Propagator.cpp Manual

David Hoffmann, Imperial College, Aug 2013

Contents

1. Introduction

This document goes some way to describing the Propagator.cpp code (written in C++). Before using this code it is necessary to read the companion document Propagation.pdf with describes the theory behind the propagation model in depth. Also included is a few pages describing the strong-field approximation (SFA) and quantum orbit (QO) models for calculating single-atom dipole-response.

2. Code Structure

This section gives a brief overview of the source files which comprise the Propagator code.

List of source files included:

Propagator.cpp
Constants_Conversions.h
Diffraction.h
Dipole_Response.h
Electric_Field.h
Environment.h
Far_Field.h
Gas_Jet.h
Input.h
Laser_Parameters.h
Misc_Functions.h
Output.h
Plasma_Defocussing.h
QO_Response.h
SFA_Response.h

To give a brief description of these source files (in rough order of use):

Constants_Conversions.h
Basic numerical constants and conversions between S.I. and atomic units

Misc Functions.h
Basic mathematical functions and defines MPI interfaces

Input.h
Defines functions to read input parameters and options from ControlBox.conf

Environment.h
Defines classes to create spatial and temporal axes, together with absorbing boundary conditions on these axes

Gas_Jet.h

Build the Gaussian gas jet density profile along the propagation axis (constant perpendicular to this axis due to cylindrically symmetric model). Also contains functions to calculate current populations based upon ADK ionisation rate and functions to calculate harmonic reabsorption by the neutral gas.

Laser_Parameters.h
Defines laser_parameters class to hold laser parameters for the number of colours chosen. Function to determine the number of dimensions required for the resultant electric field.

Electric_Field.h
Defines parent electric_field class and child electric_field_1d, electric_field_2d classes to initialise the electric fields for propagation.
electric_field_1d is used when only a single colour is selected or if (i) only a single colour if given non-zero intensity or (ii) all colours are polarised along the same axes. In this latter case all colours are realigned to be polarised along the x axis.
electric_field_2d is used when multiple colours (limited to 2 currently) are specified with different polarisation axes. In this case we propagate the components of the total field polarised along the x and y axes separately (recalculating the total field at each propagation step when necessary).
Contains constructors with and without laser_parameters object as argument, in order to build analytic laser field (Gaussian spatial profile) and flat harmonic field prior to propagation.

Diffraction.h
Functions to simulate the diffraction of the laser and harmonic fields via a Crank-Nicolson scheme.

Plasma_Defocussing.h
Functions to simulate plasma dispersion and blue-shift due to the presence of free electrons in the interaction region due to the action of the incident laser field.

Dipole_Response.h
Defines dipole_response class to act as an interface for the single-atom dipole response as calculated by SFA or QO methods.

SFA_Response.h
Defines child sfa_response_1d and sfa_response_2d (for electric_field_1d, electric_field_2d respectively) classes to calculate the single-atom dipole response via the SFA method.

QO_Response.h
Defines child qo_response_1d and qo_response_2d (for electric_field_1d, electric_field_2d respectively) classes to calculate the single-atom dipole response via the QO method.

Far_Field.h
Calculates the harmonic far-field analytically via a Hankel transform of the near-field, either directly or with an Abel integral method.

Output.h
Miscellaneous output functions not placed in other header files.

Propagator.cpp
The main source file initialising class objects, file streams and calls functions from the header files described above for use in the propagation.
Please refer to the companion document Propagation.pdf for a full description of the theory behind the propagation model.

Figure 1 shows a basic dependencies map for fundamental source files. Note that any of these source files may also rely on any of Constants_Conversions.h, Misc_Functions.h, Input.h & Output.h (see first few lines within each file).

