

### Map to 3D arrays and interpolate to cell centers
[ReExx,ReEyy,ReEzz,ImExx,ImEyy,ImEzz] =
    preprocessFields(ReFx,ReFy,ReFz, ImFx,ImFy,ImFz)

### Create time vectors
time_array = linspace(0.,Tf,Nt)
sin_array = sin(time_array*freq*2*pi)
cos_array = cos(time_array*freq*2*pi)

### Create overlapped lattice elements for ReE and ImE
addnewegrd(...,time=time_array,data=cos_array,ex=ReExx,ey=ReEyy,ez=ReEzz)
addnewegrd(...,time=time_array,data=sin_array,bx=ImExx,by=ImEyy,bz=ImEzz)

### Repeat for B
```

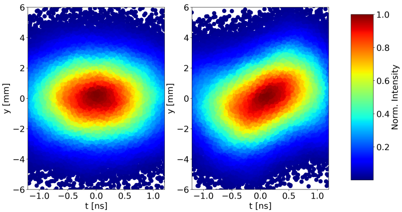
Tilting bunches

We verified that the squared cavity can crab a bunch similarly to the real DQW cavities tested in the SPS. This shows that we are injecting the bunches with the right phase.

SPS tests:



CST+Warp:



Warp-PyECLOUD-CST

Warp

Initialization:

- set up the domain

Warp-PyECLOUD-CST

Warp

Initialization:

- set up the domain
- background electrons MPs distribution

Warp-PyECLOUD-CST

Warp

Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields

read field maps

field.txt

export field maps

CST

EM 3d simulation

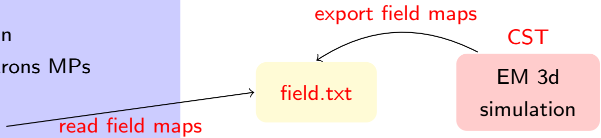
Warp-PyECLOUD-CST

Warp

Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields

For every time step:



Warp-PyECLOUD-CST

Warp

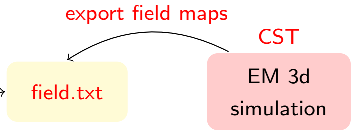
Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields

read field maps

For every time step:

- inject protons (or not)



Warp-PyECLOUD-CST

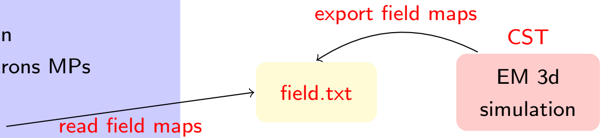
Warp

Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields

For every time step:

- inject protons (or not)
- compute MPs self-fields

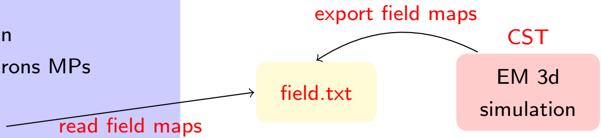


Warp-PyECLOUD-CST

Warp

Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields



For every time step:

- inject protons (or not)
- compute MPs self-fields
- superimpose RF fields

Warp-PyECLOUD-CST

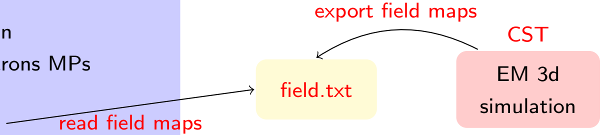
Warp

Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields

For every time step:

- inject protons (or not)
- compute MPs self-fields
- superimpose RF fields
- push MPs



Warp-PyECLOUD-CST

Warp

Initialization:

- set up the domain
- background electrons MPs distribution
- import RF fields

For every time step:

- inject protons (or not)
- compute MPs self-fields
- superimpose RF fields
- push MPs
- detect impact of electrons against the wall

read field maps

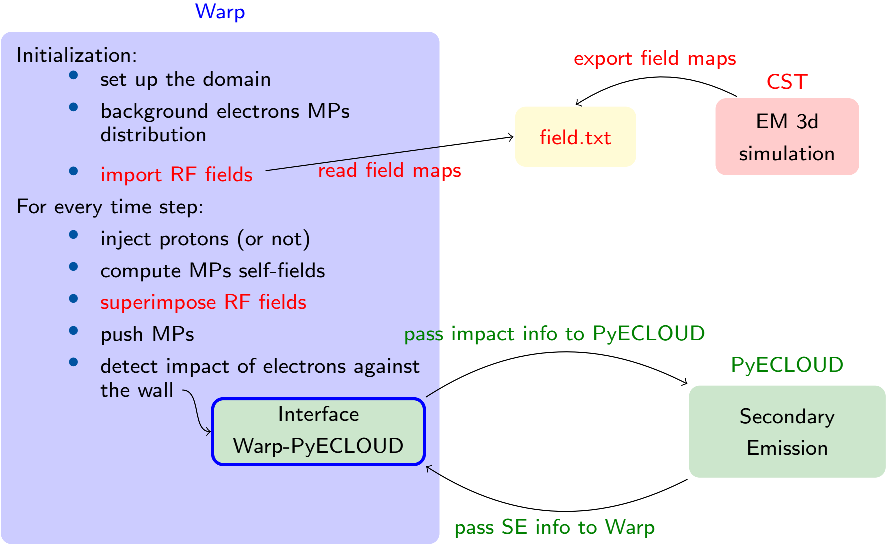
field.txt

export field maps

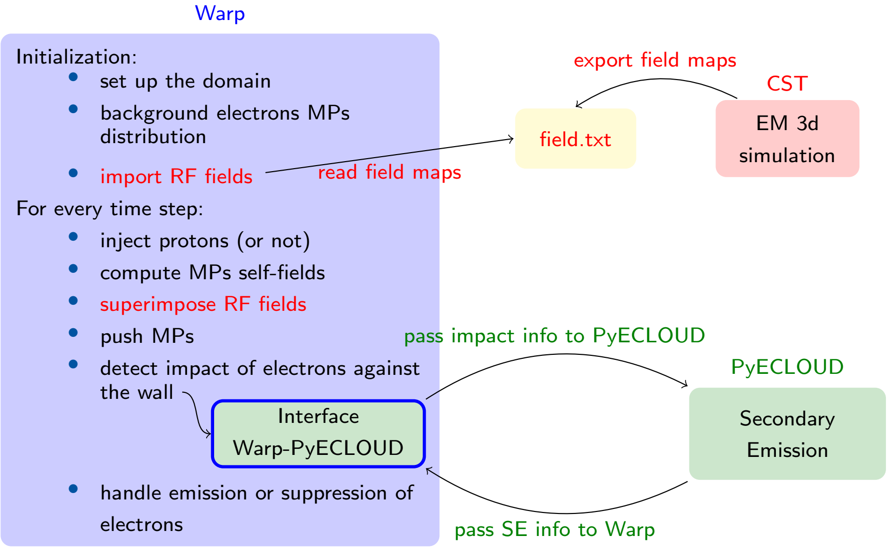
CST

EM 3d simulation

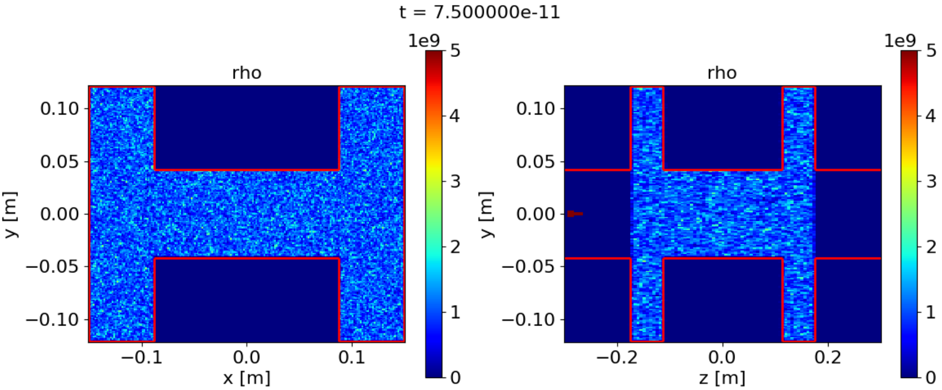
Warp-PyECLOUD-CST



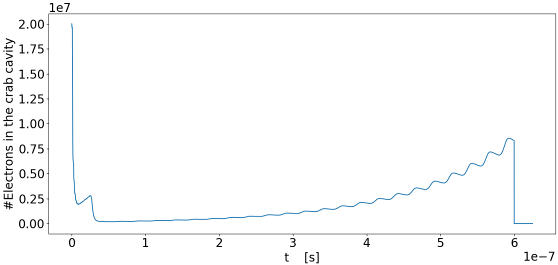
