Introduction to Octopus: Maxwell systems

Heiko Appel, Franco Bonafé

Benasque TDDFT School 2022

First, why we are solving Maxwell: Maxwell-TDDFT

Advances in Physics, 2019
Vol. 68, No. 4, 225–333, https://doi.org/10.1080/00018732.2019.1695875

REVIEW ARTICLE

Light-matter interactions within the Ehrenfest–Maxwell–Pauli–Kohn–Sham framework: fundamentals, implementation, and nano-optical applications

René Jestädt[a], Michael Ruggenthaler[a*], Micael J. T. Oliveira[a], Angel Rubio[a,b,c*] and Heiko Appel[a*]

[a]*Max Planck Institute for the Structure and Dynamics of Matter, Center for Free Electron Laser Science, 22761 Hamburg, Germany;* [b]*Center for Computational Quantum Physics (CCQ), Flatiron Institute, 162 Fifth Avenue, New York, NY 10010, USA;* [c]*Nano-Bio Spectroscopy Group and ETSF, Dpto. Fisica de Materiales, Universidad del País Vasco, 20018 San Sebastián, Spain*

Maxwell's equations

Maxwell's equations in microscopic form

- Gauss laws

$$\vec{\nabla} \cdot \vec{E}(\vec{r}, t) = \frac{1}{\epsilon_0} \rho(\vec{r}, t)$$
$$\vec{\nabla} \cdot \vec{B}(\vec{r}, t) = 0$$

- Evolution equations

$$\vec{\nabla} \times \vec{E}(\vec{r}, t) = -\partial_t \vec{B}(\vec{r}, t)$$
$$\vec{\nabla} \times \vec{B}(\vec{r}, t) = \mu_0 \epsilon_0 \partial_t \vec{E}(\vec{r}, t) + \mu_0 \vec{j}(\vec{r}, t)$$

Maxwell's equations in Riemann-Silberstein form

Maxwell's equations are solved in Octopus using the Riemann-Silberstein (RS) vector

$$\vec{F}(\vec{r},t) = \sqrt{\epsilon_0/2}\ \vec{E}(\vec{r},t) + i\sqrt{1/(2\mu_0)}\ \vec{B}(\vec{r},t)$$

- Gauss law for RS vector

$$\vec{\nabla}\cdot\vec{F}(\vec{r},t) = \frac{1}{\sqrt{2\epsilon_0}}\rho(\vec{r},t)$$

- Evolution equation for RS vector

$$i\partial_t\vec{F}(\vec{r},t) = \pm c_0\vec{\nabla}\times\vec{F}(\vec{r},t) - \frac{i}{\sqrt{2\epsilon_0}}\vec{j}(\vec{r},t)$$

Maxwell's equations in Schrödinger form

- Curl of a vector in terms of Spin-1 matrices

$$\vec{a} \times \vec{b} = -i(\vec{a} \cdot \vec{\mathbf{S}})\vec{b} \qquad \text{with} \qquad \vec{\mathbf{S}} = \left(\mathbf{S_x} \ , \ \mathbf{S_y} \ , \ \mathbf{S_z}\right)$$

- Spin-1 matrices

$$\mathbf{S_x} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{bmatrix} \qquad \mathbf{S_y} = \begin{bmatrix} 0 & 0 & i \\ 0 & 0 & 0 \\ -i & 0 & 0 \end{bmatrix} \qquad \mathbf{S_z} = \begin{bmatrix} 0 & -i & 0 \\ -i & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Maxwell's equations in Schrödinger form

- Curl of a vector in terms of Spin-1 matrices

$$\Rightarrow \vec{\nabla} \times \vec{F}(\vec{r},t) = -i(\vec{\nabla}\cdot\vec{\mathbf{S}})\vec{F}(\vec{r},t)$$

- Maxwell's equations in Schrödinger form

$$i\hbar\partial_t \ \vec{F}(\vec{r},t) = \underbrace{c_0\left(\vec{\mathbf{S}}\cdot\frac{\hbar}{i}\vec{\nabla}\right)}_{\hat{\mathcal{H}}_{\mathrm{em}}}\vec{F}(\vec{r},t) - \frac{i\hbar}{\sqrt{2\epsilon_0}}\vec{j}(\vec{r},t)$$