3. Software Requirements

Required:

1. C++ compiler
2. FFTW libraries - used in all fourier transforms: see http://www.fftw.org/
3. toms838 library - contains complex Airy function and its derivative for use in QO model
   see http://dl.acm.org/citation.cfm?id=1039819
4. FORTRAN compiler to interpret the toms838 library

Stongly recommended:

1. MPI libraries - for parallelisation options: see http://www.open-mpi.org/
2. Make utility - to compile & run

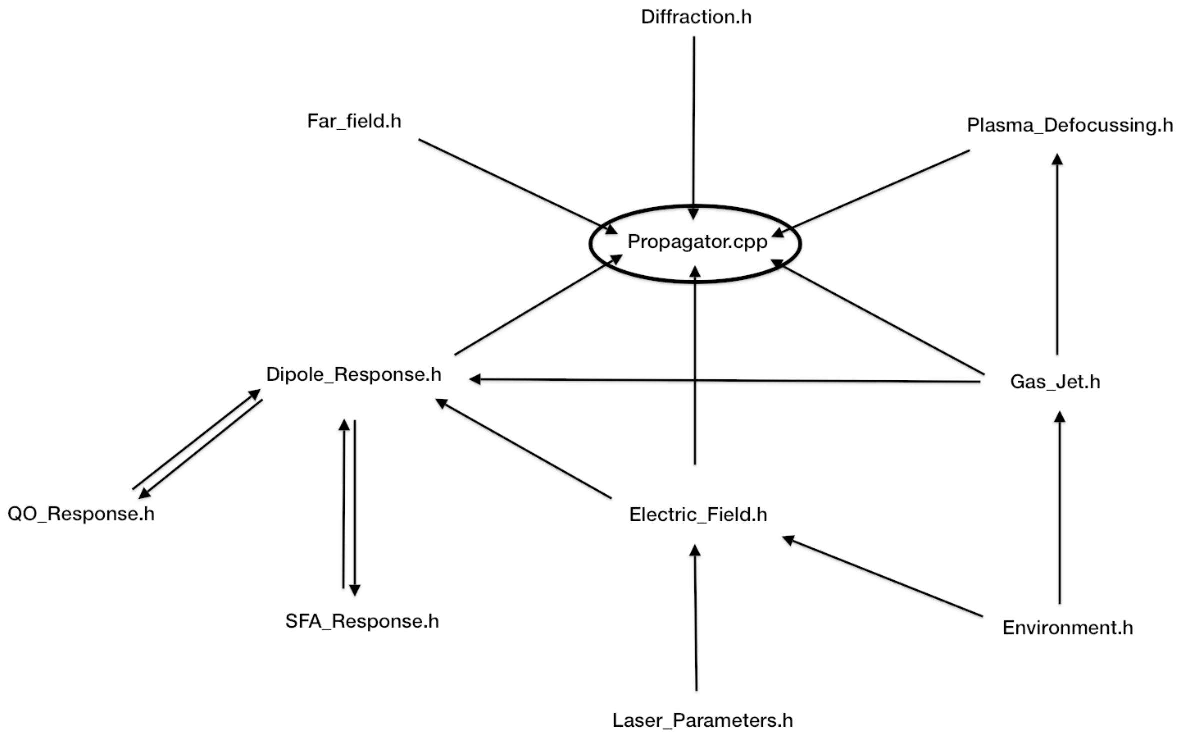

Figure 1. Dependency map of Propagator.cpp

4. Compiling & Execution (linux/unix specific)

This section describes how to compile and run the code on a local linux machine or the Imperial College HPC cluster.