Warp-PyECLOUD-CST



Simulation in the Cavity



Buildup Curve

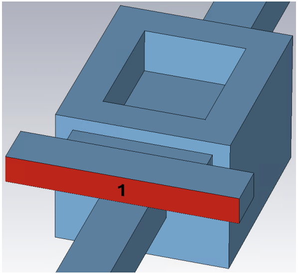


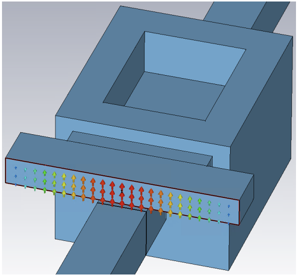
The electrons are pushed by the cavity with an energy that doesn't allow multipacting, therefore most of them are absorbed by the wall during the first RF period. On the other hand, some electrons are pushed to the adjacent drifts giving raise to multipacting in there.

Outline

Time Domain Simulations in the Crab Cavities

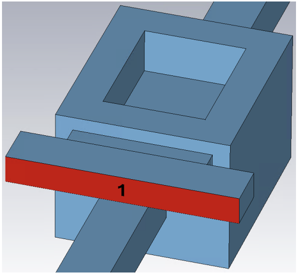
In CST the cavity can be fed with a simple rectangular Waveguide Port.

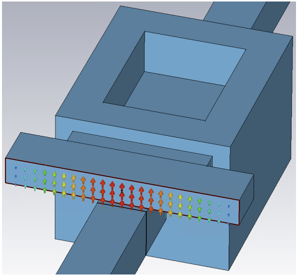




Time Domain Simulations in the Crab Cavities

In CST the cavity can be fed with a simple rectangular Waveguide Port.



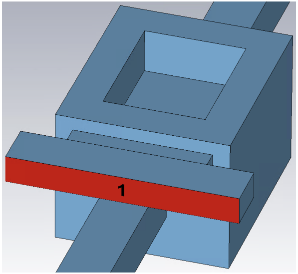


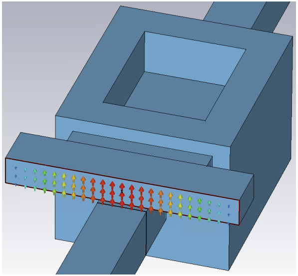
What happens in CST?

- Solve a 2D eigenmode problem to determine the field on the Waveguide Port
- Use the 2D fields as time-dependent boundary condition

Time Domain Simulations in the Crab Cavities

In CST the cavity can be fed with a simple rectangular Waveguide Port.





What happens in CST?

- Solve a 2D eigenmode problem to determine the field on the Waveguide Port
- Use the 2D fields as time-dependent boundary condition

Warp has no eigenmode solver. We need a different strategy.

