MATERIAL PLUGIN: Full tensorial frequency-independent chi2

MAY 2018

The second order nonlinear polarization

- **Simplification: non-dispersive chi2 (instantaneous response)**
- Then, for a fully tensorial chi2 nonlinear susceptibility we can write

$$P_i^{(2)}(t) = \varepsilon_o \sum_{j,k} \chi_{ijk}^{(2)} E_j(t) E_k(t)$$

- We use the d-tensor notation by writing $d_{ijk} = \frac{1}{2}\chi_{ijk}^{(2)}$ and assuming that $\chi_{ijk}^{(2)}$ is symmetric with respect to the last two indices. This assumption is valid for non-dispersive chi2 since Kleinman symmetry condition is satisfied **[Boyd, *Nonlinear Optics*]**
- Then, we can write

$$\begin{bmatrix} P_x^{(2)}(t) \\ P_y^{(2)}(t) \\ P_z^{(2)}(t) \end{bmatrix} = 2\varepsilon_0 \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} & d_{26} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} & d_{36} \end{bmatrix} \begin{bmatrix} {E_x}^2(t) \\ {E_y}^2(t) \\ {E_z}^2(t) \\ 2E_y(t)E_z(t) \\ 2E_x(t)E_z(t) \\ 2E_x(t)E_y(t) \end{bmatrix}$$

This expression can be implemented in the plugin framework!

Update equation for the plugin

- The time-domain update for each cartesian component of the polarization $\wp_i^{(2)}(t) \equiv \varepsilon_o \, P_i^{(2)}(t)$ is of the form (including a scalar linear $\chi_o^{(1)}$ term):

$$\left(U_i(\mathrm{t}_{n+1}) + \chi_0^{(1)}\right) E_i(\mathrm{t}_{n+1}) + \wp_i^{(2)}(\mathrm{t}_{n+1}) = V_i(\mathrm{t}_{n+1})$$

- Note that because of the tensorial nature of $\chi_{ijk}^{(2)}$, $\wp_i^{(2)}(t)$ depends not only on the ith cartesian component of the electric field, $E_i(t)$, but also on the other components, $E_{j\neq i}(t)$.
- We need to solve the update equation for the ith cartesian component $E_i(\mathrm{t}_{n+1})$, with knowledge of:
  - $U_i(\mathrm{t}_{n+1}), V_i(\mathrm{t}_{n+1}), E_i(\mathrm{t}_n)$ ⬅ **Always provided by the plugin**
  - $E_{j\neq i}(\mathrm{t}_n)$ ⬅ **Not provided in the regular plugin framework. We will explain how to deal with this.**

Update equation for the plugin (cont')

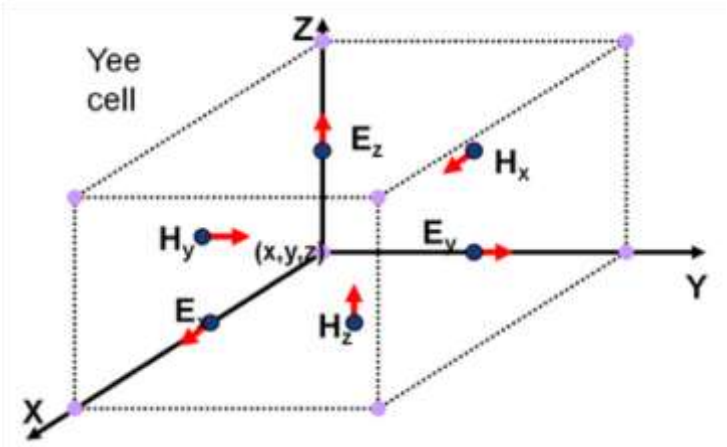
- To simplify the solution of the update equation we **approximate** $E_i(\mathrm{t}_{n+1}) \approx E_i(\mathrm{t}_n)$, which is valid for **small enough** Δt, so that

$$E_i(\mathrm{t}_{n+1}) = \frac{V_i(\mathrm{t}_{n+1}) - \wp_i^{(2)}(\mathrm{t}_{n+1})}{\left(U_i(\mathrm{t}_{n+1}) + \chi_0^{(1)}\right)}$$

$$\begin{bmatrix} \wp_x^{(2)}(\mathrm{t}_{n+1}) \\ \wp_y^{(2)}(\mathrm{t}_{n+1}) \\ \wp_z^{(2)}(\mathrm{t}_{n+1}) \end{bmatrix} = 2 \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} & d_{16} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} & d_{26} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} & d_{36} \end{bmatrix} \begin{bmatrix} {E_x}^2(\mathrm{t}_n) \\ {E_y}^2(\mathrm{t}_n) \\ {E_z}^2(\mathrm{t}_n) \\ 2E_y(\mathrm{t}_n)E_z(\mathrm{t}_n) \\ 2E_x(\mathrm{t}_n)E_z(\mathrm{t}_n) \\ 2E_x(\mathrm{t}_n)E_y(\mathrm{t}_n) \end{bmatrix}$$

Field updates in the material plugin

- The material plugin uses the same Yee cell as FDTD for updating the fields. Therefore, Ex, Ey and Ez are evaluated at different spatial positions, as shown below.



