FITtoolbox Documentation

Release 1.2

maik.holzhey

Apr 08, 2019

CONTENTS:

1 FIT module 1

2 Indices and tables 9

Python Module Index 11

CHAPTER

ONE

FIT MODULE

This module is a generic PYTHON implementation of the essential functions for doing computations using the Finite-Integration-Technique

Warning: 1/2 = 0 !integer division!

Use instead:

```
>>> 1./2 = 0.5
```

This module can be imported using

```
>>> import FIT
```

The main function calculating an example does only execute when module runs in main frame of the execution stack, which is not the case for *import* statements.

class FIT.**FIT**

This class-object represents the basic framework for any FIT calculations

boundaryIndices(*boundaryConditions*)

Boundary Indices: determine the indices of boundary components (transversal e and normal h) boundaryConditions = array of coded boundary info [xlow xhigh ylow yhigh zlow zhigh] 1 = boundaries to be considered (i.e. PEC-bounds)

PMC (Neumann) Boundary conditions are given by default with the FIT operator stencil at boundary. So only if PEC = 1 this function makes an alteration to the given operators

Initializes attributes

Parameters **boundaryConditions** (*integer array*) – array referencing [xlow xhigh ylow yhigh zlow zhigh]

Variables

- **epsBOUND** – array of indices of e-components
- **mueBOUND** – array of indices of h-components

boundaryIndicesBOR(*boundaryConditions*)

Boundary Indices: same as 'boundaryIndices' but for rotational body (rz-code)

Warning: If called, will raise NotImplementedError

buildMaterial()
Builds material matrices and stores them as attributes:

Also processes boundary information to represent the model geometry

Variables

- **Meps** – FIT epsilon matrix
- **Mepsi** – FIT epsilon matrix inverse
- **Mmue** – FIT mue matrix
- **Mmuei** – FIT mue matrix inverse

createOperators()
FIT operators

Intializes the attributes:

Variables

- **G** – gradient operator
- **S** – divergence operator
- **C** – curl operator
- **Cs** – curl operator on dual grid

createOperatorsBOR()
FIT operators: rz code

Warning: If called, will raise NotImplementedError

createP()
FIT elementary operators

Initializes the attributes:

Variables

- **Px** – Px as FIT operator
- **Py** – Py as FIT operator
- **Pz** – Pz as FIT operator

createPBOR()
FIT elementary operators: rz code

Warning: If called, will raise NotImplementedError

defaultModel(*nx*, *ny*, *nz*, *LxMin*, *LyMin*, *LzMin*, *LxMax*, *LyMax*, *LzMax*, *boundaryConditions*)
A wrapper function to have everything as quickly as possible

Parameters

- **N** (*integer*) – number of points in _
- **L_Min** (*float*) – total length in _ (minimum)
- **L_Max** (*float*) – total length in _ (maximum)

Boundary Indices: determine the indices of boundary components (transversal e and normal h) boundaryConditions = array of coded boundary info [xlow xhigh ylow yhigh zlow zhigh] 1 = boundaries to be considered (i.e. PEC-bounds)

PML (Neumann) Boundary conditions are given by default with the FIT operator stencil at boundary. So only if PEC = 1 this function makes an alteration to the given operators

Parameters **`boundaryConditions`** (*`integer array`*) – array referencing [xlow xhigh ylow yhigh zlow zhigh]

Initializes attributes

Variables

- **`epsBOUND`** – array of indices of e-components
- **`mueBOUND`** – array of indices of h-components

All of the above are then accessible via attribute calling convention

Example usage

```
>>> fitObject.nx
```

`defaultModelBOR`()

A wrapper function to have everything as quickly as possible: rz code

Warning: If called, will raise NotImplementedError

`dftMatrix`(*frequency*, *timeSignalLength*, *dtmax*)

Constructs a linear operator, the DFT matrix, with a frequency selective sampling for use in *selectiveDFT()*

Parameters

- **`frequency`** (*`numpy array`*) – array with frequencies to be sampled
- **`timeSignalLength`** (*`number`*) – length of timeSignal to norm result
- **`dtmax`** – time step (sampling)

Returns dft matrix

`fft`(*dtmax*, *timeSignal*)

Normal Fast Fourier Transform for time signals.