Light-matter interactions within the Ehrenfest–Maxwell–Pauli–Kohn–Sham framework: fundamentals, implementation, and nano-optical applications; René Jestädt, Michael Ruggenthaler, Micael J. T. Oliveira, Angel Rubio, and Heiko Appel, Advances in Physics, 68, 225-333, https://doi.org/10.1080/00018732.2019.1695875

Time-evolution of the electromagnetic field

- Time-evolution operator for the free electromagnetic field

$$\hat{U}^{(0)}_{\mathrm{em}}(t+\Delta t,t)=\exp\Big[-\frac{i}{\hbar}\int_t^{t+\Delta t}\hat{\mathcal{H}}_{\mathrm{em}}(\vec{r})d\tau\Big]=\exp\Big[-\frac{i}{\hbar}\hat{\mathcal{H}}_{\mathrm{em}}(\vec{r})\Delta t\Big]$$

$$\hat{\mathcal{H}}_{\mathrm{em}}(\vec{r})=c_0\Big(\vec{\mathbf{S}}\cdot\frac{\hbar}{i}\vec{\nabla}\Big)$$

- Time-evolution operator for inhomogeneous Maxwell's equations

$$\hat{U}_{\mathrm{em}}(t+\Delta t,t)=\hat{U}^{(0)}_{\mathrm{em}}(t+\Delta t,t)\Big(F^{(0)}(\vec{r},t)-\int_t^{t+\Delta t}\hat{U}^{(0)}_{\mathrm{em}}(\tau,t)\frac{i\hbar}{\sqrt{2\epsilon_0}}\vec{j}(\vec{r},t)d\tau\Big)$$

- Only gauge-invariant quantities are propagated

Dispersive media

- Constitutive relation for dispersive medium

$$\vec{D}(\vec{r},\omega) = \epsilon(\vec{r},\omega)\vec{E}(\vec{r},\omega)$$

- In time-domain this is a convolution

$$\begin{aligned}\vec{D}(\vec{r},t) = \chi(\vec{r},t) * \vec{E}(\vec{r},t) &= \int_0^\infty \chi(\vec{r},t-\tau)\vec{E}(\vec{r},\tau)d\tau \\ &= \epsilon_0\vec{E}(\vec{r},t) + \vec{P}(\vec{r},t)\end{aligned}$$

- Polarization current

$$\vec{J}_{\mathrm{P}}(\vec{r},t) = \partial_t\vec{P}(\vec{r},t) = \epsilon_0\partial_t\int_0^t \chi_e(\vec{r},\tau)\vec{E}(t-\tau)d\tau$$

Drude dispersive media

- Drude susceptibility (for one pole)

$$\chi(\omega) = \frac{\omega_p^2}{\omega^2 - i\omega\gamma_p}$$

- Derivative in Fourier domain $f'(t) = i\omega\tilde{f}(\omega)$
- Drude polarization current

$$\vec{J}_{\mathrm{P}}(\omega) = i\omega\vec{P}(\omega) = i\omega\epsilon_0\chi(\omega)\vec{E}(\omega) = i\omega\epsilon_0\frac{\omega_p^2}{i\omega(i\omega+\gamma_p)}\vec{E}(\omega)$$

$$(i\omega+\gamma_p)\vec{J}_{\mathrm{P}}(\omega) = \epsilon_0\omega_p^2\vec{E}(\omega)$$

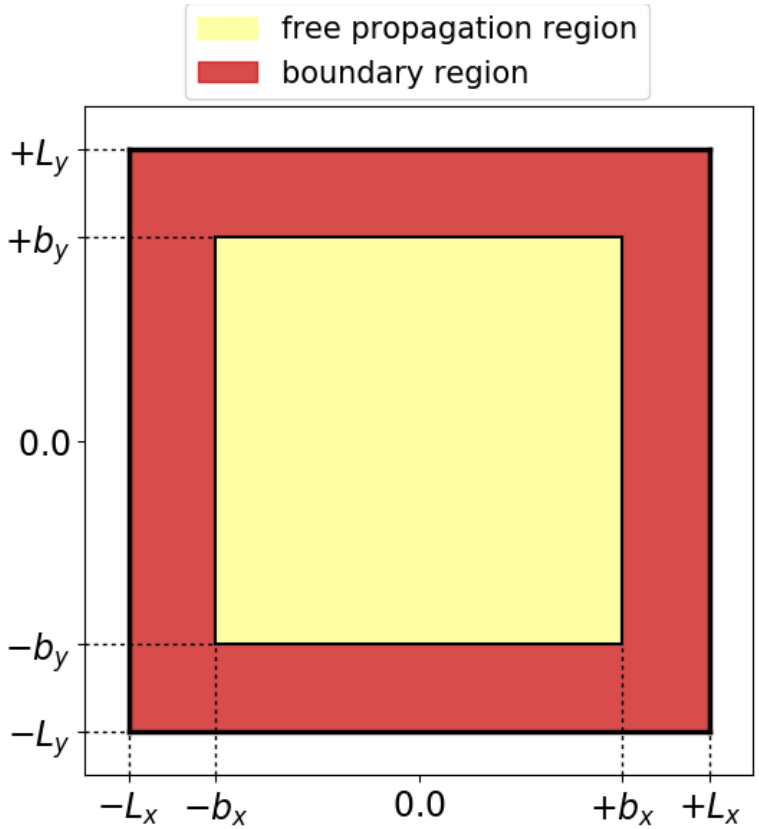
- Back in time-domain

$$(\partial_t+\gamma_p)\vec{J}_{\mathrm{P}}(t) = \epsilon_0\omega_p^2\vec{E}(t)$$

Solving this auxiliary equation for $\vec{J}_{\mathrm{P}}(t)$ simultaneously coupled to Maxwell's equations, allows to account for the effect of non-magnetic dispersive media.

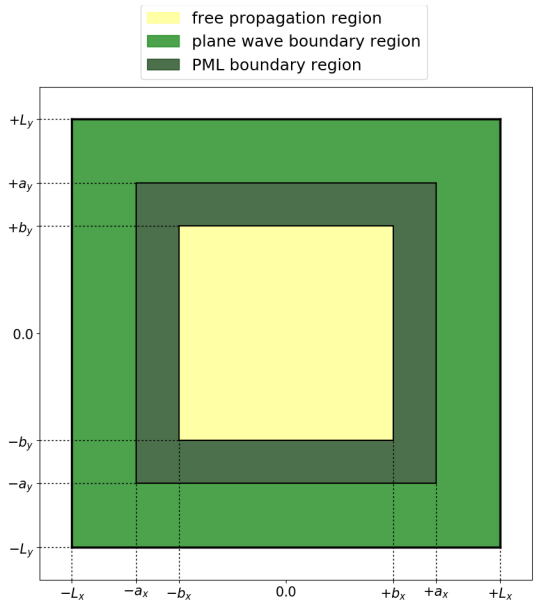
Boundary regions

- Maxwell simulation box has to be a parallelepiped
- Box boundaries are divided into inner and boundary region
- Inner region: Numerical propagation of Maxwell's equation on grid
- Outer region: Used to set the proper boundary conditions



Absorbing boundaries and incident waves

- Available absorbing boundaries are mask and perfectly matched layer (PML)
- The mask multiplies the EM field values in the boundary region with a decaying function which drops from unity in the inner region down to zero at the outer box boundaries
- The PML is a carefully crafted linear medium that absorbs all waves
- In addition, also plane wave function values can be set as Dirichlet boundary condition in the boundary region



Coupling of light and matter

- Multipole expansion (around expansion center $\vec{R}$)

$$\hat{H} = \hat{H}_{\mathrm{KS}} + \hat{H}_{\mathrm{ed}} + \hat{H}_{\mathrm{md}} + \hat{H}_{\mathrm{eq}} + ...$$