Local machine:

To run the code on a local machine you will need to edit & use supplied gcc Makefile (see figure 2).
You will need to edit the DIR path to match with you run directory.
Without MPI you are limited to **make it** and **make run** commands to compile and run on a single core**.**
With MPI use **make mpi** to compile and **make runN** to run on N cores (N=8 in the example below - edit if necessary).
For open-MPI versions prior to 1.3.2 (not recommended) you will need to add a -DoldMPI flag to the mpi compile line.

```
# GCC Makefile
# =================

DIR = /Users/dhoffman/Work/Code/Propagation/
TARGET = Propagator

it: $(DIR)$(TARGET).cpp
		g++ $(DIR)$(TARGET).cpp -o $(TARGET).x -I$(DIR)Include/ -L$(DIR)Libraries/ -lm -lfftw3 -lgfortran -lairy838 -O3

mpi: $(DIR)$(TARGET).cpp
		mpicxx $(DIR)$(TARGET).cpp -o $(TARGET).x -I$(DIR)Include/ -L$(DIR)Libraries/ -lm -lfftw3 -lgfortran -lairy838 -DuseMPI -O3

run: $(TARGET).x
		rm -f *.dat ;
		./$(TARGET).x

run8: $(TARGET).x
		rm -f *.dat ;
		mpiexec -n 8 ./$(TARGET).x
```

Figure 2. Example of a Makefile for use on a local (8 core) machine.

Imperial College HPC cluster:

To run the code on the Imperial College HPC you will need to gain access by contacting Simon Burbidge via s.burbidge@imperial.ac.uk You can then use the executable file supplied.

Alternatively, to compile the code yourself you can use the Makefile shown in figure 3.

```
# GCC Makefile
# =================

TARGET = Propagator
DIR = $(HOME)/Code/Propagation/
INCLUDE = $(DIR)Include/
LIBS = $(DIR)Libraries/

mpi: $(DIR)$(TARGET).cpp
		icc $(MPI_LIBS) -O2 -axT -xT -i-static $(DIR)$(TARGET).cpp -o $(TARGET).x -I$(INCLUDE) -I/apps/fftw/3.2.2-double/include/ -L$(LIBS) -L/usr/lib64/ -L/apps/fftw/3.2.2-double/lib/ -I$(MPI_INCLUDE) -lgfortran -lairy838 -lfftw3 -lm -DuseMPI
```

Figure 3. Example of a Makefile for use on the Imperial College HPC cluster.

Before compiling you will first need to load the relevant modules via
module load intel-suite
module load fftw
module load mpi (if you are using the cluster then you will be using the parallel version!)

After compiling the code you will need to submit jobs to the cluster queue rather than running them directly. To do this we use the qsub routine in conjunction with the Propagator.pbs file (see figure 4) through the command **qsub Propagator.pbs**

```
#!/bin/sh
#PBS -q pqlaser
#PBS -l walltime=5:00:00
#PBS -l select=1:ncpus=8:mem=1000mb
#PBS -m abe

module load intel-suite
module load mpi
module load fftw

rundir=$PBS_O_WORKDIR

rm $rundir/*.dat
rm $rundir/*.pbs.*

mpiexec $rundir/Propagator.x -C $rundir/Config/ > $rundir/console.out

cp -a $TMPDIR/*.dat $rundir/
pbsdsh2 cp -a "$TMPDIR/*_C*.dat $rundir/"
```

Figure 4. Example of the Propagator.pbs file for use on the Imperial College HPC cluster.

The requested walltime, number of cores and memory per core is specified in the top section of the file. The running output of the code is piped into the file console.out in the run directory.

