



3D Electromagnetic Time-Domain wake and impedance solver

14th International Computational Accelerator Physics Conference
Lufthansa Seeheim Conference Hotel, Seeheim-Jugenheim, Germany 2-5 October 2024

Elena de la Fuente^{1,2}

Lorenzo Giacomel¹, Giovanni Iadarola¹, Carlo Zannini¹, Manuel Cotelo²

¹CERN, Geneva Switzerland

²IFN-GV, Polytechnic University of Madrid, Madrid, Spain

Outline

1. Introduction to Beam-Coupling Impedance simulations Slides 1-4
2. **wakis** code framework and architecture Slides 5-7
3. The Finite Integration Technique (FIT): Slides 8-17
 - 3.1. Numerical Algorithm in free-space
 - 3.2. Geometry definition: STL importer
 - 3.3. Materials ε, μ, σ and EM sources
 - 3.4. Wake potential and impedance
 - 3.5. Low- β simulations
4. Conclusions, Present & Future work Slides 18-19

Outline

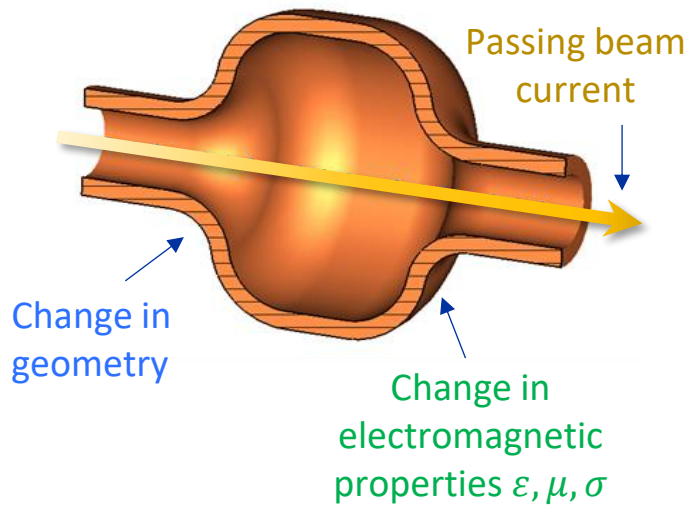
1. Introduction to Beam-Coupling Impedance simulations Slides 1-4
2. *wakis* code framework and architecture Slides 5-7
3. The Finite Integration Technique (FIT): Slides 8-17
 - 3.1. Numerical Algorithm in free-space
 - 3.2. Geometry definition: STL importer
 - 3.3. Materials ε, μ, σ and EM sources
 - 3.4. Wake potential and impedance
 - 3.5. Low- β simulations
4. Conclusions, Present & Future work Slides 18-19

Beam-coupling impedance

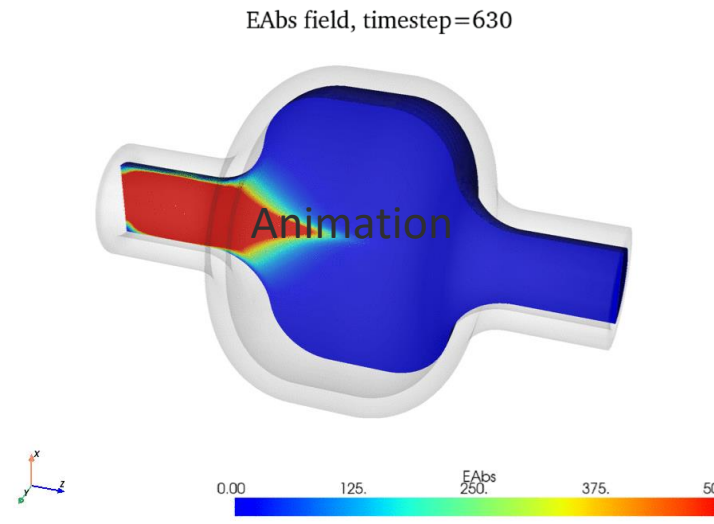
- Electromagnetic wakefields are generated as the **particle beam** traverses the different accelerator devices with **discontinuities in the geometry** (cavities, transitions...) or the **electromagnetic properties** (ϵ, μ, σ). These wakefields will affect the trailing particles/bunches and can **generate instabilities**
- The beam-coupling impedance Z is a **frequency-dependent property of each accelerator device**, used to quantify the wakefields' effects

$$Z_{\parallel}(\omega) = - \frac{\int_{-\infty}^{\infty} W_{\parallel}(s) e^{-i\omega s} ds}{\int_{-\infty}^{\infty} \beta c \lambda(s) e^{-i\omega s} ds}$$

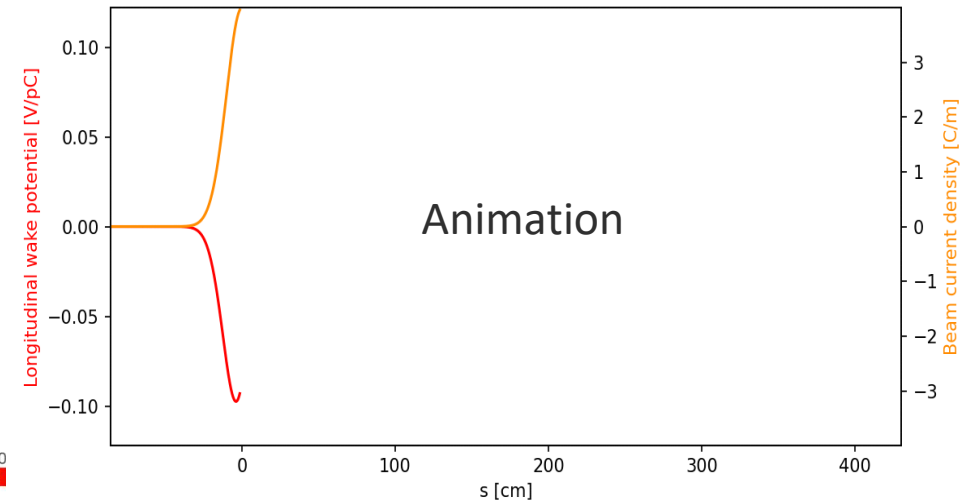
Accelerator device



Electromagnetic fields



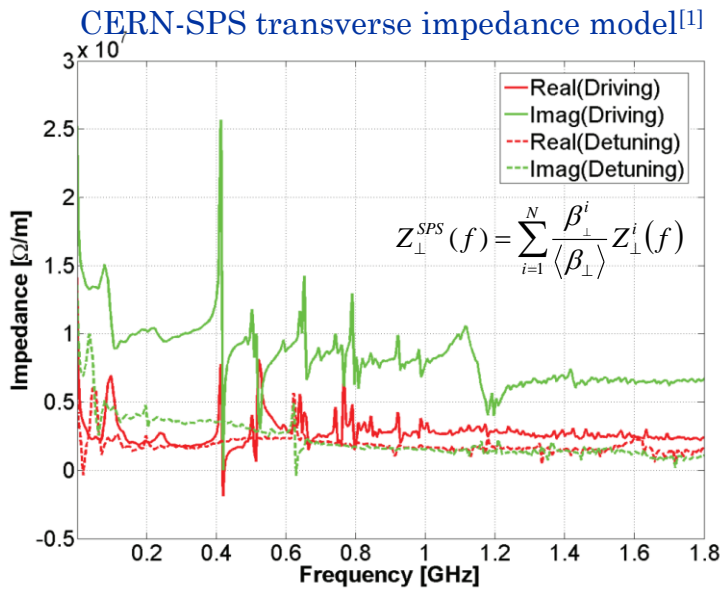
Wake potential & Impedance



Importance of beam-coupling impedance

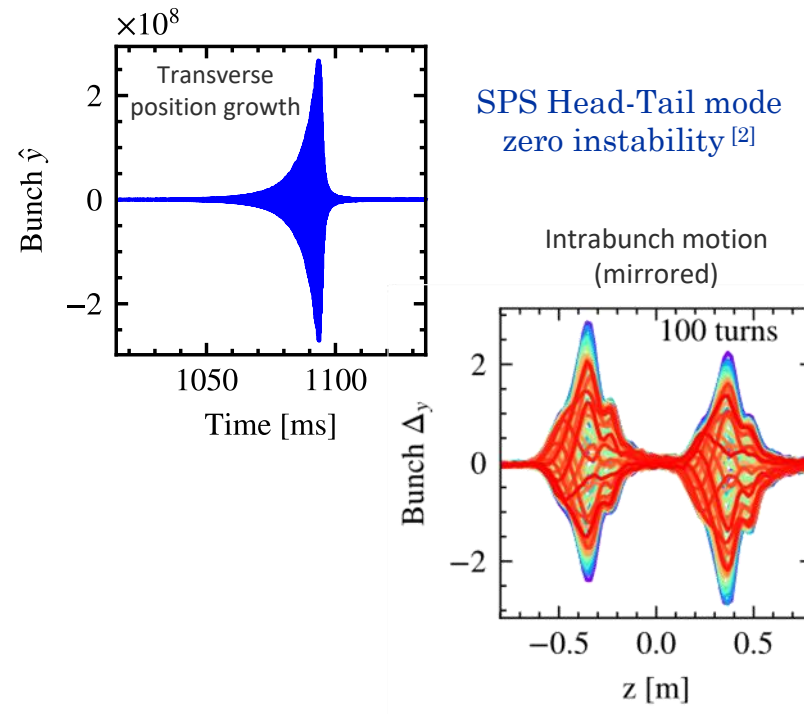
Impedance model of the accelerator

The impedance of all relevant accelerator components is gathered in each accelerator's impedance model



Beam stability

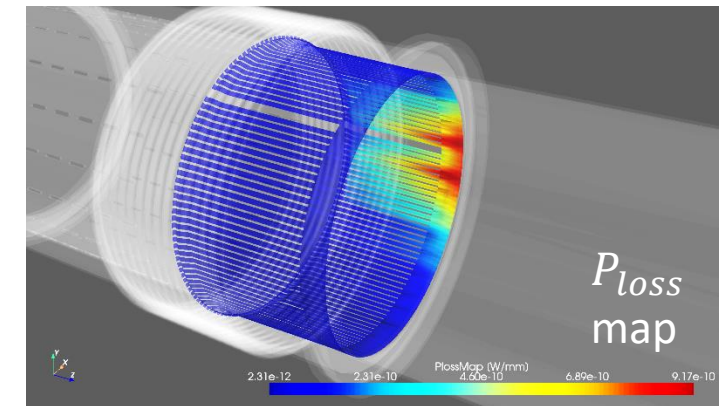
And used in beam-dynamics codes (e.g., pyHeadtail, Xsuite) to predict beam behaviour and stability



Beam induced heating

Assess beam-induced heating of individual accelerator components and propose mitigation solutions

LHC Warm Vacuum modules Power lost map^[3]



How to compute beam-coupling impedance?