- In the regular plugin framework each Cartesian component is updated independently. For example, when updating Ex at a particular location we don't know anything about Ey and Ez. However, this is a requirement for dealing with off-diagonal terms of the chi2 tensor. A work-around is explained next.

Work-around to access Ex, Ey and Ez

Requirements:

(1) Group Ex, Ey and Ez, which are evaluated at different spatial locations, and associate them with a common reference point in space:

- Safer than performing spatial interpolation, which (without care) can lead to numerical instabilities (especially in nonlinear problems).
- The choice for grouping them is arbitrary (there are $2^3=8$ possible groupings of E field components). For simplicity, we group them as in the Yee cell diagram of the previous slide, so that the common reference point for each group of components is the origin of the Yee cell as drawn there. We will simply refer to this origin as the *mesh point*.
- The choice introduces an asymmetry and we lose 2^{nd} order accuracy. However, we retain 1^{st} order accuracy and must understand we may see some asymmetry in the results.

(2) To associate correctly the group of field components with its common reference point it is necessary to have the same number of Ex, Ey and Ez components in the entire region covered by the material plugin. This is explained next.

Work-around to access Ex, Ey and Ez (cont')

- The E fields are updated in order of Ex, Ey and then Ez.
- At each field update the plugin framework provides access to a float number array (or *storage field array*) associated with each field component evaluated inside the material plugin.
  - These arrays are stored in separate memory locations.
  - We use the memory address of the storage field array associated with the first Ex component inside the plugin to determine when a cycle of Ex field updates has been completed. This acts a tag for the first mesh point inside the plugin.
  - We can follow the same procedure for the other field components.
- Each field update happens in a different C++ function: *calculateEx*, *calculateEy* and *calculateEz*.
  - We only need to do the calculation of $\wp_x^{(2)}(t_{n+1})$, $\wp_y^{(2)}(t_{n+1})$ and $\wp_z^{(2)}(t_{n+1})$ in one of these functions; for simplicity we choose *calculateEx*.
  - The calculations in *calculateEx* require all field components. Ex is already available as an input to the function, while Ey and Ez come from *calculateEy* and *calculateEz.*
  - We store Ey and Ez in float type arrays (**Eyn** and **Ezn**) that are defined globally so that they can be used by *calculateEx.*
  - We store $\wp_y^{(2)}(t_{n+1})$ and $\wp_z^{(2)}(t_{n+1})$ from *calculateEx* in float type arrays (**Pynp1** and **Pznp1**) that are defined globally so that they can be used by *calculateEy* and *calculateEz.*

lumerical

Work-around to access Ex, Ey and Ez (cont')