- Lowest order: Electric dipole Hamiltonian

$$\hat{H}_{\mathrm{ed}} = \vec{r} \cdot \vec{E}_{\perp}(\vec{R}, t)$$

- Next order: Magnetic dipole Hamiltonian, Electric quadrupole Hamiltonian

$$\hat{H}_{\mathrm{md}}(\vec{r}, \vec{R}, t) = -\mathrm{i}\frac{e\hbar}{2m}\vec{r} \times \vec{B}(\vec{R}, t) \cdot \vec{\nabla}$$

$$\hat{H}_{\mathrm{eq}}(\vec{r}, \vec{R}, t) = -\frac{1}{2}e\left(\vec{r} \cdot \vec{\nabla}\right)\left(\vec{r} \cdot \vec{E}_{\perp}(\vec{R}, t)\right)$$

- Lorentz Force for classical nuclei:

$$\vec{F}_j(t) = q\,\vec{E}(\vec{r_j}, t) + q\,\vec{v}_j \times \vec{B}(\vec{r_j}, t)$$

The tutorials

You can find the tutorials under this link:
`https://octopus-code.org/documentation/12/tutorial/`

Maxwell series:

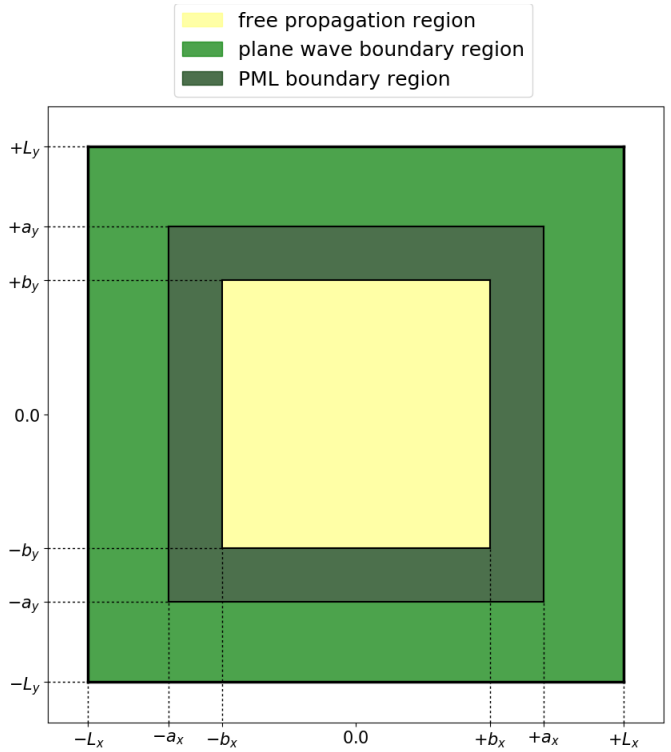
- Lesson 1: Maxwell input file
- Lesson 2: Plane waves in vacuum
- Lesson 3: External currents and PML
- Lesson 4: Non-dispersive linear media
- Lesson 5: Dispersive linear media

The tutorials

You can find the tutorials under this link:
`https://octopus-code.org/documentation/12/tutorial/`

Maxwell input file: simulation box

```
CalculationMode = td
ExperimentalFeatures = yes
%Systems
  'Maxwell' | maxwell
%

Maxwell.Dimensions = 3                  # Default
Maxwell.BoxShape = parallelepiped       # Default

lsize_mx = 20.0  # (13 inner box + 2 inc waves + 5 absorbing)
dx_mx = 0.5
%Maxwell.Lsize
lsize_mx | lsize_mx | lsize_mx
%
%Maxwell.Spacing
dx_mx | dx_mx | dx_mx
%

%MaxwellBoundaryConditions
plane_waves | plane_waves | plane_waves
%
%MaxwellAbsorbingBoundaries
cpml | cpml | cpml
%
MaxwellABWidth = 5.0
# Parameters to tune the PML (they have safe defaults)
MaxwellABPMLPower = 2.0
MaxwellABPMLReflectionError = 1e-16
```



Maxwell input file: initial EM field conditions

```
TDSystemPropagator = exp_mid
TDTimeStep = 0.002                    # <= Courant criterion, S_Courant = dx_mx / (c*sqrt(3))
TDPropagationTime = 0.4               # Total simulation time

# Option 1: Plane waves (MaxwellBoundaryConditions must be plane_waves, Tutorials 1, 3, 4)
%MaxwellIncidentWaves
plane_wave_mx_function | Ex1 | Ey1 | Ez1 | "plane_waves_func_1"
%
%MaxwellFunctions
"plane_waves_func_1" | mxf_cosinoidal_wave | kx1 | ky1 | kz1 | psx1 | psy1 | psz1 | pw1
%
%UserDefinedInitialMaxwellStates     # Initial EM field inside the box
  use_incident_waves
%

# Option 2: Switch on external current density (Tutorial 2)
ExternalCurrent = yes
%UserDefinedMaxwellExternalCurrent
current_td_function | "jx(x.y,z)" | "jy(x,y,z)" | "jz(x,y,z)" | omega | "env_func"
%
%TDFunctions
"env_func" | tdf_gaussian | 1.0 | tw | t0
%

# Option 3: uniform time-dependent field (MaxwellBoundaryConditions must be constant)
%UserDefinedConstantSpatialMaxwellField
0 | 0 | Ez | 0 | By | 0 | "time_function"
%
%TDFunctions
"time_function" | tdf_logistic | 1.0 | pulse_slope | pulse_width | pulse_shift
%
```

Maxwell input file: output options

```
# Full space-resolved outputs, to be written to the Maxwell/output_iter folder,
# every MaxwellOutputInterval steps (Check variable documentation for full list of possible options)

%MaxwellOutput
  electric_field | axis_z
  magnetic_field
  poynting_vector
%
MaxwellOutputInterval = 10
OutputFormat = axis_x + plane_y


# Outputs of the scalar variables for each time step, written into the Maxwell/td.general folder
# (the fields are evaluated at all MaxwellFieldsCoordinate points)

MaxwellTDOutput = maxwell_energy + maxwell_total_e_field

# Coordinates of the points in which field values are
# written into td.general/maxwell_fields
%MaxwellFieldsCoordinate
0.00 | 0.00 | 0.00
1.00 | 0.00 | 1.00
%
```

Tutorial 1: Plane waves in vacuum

Go to: https://www.octopus-code.org/documentation/12/tutorial/maxwell/

```
CalculationMode = td
ExperimentalFeatures = yes
%Systems
 'Maxwell' | maxwell
%

lsize_mx = 12.0  # (10 inner + 2 inc. waves)
dx_mx    = 0.5
%Maxwell.Lsize
 lsize_mx | lsize_mx | lsize_mx
%
%Maxwell.Spacing
 dx_mx | dx_mx | dx_mx
%
%MaxwellBoundaryConditions
 plane_waves | plane_waves | plane_waves
%

TDSystemPropagator = exp_mid
timestep = dx_mx / ( sqrt(3.0) * c)
TDTimeStep = timestep
TDPropagationTime = 150*timestep
```

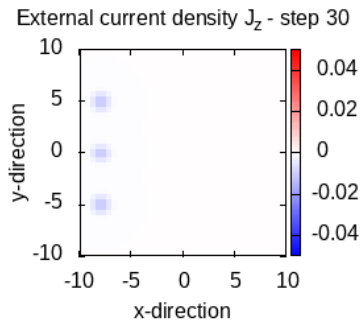
```
omega1  = 2 * pi * c /  10.0
k1_x    = omega1 / c
E1_z    = 0.05
pw1     = 10.0
ps1_x   = - 25.0

%MaxwellIncidentWaves
 plane_wave_mx_function | 0 | 0 | E1_z | "pwf1"
%
%MaxwellFunctions
 "pwf1" | mxf_cosinoidal_wave | k1_x | 0 | 0 | ps1_x | 0 | 0 | pw1
%

OutputFormat = plane_z
%MaxwellOutput
 electric_field
%
MaxwellOutputInterval = 50
MaxwellTDOutput = maxwell_energy + maxwell_total_e_field
```