Use for long time signals only, otherwise spectrum does not look very nice or use zero padding to increase frequency resolution. It is the fastest way to compute a frequency curve for an application. It uses a blackman window to mitigate wrong frequency modulation due to finite length time signal.

Parameters

- **`dtmax`** (*`number`*) – time step (“sampling”)
- **`timeSignal`** (*`numpy array`*) – time signal to be analysed

`fieldEnergy`(*em1*, *e*, *h*)

Calculates the field energy of the discrete system

Parameters

- **`e,h`** (*`sparse array csc_matrix`*) – e,h field column vector in sparse (FIT algebraic dimension)

- **em1** (*sparse array csc_matrix*) – leapfrog update n-1 e field vector (FIT algebraic dimension)

Returns scalar representing the momentary field energy

fieldMonitor (*data*, *dataMonitor*, *frequency*, *lenTimeSignal*, *stepIndex*, *dtmax*)
Field Monitor to assess field solution at given frequency point. Needs to be placed within for-loop of time domain simulation. Essentially performs DFT at a given frequency point. Take care to run as many time steps so that the monitored frequency can be resolved.

dataMonitor is returned and needs to be stored. If it is an attribute of a FIT object this function would not automatically modify the content of the dataMonitor container

Parameters

- **data** (*array like (sparse vector, same as field type)*) – data each time step
- **dataMonitor** (*array like (sparse vector, same as field type)*) – data container for monitored field
- **frequency** (*number*) – frequency to be monitored
- **lenTimeSignal** – length of timeSignal
- **stepIndex** – step index within for loop
- **dtmax** – time step (“sampling”)

Return dataMonitor field monitor result

Rtype dataMonitor array like (sparse vector, same as field type)

fieldProbe (*data*, *dataMonitor*, *spaceIndex*, *stepIndex*)
Field Monitor in time domain for subsequent frequency analysis

dataMonitor container needs to be attribute of a FIT object. Function DOES NOT return anything.

Parameters

- **data** (*array like*) – data each time step
- **dataMonitor** (*array like*) – data container for monitored field
- **spaceIndex** (*array like*) – FIT canonical index
- **stepIndex** (*number*) – iterate time step

Example canonical index

```
>>> spaceIndex = np.array([i for i in range(a.nx*a.ny)],dtype=int)+2*a.np
```

gaussin (*tdt*, *fmin*, *fmax*)
Standard Gaussian-spectrum modulated excitation signal

Parameters

- **tdt** (*array like*) – time signal
- **fmin** (*float*) – minimum frequency
- **fmax** (*float*) – maximum frequency

Return sig gaussin modulated signal

Return type array like

Return npuls normed puls end without trailing zeros

Rtype npuls array like

`geomMatrices`()

Geometrical data for FIT to build material matrices

It initializes the attributes:

Variables

- **`ds`** – primary edges
- **`da`** – primary facets
- **`dsd`** – dual edges
- **`dad`** – dual facets

`geomMatricesBOR`()

Geometrical data for FIT: rz code

Warning: If called, will raise NotImplementedError

`makeMesh`()

Generates meshgrid for cartesian grid

Sets the object's attributes:

Variables

- **`xmesh`** – xmesh as linspace spanning L_Min-L_Max in n points
- **`ymesh`** – ymesh as linspace spanning L_Min-L_Max in n points
- **`zmesh`** – zmesh as linspace spanning L_Min-L_Max in n points

`makeMeshBOR`()

Generates meshgrid for rz grid

Warning: If called, will raise NotImplementedError

`nulldiv`(*a*, *b*)

Calculates a/b elementwise if b contains 0

Division doing multiplication with nullinv essentially

can handle some python sparse data, attribute a.shape is a necessity

Parameters

- **a** (*array like*) – numerator
- **b** (*array like*) – denominator

Returns pseudo divison

Return type array like

`nullinv`(*a*)

Matrix (vector) inversion by replacing diagonal entries with its reciproke value v = 1/v

can handle some python sparse data, attribute a.shape is a necessity

Parameters **a** (*array like*) – input to be inverted

Returns pseudo inverse

Return type array like

plotField (*X*, *Y*, *field*, *figureNumber*, *title*)

This function produces a 3D plot for a field quantity in one cutting plane X,Y

Parameters

- **X,Y** (*array like*) – meshgrid representing a 2D cutting plane
- **field** (*array like (dense matrix)*) – field quantity scalar representation fitting mesh X,Y
- **title** (*string*) – title of the plot figure

FigureNumber convenience to assigne dedicated plot window

Returns axis handler of python plot axis3D object

Return type matplotlib.axes._subplots.Axes3DSubplot

selectiveDFT (*dftMatrix*, *timeSignal*)

A specalized version of the discrete fourier transform, giving only frequency selective result for a given sampling vector in frequency domain. It is well suited for short time signals, that therefore have poor frequency resolution using a normal FFT, for long time signal the effort is excessive. It uses a linear operator, the DFT matrix, to optimize the calculation speed. This was made for UQ implementations where a frequency curve needs to be calculated for each of several runs with a time signal of equal length. Cutting the linear operator allows DFT of parts of the time signal.

Warning: Pay attention to sampling. Time Signal needs to be long enough to give the correct frequency representation t_max > 1/f_min

Parameters

- **dftMatrix** (*numpy matrix*) – DFT matrix generated for discrete, selective frequency points
- **timeSignal** (*numpy array*) – time signal generated with *fieldMonitorTD()*

Returns frequency representation of timeSignal

Return type numpy matrix

setGeometry (*nx*, *ny*, *nz*, *LxMin*, *LyMin*, *LzMin*, *LxMax*, *LyMax*, *LzMax*)

This functions sets the basic geometrical shape representation.

Parameters

- **N** (*integer*) – number of points in _
- **L_Min** (*float*) – total length in _ (minimum)
- **L_Max** (*float*) – total length in _ (maximum)

All of the above are then accessible via attribute calling convention

Example usage

```
>>> fitObject.nx
```

`FIT.`**`main`**`()`

Example to calculate steady state in rectangular box resonator

main frame protected execution (means: only runs if directly executed. Does not run at import modul statements)

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

FIT, 1

INDEX

boundaryIndices()FIT.FIT method, 1
boundaryIndicesBOR()FIT.FIT method, 1
buildMaterial()FIT.FIT method, 1

createOperators()FIT.FIT method, 2
createOperatorsBOR()FIT.FIT method, 2
createP()FIT.FIT method, 2
createPBOR()FIT.FIT method, 2

defaultModel()FIT.FIT method, 2
defaultModelBOR()FIT.FIT method, 3
dftMatrix()FIT.FIT method, 3

fft()FIT.FIT method, 3
fieldEnergy()FIT.FIT method, 3
fieldMonitor()FIT.FIT method, 4
fieldProbe()FIT.FIT method, 4
FITclass in FIT, 1
FITmodule, 1

gaussin()FIT.FIT method, 4
geomMatrices()FIT.FIT method, 5
geomMatricesBOR()FIT.FIT method, 5

main()in module FIT, 6
makeMesh()FIT.FIT method, 5
makeMeshBOR()FIT.FIT method, 5

nulldiv()FIT.FIT method, 5
nullinv()FIT.FIT method, 5

plotField()FIT.FIT method, 6

selectiveDFT()FIT.FIT method, 6
setGeometry()FIT.FIT method, 6