Computation of the RF fields in Warp

Useful feature: **laser antenna**

Computation of the RF fields in Warp

Useful feature: **laser antenna**

Used in laser-plasma acceleration simulations to inject the laser

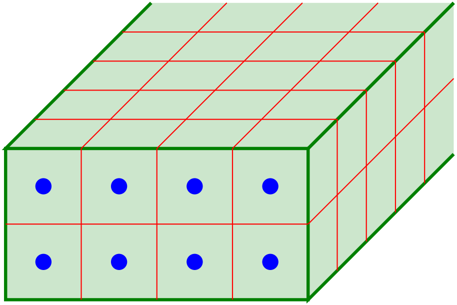
Computation of the RF fields in Warp

Useful feature: **laser antenna**

Used in laser-plasma acceleration simulations to inject the laser

In addition to the standard PIC iteration:

- Place a macro-particle with unitary charge in each mesh cell of a given region
- Assign the macro-particles a velocity
- Scatter current to the mesh
- Compute the EM field in the whole space



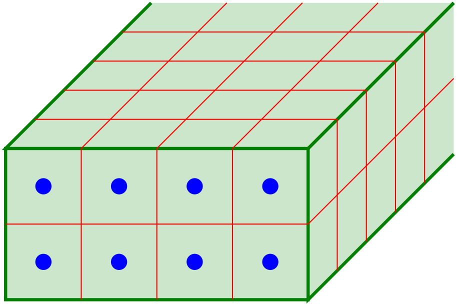
Computation of the RF fields in Warp

Useful feature: **laser antenna**

Used in laser-plasma acceleration simulations to inject the laser

In addition to the standard PIC iteration:

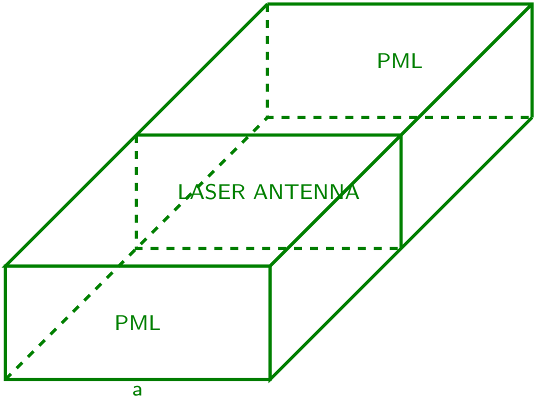
- Place a macro-particle with unitary charge in each mesh cell of a given region
- Assign the macro-particles a velocity
- Scatter current to the mesh
- Compute the EM field in the whole space



By choosing properly the velocity distribution of the macro-particles we can excite a specific mode of a waveguide.

TE10 in a Rectangular Waveguide

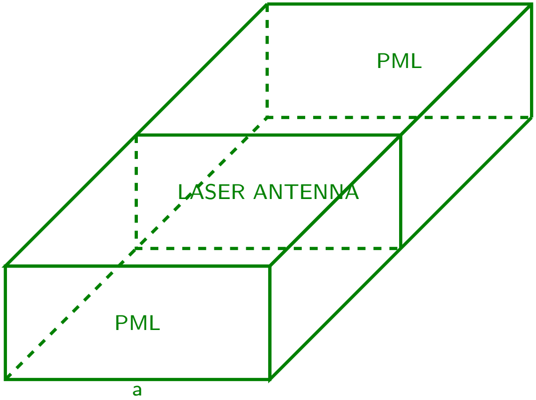
Numerical experiment: excite a TE10 mode in a rectangular waveguide.



TE10 in a Rectangular Waveguide

Numerical experiment: excite a TE10 mode in a rectangular waveguide.

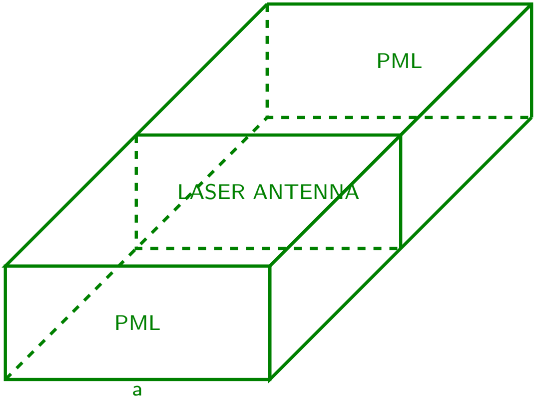
- Perfectly Matched Layers (PMLs) at the open boundaries



TE10 in a Rectangular Waveguide

Numerical experiment: excite a TE10 mode in a rectangular waveguide.