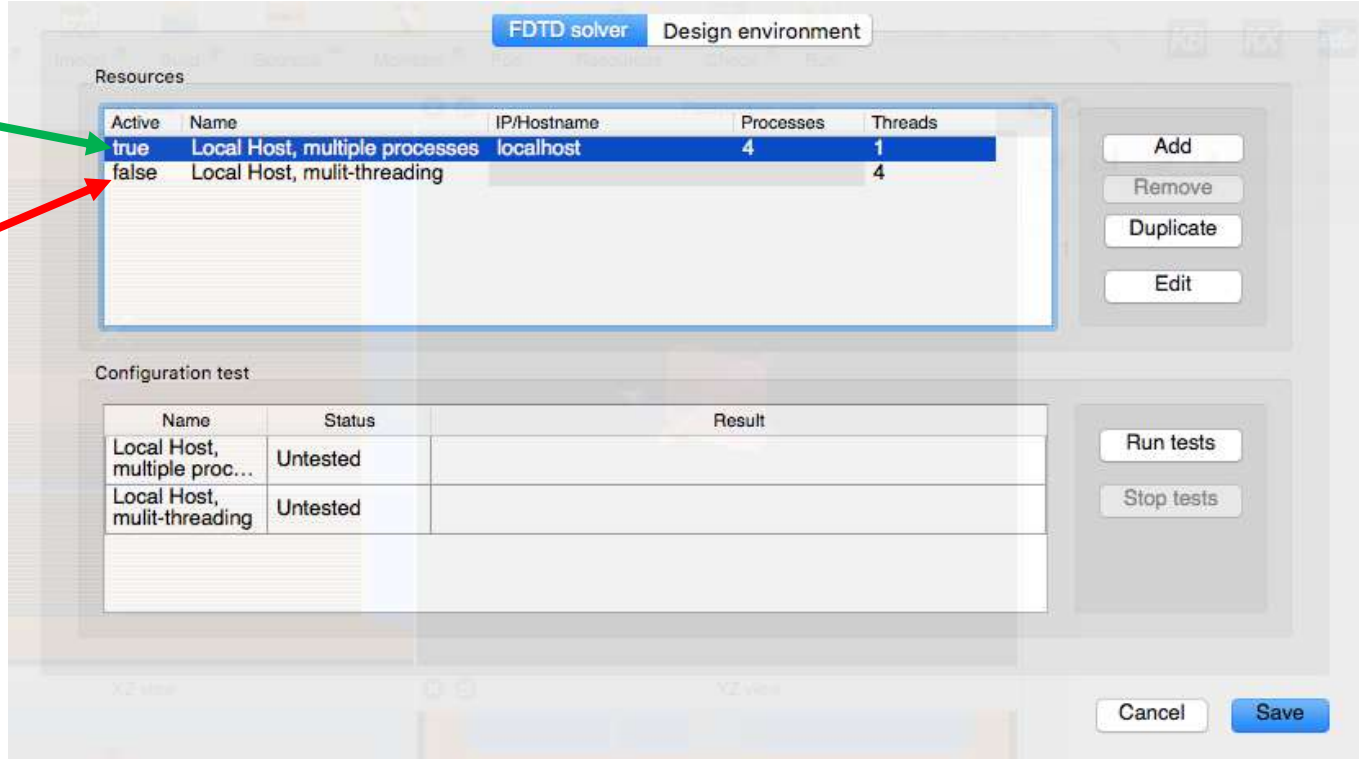
- **Safety check:** If there is a mismatch in the number of array elements between Ey, Ez and Ex, the function *causeDivergence* will return a NaN to force an instant stop to the simulation:
  - Simulation will report divergence in the first time step.
  - Put -99 in storage[0]

Setup for enforcing work-around requirements

- Consider the grid point defined by the intersection of the orange mesh lines in the GUI. We will describe a way to enforce the requirements described before.
- A grid point inside the material at position (x,y,z) must be surrounded by the plugin material within a rectangle of sides dx, dy and dz (the mesh steps), offset with respect to the grid point by (-dx/4,-dy/4,-dz/4).
- This can be easily done with a rectangular object, as shown in the test file chi2_dtensor _test.fsp, by using a mesh override region around the object and setting the dimensions of the object carefully.
- For more complicated geometries with curved surfaces the best approach is to “pixelate” the surface of the object with rectangles of sides dx, dy and dz. An example is shown in cylinder_covering_yee_cell.fsp, where an analysis group is used to start from a cylinder and create a structure that perfectly covers the correct positions of the Yee cell. There is also a script that shows how the correct coverage can be tested.

Setup for enforcing work-around requirements

- Besides the geometrical considerations in the previous slide, it is important to configure the FDTD Solver resources correctly. Multi-threading is not supported by this plugin; however, multiple processes are allowed:



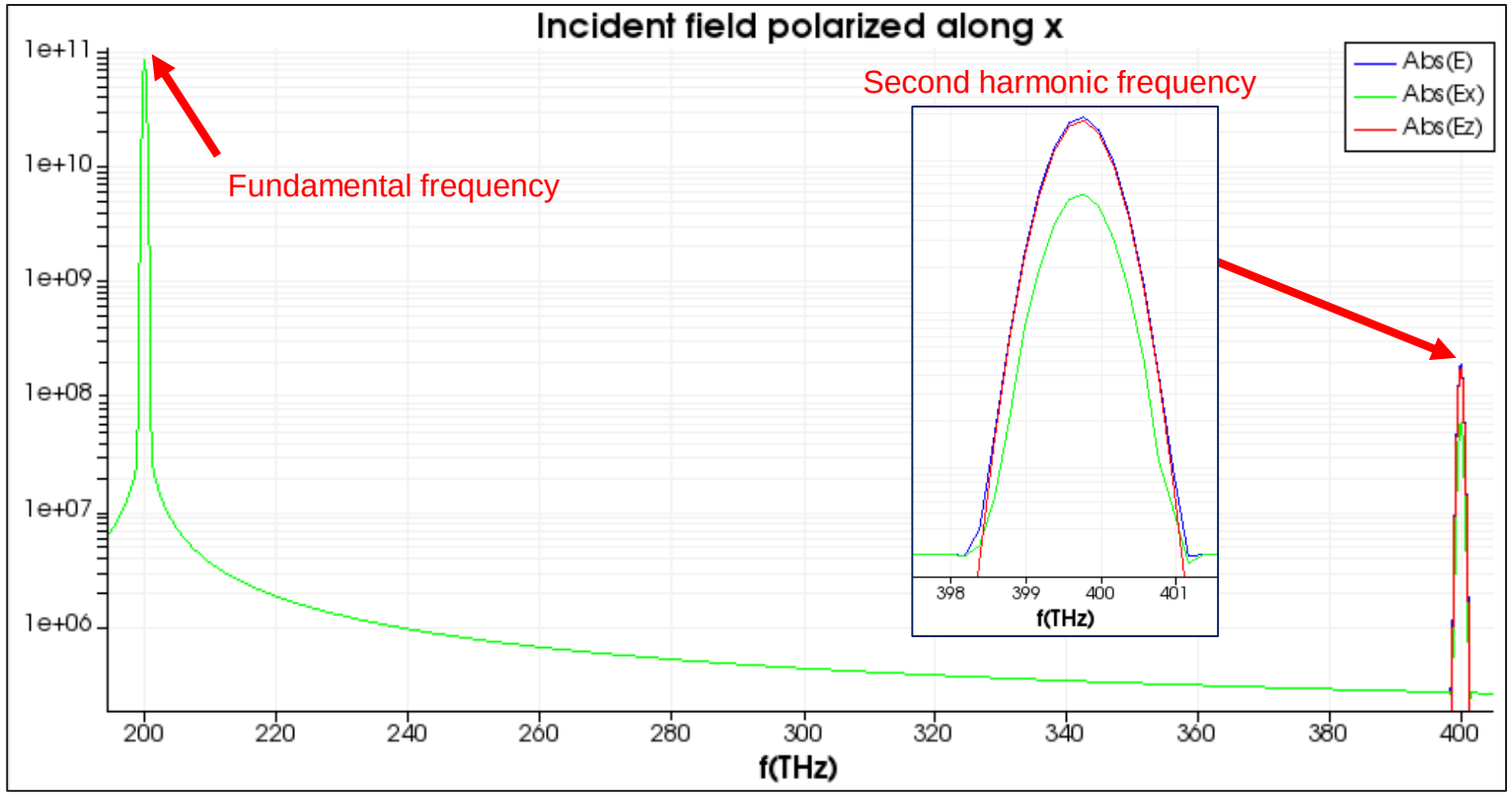
Example

- Compile and install the plugin material as explained in:
  - https://www.lumerical.com/support/webinars/videos/material_plugins_practical_implementation.html
  - https://kx.lumerical.com/t/compiling-your-plugin-material-in-windows/8102
- Open the attached file **chi2_dtensor_test.fsp**, which is based on the SHG example in https://kb.lumerical.com/en/index.html?nonlinear_harmonic_generation.html
- Create a new material “Chi2 with d tensor (version 1)”. In the Material Properties select Anisotropy=Diagonal and Base Material=none. Use the following parameter values:

Physical parameter	Material property	Value for test 1	Value for test 2
$\chi_0^{(1)}$	chi1	[2.11703, 2.11703, 2.11703]	[2.11703, 2.11703, 2.11703]
d_{11}, d_{21}, d_{31}	di1	[3e-10, 0, 9e-10]	[3e-10, 0, 0]
d_{12}, d_{22}, d_{32}	di2	[0, 3e-10, 0]	[0, 3e-10, 0]
d_{13}, d_{23}, d_{33}	di3	[0, 0, 0]	[0, 0, 0]
d_{14}, d_{24}, d_{34}	di4	[0, 0, 0]	[0, 0, 0]
d_{15}, d_{25}, d_{35}	di5	[0, 0, 0]	[0, 0, 9e-10]
d_{16}, d_{26}, d_{36}	di6	[0, 0, 0]	[0, 0, 0]