5. Input Parameters

This section describes the procedure for specifying input parameters for the code.
Input parameters located in file [RunDir]/Config/**ControlBox.conf** (see figure 5).
* changes to this path require editing of path for -C option in run command line (if present)

```
%  ---------------------------------------------------
%       Input parameters for Propagator(...).cpp
%  ---------------------------------------------------

%  compile and run using Makefile in RunDir subdirectory

%  ---------------------------------------------  Define Grid  ---------------------------------------------------

            Nr = 128                                 %  number of radial points
            rMax = 120.0e-6          [m]             %  maximum radial value

            Nz = 100                                 %  number of points along propagation axis
            zMin = -1.0e-3           [m]             %  initial z value, relative to centre of the gas jet
            zMax = 1.0e-3            [m]             %  final z value, relative to centre of the gas jet
            zRel = y                                 %  y if z is defined relative to centre of the gas jet; n if otherwise
            zFar = 1.0               [m]             %  near- to far-field propagation length (through vacuum)

            Nt = 8192                                %  number of temporal points (currently must be divisible by mpi_size)
            tRange = 150.0e-15       [s]             %  total temporal duration (centred at t = 0)

            rAB = 0.9                                %  fraction of maximum radius at which absorbing boundary appears
            tAB = 0.85                               %  fraction of temporal duration at which absorbing boundary appears

%  ------------------------------------------  Gas Jet Parameters  ------------------------------------------------

            zj = 2.0e-3              [m]             %  jet centre along propagation axis
            zw = 0.3e-3              [m]             %  jet full-width half-max along propagation axis
            rj = 2.5e-3              [m]             %  jet radial distance off-axis
            nd0 = 1.0e16             [cm^-3]         %  peak density (cm^-3 -> au^-3)

            GasName = Ar                             %  gas particle type (name or symbol) - dictates X-Ray absorption filename
            nGS = 1                                  %  n quantum number of ground state
            lGS = 0                                  %  l quantum number of ground state
            mGS = 0                                  %  m quantum number of ground state

%  ------------------------------------------  Laser Parameters  ------------------------------------------------

            colours = 1                              %  number of components comprising the driving field
                                                     %  program will seek this number of components below

%           ---- colour1 ----        ---- colour2 ----

            I0 = 1.4e14              I0 = 0.0e14              [Wcm^-2]            %  peak intensity
            lambda = 800.0e-9        lambda = 400.0e-9        [m]                 %  central wavelength
            Ifwhm = 30.0e-15         Ifwhm = 30.0e-15         [s]                 %  full-width half-max of intensity envelope
            t0 = 0.0e-15             t0 = 0.0e-15             [s]                 %  time at envelope peak
            cep = 0.0                cep = 0.0                [pi rad]            %  carrier envelope phase
            envshape = c             envshape = g                                 %  shape of field envelope (Gaussian, cos^2, sec)
            zFocus = 0.0e-3          zFocus = 0.0e-3          [m]                 %  focal position
            beamwaist = 40.0e-6      beamwaist = 20.0e-6      [m]                 %  beam waist
            polAxis = 0.0            polAxis = 0.5            [pi rad]            %  polarisation axis (polar coords)

%  ------------------------------------------  Control Parameters  ----------------------------------------------

            Propagate = y                            %  perform propagation (debugging)? y/n

            WhereLaserDef = j                        %  laser parameters valid at: f=focus, j=centre of jet, i=initial z

            RKOrder = 4                              %  specifies order of Runge-Kutta algorithm in free-electron dispersion step

            minI0 = 1.0e13    [Wcm^-2]               %  specifies min (temporal-peak) intensity for which dipole response is calculated
            DRMethod = S                             %  specifies method for calculating dipole response: S=SFA, Q=QuantumOrbit

            SFAcycles = 2.0                          %  specifies maximum excursion time for SFA in laser cycles

            QOtrackAll = y                           %  track central wavelength and full-width half-max (as well as intensity, phase and peak time)
            QOmethod = U                             %  method for calculating quantum orbit response (S=saddle-point, U=uniform, C=combined)
            QOtraj = B                               %  which orbits to track (S=short, L=long, B=both, A=all - includes low-energy paths)
            QOcolour = 0                             %  sets which colour's temporal period the subsequent four parameters are defined with respect to
            QOminIon = 0                             %  sets earliest ionisation time in terms of laser cycles wrt envelope peak
            QOmaxIon = 0                             %  sets latest ionisation time in terms of laser cycles wrt envelope peak
            QOminTau = 0                             %  sets minimum excursion time in terms of laser cycles
            QOmaxTau = 1                             %  sets maximum excursion time in terms of laser cycles

            FPMethod = D                             %  specifies method for far-field propagation: D=DirectIntegration, A=AbelIntegral

%  --------------------------------------------  Output Options  ------------------------------------------------

            OutFilter = 8                            %  specifies temporal density for temporospatial outputs (every nth point)

            PulseTime = n                            %  dictates whether driving pulse outputted in time domain through z
            PulseFreq = n                            %  dictates whether driving pulse outputted in frequency domain through z
            HarmFreq = n                             %  dictates whether harmonics outputted in frequency domain through z

            OutFreqCmpts = y                         %  outputs intensity & phase of individual frequencies vs r & z (see below)
            NFreqs = 4                               %  number of frequency components outputted
            FreqRef = 1                              %  reference point for min/max freqs (0=wMax, n>0=classical cut-off freq of colour n)
            minFreq = 0.25                           %  min frequency wrt above reference
            maxFreq = 1.0                            %  max frequency wrt above reference
            FreqShift = o                            %  shift frequencies (n = no change, h/o/e = nearest harmonic/odd harmonic/even harmonic)

%  ---------------------------------------------------------------------------------------------------------------
```

Figure 5. Example of ControlBox.conf containing the input parameters for Propagator.

A brief description of each of the options is given on the right-hand side of ControlBox.conf

The first section titled **Define Grid** is for parameters which are used to create the environment of the propagation. If *zRel* is set to 'y' then the propagation axis will run from $z_S = zj\text{-}zMin$ to $z_F = zj\text{-}zMax$ (where *zj* is specified in the Gas Jet Parameters section). Otherwise the propagation axis simple runs from $z_S = zMin$ to $z_F = zMax$.

The **Gas Jet Parameters** section specifies parameters used to create the Gaussian density profile to model the gas jet.

The **Laser Parameters** section first specifies the number of colours whose parameters are searched for the code. The subsequent columns are then scanned to input these parameters to form the Gaussian beam. If only one column has non-zero intensity then a single-colour field is formed irrespective of the value of the *colours* parameter. Alternatively, if all colours are found to have the same polarisation axis then all are realigned along the x-axis (polAxis = 0) and a one-dimensional electric field is propagated.

Additional *----colourN----* columns may be added when *colours* > 2 (only if they are all polarised along the same axis).

The **Control Parameters** section lists various options controlling methods used in the propagation.
The *WhereLaserDef* option controls where the previously listed laser parameters are valid. If set to 'f' then the electric field at z_S is calculated analytically via the Gaussian beam equations. If *WhereLaserDef* = 'j' then the field is diffracted from *zj* to *zf* and reanalysed to obtain the laser parameters at the focus. Note that at high gas jet densities and/or very intense laser fields there will be a high free electron density across the interaction region. This means will that the field at the centre of the gas jet will significantly differ from the field described by the parameters in ControlBox.conf

The QO options specify the range across quantum orbit label as specified in SFA_QO.pdf

The code outputs various the laser and harmonic fields in various forms at the end of the interaction region, and in the far-field. The **Output Options** section controls additional outputs where the fields are tracked through the interaction region. *PulseTime* specifies whether the complete radially and temporally dependent laser field is outputted in the time domain. *PulseFreq* does the same for the frequency domain, as does *HarmFreq* for the harmonic field. In each case the temporal resolution is lowered by a factor of *OutFilter*.

The next set of parameters allow for the output of specific energies through the interaction region in (r,z) coordinates. *Nfreqs* energies are outputted, equally spaced from $minFreq{*}E_{ref}$ to $maxFreq{*}E_{ref}$ where E_{ref} is (according to *FreqRef*) either the maximum frequency simulated or the classical cut-off given by the parameters of a particular colour given in the Laser Parameters section. The *FreqShift* option shifts the selected energies to equal the closest harmonic / odd harmonic / even harmonic of the colour specified by *FreqRef.*

6. Outputs

This section discusses the output files provided by the code.

Full list of outputted files (all outputs in atomic units):

AxisR.dat	radial (r-)axis for the interaction region
AxisRp.dat	linear radial axis and divergence at the far-field position
AxisTp.dat	temporal axis
AxisW.dat	corresponding frequency axis
AxisZ.dat	propagation (z-)axis
GasJet.dat	Gaussian gas jet density profile along z-axis

Each of the following filenames are of the form [...]X.dat, indicating fields polarised along the x-axis. For two-dimensional propagation, each will have a corresponding [...]Y.dat file giving the fields polarised along the y-axis.

HarmFarRTpX.dat	Real component of the harmonic far-field vs (t,rp)
HarmFarRWnormX.dat	Norm of the harmonic far-field vs (rp,w)
HarmFarRWphaseX.dat	Phase of the harmonic far-field vs (rp,w)
HarmNearRTpX.dat	Real component of the harmonic near-field (at z_F) vs (t,r)
HarmNearRWnormX.dat	Norm of the harmonic near-field vs (r,w)
HarmNearRWphaseX.dat	Phase of the harmonic near-field vs (r,w)
PulseRTpX.dat	Real component of the laser near-field vs (t,r)
PulseRWnormX.dat	Norm of the laser near-field vs (r,w)
PulseRWphaseX.dat	Phase of the laser near-field vs (r,w)
PulseRZX.dat	Real component of laser near-field at t=0 vs (r,z)
PulseTpX.dat	Real component of on-axis (r=0) laser near-field vs t
PulseWnormX.dat	Norm of on-axis laser near-field vs w
PulseWphaseX.dat	Phase of on-axis laser near-field vs w
PulseZTpX.dat	Real component of on-axis laser near-field vs (t,z)

Note that in the time domain we output spatially dependent fields with radial dependence along row number and temporal dependence along column number. Conversely in the frequency domain radial dependent is along column number and frequency dependence along row number. This is done to more efficiently parallel the calculation across multiple cores.

Optional outputs:

For these outputs use options in ControlBox.conf

HarmNearRZ[N]eVnormX.dat	Norm of energy [N]eV of the harmonic near-field vs (r,z)
HarmNearRZ[N]eVphaseX.dat	Phase of energy [N]eV of the harmonic near-field vs (r,z)
HarmNearRZWnormX_C[N].dat	Norm of harmonic near-field vs (r,w,z), w-axis split across cores 0 to N-1
HarmNearRZWphaseX_C[N].dat	Phase of harmonic near-field vs (r,w,z), w-axis split across cores 0 to N-1
PulseNearRZTpX_C[N].dat	Real component of laser near-field vs (t,r,z), r-axis split across cores 0 to N-1
PulseNearRZWnormX_C[N].dat	Norm of laser near-field vs (r,w,z), w-axis split across cores 0 to N-1
PulseNearRZWphaseX_C[N].dat	Norm of laser near-field vs (r,w,z), w-axis split across cores 0 to N-1

For plotting outputted data, see supplied MATLAB scripts **PlotPulse.m** and **Wavelet.m** Figure 6 shows an example of the spatially-resolved far-field spectrum produced from by the stated laser and gas jet parameters. The SFA method was used to calculate the single-atom dipole response in this case.

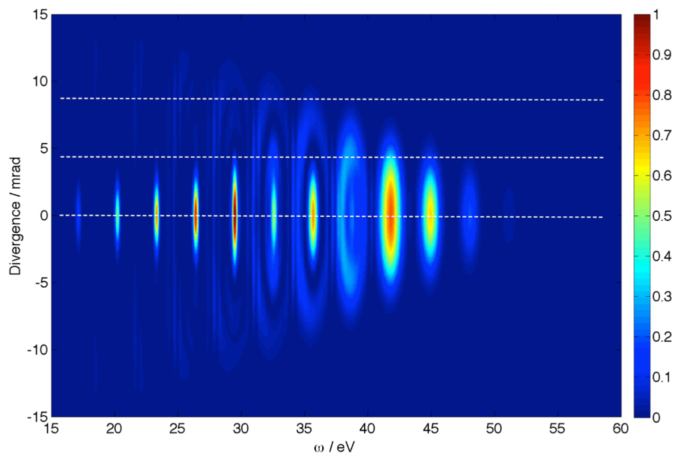

Figure 6. Far-field harmonic spectrum for a 30μm (fwhm), 10^{16} atoms/cm^3 Argon gas jet situated 1mm downstream of the focus of an 800nm, 2x10^{14}Wcm^{-2}, 20fs laser pulse focused to 30μm.

Figure 7 shows the results of wavelet analyses of figure 6, through radial cuts at 0 and 8mrad.

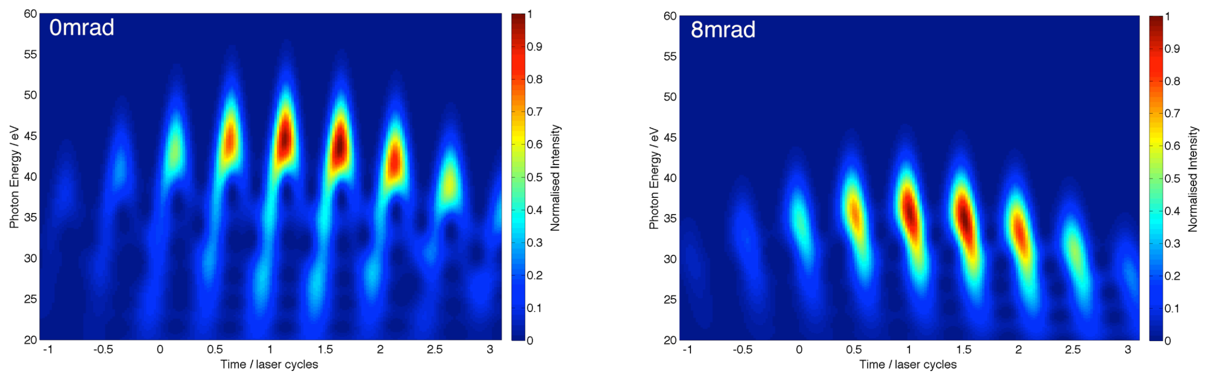

Figure 7. Wavelet analyses of figure 6 through radial cuts at 0 and 8mrad.

For reference the console output for a successful propagation run is given in figure 8.

```
Found 2 colour(s) ... creating 2-dimensional laser field
Pulse energy at z = -1mm is calculated as E = 0.123757 mJ
colours = 2
I0 = 1.4e14Wcm^-2, lambda = 800nm, t0 = 0fs, fwhm = 30fs, cep = 0pi rad
I0 = 0.7e14Wcm^-2, lambda = 400nm, t0 = 0.47fs, fwhm = 30fs, cep = 0pi rad
I0 = 1.36825e14Wcm^-2, lambda = 800nm, t0 = -0.0732511fs, fwhm = 30fs, cep = -0.105626pi rad
I0 = 0.551693e14Wcm^-2, lambda = 400nm, t0 = 0.439507fs, fwhm = 30fs, cep = -0.129158pi rad
I0 = 1.36825e14Wcm^-2, lambda = 799.988nm, t0 = -0.0732511fs, fwhm = 30.0134fs, cep = -0.105626pi rad
I0 = 0.551693e14Wcm^-2, lambda = 399.99nm, t0 = 0.439507fs, fwhm = 29.991fs, cep = -0.129158pi rad

Commencing propagation through target medium:
Pulse energy at z = 0mm is calculated as E = 0.123757 mJ (numerically) & 0 mJ (estimate)
I0 = 1.36959e14Wcm^-2, lambda = 800nm, t0 = -0.0732511fs, fwhm = 30fs, cep = -0.104627pi rad
I0 = 0.554923e14Wcm^-2, lambda = 400nm, t0 = 0.439507fs, fwhm = 30fs, cep = -0.127702pi rad
I0 = 1.36959e14Wcm^-2, lambda = 799.988nm, t0 = -0.0732511fs, fwhm = 30.0161fs, cep = -0.104627pi rad
I0 = 0.554923e14Wcm^-2, lambda = 399.99nm, t0 = 0.439507fs, fwhm = 29.9916fs, cep = -0.127702pi rad
Propagation runtime initially estimated as 5m23s
Propagation 5% complete in 0m20s     ...          time remaining estimated as 5m13s
Propagation 10% complete in 0m36s    ...          time remaining estimated as 4m51s
Propagation 15% complete in 0m53s    ...          time remaining estimated as 4m38s
Propagation 20% complete in 1m9s     ...          time remaining estimated as 4m19s
Propagation 25% complete in 1m26s    ...          time remaining estimated as 4m4s
Propagation 30% complete in 1m43s    ...          time remaining estimated as 3m49s
Propagation 35% complete in 2m0s     ...          time remaining estimated as 3m33s
Propagation 40% complete in 2m16s    ...          time remaining estimated as 3m15s
Propagation 45% complete in 2m33s    ...          time remaining estimated as 2m59s
Propagation 50% complete in 2m49s    ...          time remaining estimated as 2m42s
Pulse energy at z = 0.010101mm is calculated as E = 0.123752 mJ (numerically) & 0 mJ (estimate)
I0 = 1.40185e14Wcm^-2, lambda = 800nm, t0 = 0fs, fwhm = 30fs, cep = 0.00231885pi rad
I0 = 0.708359e14Wcm^-2, lambda = 400nm, t0 = 0.476132fs, fwhm = 30fs, cep = 0.0122476pi rad
I0 = 1.40185e14Wcm^-2, lambda = 799.955nm, t0 = 0fs, fwhm = 30.0031fs, cep = 0.00231885pi rad
I0 = 0.708359e14Wcm^-2, lambda = 399.996nm, t0 = 0.476132fs, fwhm = 29.9909fs, cep = 0.0122476pi rad
Propagation 55% complete in 3m6s     ...          time remaining estimated as 2m26s
Propagation 60% complete in 3m23s    ...          time remaining estimated as 2m9s
Propagation 65% complete in 3m40s    ...          time remaining estimated as 1m53s
Propagation 70% complete in 3m58s    ...          time remaining estimated as 1m37s
Propagation 75% complete in 4m15s    ...          time remaining estimated as 1m20s
Propagation 80% complete in 4m33s    ...          time remaining estimated as 1m4s
Propagation 85% complete in 4m50s    ...          time remaining estimated as 0m47s
Propagation 90% complete in 5m8s     ...          time remaining estimated as 0m30s
Propagation 95% complete in 5m26s    ...          time remaining estimated as 0m13s
Pulse energy at z = 0.959596mm is calculated as E = 0.12375 mJ (numerically) & 0 mJ (estimate)
I0 = 1.36804e14Wcm^-2, lambda = 800nm, t0 = 0.0366256fs, fwhm = 30fs, cep = 0.077141pi rad
I0 = 0.777402e14Wcm^-2, lambda = 400nm, t0 = 0.512758fs, fwhm = 30fs, cep = 0.160707pi rad
I0 = 1.36804e14Wcm^-2, lambda = 799.979nm, t0 = 0.0366256fs, fwhm = 30.0379fs, cep = 0.077141pi rad
I0 = 0.777402e14Wcm^-2, lambda = 400.001nm, t0 = 0.512758fs, fwhm = 30.0248fs, cep = 0.160707pi rad
Propagation completed in 5m36s

Commencing propagation to far-field ... completed in 0m3s

Outputting remaining data files
Freeing up memory & exiting
Total time elapsed = 5m49s
```

Figure 8. Console output for Propagator.cpp