Supplementary material for STAR-FDTD : Space-time modulated acousto-optic guidestar in disordered media

Michael Raju[1,2,*], Baptiste Jayet[1]& Stefan Andersson-Engels[1,2]

[1] Tyndall National Institute, Lee Maltings Complex, Dyke Parade, Cork, Ireland, T12 R5CP
[2] Department of Physics, University College Cork, College Road, Cork, Ireland, T12 K8AF

[*] Author to whom any correspondence should be addressed.
Email : michaelraju@umail.ucc.ie

Abstract

This tutorial document discusses how to use the open-source (MIT license) wave FDTD code packages associated with the methods presented in our paper titled 'STAR-FDTD : Space-time modulated acousto-optic guidestar in disordered media'. The main code package consists of a collection of independent folders associated with their own applications. The FDTD codes are developed in MATLAB® with a provision to be used with or without GPU acceleration involving Parallel Computing Toolbox®. For the faster running of the FDTD time-stepping, the visualization flag may be switched off so that the final state of the wave fields are obtained. Although switching on the visualization flag leads to slower running of the code, it has the pedagogic value of plotting the progression of the time-stepping process. Towards the beginning of the tutorial, the focussing outside of a stationary disorder via phase conjugation (without any space-time modulation) is discussed. It gives an introduction to the basic FDTD code units contributing to the working of FDTD method. This is followed by the other packages containing the space-time modulation for demonstrating the one-time and iterative optical phase conjugations to focus outside and inside a disordered medium.

1 Introduction

The main code folder 'STAR-FDTD' contains four different code packages as shown in the figure 1. All four code packages, namely, Code-Package-1 to Code-Package-4, are independent and self-contained, intended for different applications. In order to generate the results presented in the main paper which involves the Space-Time Modulation (STM), Code-Package-2 to Code-Package-4 could be used. Code-Package-1 demonstrates the functionalities of some of the basic code units used in all the four packages. Code-Package-1 doesn't involve STM, but it demonstrates the code functionalities by doing three things. First is the forward propagation of waves emerging from a point source placed on the left side of a stationary disorder. Second is the estimation of the amplitude and phase of the transmitted wave on the right side of the disorder using the phase sensitive lock-in detection. Third is the subsequent time-reversal from the right side for focusing the wave back at the original source position used for forward propagation. Code-Package-1 , therefore acts as an introduction to the other code packages as it covers many basic aspects of the FDTD time stepping. This includes the initialization of various FDTD parameters, the time stepping for-loop and the associated boundary conditions. It is therefore recommended to go through Code-Package-1 first, before going through the other code packages.

Every FDTD simulations should attain steady state by time-stepping sufficient number of steps. This is assigned by setting the value for `NT_steady_state_forward_propagation` or `NT_steady_state_PC` . Such a number indicating the total number of time steps should be manually set by the user by considering the size of the disorder and the degree of multiple scattering so that the flux conservation term $R+T$ converges to unity. Here, R is the total reflection and T the total transmission. If the size of the disorder is large or the degree of multiple scattering high, one should expect to set a large value for `NT_steady_state_forward_propagation` of the order of 35,000 or more so that $R+T$ converges to unity for the disorders. The flag used for visualization

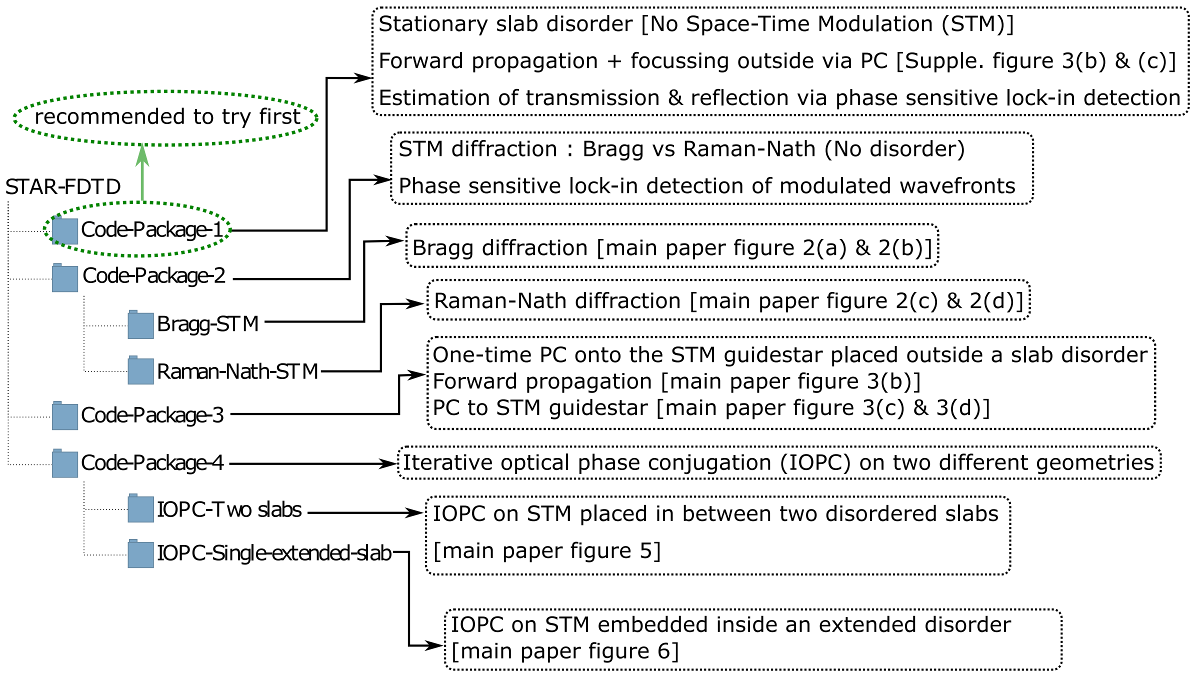
Stationary slab disorder [No Space-Time Modulation (STM)]
Forward propagation + focussing outside via PC [Supple. figure 3(b) & (c)]
Estimation of transmission & reflection via phase sensitive lock-in detection
recommended to try first
STM diffraction : Bragg vs Raman-Nath (No disorder)
Phase sensitive lock-in detection of modulated wavefronts
STAR-FDTD
Code-Package-1
Code-Package-2
Bragg-STM
Raman-Nath-STM
Code-Package-3
Code-Package-4
IOPC-Two slabs
IOPC-Single-extended-slab
Bragg diffraction [main paper figure 2(a) & 2(b)]
Raman-Nath diffraction [main paper figure 2(c) & 2(d)]
One-time PC onto the STM guidestar placed outside a slab disorder
Forward propagation [main paper figure 3(b)]
PC to STM guidestar [main paper figure 3(c) & 3(d)]
Iterative optical phase conjugation (IOPC) on two different geometries
IOPC on STM placed in between two disordered slabs
[main paper figure 5]
IOPC on STM embedded inside an extended disorder
[main paper figure 6]