- Perfectly Matched Layers (PMLs) at the open boundaries
- PEC all around



TE10 in a Rectangular Waveguide

Numerical experiment: excite a TE10 mode in a rectangular waveguide.

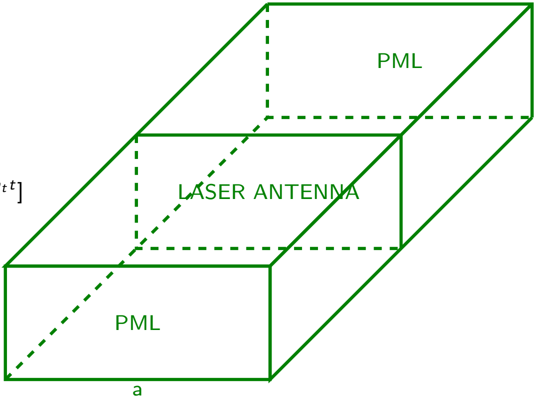
- Perfectly Matched Layers (PMLs) at the open boundaries
- PEC all around
- Field on the antenna plane:

$$\mathbf{E}(\mathbf{x}) = \mathrm{Re}[E_0 \sin \frac{\pi x}{a} e^{-i\omega_z z} e^{-i\omega_t t}]$$

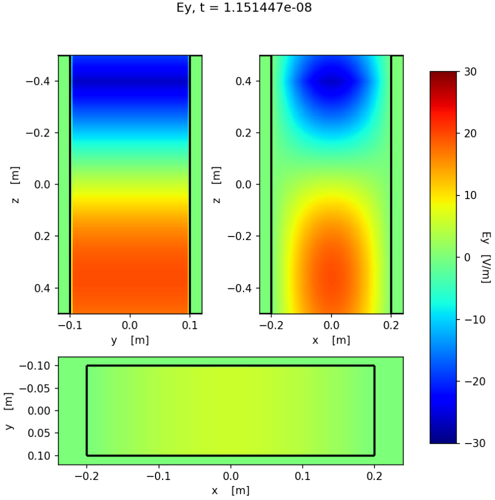
$$\omega_t = 2\pi f$$

$$f = 400 MHz$$

$$\omega_z = c\sqrt{\left(\frac{\omega_t}{c}\right)^2 - \left(\frac{\pi}{a}\right)^2}$$



TE10 in a Rectangular Waveguide - Results



Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity
- carry out **self-consistent** simulations

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity
- carry out **self-consistent** simulations

Some new challenges are already on for the future:

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity
- carry out **self-consistent** simulations

Some new challenges are already on for the future:

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity
- carry out **self-consistent** simulations

Some new challenges are already on for the future:

- 3D simulations produce huge amount of data.
  How can we handle them in order to enable
  post-processing?

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity
- carry out **self-consistent** simulations

Some new challenges are already on for the future:

- 3D simulations produce huge amount of data.
  How can we handle them in order to enable **post-processing**?
- Can we improve Warp electromagnetic solvers to resolve correctly **curved boundaries**?

Future Steps

We have all the fundamental tools we need to accomplish self-consistent simulations in RF structures...

...but we still need to:

- study more deeply the **transient behavior** of the rectangular waveguide
- plug the rectangular waveguide to the cavity
- carry out **self-consistent** simulations

Some new challenges are already on for the future:

- 3D simulations produce huge amount of data. How can we handle them in order to enable **post-processing**?
- Can we improve Warp electromagnetic solvers to resolve correctly **curved boundaries**?

CERN ETH zürich

home.cern