Analytical calculations



Beam-coupling impedance can be derived analytically for **simple geometries and material configurations**^[4]:

- e.g., cylindrical pillbox cavities, pipe transitions, single-layer resistive wall

Numerical computations



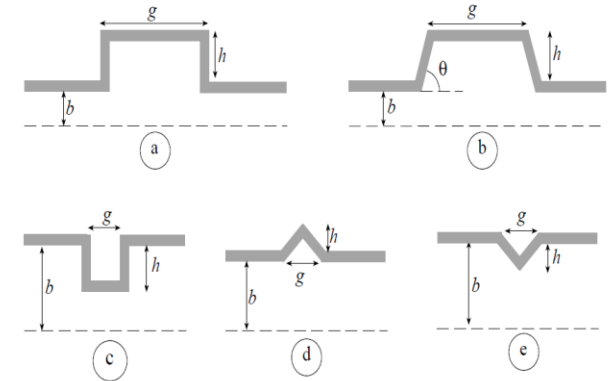
For more complex accelerator devices, the **numerical solution of Maxwell's equations** in frequency or time-domain is required.

- In frequency domain:

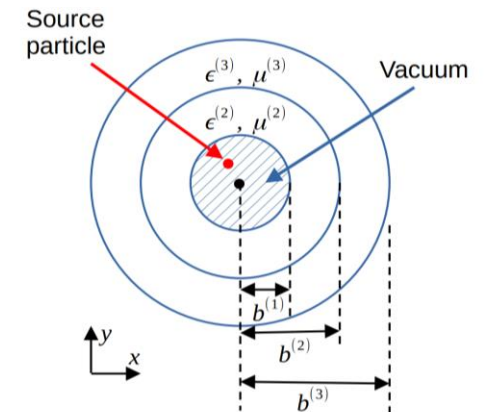
- 2D models for vacuum chamber multi-layered elements e.g., **ImpedanceWake2D**^[5]
- Commercial **CST® Eigenmode solver** → loss-less resonant modes of general 3D structures without excitation + impedance in post-processing

- In time-domain:

- Time-domain allows for **full-wave electromagnetic simulations** including losses and excitation bunch. Several codes available following different approaches e.g.:
 - ABCI (axi-symmetric), CST® Wakefield solver (3D), GdfidL®, PBCI, ECHO-3D (3D, moving window) → **not on an open-source platform**



[4] Wake fields and impedance, L. Palumbo, V.G. Vaccaro, CERN-1995-006.331



[5] ImpedanceWake2D, N. Mounet, TU01C02 HB'10 <https://gitlab.cern.ch/IRIS/IW2D>

About the project:

Project start:

wakis as **post-processing tool**

Computation of **wake potential and impedance** from **pre-computed fields** from other EM solvers

Longitudinal $Z_{||}$ and transverse Z_{\perp} (dipolar & quadrupolar) successfully benchmarked with CST® fields

Towards open-source Wakefield solver:

Coupling post-processing **wakis with open-source PIC-EM solver WarpX^[6]**

Successfully benchmarked lossless pillbox cavity below cutoff^[7]

2023

Exploring FIT mock-up in Python

Implementing the **Finite Integration Technique (FIT) fully in Python** to explore computational viability and benefit from the **open-source community**

Present:

wakis as an open-source **3D time-domain electromagnetic wake and impedance solver** as the foundation to address computational challenges

2022

Don't
reinvent the
wheel!




[6] ECP-WarpX: <https://ecp-warpx.github.io/>

[7] E. de la Fuente, WEPL170 IPAC'23

Outline

1. Introduction to Beam-Coupling Impedance simulations Slides 1-4
2. **wakis** code framework and architecture Slides 5-7
3. The Finite Integration Technique (FIT): Slides 8-17
 - 3.1. Numerical Algorithm in free-space
 - 3.2. Geometry definition: STL importer
 - 3.3. Materials ε , μ , σ and EM sources
 - 3.4. Wake potential and impedance
 - 3.5. Low- β simulations
4. Conclusions, Present & Future work Slides 18-19

Available on GitHub

wakis Public 

Edit Pins Unwatch 1 Fork 5 Starred 8

main 7 Branches 2 Tags

Go to file Add file Code

elenafuengar adding 3d plot <https://github.com/ImpedanCEI/wakis> 423e8ca · 1 hour ago 382 Commits

| | | |
|------------------|----------------------------------------------------------|--------------|
| benchmarks | adding 3d plot | 1 hour ago |
| docs | change logo to high resolution png | last week |
| examples | Clean old tests | 5 days ago |
| tests | add interactive flag and set to False | 4 days ago |
| wakis | new feautres in plot3DonSTL: field on plane capabilities | 2 days ago |
| .gitattributes | remove .sh from language statistics | 5 days ago |
| .gitignore | ignore distribution files | 4 days ago |
| LICENSE | add LICENSE | 4 months ago |
| MANIFEST.in | prepare for deployment | 5 days ago |
| README.md | add license badge | 4 days ago |
| pyproject.toml | change to setup.py to follow xsuite approach | 4 days ago |
| readthedocs.yml | documentation setup | 4 months ago |
| release.sh | change to setup.py to follow xsuite approach | 4 days ago |
| requirements.txt | add pytest to dependencies | 5 days ago |
| setup.py | change to setup.py to follow xsuite approach | 4 days ago |

About

3D electromagnetic time-domain solver, specialized in wake potential and beam-coupling impedance computation for particle accelerators

wakis.readthedocs.io/

gpu time-domain 3d impedance wakefield accelerator-physics electromagnetic-simulation

Readme View license Activity Custom properties 8 stars 1 watching 5 forks Report repository

Releases

2 tags [Create a new release](#)

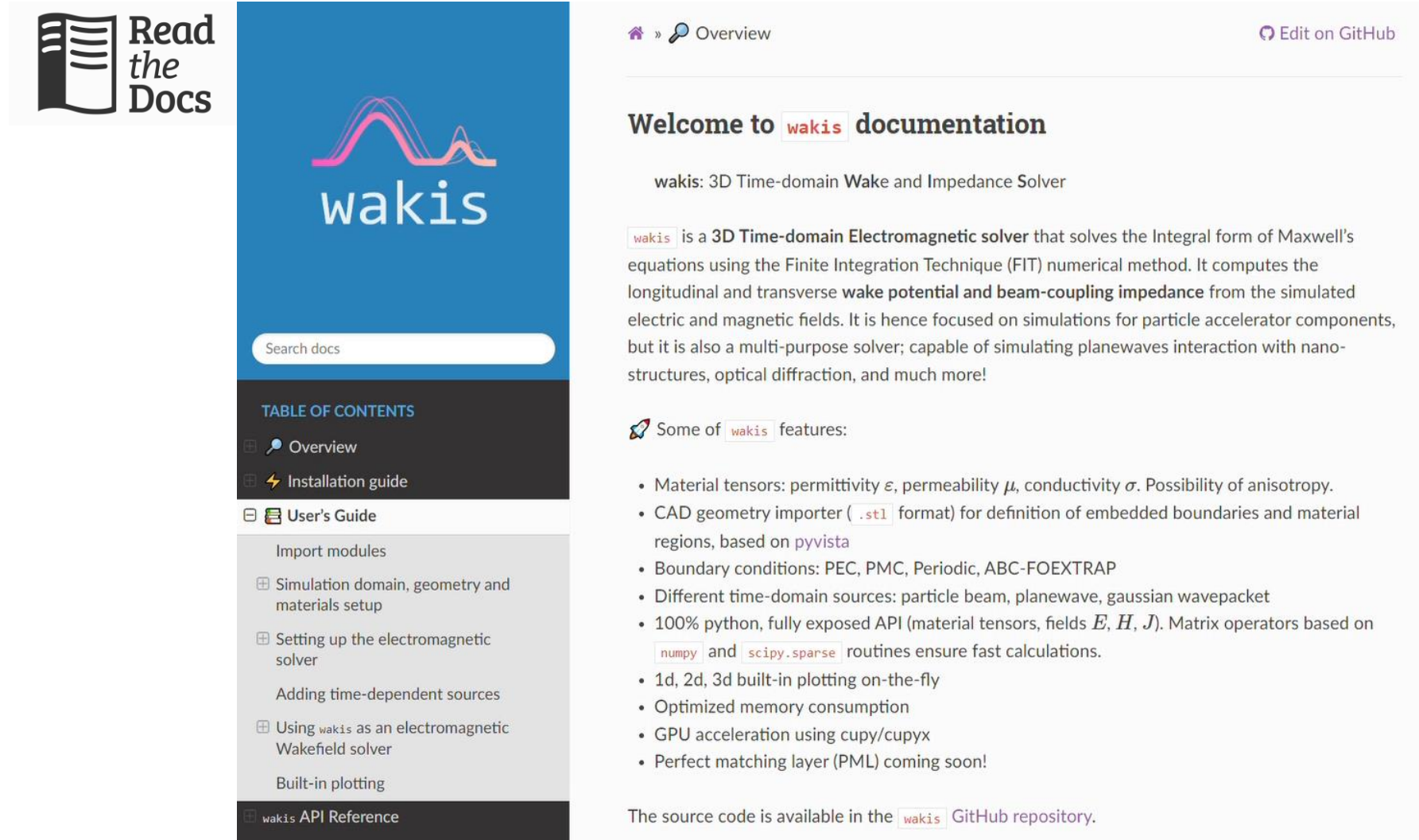
Benchmarks against CST studio®

Simulation examples
⌚ ~2' to 20'

Source code

Documentation on ReadTheDocs

<https://wakis.readthedocs.io/>



Read the Docs

wakis

Search docs

TABLE OF CONTENTS

- Overview
- Installation guide
- User's Guide
 - Import modules
 - Simulation domain, geometry and materials setup
 - Setting up the electromagnetic solver
 - Adding time-dependent sources
 - Using wakis as an electromagnetic Wakefield solver
 - Built-in plotting
- wakis API Reference