Figure 1: **Overview of the code directory structure.** The four Code-Packages are independent with respect to each other and cover their own individual applications as shown. It is recommended to go through Code-Package-1 which demonstrates the basic code functionalities by focusing outside a disorder via phase conjugation, without any STM involved. Code-Package-2 to Code-Package-4 contain STM modulation and it is built on the top of Code-Package-1.

while running the FDTD can be switched on or off by setting the `visualisation_flag` to 1 or 0 in the main.m file. As expected, switching on the `visualisation_flag` would increase the total run time, but it has the pedagogic value of visualizing the evolution of the wave propagation on the computational grid. As we had access to the Parallel Computing Toolbox® in Matlab®, Nvidia GPU acceleration involving gpuArray function was implemented to generate the results in the main paper. Such a functionality can be switched on or off by setting up the flag `GPU_flag` to unity or zero. One could also run the code package without the GPU acceleration as well.

For FDTD code packages containing a disordered slab, each code run from the beginning would initialize a random disorder. Therefore, one should expect sample-to-sample mesoscopic fluctuations for the transmitted and reflected waves with every new run of the code. Many code runs involving different realizations of disorders are needed for ensemble-averaging various trends. Such an ensemble averaging is not included in this code package. In this package, a comparatively small slab is initialized. Depending upon one's computational capacity, one could try larger slabs or higher degree of scattering which may take larger computational time for attaining steady-state. For estimating the total transmission and reflection, it is recommended to adopt a slab like geometry, where the transverse size W (along the Y axis) is larger compared to the thickness L (along the Z axis) of the slab. This would help to reduce transverse oscillation of the dipole-source wave getting reflected from the transverse boundaries, taking more time for the convergence of the flux conservation relation that $R + T = 1$.

Note that the flux conservation is mainly used in the context of profiling the transport regime of the disorder used, without any STM modulation in the computational domain. Estimating the total unshaped transmission (T) would characterize whether or not the disorder is in the diffusion regime. As the paper focus on the wavefront shaping aspects through diffusive or ballistic-to-diffusion transition regimes, it is recommended to stick to their corresponding unshaped transmission.

Most of the code lines are commented within the Matlab files and hence one could always refer the code comments to figure out the purpose, starting from Code-Package-1. Therefore, the purpose of this user manual is to give an overview of the code structure from an operational point-of-view. Code-Package-1 is considered first as the following.

2 Code-Package-1

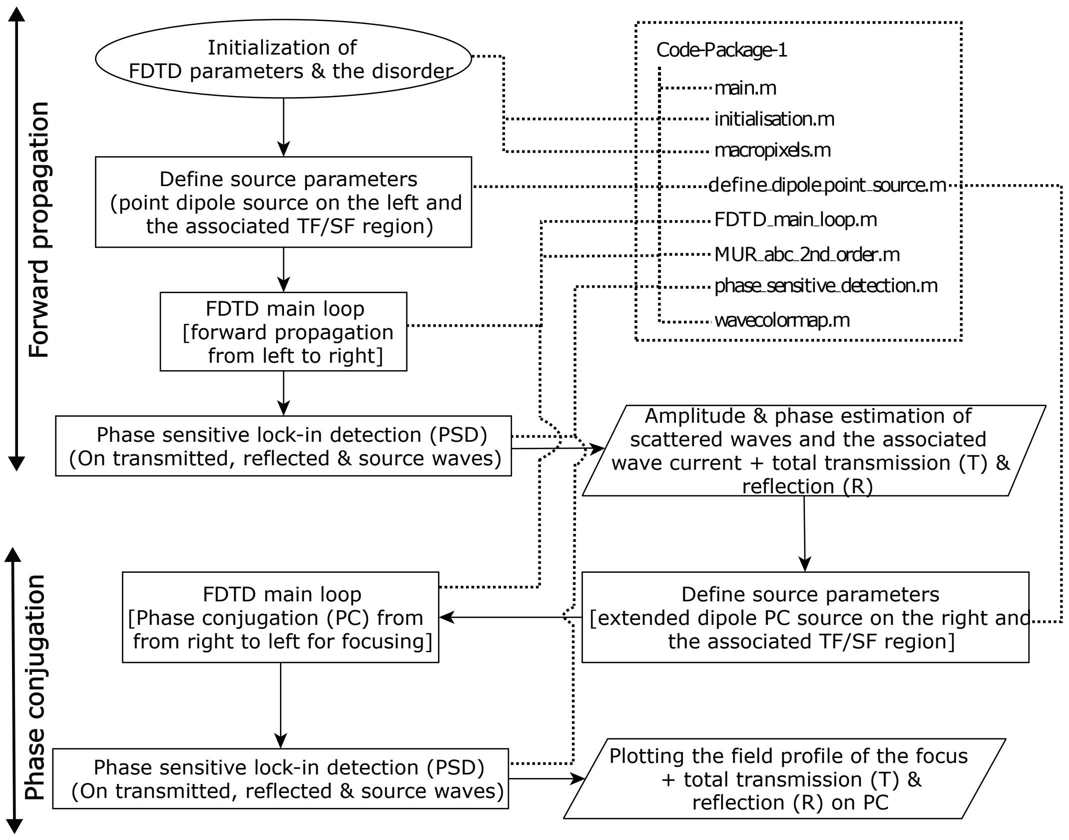
Forward propagation
Initialization of FDTD parameters & the disorder
Define source parameters (point dipole source on the left and the associated TF/SF region)
FDTD main loop [forward propagation from left to right]
Phase sensitive lock-in detection (PSD) (On transmitted, reflected & source waves)
Amplitude & phase estimation of scattered waves and the associated wave current + total transmission (T) & reflection (R)
Code-Package-1
main.m
initialisation.m
macropixels.m
define_dipole_point_source.m
FDTD_main_loop.m
MUR_abc_2nd_order.m
phase_sensitive_detection.m
wavecolormap.m
Phase conjugation
FDTD main loop [Phase conjugation (PC) from from right to left for focusing]
Define source parameters [extended dipole PC source on the right and the associated TF/SF region]
Phase sensitive lock-in detection (PSD) (On transmitted, reflected & source waves)
Plotting the field profile of the focus + total transmission (T) & reflection (R) on PC

Figure 2: **Flow chart of the FDTD operations and the associated file structure of Code-Package-1**. Both the forward propagation and the subsequent phase conjugation (without any STM present) use the same set of functions contained in the code package as shown. Such a code structure is adopted to reuse the basic code blocks for different applications.

The operational flowchart and the associated file structure of Code-Package-1 is shown in figure 2. The goal of Code-Package-1 is to introduce the basic functionalities of the FDTD operation by performing the initialization of the disorder (shown in figure 3(a)), forward propagation (shown in figure 3(b)) and a phase-conjugation (shown in figure 3(c)) to focus outside with disorder, without any STM present in the computational domain.

2.1 Initialization

The file main.m starts with the initialization of the FDTD method. The number of time steps for forward propagation and phase conjugation can be set as the following.

```
%——————————————— initialisation : fundamental parameters ———————————
NT_steady_state_forward_propagation=35000;
 % No of time steps for forward  propagation
NT_steady_state_PC=35000;
 % No of time steps for phase conjugation
```

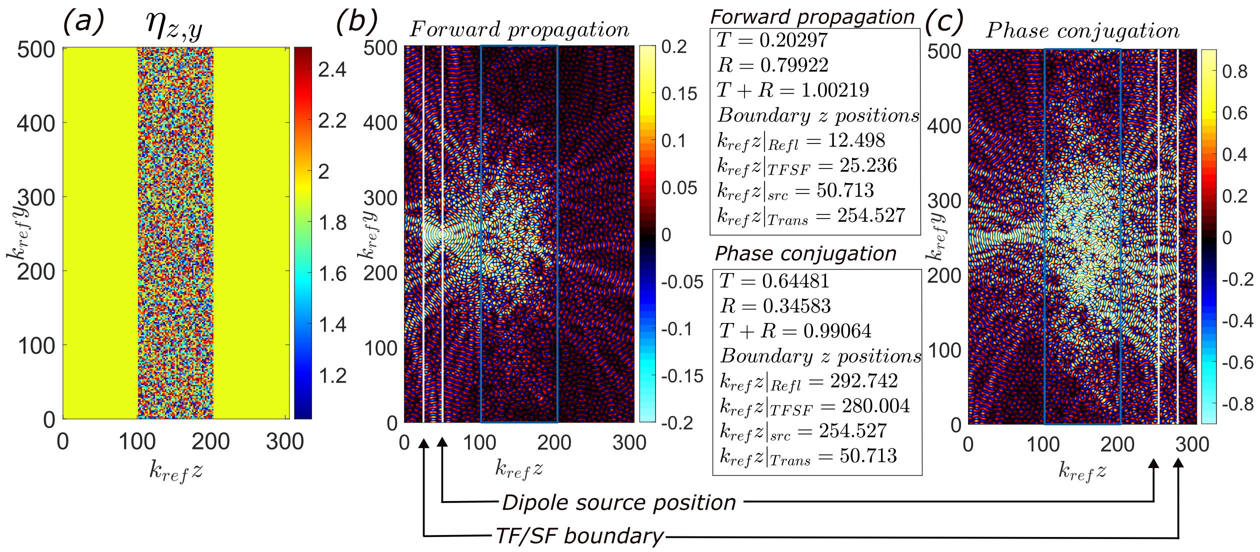

Figure 3: **Code-Package-1 performs forward propagation from left to right of the slab disorder and phase conjugation from right to left**. **(a)** Slab disorder realization in terms of the refractive index $\eta_{z,y}$ **(b)** Forward propagation due to a point dipole source placed on the left side of the disorder at $k_{ref}z|_{src}$. In order to separate the source wave and the reflected wave, a total-field/scattered-field (TF/SF) boundary is placed on the left side of the source at $k_{ref}z|_{TFSF}$ as shown. Transmitted and reflected wavefronts are estimated at $k_{ref}z|_{Trans}$ and $k_{ref}z|_{Refl}$ respectively. **(c)** Phase conjugation due to an array of dipole source placed on the right side of the slab. TF/SF boundary is shown on the right side of the dipole array source.

These numbers should be set by the user to attain steady-state of flux conservation such that $R + T = 1$. Increase the value if it is a larger slab with higher degree of scattering. Next is the geometrical aspect of the disordered slab.

```
krefL=100;              % Dimensionless longitudinal thickness of the
                       % scattering slab along z, ie kref*L
aspect_ratio=5;  % Aspect ratio of the slab size = W/L;
% Recommended to take the aspect ratio value larger than 3
z_domain_size_factor=3; % kref*Z=z_domain_size_factor*krefL
                             % Number of times krefL, is the domain z size.
krefW= aspect_ratio*krefL;  % Initial value for the dimensionless transverse
                             % length of the slab along y, ie kref*W
```

Here, `krefL` is the dimensionless thickness of the slab along Z axis, `aspect_ratio` controls the transverse size (along Y axis) of the slab disorder and `z_domain_size_factor` controls the thickness of the entire computational domain along Z axis. In other words, these terms control the size of the slab disorder in a safe manner without overflowing the computational domain. Depending on one's computational capacity, one could increase or decrease the value `aspect_ratio` or `z_domain_size_factor` for an FDTD run. As expected a large value for these terms implies longer time for attaining steady-state.

```
nref=1.9;               % Background(reference medium) refractive index
sigma=0.98;             % A parameter, usually < 1 used to control the
                         % degree of random scattering.
```

To control the degree of scattering `nref` and `sigma` could be adjusted. The disordered slab is refractive index matched with the surrounding to give prominence to diffusive scattering. These terms are frequently used to control the size and the degree of random scattering of the slab. For the initialization of all the other FDTD parameters, the following function is used, which returns the structure, `init_data`. Note that the other FDTD parameters are initialized within the `initialisation(....)` function and not in main.m. The main.m contains initialization values that is a bit more higher level of abstraction than the `initialisation(....)` function.

```
% Call initialisation script for initialising rest of the parameters and
%    return an initialisation data structure.
init_data=initialisation(GPU_flag,NT_steady_state_forward_propagation,krefL,...
krefW,nref,sigma,z_domain_size_factor,no_of_PSD_cycles);
```

The function `initialisation(....)` returns the structure `init_data` and it contains the following data fields.

```
init_data.Ny : No of grid points along Y
init_data.Nz : No of grid points along Z
init_data.kref : Magnitude of the  wave vector in the reference (background) medium
init_data.dz : Grid resolution differential element size along Z
init_data.dy : Grid resolution differential element size along Y
init_data.dt : FDTD time-stepping time differential element
init_data.start_slab1 : Grid position where the slab starts
init_data.scat_width : Scattering slab width in terms of no of grid elements
init_data.mirror_pos : Transverse offset from the mirror boundaries
init_data.NT_steady_state : Number of FDTD time steps to achieve steady state
init_data.w0 : The w0 FDTD prefactor  (defined as per the main paper)
init_data.wsrc0 : The w0 FDTD prefactor in the context of the source wave without any
    disorder
init_data.fsrc : Source temporal frequency
init_data.c0: Wave velocity in free space
init_data.eps_avg : Average dielectric constant of the reference medum
init_data.epsilon_ij : dielectric constant associated with the disorder stored as a matrix
init_data.nref : Refractive index of the reference (background) medium
init_data.lambda0 : Free space wavelength
init_data.lam_fact : Factor used to set the grid resolution
init_data.time_fact : Factor used to set the temporal resolution to satisfy the Courant-
    Friedrichs-Lewy stability condition
init_data.lambda_ref : Wavelength in the reference medium
init_data.k0 :  Free space wave vector
init_data.no_of_PSD_cycles : No of wave cycles used to average and estimate the amplitude
    and phase of the scattered waves. Although one cycle is theoretically enough, several
    cycles would help to numerically average out the values.
```

2.2 Setting up the source

Next is to initialize the source structure for defining the properties of the source to inject the incident wave into the computational domain. Here, the function `define_dipole_point_source(....)` returns the structure `source_data` as given below which is used for defining the source for both the forward propagation and the phase conjugation.

```
[source_data] = define_dipole_point_source(GPU_flag,init_data);
                        % Returns the source data structure.
source_data.clip_lim=0.2; % Colorbar clip limit for
                          % better contrast to begin with
source_data.t_sign=+1;
                        % +1 for forward propagating source and
                        % -1 for time reversed version while performing phase conjugation
```

The contrast of the wave transport images can be adjusted by setting the colour map clipping limit value `source_data.clip_lim`. The custom colormap used in these code packages are contained in wavecolormap.m. Within the code packages, `source_data.t_sign` is set to `+1` during forward propagation and

-1 in case of phase conjugation whenever it is needed to generate the intended results. Next, for the forward propagation, various boundaries associated with the source are initialized.

```
src_z_start=floor(init_data.start_slab1/2);
        % Source position z value. It is set as half way
        %            between slab and the  left boundary.
source_data.src_z=[src_z_start src_z_start+1];
 % Dipole source location
source_data.TFSF_Nz=floor(source_data.src_z(1)/2);
% Total—field Scattered field (TF/SF) boundary location
% Set as half way between the source and the left boundary
source_data.TFSF_region=1:source_data.TFSF_Nz; % Defining the TF/SF region
```

Referring to figure 3(b), the z position of the source and the TF/SF boundary are set using `src_z_start` and `source_data.TFSF_Nz`, respectively.

2.3 FDTD time stepping and phase sensitive lock-in detection for forward propagation

Next, the FDTD time stepping is implemented by the function `FDTD_main_loop`, which returns with final wave field states `Efield_array_collection` upon achieving steady state. `FDTD_main_loop` also access MUR_abc_2nd_order.m which implements the outgoing Mur absorbing boundary conditions as described in the main paper. The FDTD loop implements the time-stepping of the wave equation as described in the main paper, without the STM modulation.

```
%— Forward FDTD wave propagation : Left to right ——————————————————————————
[Efield_array_collection] = FDTD_main_loop(GPU_flag,visualisation_flag,...
init_data,source_data); % Main FDTD time—stepping loop
```

The final wave field states `Efield_array_collection` attained upon steady state is passed onto the next function, which is the function for the phase sensitive lock-in detection. Such a detection method is explained as the following.

```
%——— Phase sensitive detection (PSD) for the forward propagation —————————
init_data.detec_trans_position=...
floor(((init_data.start_slab1 + init_data.scat_width) + init_data.Nz)/2);
% Transmission measurement z index location
% Half way between the end of the slab and the right boundary
init_data.detec_refl_position=floor(source_data.TFSF_Nz/2);
% Reflection measurement z index location
% Half way between the TF/SF boundary and the left boundary
init_data.detec_src_position=init_data.start_slab1;
% Source transmission measurement z index location
% Placed at the slab surface incidence

% Next is the PSD of the amplitude and phase of the scattered and
%    the source wave, followed by the estimation of the total transmission
%    and reflection
PSD_array_collection= ...
phase_sensitive_detection(GPU_flag,visualisation_flag,Efield_array_collection,...
init_data,source_data);
% Returns a data structure containing amplitude
% and phase of transmitted, reflected and source
% waves along with total transmission and reflection.
```

Here, `init_data.detec_trans_position`, `init_data.detec_refl_position` and `init_data.detec_src_position` are used to define the planes at which the amplitude and the phase of the

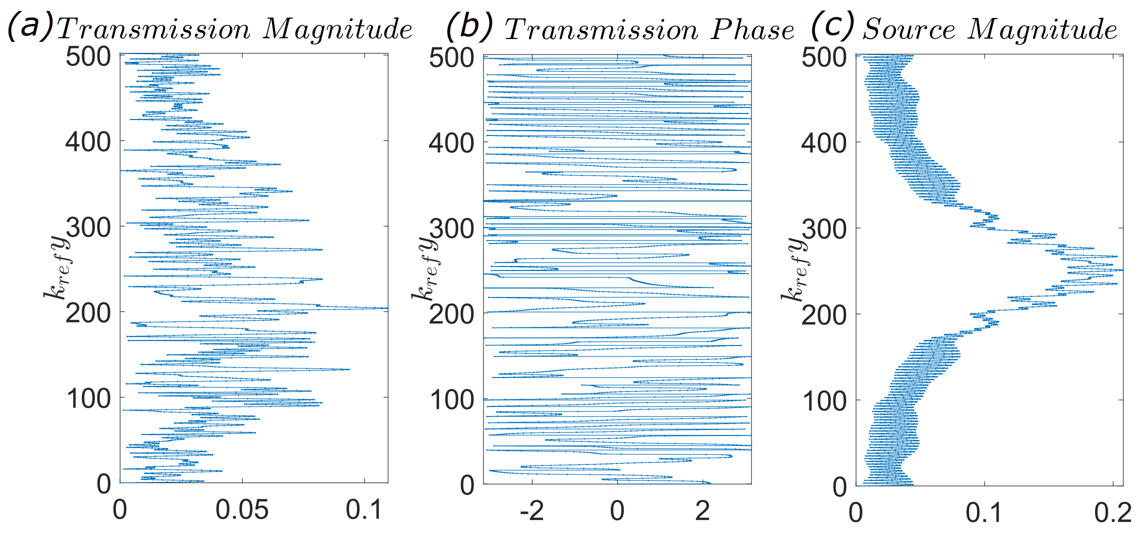

Figure 4: **Phase sensitive lock-in detection of the wavefronts** (a) The amplitude and (b) phase of the transmitted wave during the forward propagation scenario shown in figure 3(b). (c) Plotting just the amplitude of the source wave (without any disorder) to be incident at the slab surface shown in figure 3(b).

transmitted, reflected and the source waves are estimated. The function `phase_sensitive_detection(...)` contains the method for the phase sensitive detection as described in the main paper for estimating the amplitude and phase of the scattered waves as shown in figure 4. `phase_sensitive_detection(...)` returns the structure `PSD_array_collection`. It contains the following fields.

```
PSD_array_collection.E_trans_sig : Transmitted field amplitude
PSD_array_collection.theta_trans_sig : Transmitted field phase
PSD_array_collection.E_refl_sig : Reflected field amplitude
PSD_array_collection.theta_refl_sig : Reflected field phase
PSD_array_collection.E_source_sig : Transmitted source amplitude
PSD_array_collection.theta_source_sig : Transmitted source phase
PSD_array_collection.R : Total reflection from the slab
PSD_array_collection.T : Total transmission from the slab
```

Using these field values, the `PSD_array_collection.R` and `PSD_array_collection.T` are estimated from the Z component of the wave current as given in the main paper. The numerical implementation is given below.

2.3.1 Numerical estimation of the wave current for evaluating the total transmission and reflection

The wave current density $\vec{J}(r,t)$ at a given boundary Γ is given as

$$\vec{J}(r,t)|_{r\in\Gamma} = \left(\frac{-\partial E(r,t)}{\partial t}\nabla E(r,t)\right)|_{r\in\Gamma} \text{ where } E(r,t) = E_0(r)\cos\left(\omega_0 t + \phi(r)\right). \tag{1}$$

Here, the amplitude $E_0(r)|_{r\in\Gamma}$ and the phase $\phi(r)|_{r\in\Gamma}$ of the scattered wave are estimated by the phase sensitive detection method as described before. For estimating the total transmission and reflection, the magnitude of the Z component of the current density is used. Therefore,

$$J_z(r,t)|_{r\in\Gamma} = \left(\underbrace{\frac{-\partial E(r,t)}{\partial t}}_{\text{term 1}}\underbrace{\frac{\partial E(r,t)}{\partial z}}_{\text{term 2}}\right)|_{r\in\Gamma}, \tag{2}$$

where the term 1 and term 2 can be simplified using the expression for $E(r,t) = E_0(r)\cos(\omega_0 t + \phi(r))$. Using the first order centered difference scheme for the derivatives, $J_z(r,t)|_{r\in\Gamma}$ can be approximated as,

$$J(r,t)|_{r\in\Gamma} \approx \frac{\omega_0}{2\Delta_z}\left\{E_0(r)|_{r\in\Gamma}\sin(\omega_0 t + \phi(r))\right\}\left\{E_0\left(y_\Gamma, z_\Gamma + \Delta_z\right)\cos\left(\omega_0 t + \phi\left(z_\Gamma + \Delta_z\right)\right) - E_0\left(y_\Gamma, z_\Gamma - \Delta z\right)\cos\left(\omega_0 t + \phi\left(z_\Gamma - \Delta_z\right)\right)\right\}. \tag{3}$$

Hence, the total transmission and reflection can be estimated by temporally and spatially integrating $J(r,t)|_{r\in\Gamma}$ as given in the main paper. This completes the forward propagation associated with the first half of the main.m file. Next, the transmitted wave is phase conjugated from the right side.

2.4 Phase conjugation of the transmitted wave from the right side

For the phase conjugation of the transmitted wave from the right side as shown in figure 3(c), first the source placed on the right side of the slab has to be configured.

```
%------ phase conjugation : Right to Left ------------------------------
init_data.NT_steady_state=NT_steady_state_PC;
source_data.t_sign=-1; % Setting the sign flag for phase conjugation.
% -1 implies time-reversal
```

The sign flag `source_data.t_sign` has to be set to -1 for time-reversal operation of the time-stepping process. This is followed by the setting of the phase conjugate wave source position `source_data.src_z` to be the same position at which the transmission was measured during the forward propagation.

```
% Defining the dipole source location
source_data.src_z=[init_data.detec_trans_position init_data.detec_trans_position+1];
source_data.src_y=2:init_data.Ny+1; % Setting the active y region of the
% source
source_data.TFSF_Nz=floor((source_data.src_z(2)+init_data.Nz)/2);
% Setting the TFSF boundary location for the new source,
% this time for phase conjugation.
source_data.TFSF_region=source_data.TFSF_Nz:init_data.Nz; % Setting the
% TFSF region while performing phase conjugation
```

Next, the amplitude and phase of the phase conjugate wave is set by assigning the relevant values to the `source_data` structure. This is based on the transmitted field amplitude and phase values obtained during the phase sensitive detection of the forward propagation.

```
% Setting the dipole source amplitude for phase conjugation, which is
% based on the transmitted wave itself
source_data.wave_amp= ...
repmat(PSD_array_collection.E_trans_sig(2:end-1),1,2);

source_data.wave_amp= ...
0.1.*source_data.wave_amp./max(source_data.wave_amp);
% Normalising for better contrast.
% Next, set the dipole source phase for phase conjugation, which is
% based on the transmitted wave itself
source_data.wave_phase= ...
repmat(PSD_array_collection.theta_trans_sig(2:end-1),1,2);
source_data.wave_phase(:,2)=source_data.wave_phase(:,2)+pi;
% dipole phase factor pi is added to the second component of
% the dipole source for actually making it a dipole source.
source_data.clip_lim=gather(max(source_data.wave_amp(:)))+0.8;
%  clip_lim is the limit value for the colour map clip to
% adjust the contrast of the field images.
```

Next, the FDTD time stepping is performed for the phase conjugation part using the same time stepping function `FDTD_main_loop(...)` which gets the updated `source_data` structure being passed on to it. This is followed by the phase sensitive detection of the phase conjugate wavefront, similar to that of the forward propagation.

```
[Efield_array_collection] = ...
FDTD_main_loop(GPU_flag,visualisation_flag,init_data,source_data);
% Main FDTD time-stepping loop
%------------ Phase sensitive detection ------------------------------------
init_data.detec_trans_position=src_z_start;
% Setting the transmission detection location
% same as that of the original point source location
init_data.detec_refl_position=floor((source_data.TFSF_Nz+init_data.Nz)/2);
% Reflection detection location
% Set as half way between the TF/SF boundary and the right boundary
init_data.detec_src_position= ...
init_data.start_slab1+init_data.scat_width+1;
% Source transmission detection location
% Set as the RHS incident surface of the slab

% Next, obtain the PSD for the phase conjugation process and estimate
% the associated total transmission and reflection
PSD_array_collection=phase_sensitive_detection(GPU_flag,visualisation_flag,...
Efield_array_collection,init_data,source_data);
```

Finally, the field intensity profile at the focal region during the phase conjugation is plotted as shown in figure 5.

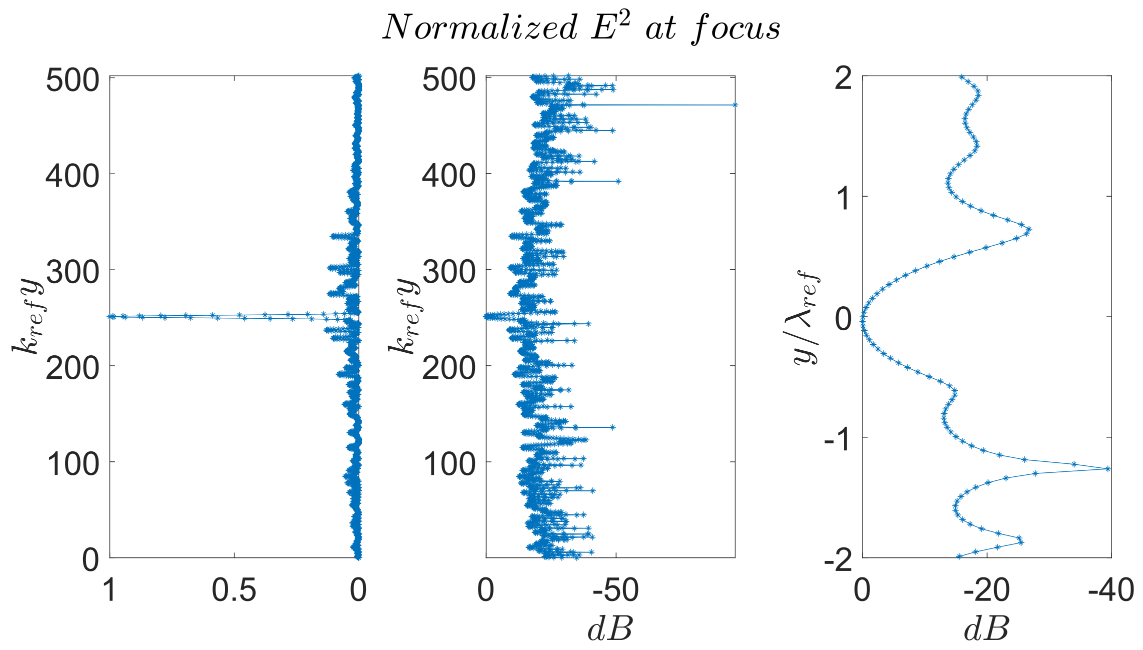

Figure 5: **The same field intensity profile at the focal region (during the phase conjugation shown in figure 3(c))) plotted in different scales.**

3 Code-Package-2

In this section the code structure of Code-Package-2 is discussed, which is built on the top of Code-Package-1. Hence, many code units contained in Code-Package-1 find its use for other code packages as well. So, the discussion of the code units which we already did for Code-Package-1 is avoided in this section. Hence, new code units associated with the STM modulation are covered here. The file structure of Code-Package-2 is

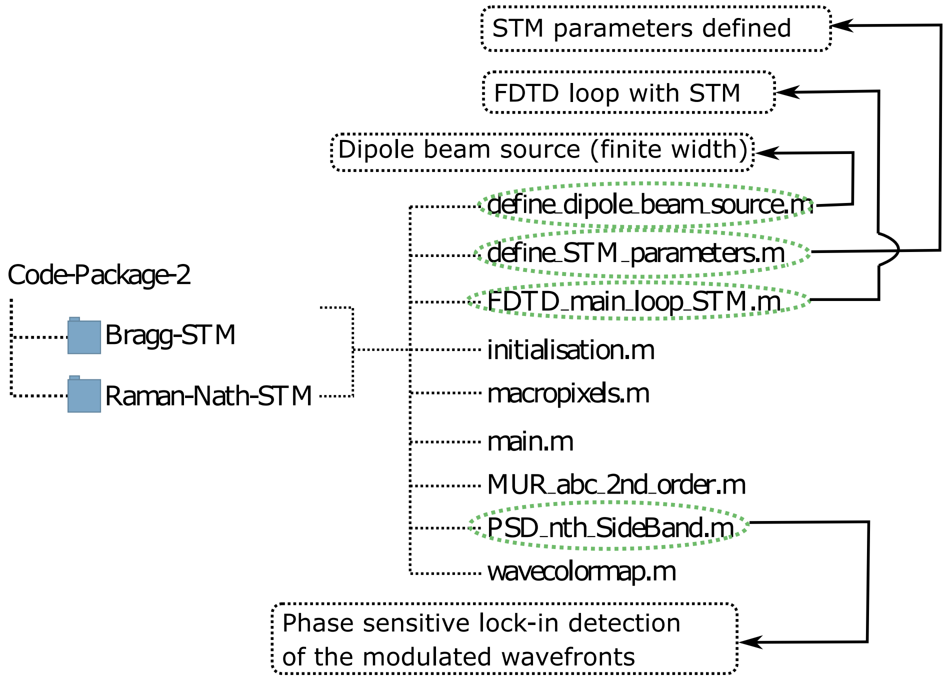

Figure 6: **The file structure of Code-Package-2**. This code-package demonstrates the two STM regimes without any disorder present. The initialized STM parameters are used by the main FDTD loop with time varying dielectric constant. Phase sensitive lock-in detection scheme is utilized to estimate the modulated wavefronts.

given in the figure 6. This is to model the phase sensitive lock-in detection of the Bragg vs Raman-Nath diffraction without any disorder present in the computational domain. Therefore, sigma is set to zero as

```
sigma=0.0;               % A parameter, usually < 1 used to control the
%       degree of random scattering.
% sigma is set to zero as there is no disorder slab in this case.
```

Although there is no need of a slab disorder to be defined here, an empty slab is defined with zero scattering so that the code could be reused for the next code package by bringing in a slab disorder in front of the Raman-Nath STM. Both the Bragg-STM and Raman-Nath-STM folders have the same set of files listed in figure 6. The main.m file has the following STM initialization part for the Raman-Nath diffraction,

```
%—————————————————————————— STM initialisation ——————————————————————————
del_eps=0.5; % dielectric constant modulation by a space—time modulated curve
spat_freq_fact=0.4;          % spatial frequency of modulation
f_mod=0.2*(init_data.fsrc);  % temporal frequency of the modulation
% keep the fraction less than or equal to 0.2
grat_width=ceil(3.0*pi/(init_data.kref*init_data.dz)); % space—time modulated
                                        %      grating width
```

and the following for the Bragg diffraction,

```
 %—————————————————————————— STM initialisation ——————————————————————————
 del_eps=0.068; % dielectric constant modulation by a space−time modulated curve
 spat_freq_fact=0.73; % spatial frequency of modulation
 f_mod=0.2*(init_data.fsrc);  % temporal frequency of the modulation
 % keep the fraction less than or equal to 0.2
 grat_width=ceil(88/(init_data.kref*init_data.dz)); % space−time modulated grating width
```

For STM diffraction, an incident beam of finite width is used as shown in the main paper for which the following function is used.

```
%−−−−−−−−−−−−−− Define the source −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
[source_data] = define_dipole_beam_source(GPU_flag,init_data);
```

Temporal diffraction order numbers can be selected by setting the values of `norder_array`. As an example, for the Raman-Nath regime,

```
norder_array=[+3,+2,+1,0,−1];  % Various modulated temporal order number
```

which is given as an input for performing the phase sensitive detection of the modulated wavefronts as the following.

```
for ncount=1:length(norder_array)
nth=norder_array(ncount);
PSD_array_collection=PSD_nth_SideBand(GPU_flag,visualisation_flag,...
Efield_array_collection,init_data,source_data,STM_data,nth);
title(sprintf('$Phase~sensitive~detection~: \\bf n=%d$',nth),'Interpreter','Latex')
PSD_structure_collection(ncount)=PSD_array_collection;
end
```

4 Code-Package-3

Code-Package-3 is similar to that of Code-Package-2, except that a slab disorder is placed in front of the Raman-Nath STM grating and phase conjugation is performed. Here, the STM grating gets directly excited by the incident beam and the modulated wavefront propagates through the disorder as described in the main paper. This is followed by the phase conjugation of the upconverted wavefront and the downconverted wavefront in two separate runs. If one needs to profile the transmission of the disordered slab (without any STM modulation) one may set the flag `estimate_disorder_transmission_flag` to unity.

```
estimate_disorder_transmission_flag=1; % Set the flag 1 if the transport
%               regime of the disorder needs to be
%               profiled.

if estimate_disorder_transmission_flag==1
estimate_disorder_transmission % Estimate the disorder transmission
                                % conditionally
figure(2)
    title('$Forward~propagation~without~STM$','Interpreter','Latex')
end
```

5 Code-Package-4

Code-Package-4 performs iterative optical phase conjugation (IOPC). It has two subfolders as shown in figure 1 corresponding to two different scenarios. Both uses the same set of files with the only change being the geometry of the problem. The first part addresses IOPC when the STM is placed in between the two slabs. In the second part, the two slabs with increased scattering are merged together with the Raman-Nath STM region embedded inside the the combination of the two slabs. This is to study the role of multiple scattering on IOPC through an STM region embedded inside a disordered medium in the diffusion regime. The associated common file structure is shown in figure 7. The scripts phase_conjugation_STM_Right_to_left and phase_conjugation_STM_Right_to_left are looped in main.m for performing IOPC as the following.

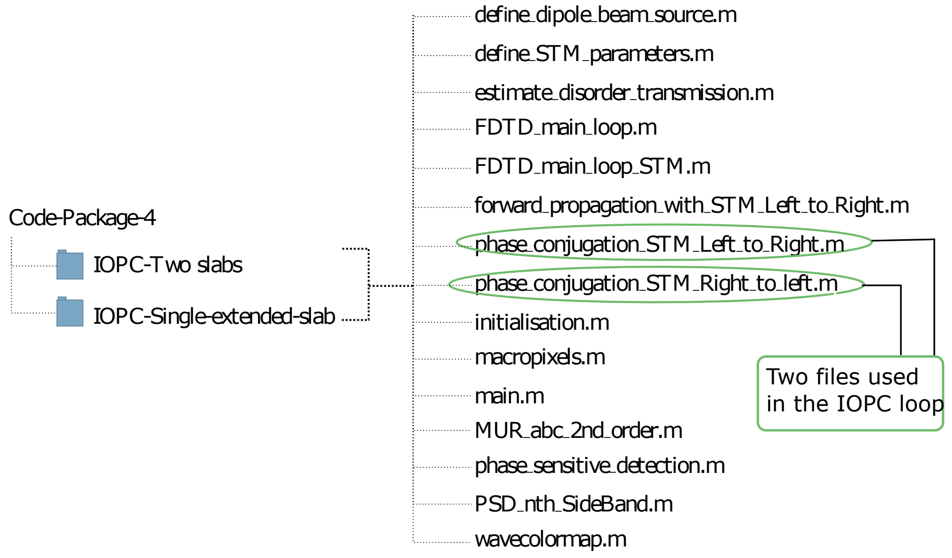
Code-Package-4
IOPC-Two slabs
IOPC-Single-extended-slab
define_dipole_beam_source.m
define_STM_parameters.m
estimate_disorder_transmission.m
FDTD_main_loop.m
FDTD_main_loop_STM.m
forward_propagation_with_STM_Left_to_Right.m
phase_conjugation_STM_Left_to_Right.m
phase_conjugation_STM_Right_to_left.m
initialisation.m
macropixels.m
main.m
MUR_abc_2nd_order.m
phase_sensitive_detection.m
PSD_nth_SideBand.m
wavecolormap.m
Two files used in the IOPC loop

Figure 7: **The file structure of Code-Package-4 for performing iterative optical phase conjugation (IOPC) through a Raman-Nath STM.**

```
%------------------------------ Perform IOPC ------------------------------------
for r_count=1:no_of_roundtrips
%---- Section 3 : Phase conjugation of the upper sideband from the right --
nchosen_R2L=+1; % Choose the upconverted wavefront
% for the phase conjugation from the right
phase_conjugation_STM_Right_to_left
% The script used to perform phase conjugation from right to left, in presence
% of the STM modulation and detect the n=+1 sideband (upper).

%----- Section 4 : Phase conjugation of the omega_0 wavefront from the left
nchosen_L2R=0; % Choose the center frequency omega_0 corresponding to
% n=0, to be phase conjugated from the left.
% Such a wavefront at omega_0, actually emerged on the left,
% due the STM modulation of the phase conjugated upperside band.
% The STM region, downconverted the phase conjugated upperside band from the
% right, into the center frequency, while passing through the STM region.
phase_conjugation_STM_Left_to_Right;
end
```