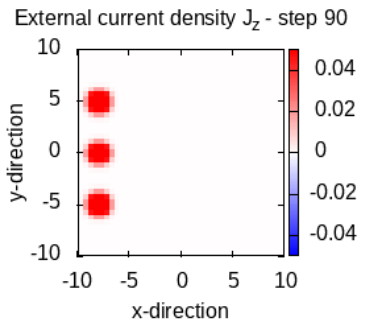
External currents

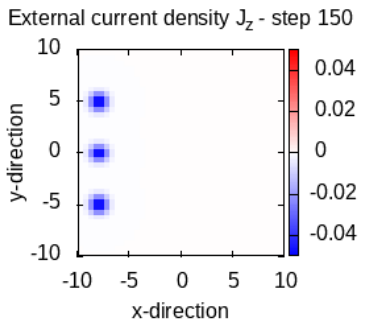
```
CalculationMode   = td
ExperimentalFeatures = yes
%Systems
  'Maxwell' | maxwell
%
lsize_mx = 10.0  # (10.0 is inner box)
dx_mx    = 0.5
Spacing = dx_mx
%Maxwell.Lsize
 lsize_mx | lsize_mx | lsize_mx
%
%MaxwellBoundaryConditions
 zero | zero | zero
%
%MaxwellAbsorbingBoundaries
 not_absorbing | not_absorbing | not_absorbing
%
TDSystemPropagator = exp_mid
timestep           = 1 / ( sqrt(c^2/dx_mx^2 + c^2/dx_mx^2 + c^2/dx_mx^2) )
TDTimeStep         = timestep
TDPropagationTime  = 180 * timestep

OutputFormat = axis_x + plane_x + plane_y + plane_z
MaxwellOutputInterval = 10
MaxwellTDOutput  = maxwell_energy + maxwell_total_e_field
%MaxwellOutput
 electric_field
 external_current | "output_format" | plane_z | "output_interval" | 2
%
%MaxwellFieldsCoordinate
 0.00 | 0.00 | 0.00
```

External currents

```
ExternalCurrent = yes
t1 = (180 * timestep) / 2
tw = (180 * timestep) / 6
j = 1.0000
sg = 0.5
lambda = 5.0
om = 2 * pi * c / lambda

%UserDefinedMaxwellExternalCurrent
current_td_function | "0" | "0" | "j*exp(-(x+8)^2/2/sg^2)*exp(-(y-5)^2/2/sg^2)*exp(-z^2/2/sg^2)" | om | "envf"
current_td_function | "0" |" 0" | "j*exp(-(x+8)^2/2/sg^2)*exp(-y^2/2/sg^2)*exp(-z^2/2/sg^2)"     | om | "envf"
current_td_function | "0" | "0" | "j*exp(-(x+8)^2/2/sg^2)*exp(-(y+5)^2/2/sg^2)*exp(-z^2/2/sg^2)" | om | "envf"
%

%TDFunctions
 "envf" | tdf_gaussian | 1.0 | tw | t1
%
```







The perfectly matched layer (PML)

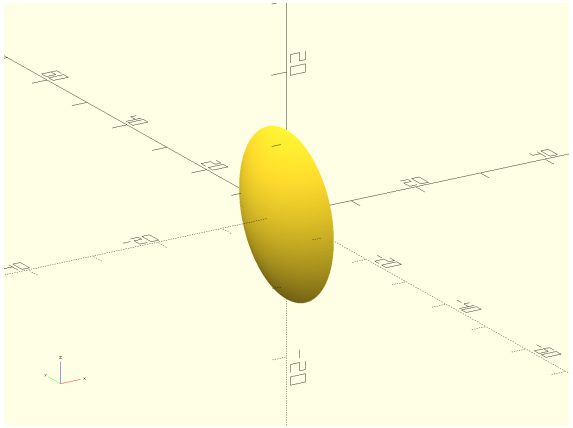
```
%MaxwellAbsorbingBoundaries
 cpml | cpml | cpml
%
MaxwellABWidth = 5.0

lsize_mx = 15.0
# (10.0 inneer + 5.0 PML)
```

Non-dispersive linear media: building the object

1. Install openscad:
   `sudo apt-get install openscad`
2. Build the lens either using the GUI or the command line (with a script). In either case, this is the code to create a lens:

```
$fs=2.0;
$fa=2.0;
translate([-10,0,0]) {
intersection(){
  translate([-8,0,0]) sphere(r=10);
  translate([ 8,0,0]) sphere(r=10);
};
};
```

3. Octopus can read OFF files, so we render the structure and store with this format:
   `openscad -o lens.off lens.scad`



Non-dispersive linear media: dynamics

```
CalculationMode = td
ExperimentalFeatures = yes
%Systems
 'Maxwell' | maxwell
 'Medium'  | linear_medium
%

# 20 = free box size (13.0) + incident wave boundaries (2.0) + absorbing (5.0)
# incident waves = der_order * dx_mx (here: der_order = 4, default)
lsize_mx = 20.0
dx_mx    = 0.5
%Lsize
 lsize_mx | lsize_mx | lsize_mx
%
Spacing = dx_mx

LinearMediumBoxShape = medium_box_file
LinearMediumBoxFile = "lens.off"
%LinearMediumProperties
 5.0 | 1.0 | 0.0 | 0.0          # epsilon_r | mu_r | sigma_e | sigma_m
%

MaxwellHamiltonianOperator = faraday_ampere_medium
%MaxwellBoundaryConditions
 plane_waves | plane_waves | plane_waves
%
%MaxwellAbsorbingBoundaries
 mask | mask | mask
%
MaxwellABWidth = 5.0
(...)
```

Non-dispersive linear media: dynamics

```
(...)
Maxwell.TDSystemPropagator = exp_mid
timestep = 1 / ( sqrt(c^2/dx_mx^2 + c^2/dx_mx^2 + c^2/dx_mx^2) )
Maxwell.TDTimeStep = timestep
Medium.TDTimeStep = timestep/2          # hack for the moment, but needed
TDPropagationTime = 150*timestep

lambda1 = 10.0
omega1  = 2 * pi * c / lambda1
k1_x    = omega1 / c
E1_z    = 0.05
pw1     = 10.0
ps1_x   = - 25.0
%MaxwellIncidentWaves
  plane_wave_mx_function | 0 | 0 | E1_z | "plane_waves_function_1"
%
%MaxwellFunctions
  "plane_waves_function_1" | mxf_cosinoidal_wave | k1_x | 0    | 0 | ps1_x | 0     | 0 | pw1
%

OutputFormat = plane_z + axis_x
%MaxwellOutput
 electric_field
%
MaxwellOutputInterval = 25
MaxwellTDOutput = maxwell_energy + maxwell_total_e_field
```

Dispersive linear media

```
CalculationMode = td
ExperimentalFeatures = yes
%Systems
 'Maxwell' | maxwell
 'NP' | dispersive_medium
%

l_zero = 550*nm     # central wavelength
lsize_mx = 1.25*l_zero + 0.25*l_zero
lsize_myz = 0.5*l_zero + 0.25*l_zero
dx_mx    = 20*nm
Spacing = dx_mx
%Lsize
 lsize_mx | lsize_myz | lsize_myz
%

LinearMediumBoxShape = medium_box_file
LinearMediumBoxFile = "gold-np-r80nm.off"
MediumPoleEnergy = 7.87*ev
MediumPoleDamping = 0.053*ev
MediumDispersionType = drude_medium

%MaxwellBoundaryConditions
 plane_waves | zero | zero
%
%MaxwellAbsorbingBoundaries
 cpml | cpml | cpml
%
MaxwellABWidth = 0.25*l_zero
```

```
TDSystemPropagator = exp_mid
timestep                = 0.1*dx_mx/c     # S_c = 0.1
TDTimeStep              = timestep
TDPropagationTime       = 240*timestep

omega  = 2 * pi * c / l_zero
kx     = omega / c
Ez     = 1.0
sigma = 40.0*c
p_s     = -(lsize_mx-0.25*l_zero)*1.2   # already !=0 inside
%MaxwellIncidentWaves
 plane_wave_mx_function | 0 | 0 | Ez | "plane_waves_function"
%
%MaxwellFunctions
 "plane_waves_function"|mxf_gaussian_wave|kx|0|0|p_s|0|0|sigma
%
%UserDefinedInitialMaxwellStates
 use_incident_waves
%

%MaxwellOutput
 electric_field | axis_x + plane_y
%
MaxwellOutputInterval = 20
MaxwellTDOutput       = maxwell_energy + maxwell_total_e_field
%MaxwellFieldsCoordinate
  -120.0*nm | 0.0 | 0.0  (...)
%
%MediumCurrentCoordinates
 -120.0*nm | 0.0 | 0.0 (...)
%
```