Overview

Edit on GitHub

Welcome to wakis documentation

wakis: 3D Time-domain Wake and Impedance Solver

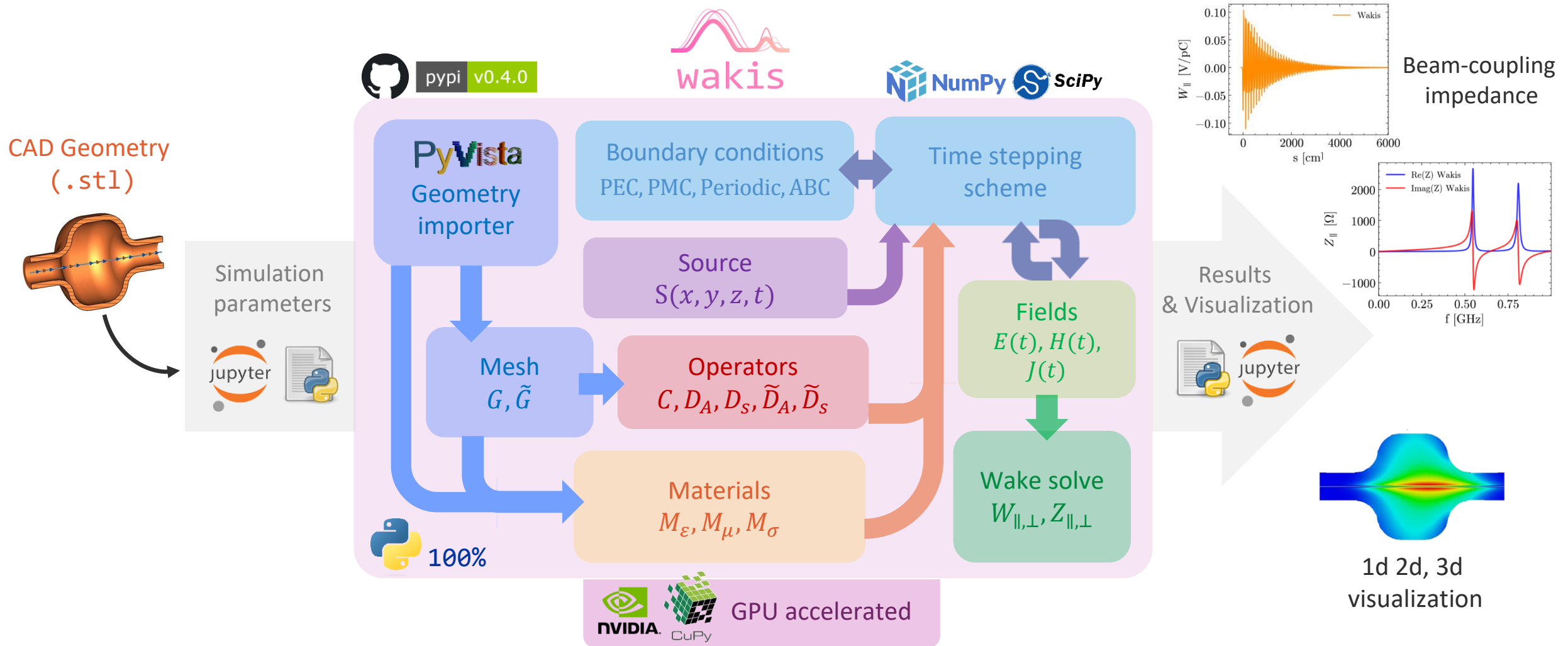
wakis is a 3D Time-domain Electromagnetic solver that solves the Integral form of Maxwell's equations using the Finite Integration Technique (FIT) numerical method. It computes the longitudinal and transverse **wake potential and beam-coupling impedance** from the simulated electric and magnetic fields. It is hence focused on simulations for particle accelerator components, but it is also a multi-purpose solver; capable of simulating planewaves interaction with nano-structures, optical diffraction, and much more!

Some of wakis features:

- Material tensors: permittivity ϵ , permeability μ , conductivity σ . Possibility of anisotropy.
- CAD geometry importer (.stl format) for definition of embedded boundaries and material regions, based on `pyvista`
- Boundary conditions: PEC, PMC, Periodic, ABC-FOEXTRAP
- Different time-domain sources: particle beam, planewave, gaussian wavepacket
- 100% python, fully exposed API (material tensors, fields E, H, J). Matrix operators based on `numpy` and `scipy.sparse` routines ensure fast calculations.
- 1d, 2d, 3d built-in plotting on-the-fly
- Optimized memory consumption
- GPU acceleration using `cupy/cupyx`
- Perfect matching layer (PML) coming soon!

The source code is available in the wakis GitHub repository.

wakis solver architecture



Outline

1. Introduction to Beam-Coupling Impedance simulations Slides 1-4
2. **wakis** code framework and architecture Slides 5-7
- 3. The Finite Integration Technique (FIT):** Slides 8-17
 - 3.1. Numerical Algorithm in free-space
 - 3.2. Geometry definition: STL importer
 - 3.3. Materials ε, μ, σ and EM sources
 - 3.4. Wake potential and impedance
 - 3.5. Low- β simulations
4. Conclusions, Present & Future work Slides 18-19

Examples & benchmarks

Finite Integration Technique

Maxwell Equations (Integral form)

$$\oint_{\partial A} \mathbf{E} \cdot d\mathbf{s} = - \iint_A \frac{\partial \mathbf{B}}{\partial t} \cdot d\mathbf{A}$$

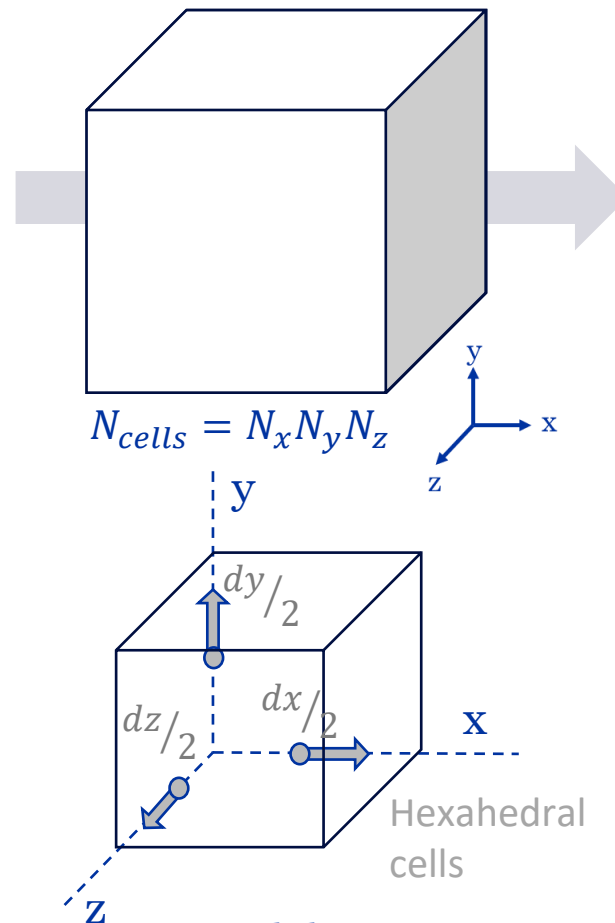
$$\oint_{\partial A} \mathbf{H} \cdot d\mathbf{s} = - \iint_A \left(\frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \right) \cdot d\mathbf{A}$$

$$\oiint_{\partial V} \mathbf{B} \cdot d\mathbf{A} = 0$$

$$\oiint_{\partial V} \mathbf{D} \cdot d\mathbf{A} = \iiint_V \rho dV$$

$$\mathbf{D} = \epsilon \mathbf{E}, \quad \mathbf{B} = \mu \mathbf{H}, \quad \mathbf{J} = \sigma \mathbf{E} + \rho \mathbf{v}$$

1st approximation
Domain discretization
 dx, dy, dz



Maxwell Grid Equations

$$\mathbf{C} \mathbf{D}_s \mathbf{e} = - \mathbf{D}_A \frac{\partial \mathbf{b}}{\partial t}$$

$$\tilde{\mathbf{C}} \tilde{\mathbf{D}}_s \mathbf{h} = \tilde{\mathbf{D}}_A \left(\frac{\partial \mathbf{d}}{\partial t} + \mathbf{j} \right)$$

$$\mathbf{S} \mathbf{D}_A \mathbf{b} = 0$$

$$\tilde{\mathbf{S}} \tilde{\mathbf{D}}_A \left(\frac{\partial \mathbf{d}}{\partial t} + \mathbf{j} \right) = 0$$

$$\mathbf{d} = \tilde{\mathbf{D}}_\epsilon \mathbf{e}, \quad \mathbf{b} = \mathbf{D}_\mu \mathbf{h}, \quad \mathbf{j} = \tilde{\mathbf{D}}_\sigma \mathbf{e} + \mathbf{j}_{src}$$

- Operators
- Grid areas and lengths
- Materials

Finite Integration Technique (II)

Maxwell Grid Equations

$$\begin{aligned} \mathbf{C}\mathbf{D}_s\mathbf{e} &= -\mathbf{D}_A\frac{\partial\mathbf{b}}{\partial t} \\ \tilde{\mathbf{C}}\tilde{\mathbf{D}}_s\mathbf{h} &= \tilde{\mathbf{D}}_A\left(\frac{\partial\mathbf{d}}{\partial t} + \mathbf{j}\right) \\ \mathbf{d} &= \tilde{\mathbf{D}}_\epsilon\mathbf{e}, \quad \mathbf{b} = \mathbf{D}_\mu\mathbf{h}, \\ \mathbf{j} &= \tilde{\mathbf{D}}_\sigma\mathbf{e} + \mathbf{j}_{src} \end{aligned}$$