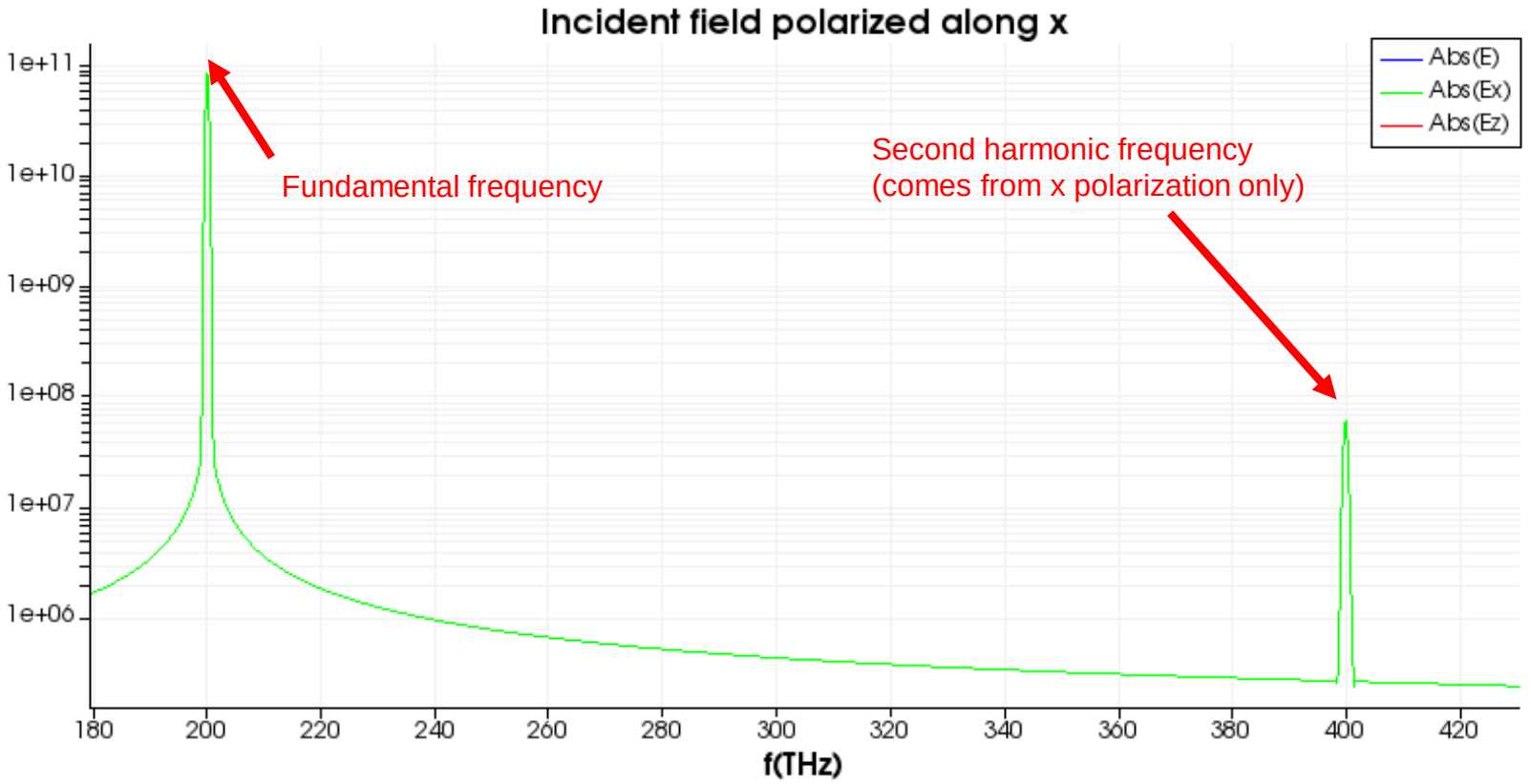
Example: Test 1

- In test 1 the only contribution to $P_z^{(2)}(t)$ comes from Ex. This contribution to SHG can only be obtained with the new plugin.
- The spectrum measured by monitor2 is shown below for x-polarized light (polarization angle 90 deg):



Example: Test 2

- In test 2 the only contribution to $P_z^{(2)}(t)$ comes from the product of Ex and Ez. Therefore, this can only be obtained when the incident field has both x and z components.
- The spectrum measured by monitor2 is shown below for x-polarized light (polarization angle 90 deg):



Example: Test 2

- In test 2 the only contribution to $P_z^{(2)}(t)$ comes from the product of Ex and Ez. Therefore, this can only be obtained when the incident field has both x and z components.
- The spectrum measured by monitor2 is shown below for 45 deg polarization angle: