An introduction to lightlabsFDS

- The lightlabsFDS mission
- Solver
- Interface
- Hardware
- End it

The lightlabsFDS mission

To enable engineers and scientists to characterize optical structures quickly, simply, and cost-effectively.

Innovate on three fronts to make this a reality:

- Solve for electromagnetic fields in the frequency domain,
- Run simulations from pre-installed scientific software (Matlab),
- Offload computation to centralized custom-tuned hardware.

Solving electromagnetics in the frequency domain

- lightlabsFDS: Frequency-Domain Solver

- Solves

$$\nabla \times \mu^{-1}\nabla \times E - \omega^2 \epsilon E = -i\omega J. \tag{1}$$

  - Inputs: frequency (ω), structure (μ, ϵ), and excitation (J).
  - Outputs: electromagnetic fields (E, H, D, B).

- *Many* practical advantages over existing time-domain solvers.

Example

Time-domain issues include

- Input: clean excitation at input requires an auxiliary simulation
- Device: approximations required for material dispersion
- Output: overlap integrals must be repeatedly calculated *during* the simulation

- Fundamental problem: trying to use a time-domain solver as a frequency-domain solver.

- Additionally, no method to measure simulation error!

- Allow direct access to frequency-domain data.
- Take care of input and output fields outside of the simulation.
- Material dispersion explicitly defined.
- Simulation error well defined.

Picture from paper here.

- Frequency-domain solver made possible by correct choice of PML and linear algebra algorithm.

- See: Wonseok Shin, Shanhui Fan, "Choice of the perfectly matched layer boundary condition for frequency-domain Maxwell's equations solvers", Journal of Computational Physics (January 2012).

Interface

We want engineers and scientists to absolutely love using FDS and, at the same time, to forget about it because it just works.

- No installation, just start up Matlab and

```
>>> [E, H] = fds(omega, epsilon, J); % Done.
```

- Helper functions to
  - construct the optical structures (ϵ, μ),
  - define the input excitations (J), and
  - analyze and visualize the output fields (E, H, D, B)

  are all included and open-sourced.

- Additionally, lots of examples and tutorials.

Hardware

Hub and spokes picture here.

- Centralized, shared, custom-tuned server able to deliver the performance of a large cluster

- Performance achieved via heavily optimized GPU code

- Multiple servers can be clustered with nearly 100% computational efficiency.

Current status

- Algorithm: Implemented on GPUs, still optimizing (Jesse)
- Interface: (Wonseok)
- Hardware: Prototype ordered and being built (Jesse)