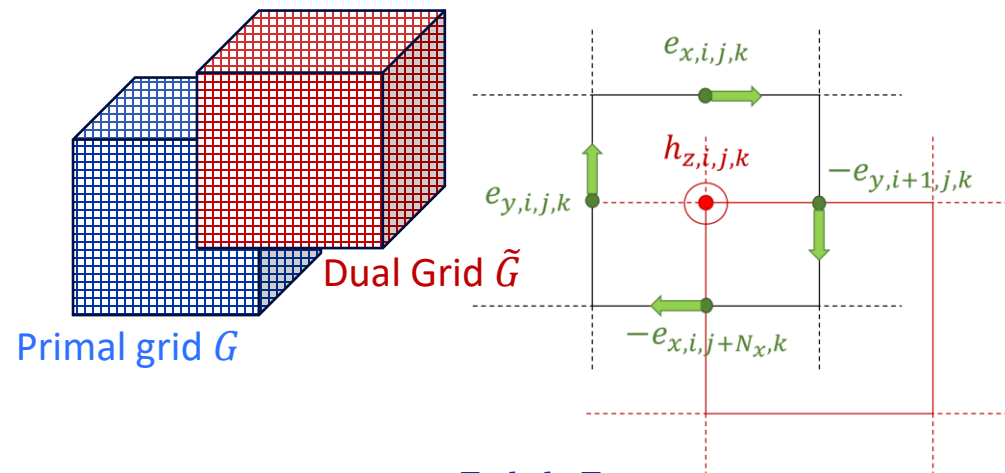
$$\begin{aligned} \mathbf{S}\mathbf{D}_A\mathbf{b} &= \mathbf{0} \\ \tilde{\mathbf{S}}\tilde{\mathbf{D}}_A\left(\frac{\partial\mathbf{d}}{\partial t} + \mathbf{j}\right) &= \mathbf{0} \end{aligned}$$

2nd approximation
Evolution in timestep Δt

$$\frac{\partial a}{\partial t} = \frac{a^{n+1} - a^n}{\Delta t}$$



Domain discretization
With Yee grid



Time-stepping scheme (leapfrog)

$$\begin{aligned} \mathbf{h}^{n+1} &= \mathbf{h}^n - \Delta t \tilde{\mathbf{D}}_s \mathbf{D}_\mu^{-1} \mathbf{D}_A^{-1} \mathbf{C} \mathbf{e}^{n+0.5} \\ \mathbf{e}^{n+1.5} &= \mathbf{e}^{n+0.5} + \Delta t \mathbf{D}_s \tilde{\mathbf{D}}_\epsilon^{-1} \tilde{\mathbf{D}}_A^{-1} \tilde{\mathbf{C}} \mathbf{h}^n - \tilde{\mathbf{D}}_\epsilon^{-1} \mathbf{j}_{src}^n \\ &\quad - \tilde{\mathbf{D}}_\epsilon^{-1} \tilde{\mathbf{D}}_\sigma \mathbf{e}^{n+0.5} \end{aligned}$$

- ✓ 2nd order convergence in time
- ✓ No matrix inversion needed during time loop
- ✓ Can pre-compute most of the quantities: ↑ speed
 - Taste of speed: 20M cells, 22.3 it/s on Titan V GPU, 11.5k MB memory
- ✓ CFL stability condition
- Gauss Laws (for E and H) considered satisfied* $\forall t$ for no charges j_{ext}

Finite Integration Technique (III)

$$h^{n+1} = h^n - \Delta t \tilde{D}_s D_\mu^{-1} D_A^{-1} C e^{n+0.5}$$

$$e^{n+1.5} = e^{n+0.5} + \Delta t D_s \tilde{D}_\varepsilon^{-1} \tilde{D}_A^{-1} \tilde{C} h^n - \tilde{D}_\varepsilon^{-1} j_{src}^n - \tilde{D}_\varepsilon^{-1} \tilde{D}_\sigma e^{n+0.5}$$



Operators

$C, D_A, D_s, \tilde{D}_A, \tilde{D}_s$



Fields

$E(t), H(t), J(t)$



Materials

$D_\varepsilon^{-1}, D_\mu^{-1}, D_\sigma$

- **C, \tilde{C} Curl operator:** $[3N_{\text{cell}} \times 3N_{\text{cell}}]$ sparse matrix that relates the E and H fields and defines PEC, PMC or Periodic boundary conditions (BCs)
- **$D_A, D_s, \tilde{D}_A, \tilde{D}_s$:** $3N_{\text{cell}}$ diagonal matrices containing the grid (primal and dual) discretization in terms of cell lengths and areas

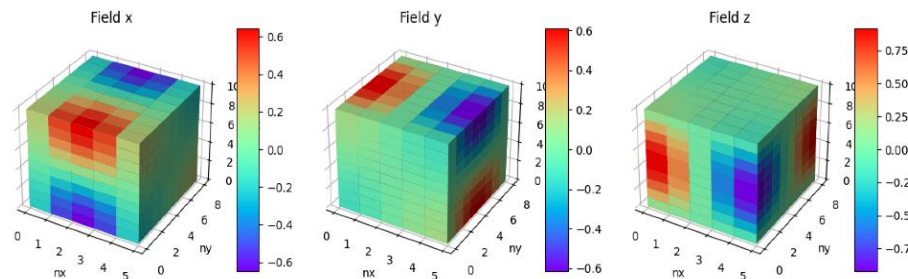
- **$e^{n+1.5}, h^{n+1}, j^n$ Fields:** saved in memory as $[3N_{\text{cell}}]$ lexicographic arrays, updated every simulation timestep.

Fully-exposed, modifications on-the-fly (e.g. addition of sources or initial conditions)

- **$\tilde{D}_\varepsilon^{-1}, D_\mu^{-1}$:** $3N_{\text{cell}}$ diagonal matrices for the anisotropic inverse permittivity ε and permeability μ rank-2 tensors

- **\tilde{D}_σ :** $3N_{\text{cell}}$ diagonal matrix for the anisotropic electric conductivity rank-2 tensor. The current can be computed as: $j^{n+1} = \tilde{D}_\sigma e^{n+0.5} + j_{src}^n$

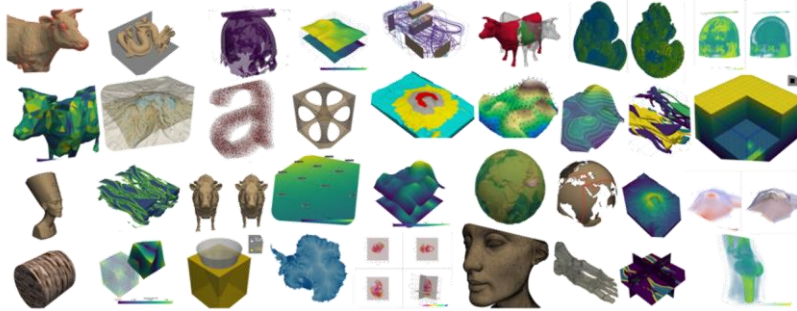
wakis E.inspect3d() method



Importing CAD geometry and visualization

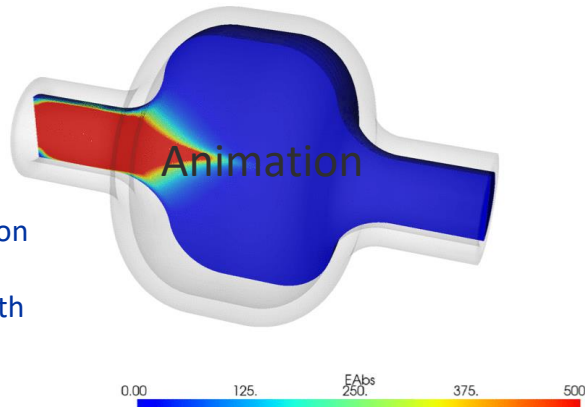
PyVista

3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK)



<https://github.com/pyvista/pyvista>

EAbs field, timestep=630

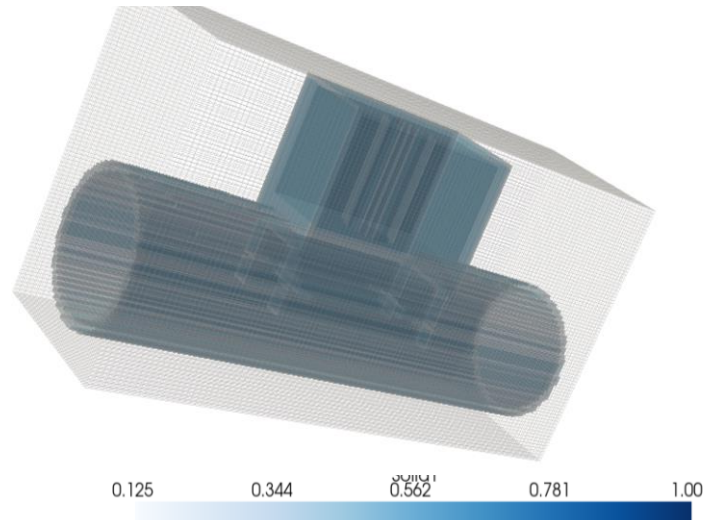


Example: *wakis* simulation of a lossy pillbox cavity plotted every 30 steps with `solver.plot3d()` method

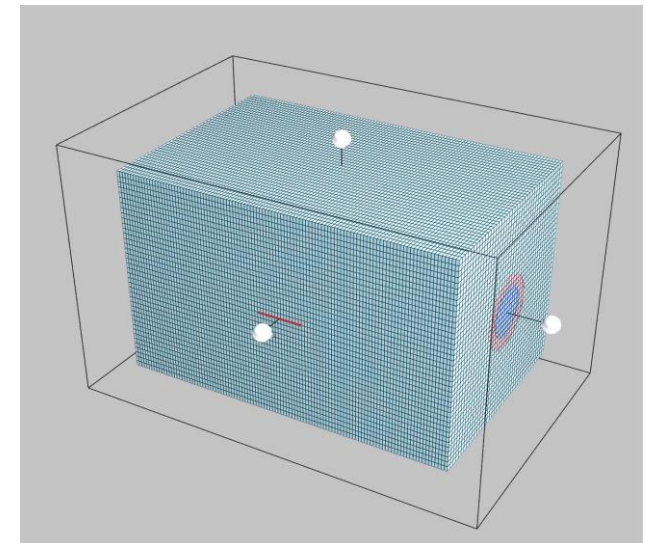
Using *PyVista* capabilities, *wakis* can:

- Import CAD geometry: `geometry = pv.read(CAD_file)`
- Generate the domain grid: `grid = pv.StructuredGrid(X, Y, Z)` and find the domain cells inside the CAD solids with optimized **collision filters**
- Assign material properties to each cell inside solid and subpixel smoothing to mitigate corner artifacts
- State-of-the-art interactive 3d plotting

Example: importing the LHC TCPC Crystal goniometer CAD file

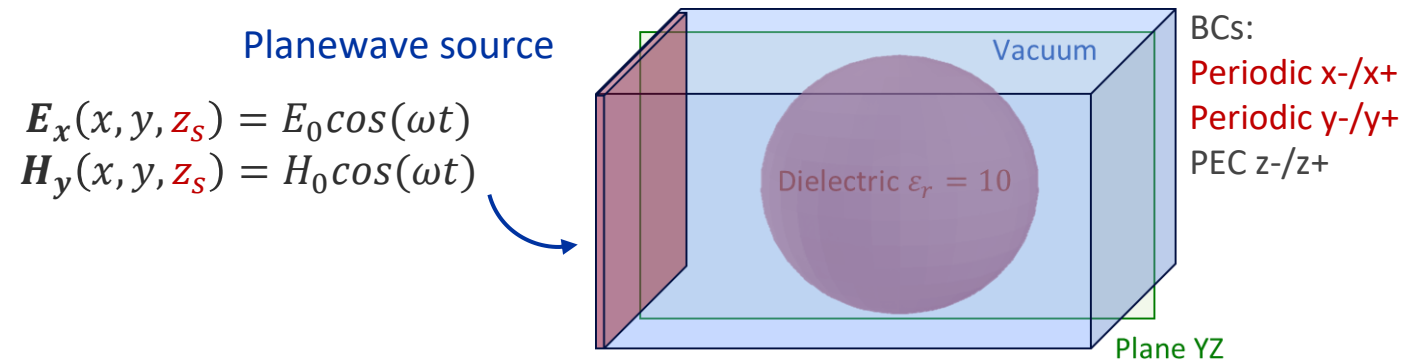


Example: *wakis* interactive `grid.inspect()` method

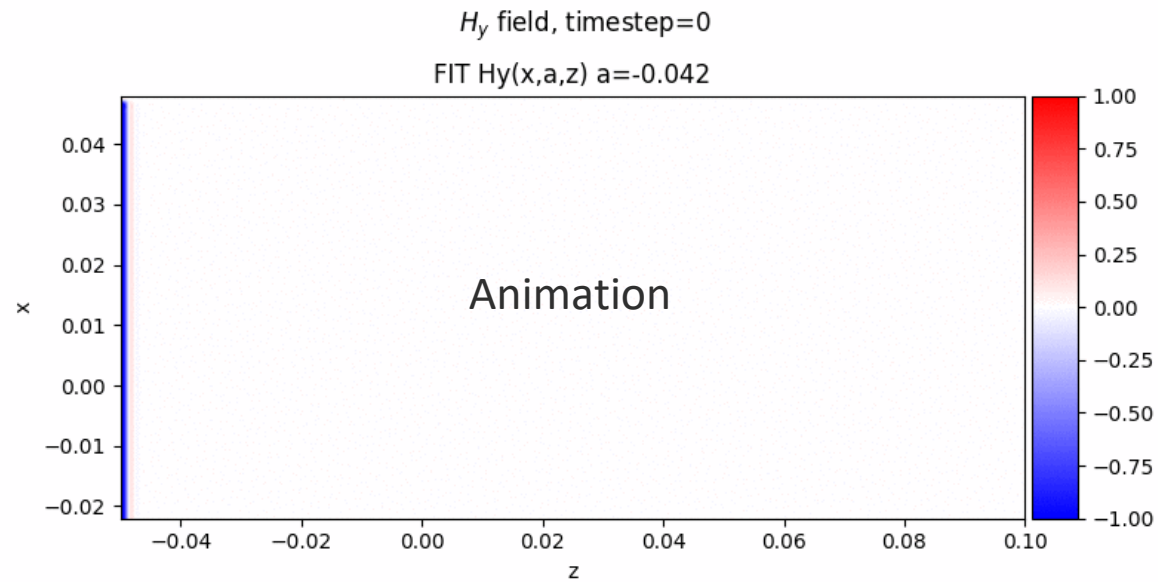


Simulations with materials and sources

Example 1: Dielectric sphere interacting with planewave



Wakis
simulation output



Mie Scattering
benchmark^[6]

[6] MEEP, Mie Scattering of a lossless dielectric sphere [\[link\]](#)

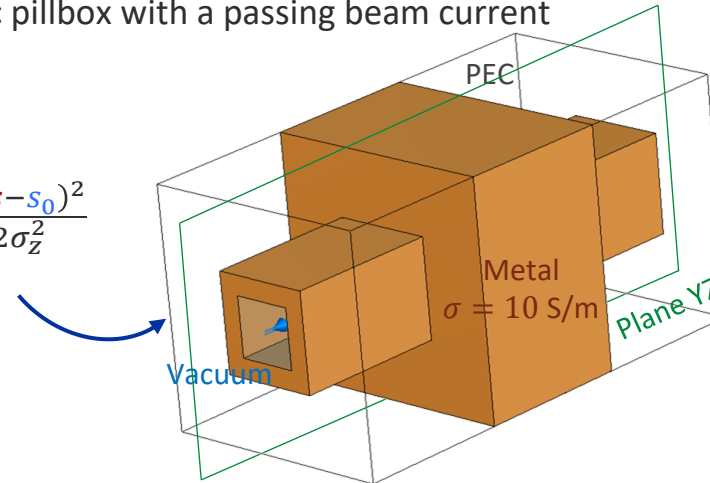
Simulations with materials and sources (II)

Example 2: Lossy cubic pillbox with a passing beam current

Beam current source

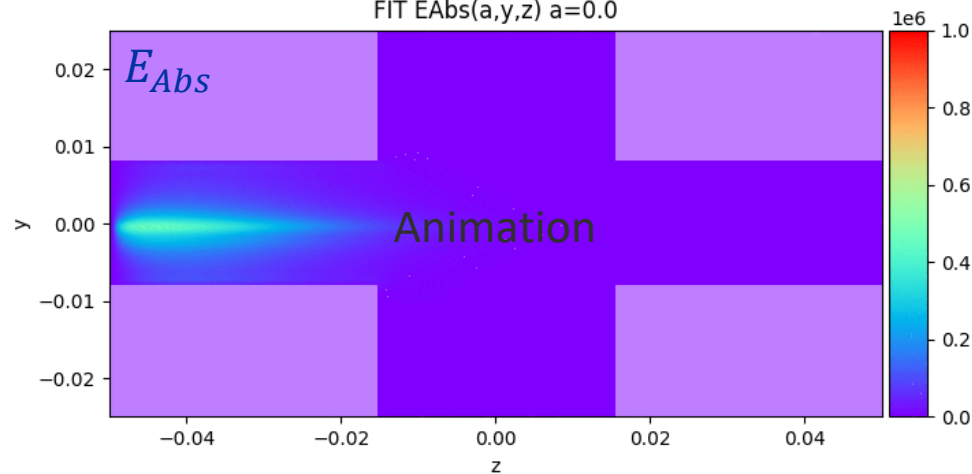
$$J_z(x_s, y_s, z) = \frac{qc}{\sqrt{2\pi\sigma_z}} e^{-\frac{(s-s_0)^2}{2\sigma_z^2}}$$

$$s = z - ct; \quad s_0 = z_{min} - ct$$



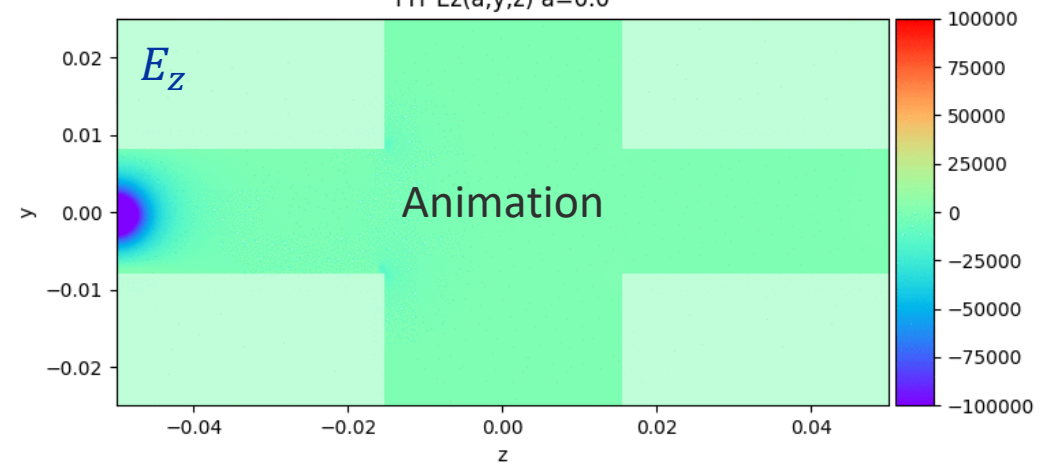
E_{Abs} field, timestep=1050

FIT EAbs(a,y,z) a=0.0



E_z field, timestep=1050

FIT Ez(a,y,z) a=0.0



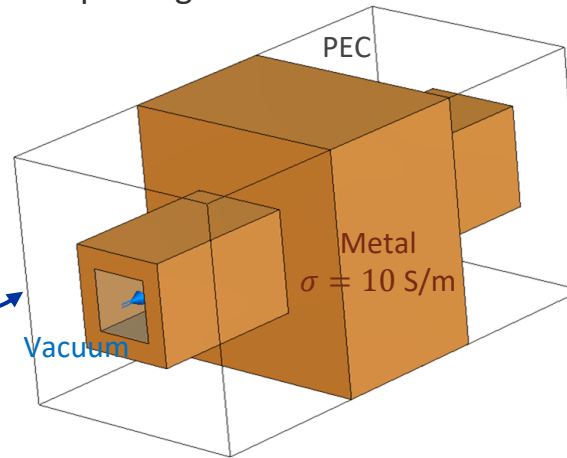
Wake potential and impedance

Example 2: Lossy cubic pillbox with a passing beam current

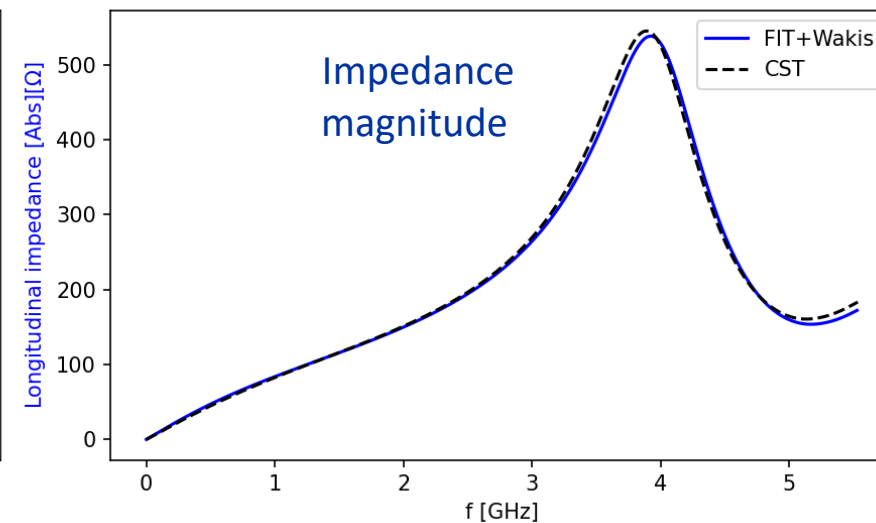
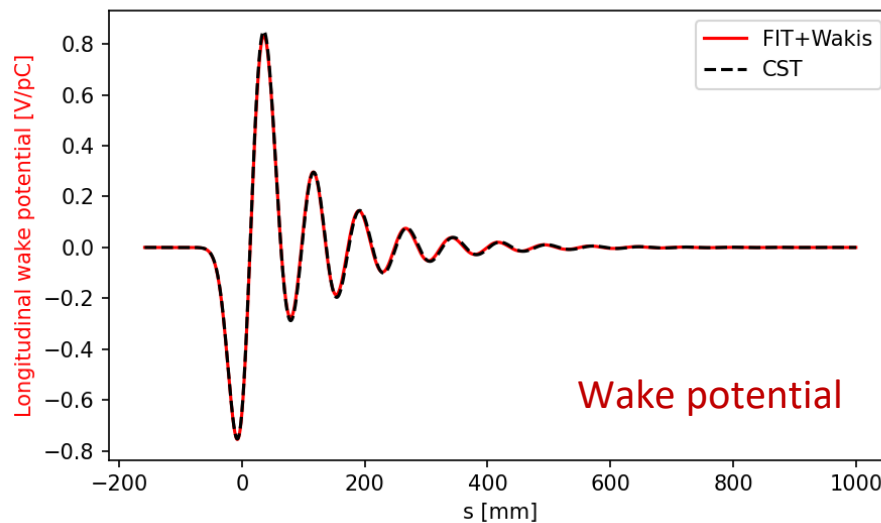
Beam current source

$$J_z(x_s, y_s, z) = \frac{qc}{\sqrt{2\pi\sigma_z}} e^{-\frac{(s-s_0)^2}{2\sigma_z^2}}$$

$$s = z - ct; \quad s_0 = z_{min} - ct$$



Benchmark with CST Wakefield Solver



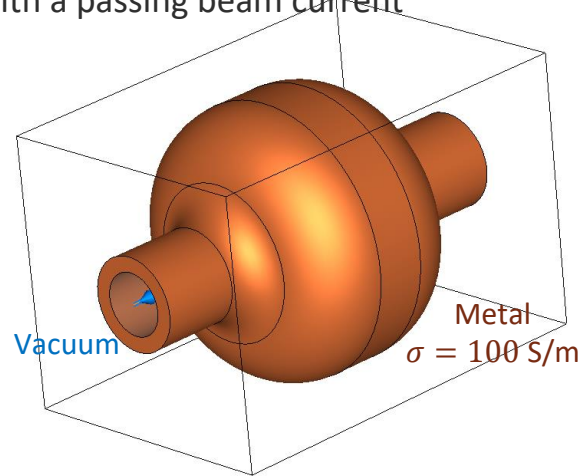
Wake potential and impedance (II)

Example 2: Lossy cubic pillbox with a passing beam current

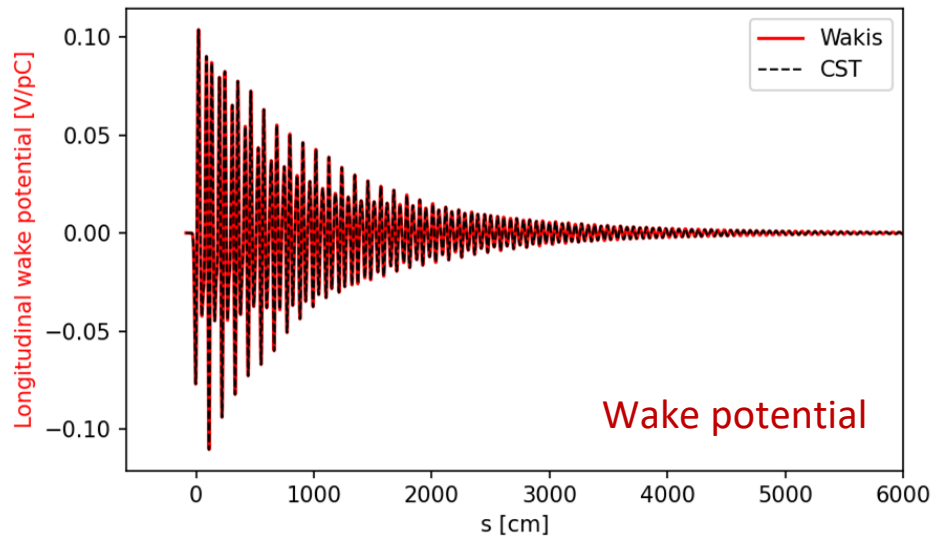
Beam current source

$$J_z(x_s, y_s, z) = \frac{qc}{\sqrt{2\pi\sigma_z}} e^{-\frac{(s-s_0)^2}{2\sigma_z^2}}$$

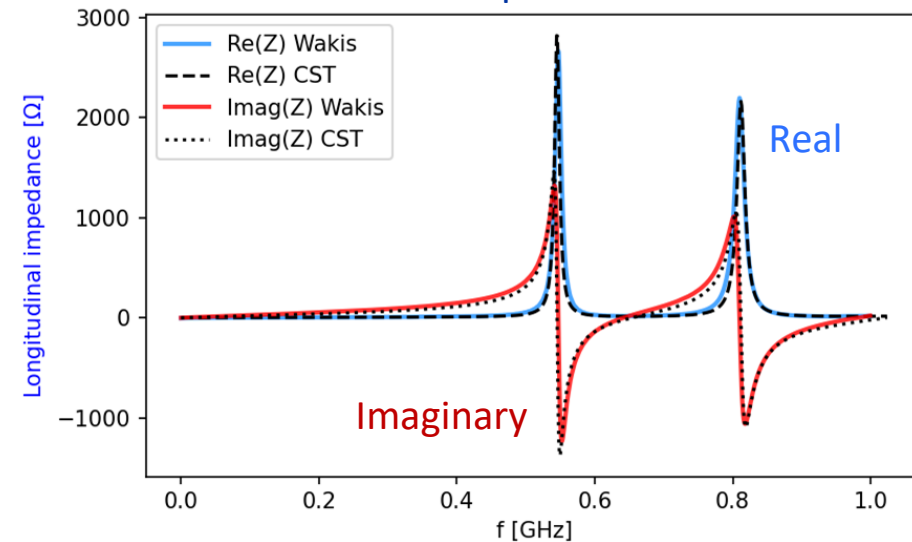
$$s = z - ct; \quad s_0 = z_{min} - ct$$



Benchmark with CST Wakefield Solver



Impedance



Low- β simulations

Beam current source **with β**

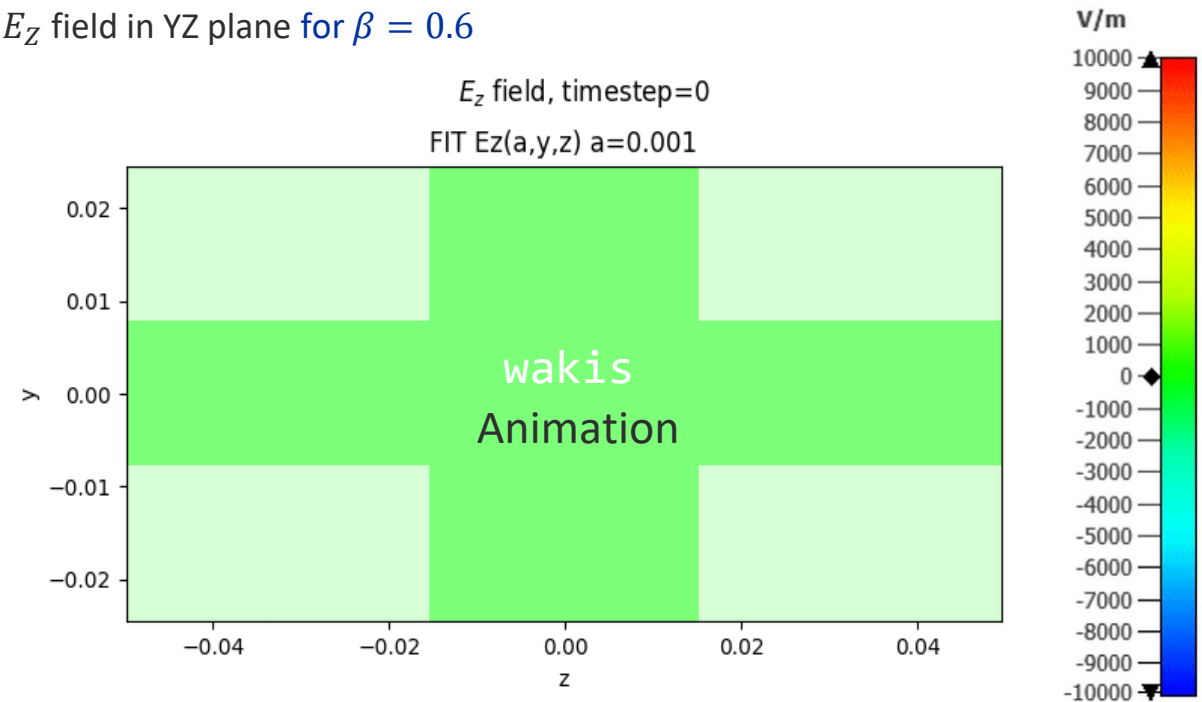
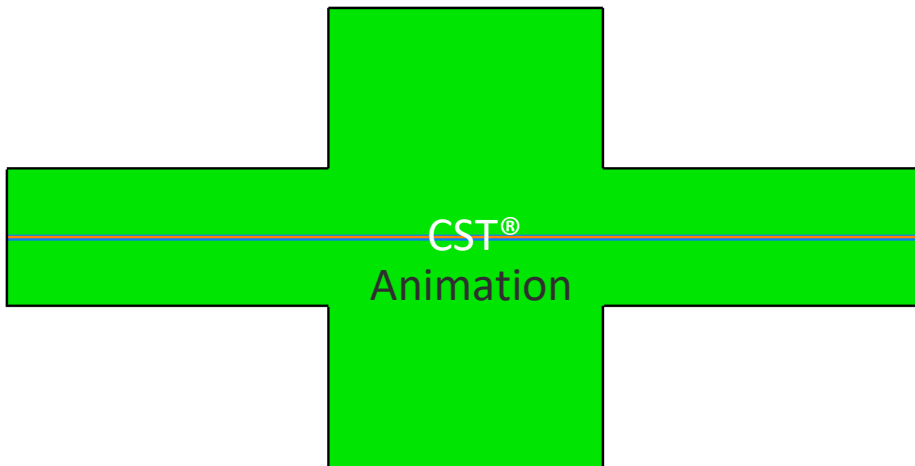
$$J_z(x_s, y_s, z) = \frac{q\beta c}{\sqrt{2\pi\sigma_z}} e^{-\frac{(s-s_0)^2}{2\sigma_z^2}}$$

$$s = z - \beta ct; \quad s_0 = z_{min} - \beta ct$$

Simulations with a **non-ultra relativistic beam** have proven **challenging** with current simulation tools due direct & indirect space charge effects \rightarrow finer mesh

- \triangleright **wakis** imulations for relativistic $\beta \in \{0.4, 1\}$ have been benchmarked with CST[®] Wakefield solver

Example 4: Round pillbox E_z field in YZ plane for $\beta = 0.6$



Low- β simulations (II)

Beam current source **with β**

$$J_z(x_s, y_s, z) = \frac{q\beta c}{\sqrt{2\pi\sigma_z}} e^{-\frac{(s-s_0)^2}{2\sigma_z^2}}$$

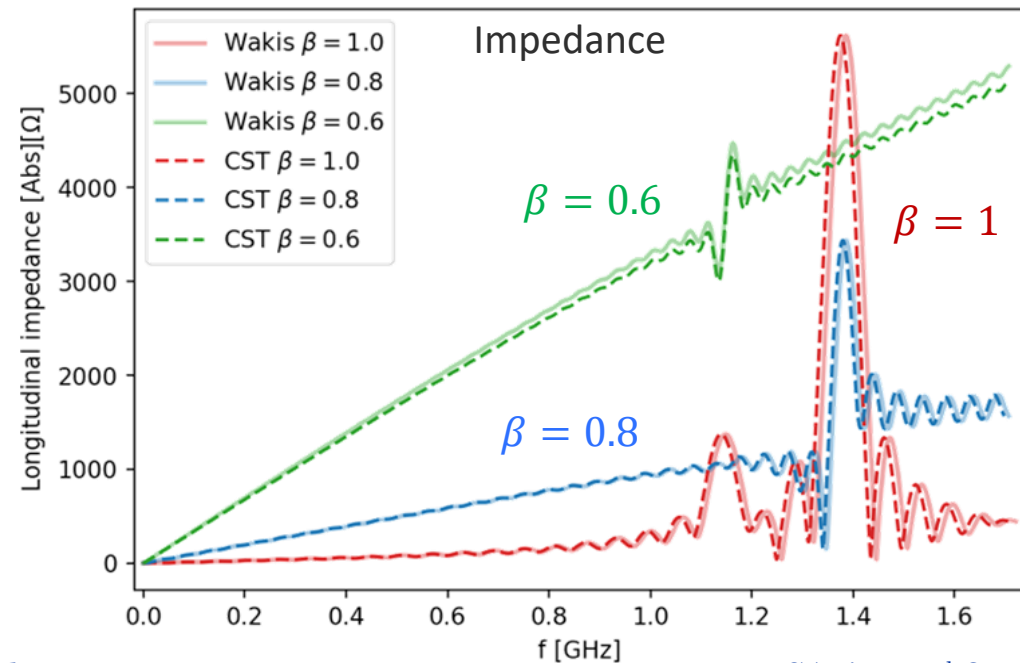
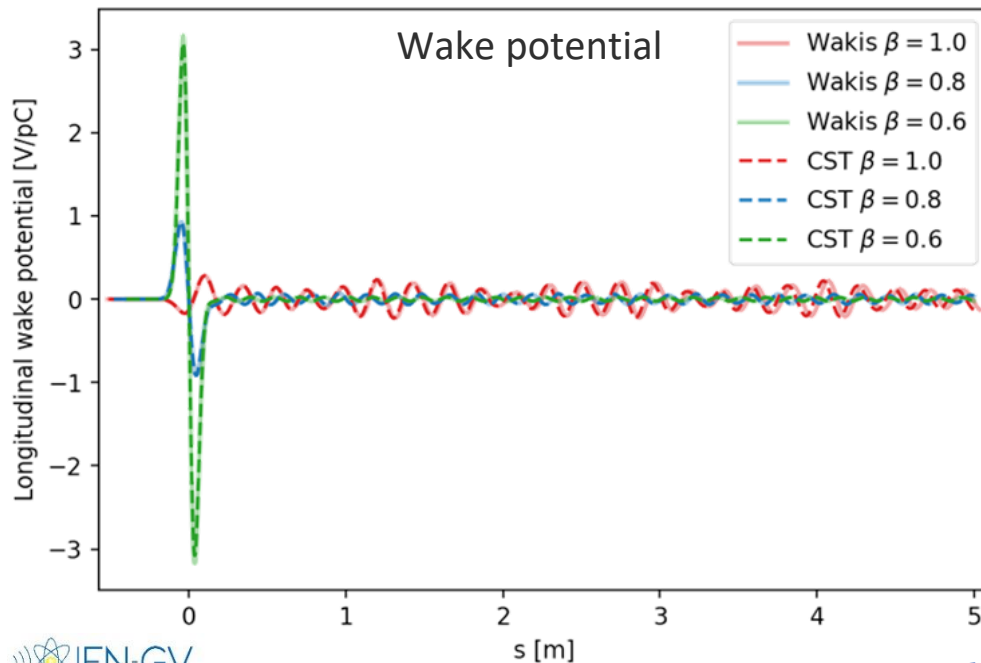
$$s = z - \beta ct; \quad s_0 = z_{min} - \beta ct$$

Simulations with a **non-ultra relativistic beam** have proven **challenging** with current simulation tools due direct & indirect space charge effects \rightarrow finer mesh

\triangleright **Wakis** simulations for relativistic $\beta \in \{0.4, 1\}$ have been benchmarked with CST[®] Wakefield solver

Example 5: Comparison of $W_{||}$ and $Z_{||}$ for different values of beam β

Benchmark with CST Wakefield Solver



Outline

1. Introduction to Beam-Coupling Impedance simulations Slides 1-4
2. **wakis** code framework and architecture Slides 5-7
3. The Finite Integration Technique (FIT): Slides 8-17
 - 3.1. Numerical Algorithm in free-space
 - 3.2. Geometry definition: STL importer
 - 3.3. Materials ε, μ, σ and EM sources
 - 3.4. Wake potential and impedance
 - 3.5. Low- β simulations
4. Conclusions, Present & Future work Slides 18-19

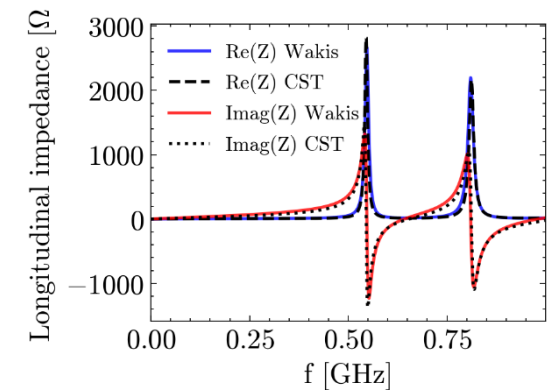
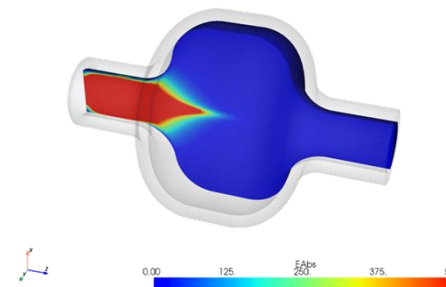
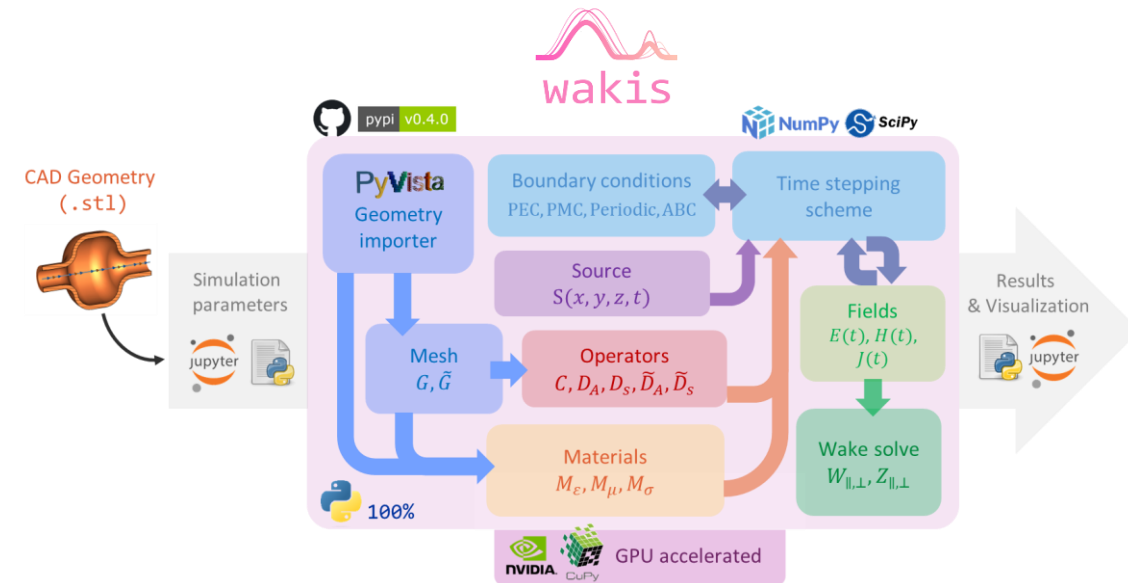
Conclusions

○ Developed a **3D Electromagnetic and Wake Solver in time domain**, 100% in python, based on the Finite Integration Technique:


- Based on numpy & scipy
- BCs: PEC, PMC, Periodic, PML*
- Anisotropic materials ϵ, μ, σ
- State of the art 2d, 3d plotting
- Available on GitHub and PyPI
- CAD geometry import
- EM Sources & beam current with β
- GPU accelerated
- Fully exposed API
- Documentation with Sphinx

○ **wakis** solver has been **benchmarked with CST Wakefield Solver®** with pillbox cavities of different materials and geometries

- **Stepping-stone** towards a collaborative open-source electromagnetic & wakefield solver capable of addressing present and future Impedance challenges



Present & Future work

- The **code is under continuous development and optimization** to accommodate current **impedance challenges**: FCC-ee & CLIC impedance model, Muon collider ionization cooling studies,...
- Some **present work** includes:
 - Optimization of the existing **PML boundaries**
 - Implementation of **Shchukin-Leontovich condition** for good conductors
 - Addition of **dispersive materials**
 - Multi-core **parallelization with mpi4py** 
 - Student M. Raschke working on **wake extrapolation** of partially decayed wakes
- Some **future work** considers:
 - Exploring the implementation of **co-moving window** for short-range wakes
 - Improving grid discretization and **mesh refinement**
 - Correct **numerical dispersion**



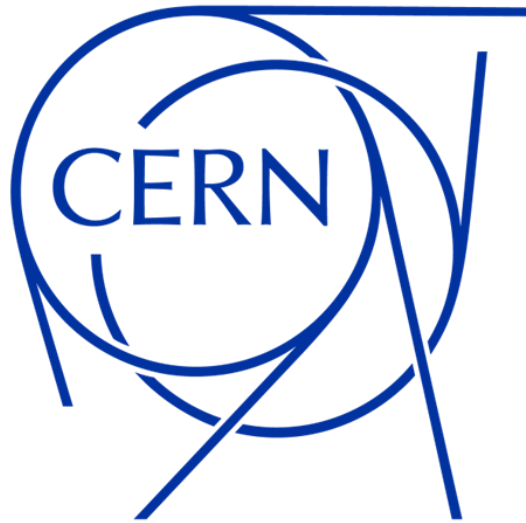
Wakis forks



Wakis developers



Thank you for the attention 😊



Wakis:

3D Electromagnetic Time-Domain
wake and impedance solver

Elena de la Fuente García

Bibliography sources

- I. T. Weiland and R. Wanzenberg, “Wake fields and impedances,” *Lect. Notes Phys.*, vol. 400, pp. 39–79, 1992, doi: [10.1007/3-540-55250-2_26](https://doi.org/10.1007/3-540-55250-2_26)
- II. I. Zagorodnov and T. Weiland, “TE/TM field solver for particle beam simulations without numerical Cherenkov radiation,” *Phys. Rev. ST Accel. Beams*, vol. 8, p. 042001, 2005, doi: [10.1103/PhysRevSTAB.8.042001](https://doi.org/10.1103/PhysRevSTAB.8.042001)
- III. M. C. Balk, R. Schuhmann, and T. Weiland, “Open boundaries for particle beams within fit-simulations,” *Nuclear Instruments and Methods in Physics Research*, vol. 558, no. 1, pp. 54–57, 2006, doi: <https://doi.org/10.1016/j.nima.2005.11.076>.
- IV. K. Klopfer, “Computation of Complex Eigenmodes for Resonators Filled With Gyrotropic Materials,” PhD thesis, Technische Universität Darmstadt, Darmstadt, 2014. [Online]. Available: <http://tuprints.ulb.tu-darmstadt.de/4210/>
- V. J. A. Moreno, E. Oliva, and P. Velarde, “EMcLAW: An unsplit Godunov method for Maxwell’s equations including polarization, metals, divergence control and AMR,” *Computer Physics Communications*, vol. 260, p. e107268, Mar. 2021, doi: [10.1016/j.cpc.2020.107268](https://doi.org/10.1016/j.cpc.2020.107268).
- VI. U. Niedermayer, “Determination of Beam Coupling Impedance in the Frequency Domain,” PhD thesis, Technische Universität Darmstadt, Darmstadt, 2016. [Online]. Available: <http://tuprints.ulb.tu-darmstadt.de/5157/>

Acknowledgements:

The author would like to thank: Chiara Antuono, Giovanni Rumolo, Benoit Salvant, Szymon Lopaziuck, Leonardo Sito (CERN), Jean Luc Vay, Weiqun Zhang (LBNL) & Eduardo Oliva (UPM) for the very useful discussions

Simulation script example

\$ipython script.py



```
from wakis import GridFIT3D, SolverFIT3D, WakeSolver
import pyvista as pv

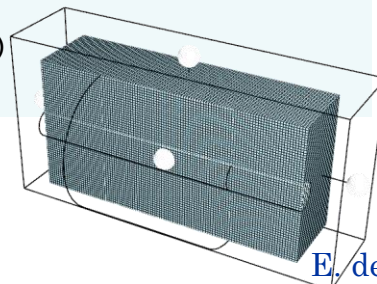
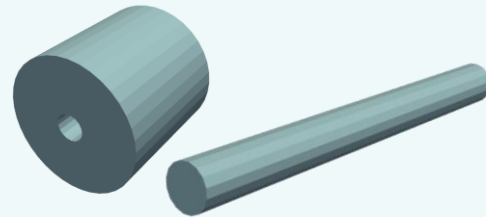
# ----- Domain and Grid setup -----
# Number of mesh cells
Nx = 57
Ny = 57
Nz = 109
#dt = 5.707829241e-12

# Geometry Import
stl_cavity = 'cavity.stl'
stl_pipe = 'beampipe.stl'
stl_solids = {'cavity': stl_cavity, 'pipe': stl_pipe}

# Materials
stl_materials = {'cavity': 'vacuum', 'pipe': 'vacuum'}
background = [1.0, 1.0, 100] # lossy metal [ $\epsilon_r$ ,  $\mu_r$ ,  $\sigma$ ]

# Domain bounds (from stl)
surf = pv.read(stl_cavity) + pv.read(stl_pipe)
xmin, xmax, ymin, ymax, zmin, zmax = surf.bounds

# Set grid and geometry
grid = GridFIT3D(xmin, xmax, ymin, ymax, zmin, zmax, Nx, Ny, Nz,
                 stl_solids=stl_solids,
                 stl_materials=stl_materials)
#grid.inspect()
```



```
# ----- Beam source -----
# Beam parameters and wake obj.
beta = 0.8 # beam relativistic beta
sigmaz = beta*6e-2 # [m] -> multiplied by beta to have f_max cte
q = 1e-9 # [C]
xs = 0. # x source position [m]
ys = 0. # y source position [m]
xt = 0. # x test position [m]
yt = 0. # y test position [m]
# tinj = 8.53*sigmaz/(beta*c) # injection time offset [s]

wake = WakeSolver(q=q, sigmaz=sigmaz, beta=beta,
                  xsource=xs, ysource=ys, xtest=xt, ytest=yt,
                  save=True, logfile=True)

# ----- Solver & Simulation -----
# boundary conditions and solver obj.
bc_low=['pec', 'pec', 'abc']
bc_high=['pec', 'pec', 'abc']
solver = SolverFIT3D(grid, wake,
                    bc_low=bc_low, bc_high=bc_high,
                    use_stl=True, bg=background,
                    use_gpu=True)

# Run wakefield time-domain simulation
wakelength = 500. #[m]
solver.wakesolve(wakelength=wakelength, add_space=add_space,
                 plot=True, plot_every=30, save_J=True,**plotkw)
```



A word on wake potential $W(s)$

Wake function $w(s)$ cannot be obtained for **distributed source of charge λ** (i.e., a beam), but instead the **wake potential $W(s)$** . For a beam traversing a device in the positive z direction:

$$W(x_s, y_s, x_t, y_t, s) = \frac{1}{q_1} \int_{-\infty}^{\infty} dz [\mathbf{E}(x_s, y_s, x_t, y_t, z, t) + c \mathbf{e}_z \times \mathbf{B}(x_s, y_s, x_t, y_t, z, t)]_{t=\frac{(s+z)}{c}}$$

Where x_s, y_s is the transverse position of the **source beam λ** , and x_t, y_t is the transverse position of the **integration path**.

We can conveniently separate the **longitudinal plane (\parallel)** and the transverse plane (\perp), having:

$$W_{\parallel}(x_s, y_s, x_t, y_t, s) = \frac{1}{q_1} \int_{-\infty}^{\infty} E_z \left(x_s, y_s, x_t, y_t, z, t = \frac{(s+z)}{c} \right) dz$$

Using the **direct integration method**, we can obtain the longitudinal wake potential from the **4D (spatial + time) electric field data $E_z(x, y, z, t)$** by computing this integral

A word on Impedance $Z(f)$

Impedance $Z(f)$ can be calculated from the wake potential $W(s)$ (longitudinal and transverse) by the **Fourier transform (FT)** of the former divided by the FT of the **charge distribution $\lambda(s)$** :

$$Z_{\parallel}(\omega) = -\frac{\int_{-\infty}^{\infty} W_{\parallel}(s)e^{-i\omega s} ds}{\int_{-\infty}^{\infty} c\lambda(s)e^{-i\omega s} ds}$$

$$Z_{\perp}(\omega) = -i\frac{\int_{-\infty}^{\infty} W_{\perp}(s)e^{-i\omega s} ds}{\int_{-\infty}^{\infty} c\lambda(s)e^{-i\omega s} ds}$$

To compute the FT, the `scipy` or `numpy` FFT algorithms are fast and reliable.

To match CST® single sided DFT, defined by [1]:

$$S(\omega) = \frac{\Delta t}{\sqrt{\pi}} \sum_{k=0}^{1000} s(k)e^{-ik\Delta t\omega}$$

We use `np.fft.fft` routine with:

- i. $f_{\max} = \frac{c}{\pi\sigma_z}$ maximum frequency the beam excites
- ii. $\Delta f = \Delta s/c$ freq. resolution defined by timestep
- iii. Length N (for zero padding) $N = 1001/(\Delta f \cdot f_{\max})$

```
# import charge distribution in z
load charge_dist
lambda = interpolate(s, z, charge_dist/q)
# or analytically
lambda = 1/((sigma_z*sqrt(pi))*exp(-(s**2)/(2*sigma_z**2)))
# calculate FFT
lambdaf = np.fft.fft(lambda*c, n=N)
WPf = np.fft.fft(WP*1e-12, n=N) #[pC]
f = np.fft.fftfreq(N, delta_f)
# calculate impedance
Z = - WPf / lambdaf
```

[1] CST Studio® Wakefield Solver Overview [[link](#)]