BOPS User Guide

Version: 3.3

Contributors:

Paul Gibbon
Anupam Karmakar
Bin Qiao

Jülich Supercomputing Centre,
Forschungszentrum Jülich GmbH
D–52425 Jülich, Germany

21 April, 2011

Contents

1 Introduction

BOPS is a one-and-three-halves (1 spatial, 3 velocity coordinates: 1D3V) particle-in-cell code originally created by this author together with Tony Bell in the Plasma Physics Group of Imperial College, London. Based on standard algorithm for a 1D electromagnetic PIC code (Birdsall & Langdon), BOPS employs a Lorentz transformation, or 'boost' along the target surface to mimic the standard 2D, periodic-in-y geometry common to much of the early PIC work on resonance absorption in high-power laser-plasma interactions.

The technique was first presented at the ECLIM conference in 1990, and later applied to absorption of femtosecond laser pulses on solid targets in PRL 68, 1535 (1992). A longer description of the method including the transformation subtleties can be found later in Section 6.

While restricting the simulations to a special class of problems – in which the light, its harmonics and any other scattered modes are reflected in the specular direction only – the reduction from 2D $\rightarrow$ 1D brings huge savings in computational effort and/or increased spatial and temporal resolution. Not surprisingly, this type of code has become a 'workhorse' for high-intensity laser-matter interaction studies, giving relatively easy access to some extremely nonlinear, kinetic plasma phenomena, such as hot electron generation, ion acceleration and high-harmonic generation from solid surfaces.

2 Prerequisites

BOPS Version 3.X is a sequential code written in Fortran 90 and requires a full-blooded f90 compiler such as the Intel ifort or GNU's gfortran. The run scripts provided are designed for generic Unix systems and will work on a Linux PC.

Linux users

Unpack the tar file with:

```
tar xvfz bopsXX.tar.gz
```

and cd to the installation directory `bops`.

Windows users

The code can be run 'under' Windows using an appropriate commercial Fortran developer environment (IDE), but these platforms are not supported. (Instructions for successful builds most welcome, however). Here is a tried-and-tested method which may require a bit of disc space.

1. First install CYGWIN (www.cygwin.com). This creates a fully-fledged Unix environment emulator under Windows, for which you will need some elementary unix know-how. In addition to the default tools/packages offered during the installation you will also need:
   (a) Gnu compilers gcc, g77, g95 etc. (Look in the optional *devel* package in Cygwin)

(b) make (*devel* package)

(c) vi, emacs (*editors* package– optional, but very handy for quick editing!)

(d) X11 libraries if you want to generate graphics directly under cygwin (eg using gnuplot)

If you forget anything first time, just click on the Cygwin installer icon to locate/update extra packages.

2. Download and unpack the bopsXX.tar.gz file with an archiving tool (eg PowerArchiver: www.powerarchiver.com). You can put this anywhere, but a convenient location is your 'home' directory under CYGWIN, e.g.:

   ```
   C:\cygwin\home\gibbon\bops
   ```

3. Open a Cygwin terminal/shell and 'cd' to the bops directory.

3 Installation

The directory structure resulting from unpacking the tar files should look like this:

`src/`	...	containing the fortran90 source code
`doc/`	...	some documentation in html and ps
`run_scripts/`	...	sample scripts for running the code
`tools/`	...	postprocessing tools
`example_plots/`	...	sample output graphics

Go to the source directory `src`, adjust and tune the flags in the Makefile to match your machine type (FC=ifort or gfortran etc) and do:

```
make
```

On machines other than a Linux-PC, you may get complaints about the timing routine `etime` in the file cputime.f90. If this happens, edit cputime.f90 and either replace etime with something which the compiler knows, or comment it out altogether - this is not essential to run the code, but handy to know how long it's going to run for.

4 Running BOPS

Once compiled, go to the base (or top) directory and edit one of the examples in `run_scripts` (e.g. resabs). Change the $BOPS variable to the directory where the bops.tar file was unpacked (e.g. $HOME/bops) and the $RUN variable to where you want the data to be placed (e.g.: `resabs1`). To run from the base directory, just type

```
run_scripts/resabs
```

This will create a new run directory 'resabs1' and start executing the code. All graphical output etc., will be generated as a series of ASCII files in the run directory. Actual graphics are NOT supplied at present, but there is a postprocessor in the `tools/gle` directory (od2gle). Running the script `odpp` with the run directory as its argument will create GLE-readable output and .eps or .jpg plots in a separate plots/ subdirectory. This method of producing graphical output is highly recommended because it gives you an automatically generated, microfiche-like overview of the simulation results which you can browse with a simple image viewer. You can get this program – Graphics Layout Engine – for Windows, MAC OS/X and various Linux flavours from:

`http://glx.sourceforge.net`

It is an absolute doddle to install and these days even comes with a nice GUI called QGLE. Further graphics is up to the user - gnuplot or xmgrace will usually suffice to get started. Some sample plots roughly corresponding to the sample input files can be found in the examples directory.

To do a series or parameter study, you might prefer to modify the script to sit inside a 'project' directory and create subdirectories for each run.

4.1 Example scripts

resabs	Long scale-length, classical resonance absorption demo
snells_law	Refractive index transition (underdense plasma)
gb_prl92	Vacuum heating demo: steep density gradient, fixed ions
foil	Thin foil
foil+ramp	Foil + exponential leading ramp
foil_fsmu	Foil simulation set up with 'experimental units' (fs, microns)
hhg	High-harmonic generation from plasma surface

5 Input parameters

The variable names below correspond to those appearing in the Fortran namelist file bops.indata read by the code at the start of the run.

5.1 Target setup

The first task for the code is to set up the initial plasma conditions – density profile, particle velocity distribution, and so on. This is done by the routine 'parload', and controlled through various options in the parameter file. The target type is chosen via two parameters: `target_config` and `inprof`. The first of these picks the general target class (fixed ions, single species, multi-species), whereas the second parameter determines the shape of the plasma (uniform slab, ramp plus slab, etc.).

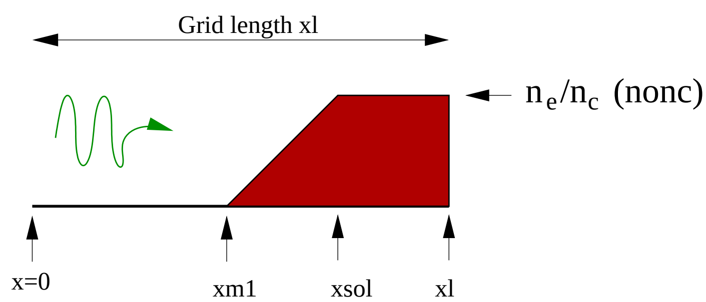

Figure 1: Setup for linear density profile.

variable name	value	meaning/selection
`target_config`	0	fixed ions
	1	single ion species
	≥ 2	additional ion layers (protons)
`nonc`		n/n_c
`xlolam`		L/λ
`xm1`		plasma edge, depending on profile choice `inprof`
`xsol`		start of 'solid', or max plasma density
`xsol2`		RH edge of plasma
`xl`		total grid length
`inprof`:	1	uniform profile
	2	linear ramp from `xm1` to `xl`
	3	linear ramp `(xm1 - xsol)` + flat top `(xsol - xl)` scalelength `xlolam`
	4	linear + flat top + trailing ramp `(xsol - xm2)`
	5	exponential ramp `xm1` to `xsol`, scalelength `xlolam`
	6	tanh ramp `(xm1 - xsol)`, scalelength `xlolam`
	7	foil, thickness `dfoil`, starting at `xm1`
	8	2 uniform layers with densities `nlayer (xm1-xm2)` and `n0 (xsol-xsol2)`
	57	foil thickness `dfoil`, with exponential ramp starting at `xm1`, scalelength `xlolam`

5.1.1 Multi-species target

Two additional multi-ion configurations are currently built in. The relative proton fraction can be varied from 0 to 100 % by choosing an appropriate value for `rho_layer`. For example, setting `nonc=10, rho_layer=5` will set up an additional layer with 50% heavy ions and 50% protons.

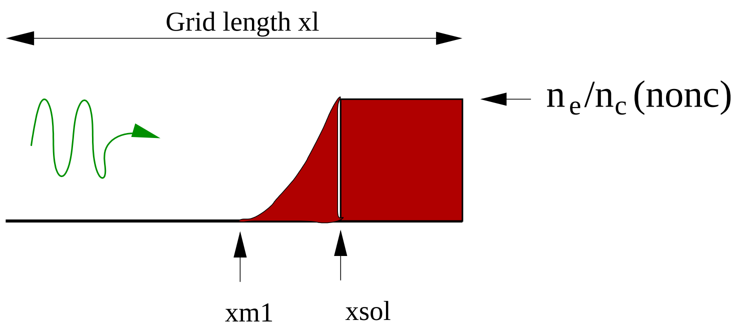

Figure 2: Setup for exponential density profile.

`target_config`	2	additional proton/mixed layer on rear of slab (`inprof=7`)
`target_config`	3	additional proton/mixed layer on front of slab
`rho_layer`		density of layer (n_p/n_c)
`x_layer`		width of layer
`mpome`		proton/electron mass ratio

5.1.2 Further plasma parameters

`ne`		number of electrons
`ni`	> 0	number of ions
	0	ions fixed, with $n_i = n_0$
`miome`		mass ratio
`Z`		ion charge (1)
`amass`		ion atomic weight (multiplies miome)
`Te`		electron temperature (keV)
`Ti`		ion temperature (keV)

5.2 Laser

`a0`	> 0	intensity in Wcm^{-2}
`a0`	< 0	pump strength v_{os}/c
`xlambda`		wavelength in microns
		(used to calculate v_{os}/c from $I\lambda^2$)
`theta0`		angle of incidence θ relative to target normal
`tpulse`		pulse duration t_{fwhm}
`tdel`		pulse delay t_d for Gaussian
`trise`		rise-time (linear)
`tfall`		fall-time (linear)
`ilas`:	1	uniform sinusoid
	2	gaussian $I(t) = I_0 \exp\{-(t-t_d)^2/t_p^2\}$; $(t_p = t_{fwhm}/2\sqrt(\log 2))$
	3	beat-wave (2-frequency pump)
	4	triangular (trise, tfall)
	5	$I(t) = I_0 \sin^2(\pi t/t_p)$; $t_p = 2t_{fwhm}$
`cpolzn`:	'P'	P-polarized light
	'S'	S-polarized
	'C'	C-polarized (circular)

5.3 Boundary conditions

`ipbc`:	1	periodic particles
	2	reflective particles
	3	absorb/reemit at both sides
	4	absorb ions at LHB, electrons only if charged
	5	absorb electrons, reflect ions at RHB
`ifbc`:	1	periodic fields
	2	bounded fields – reflective at solid, RHB

5.4 Control, diagnostics

`trun`		total run time
`nx`		number of mesh points
`igr`		frequency of graphical snapshots
`itc`		store for history plots
`iout`		printed output
`igmovie`		time-sequence snapshot frequency
`ncyc`		number of cycles for time-average plots
`igxs`		only plot every `igxs` point in 1D plots
`ipskip`		only plot every `ipskip` particle in phase-space plots
`itsk`		skip factor in k-space plots
`nsp`		number of special plots (in splot.f) at specified times
`isp[1:nsp]`		array giving times for special plots
`iunits`	0	time, length normalized to $1/\omega_0$, c/ω_0 (default)
	1	time, length normalized to $1/\omega_p$, c/ω_p
	2	time in fs, length in microns
`isubi`		subcycle ion motion
`ifreeze`		freeze-time for ions
`ioboost`	0	line plots in lab frame
	1	line plots in boost frame
`umevmax`		maximum energy for hot electron spectra
`uimax`		maximum energy for ion spectra
`upmax`		maximum energy for proton spectra
`nuav`		no. timesteps for distribution function average
`vxm, vym`		maxima for distribution function plots
`nftem`		frequency of fourier transform plots
`ift`		skip factor for FT
`omegm`		max frequency for FT plots
`ifbin`		binning for FT plots
`rhotrak`		tracking density (`nonc`)
`lrstrt`	(`.false.`)	restart switch

5.5 Particle tracking diagnostics

`ntrack`	number of particle to track
`itropt`	
`uhot`	threshold tracking energy
`xpint`	
`xpstart`	
`itstart`	
`itend`	

5.6 Units

The default unit system (and the working system of the code) is to normalise time and space variables to the laser frequency ω_0 and c/ω_0 respectively. This is achieved by setting `iunits=0`, the default. In this case, the other input parameters `trun`, `tpulse`, `trise`, `tfall`, etc. are all assumed to be in terms of $1/\omega_0$. Given the actual runtime in femtoseconds, the conversion factor is:

$$\omega_0 T_{\mathrm{run}} = \frac{1.88}{\lambda} T_{\mathrm{run}}(\mathrm{fs})$$

where λ is the laser wavelength in microns. If you specify the wavelength `xlambda` in the input file, you will see the equivalent elapsed run-time in fs in the printed output during the run. Note that the diagnostic switches `igr`, `itc`, `iout`, `nftem` and `igmovie` are also normalised to $1/\omega_0$ to make the output control easier. All grid length parameters: `xl`, `xm1`, `xsol`, `xsol2`, `dfoil`, etc. are normalised to c/ω_0. A grid length of `xl=12.57` (4π) is therefore equivalent to 2 vacuum laser wavelengths.

The other unit choices are `iunits=1`, which expects the above temporal and spatial input parameters in $1/\omega_p$ and c/ω_p respectively (sometimes useful if the laser is turned off completely or for comparison with other models); and the 'experimental' mode `iunits=2`, where time is in femtoseconds, length in microns. With the latter choice, special care needs to be taken to ensure that the simulation is set up with numerically stable parameters (particularly that $\lambda_D/\Delta x > 0.5$).

The laser intensity is specified (for historical reasons) through the parameter a0, which actually represents the normalised pump strength. This gets converted in the code via the relation:

$$a_0 = \sqrt{\frac{I_0 \lambda^2}{1.38 \times 10^{18}}}$$

Alternatively, you can specify the pump strength (v_{os}/c) directly by using a negative value for `a0`, eg. `a0=-0.5`.

5.7 Choosing simulation parameters

Like most numerical models, PIC codes have to be used with appropriate caution, otherwise they will cough up garbage at the earliest opportunity. In particular, it is important to observe certain constraints on spatial resolution and statistics in order to minimise numerical errors. Each of the sample scripts are set up to produce reasonably reliable results, even though the statistics (particle numbers) have been deliberately reduced to a minimum in the interests of efficiency. Starting from one of these scripts and scaling up the parameters appropriately should serve as a good first iteration. Here is a (far from complete) trouble-shooting checklist in case you suspect something has gone wrong:

Symptom	Remedy
Plasma heats up/expands too quickly	Is the Debye length resolved? Should have $\Delta x < 2\lambda_D$
Excessive noise in electron/ion densities or ES field	Increase particle nos. `ne, ni`
Anomalously high electron energies	Check vacuum regions large enough
Strange energy accounting (eg: -ve absorption)	Has pulse completely reflected? Increase run time

6 Scaling of simulation variables

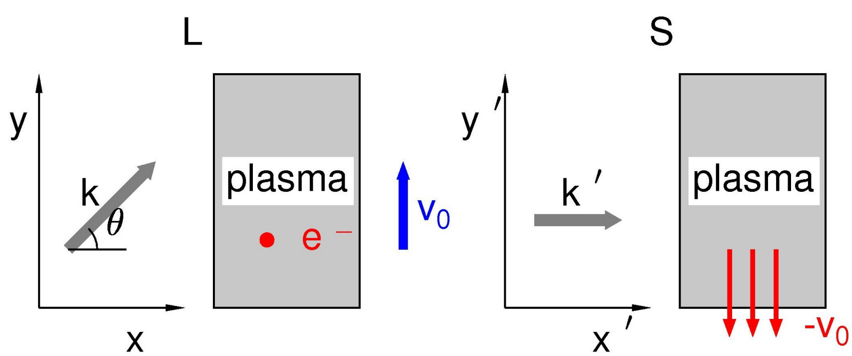

Figure 3: Boost geometry used for simulating oblique incidence interactions in BOPS.

BOPS uses a special technique for modelling oblique-incidence interactions – a case which normally requires two spatial dimensions, (x, y) say, for a target with a gradient in one direction (x). The trick is to perform a Lorentz transformation to a frame in which the wave vector is normally incident, so that the propagation is purely along the target normal – Fig.6. Denoting the boost (S) frame quantities by primes, the inverse Lorentz transformations for the wave frequency and k-vector are:

$$\begin{aligned} \omega' &= \gamma_0(\omega - v_0 k_y) \\ k'_y &= \gamma_0(k_y - \frac{v_0}{c^2}\omega) \\ k'_{x,z} &= k_{x,z} \end{aligned}$$

Since $k_y = k\sin\theta = \omega/c\sin\theta$ and $v_o = c\sin\theta$, we have:

$$\begin{aligned} k'_y &= 0 \\ \omega' &= \omega/\gamma_0 \\ k' = \mid \mathbf{k}' \mid &= k/\gamma_0 \end{aligned} \tag{6.1}$$

where $\gamma_0 = 1/\cos\theta$. For the space and time coordinates, we have (from $S \rightarrow L$)

$$
\begin{aligned}
t &= \gamma_0(t' + \frac{v_0}{c^2}y') \\
x &= x' \\
y &= \gamma_0(y' + v_0 t') \\
z &= z'
\end{aligned}
\tag{6.2}
$$

(6.3)

Likewise, noting that $B'_x = 0$, the electric and magnetic fields transform as:

$$
\left.\begin{aligned}
E_x &= \gamma_0(E'_x - v_0 B'_z) \\
E_y &= E'_y \\
B_z &= \gamma_0(B'_z - \tfrac{v_0}{c^2}E'_x)
\end{aligned}\right\} \text{p-polarized light,}
$$

$$
\left.\begin{aligned}
E_z &= \gamma_0 E'_z \\
B_x &= \gamma_0 v_0 E'_z \\
B_y &= B'_y
\end{aligned}\right\} \text{s-polarized light}
\tag{6.4}
$$

and the density and current for each particle species α (inversely) as:

$$
\begin{aligned}
\rho'_\alpha &= \gamma_0(\rho_\alpha - \frac{v_0}{c^2}j_{\alpha y}) \\
j'_{\alpha y} &= \gamma_0(j_{\alpha y} - v_0\rho_\alpha) \\
j'_{\alpha x,z} &= j_{\alpha x,z}
\end{aligned}
$$

Initially, $j_{ey} = j_{iy} = 0$, so in the boost frame, $(\rho_0)'_{e,i} = \gamma_0(\rho_0)_{e,i}$ and $(j_{y0})'_{e,i} = -(\rho_0)'_{e,i} c\sin\theta$.

In an electromagnetic PIC or Vlasov code, it is convenient to use the standard normalisations [**?**]:

$$
\begin{aligned}
\tilde{t} &= \omega t \\
(\tilde{x}, \tilde{y}) &= (kx, ky) \\
\tilde{\mathbf{E}} &= \frac{e\mathbf{E}}{m\omega c} \\
\tilde{\mathbf{B}} &= \frac{e\mathbf{B}}{m\omega} \\
\tilde{n}_{e,i} &= n_{e,i}/n_c \\
\tilde{v}_{e,i} &= v_{e,i}/c
\end{aligned}
$$

Thus, from Eqs.(6.2) and (6.1), the normalised time interval $\omega' t' = \omega t$ is an invariant, as is the wave phase $\omega t - \mathbf{k}\cdot\mathbf{r}$. On the other hand, the simulation grid length $k' x'_l = \gamma_0^{-1} k x_l$ *shrinks* with increasing θ. The scaling of ω' and k' with γ_0 and the fact that $\tilde{B}'_x = 0$ means that the *normalised* field transformations become:

$$
\begin{aligned}
\tilde{E}_x &= \tilde{E}'_x - \tilde{v}_0 \tilde{B}'_z \\
\tilde{E}_y &= \frac{\tilde{E}'_y}{\gamma_0} \\
\tilde{B}_z &= \tilde{B}'_z - \tilde{v}_0 \tilde{E}'_x, \\
\tilde{E}_z &= \tilde{E}'_z \\
\tilde{B}_x &= \tilde{v}_0 \tilde{E}'_z \\
\tilde{B}_y &= \frac{\tilde{B}'_y}{\gamma_0}
\end{aligned} \tag{6.5}
$$

Note that the critical density in the simulation frame transforms as:

$$
\frac{n'_c}{n_c} = \frac{\omega'^2}{\omega^2} = \frac{1}{\gamma_0^2}
$$

Hence the initial normalised unperturbed electron density is:

$$
\tilde{n}'_{e0} \equiv \frac{n_e(t=0)'}{n'_c} = \gamma_0^3 \tilde{n}_e
$$

Thus, to initialise the particles in the simulation frame, we give them a charge $q'_e = \gamma_0^2 q_e$, and mass $m'_e = \gamma_0^2 m_e$, which when combined with the grid length contraction $k'x'_l = kx_l/\gamma_0$, satisfies: $N_e q'_e = -n'_e x'_l$ for the upper density shelf, where N_e is the number of simulation particles. (See Ref.[**?**] for a discussion on particle loading.) To initialise the particle momenta, we first specify the thermal distribution in the lab frame, and then perform boosts:

$$
\begin{aligned}
\tilde{p}'_{x,z} &= \tilde{p}_{x,z} \\
\tilde{p}'_y &= \gamma_0(\tilde{p}_y - \tilde{v}_0\gamma) \\
\gamma' &= \gamma_0(\gamma - \tilde{v}_0 \tilde{p}_y)
\end{aligned} \tag{6.6}
$$

Finally, to launch the EM wave, we must specify its amplitude $a_0 = v_{osc}/c$ at the left-hand simulation boundary. Since $v_{osc}/c = eE_0/m\omega c$, we have for a p-polarized wave:

$$
a'_0 = \frac{eE'_y}{m\omega' c} = \gamma_0 \frac{eE_y}{m\omega c} = \frac{eE_0}{m\omega c}\gamma_0 \cos\theta = a_0 \tag{6.7}
$$

One can easily verify this invariance by using Eq.(6.5) to recover the lab-frame vacuum fields. According to Eq.(6.7), we let $\tilde{E}'_y(x'=0) = \tilde{B}'_z(x'=0) = a_0$, so since $E'_x = 0$, we obtain $\tilde{E}_x = -v_0 a_0 = -a_0 \sin\theta$, $\tilde{E}_y = a_0/\gamma_0 = a_0 \cos\theta$, and $\tilde{B}_z = a_0$, which is just what we expect for a plane wave launched at an angle θ to the density gradient according to Fig.6

7 Diagnostics

To interpret the simulation results unambiguously, it is simplest to transform the field and particle variables back to the lab frame. We note in passing, however, that some physical insight and analytical economy can be gained by examining boost frame quantities, so long as care is taken in the labeling of sources and fields [?]. For the fields, Eq.6.5 is applied before performing cycle-averages and Fourier transforms. For the particles, it is convenient to use the inverse transformations of (6.6), from which we can recover the lab frame velocities, kinetic energies and currents.

7.1 Absorption rate

The absorption rate is a little more subtle: since it is useful to be able to evaluate it in either frame, and we need to take some care over the definition. The boost frame is specially chosen so that the EM waves travel only along the x'-axis. Therefore, the absorbed wave energy is just the Poynting flux at the left-hand boundary, normalized to the energy density of the incoming wave:

$$\eta_{wave} = \frac{(F'^{+})^2 - (F'^{-})^2}{(F'^{+})^2}. \tag{7.8}$$

where $F'^{+} = E'_y + B'_z, F'^{-} = E'_y - B'_z$. In the boost frame, the Poynting flux normal to the target equals the *total* EM flux; in the lab frame, this is reduced by γ_0, ie:$P_x = P' \cos\theta = E_y B_z$. The cycle-averaged incoming energy is thus

$$U_{in} =< P_x >= \frac{1}{2} a_0^2 \cos\theta. \tag{7.9}$$

This factor is used to obtain the fractional absorption components in thermal energy, field energy, hot electrons and ions etc. For example,

$$\eta_{hot} = \frac{U_{hot}}{\int U_{in} dt}. \tag{7.10}$$

As a consistency check at the end of the simulation, we should have

$$\eta_{wave} = \sum (\eta_{hote} + \eta_{ions} + \eta_{therm} + \eta_{field}) \tag{7.11}$$

7.2 Electric and magnetic field conversion

As explained in Section 6, the electric and magnetic fields produced in the output files are, unless specified otherwise, normalised to $m\omega_0 c/e$ and $m\omega_0/e$ respectively. To convert back to SI units, the following relations should be used:

$$\begin{aligned} E &= \left(\frac{m\omega_0 c}{e}\right) \tilde{E} \\ &\simeq 3.22 \times 10^{12} \lambda_\mu^{-1} [\mathrm{V\ m^{-1}}] \end{aligned} \tag{7.12}$$

(7.13)

$$\begin{aligned} B &= \left(\frac{m\omega_0}{e}\right) \tilde{B} \\ &\simeq 1.1 \times 10^{4} \lambda_\mu^{-1} [\mathrm{T}] \end{aligned} \tag{7.14}$$

(7.15)

Similarly, a normalised field intensity $a_0^2 = 1$ corresponds to

$$I = 1.37 \times 10^{18} \lambda_\mu^{-2} \text{ Wcm}^{-2} \tag{7.16}$$

7.3 Conversion factor for hot electrons

Because BOPS is a 1D code, simulation particles actually represent charge sheets, so one cannot directly translate their numbers into physical equivalents. Neverthless, a conversion factor can be estimated by specifying the laser spot size and wavelength. The argument goes as follows:

Consider a slab of plasma with very short scale-length (as Fig. 5.1 with $L/\lambda = 0$). Since the total charge $Q = N_e q_e$ is conserved, each particle carries a fixed line density:

$$\Gamma \equiv -\tilde{\rho}_0 = \frac{N_e q_e}{\tilde{L}_p}$$

where $N_e, \tilde{L}_p, \tilde{\rho}_0, q_e$ are the number of simulation electrons, the plasma length (`xl-xm1`), normalised density ($\tilde{\rho}_0 = n_0/n_c$) and (macro-)charge respectively.

The number of *physical* electrons contained within a cylinder of length L_p and radius σ_L (laser spot size) is just:

$$N_a = n_0 L_p \pi \sigma_L^2$$

Thus using the default conversion factors

$$L_p = c/\omega_0 \tilde{L}_p,$$

or

$$\frac{L_p}{\mu\text{m}} = \frac{\lambda_\mu}{2\pi} \tilde{L}_p$$

and

$$n_0 = \left(\frac{n_0}{n_c}\right) 10^{21} \lambda_\mu^{-2} \text{cm}^{-3} = 10^9 \tilde{n}_0 \lambda_\mu^{-2} \mu m^{-3}$$

we get

$$\begin{aligned} C_s = \frac{N_a}{N_e} &= 5 \times 10^8 \lambda_\mu^{-1} \sigma_\mu^2 \left(\frac{\tilde{n}_0 \tilde{L}_p}{N_e}\right) \\ &= 5 \times 10^8 \lambda_\mu^{-1} \sigma_\mu^2 |q_e| \end{aligned} \tag{7.17}$$

This is valid for a plasma slab, but can be refined for more complex profiles by computing the total charge contained in the plasma (a procedure which is performed anyway to get q_e in the first place). To apply the conversion, just multiply by the number of simulation particles, for example:

$$N_{hot}(real) \simeq C_s N_{hot}(sim).$$

In the code output (bops.out) C_s is displayed as:

```
Charge conversion factor N/Nsi 3.5244E+07
```

8 Output files

bops.header	Summary of run parameters and some numerical checks
bops.out	Continuous run protocol (helpful for debugging)
bops.oddata	List of graphical output files produced during run

9 Graphical output

The graphical output from the code falls broadly into 6 types of plot:

1. Spatial profiles (fields, sources; instantaneous and cycle-averaged)
2. Particle distribution functions (velocity/energy)
3. Fourier spectra (time- and space-domains)
4. Phase space (x-px, x-py, px-py etc)
5. Time histories (energy diagnostics; absorption; position tracking)
6. 2D surface plots in x-y plane

The filenames of the xy plots generated during the run are listed below. Unless otherwise stated, the names consist of 4 letters plus a digit from 00-99 indicating the snapshot number, followed by the suffix .xy. A run lasting 500 time units with graphs produced every 100 units will generate 5 snapshots for each quantity. (At present, the maximum number of snapshots is 9 – this limitation will be removed in future). The time-histories all have a suffix '0' after the name.

Assuming you have GLE installed, these plots can be postprocessed with the help of a program in the `tools` subdirectory.

```
cd tools/gle
make od2gle
```

Make sure the binary generated is in your PATH. Now go back up to the top directory and run the shell script

```
./odpp resabs1
```

where `resabs1` is the name of the run directory you wish to postprocess. The program will scan the file `foil.id` for plots – the numbers correspond to the IDs in the tables below – and attempt to produce series of line graphs for these in `resabs1/plots`.

9.1 Spatial profiles

Instantaneous

ID	file	quantity
2000	rhot	net charge density (rhoe+rhoi+rhop)
2100	ninc	heavy ion density
2200	ninc	proton density
2500	phsi	electrostatic potential
3000	exsi	field Ex
4000	eysi	field Ey
3500	eysi	field Ez
4500	bzsi	field Bx
5500	bzsi	field By
5000	bzsi	field Bz
40000	tefo	forward going EM wave (boost frame P)
40500	teba	backward going EM wave (boost frame P)
44000	tmfo	forward going EM wave (boost frame S)
44500	tmba	backward going EM wave (boost frame S)
41000	jyel	electron current
43000	jyio	ion current
41500	jtot	net current
42500	ayem	EM vector potential (boost frame only)
42000	azem	EM vector potential (boost frame only)

Cycle-averaged

20000	exrm	RMS electric field in x-direction
21500	eyrm	RMS electric field in y-direction
22000	bzrm	RMS magnetic field in z-direction
21000	ezrm	RMS electric field in z-direction
30000	byrm	RMS magnetic field in y-direction
20500	exdc	DC electric field in x-direction
22500	bzdc	DC magnetic field in z-direction
23500	phdc	DC electrostatic potential
29000	jyrm	RMS net current in y-direction
28500	jydc	DC net current in y-direction
29500	redc	DC charge density
26000	vxbp	cycle-averaged $\boldsymbol{v} \times \boldsymbol{B}$ force in x-direction (P)
27000	vxbs	cycle-averaged $\boldsymbol{v} \times \boldsymbol{B}$ force in x-direction (S)
25000	edoj	RMS $\boldsymbol{E} \cdot \boldsymbol{J}$ absorption fraction

9.2 Particle distribution functions

6000	fvxe	fe(vx) - electron velocity distribution in vx
6200	fvye	fe(vy)
6300	fvze	fe(vz)
6500	fuep	fe(U) - plasma electron energy spectrum
6400	fuip	fi(U) - heavy ion energy spectrum
6600	fupp	fp(U) - proton energy spectrum
6700	fues	fh(U) - energy distribution of escape hot electrons
16800	fuin	energy distribution of re-injected electrons
6800	fhot	hot electron distribution with reinjected thermal electrons subtracted
7100	ufue	1st moment of energy distribution = U*fh(U)
7200	qoqt	cummulative integral form of U*fh(U)
	e_mom_lhb.dat	electron exit momenta (LHB)
	e_mom_rhb.dat	electron exit momenta (RHB)
	ion_mom_lhb.dat	ion exit momenta (LHB)
	exit_energies.dat	electron exit times and energies (RHB)

9.3 Fourier spectra

71000	tebs	spectrum of reflected EM wave (P-polarized light)
70000	tefs	spectrum of transmitted EM wave (P)
76000	tmbs	spectrum of reflected EM wave (S-polarized light)
74000	tmfs	spectrum of transmitted EM wave (S)
72000	escr	frequency-spectrum of electrostatic field at critical density
73000	jycr	frequency-spectrum of current at critical density
7300	uesk	k-spectrum of electrostatic energy density in plasma
	ey_back.t	time-signal of reflected EM E-field
	ez_back.t	time-signal of relfected EM B-field

9.4 Phase space (scatter plots)

1000	pxxe	electron x-momenta vs x
1500	pyxe	electron y-momenta vs x
1700	pyxe	electron z-momenta vs x
1600	pxpy	electron y-momenta vs x-momenta
1200	pxxi	ion x-momenta vs x

9.5 Time histories – mostly cycle-averaged

uinc	incoming electromagnetic wave energy (= normalised laser intensity)
ubac	outgoing EM wave energy
absr	laser absorption calculated 1-Reflectivity
uesp	spatially integrated electrostatic wave energy
ueme	integrated EM energy (TE modes)
uthp	thermal energy
uthe	electron thermal energy
uthi	ion thermal energy
urhb	cumulative energy of outgoing electrons (solid boundary)
ulhb	cumulative energy of outgoing electrons (vacuum boundary)
usys	total energy in simulation box
utot	total energy including particles lost to boundaries
abut	rate of change of total energy dUtot/dt
abuh	rate of change of energy of electrons leaving box dUhot/dt
abui	rate of change of ion energy
nihi	max ion density
xcni	position of critical surface

9.6 2D surface plots

Bz1.2D	Laser magnetic field
edens1.2D	Electron density
idens1.2D	Ion density
Jy1.2D	Electron current
Jyzoom1.2D	Zoom of electron current

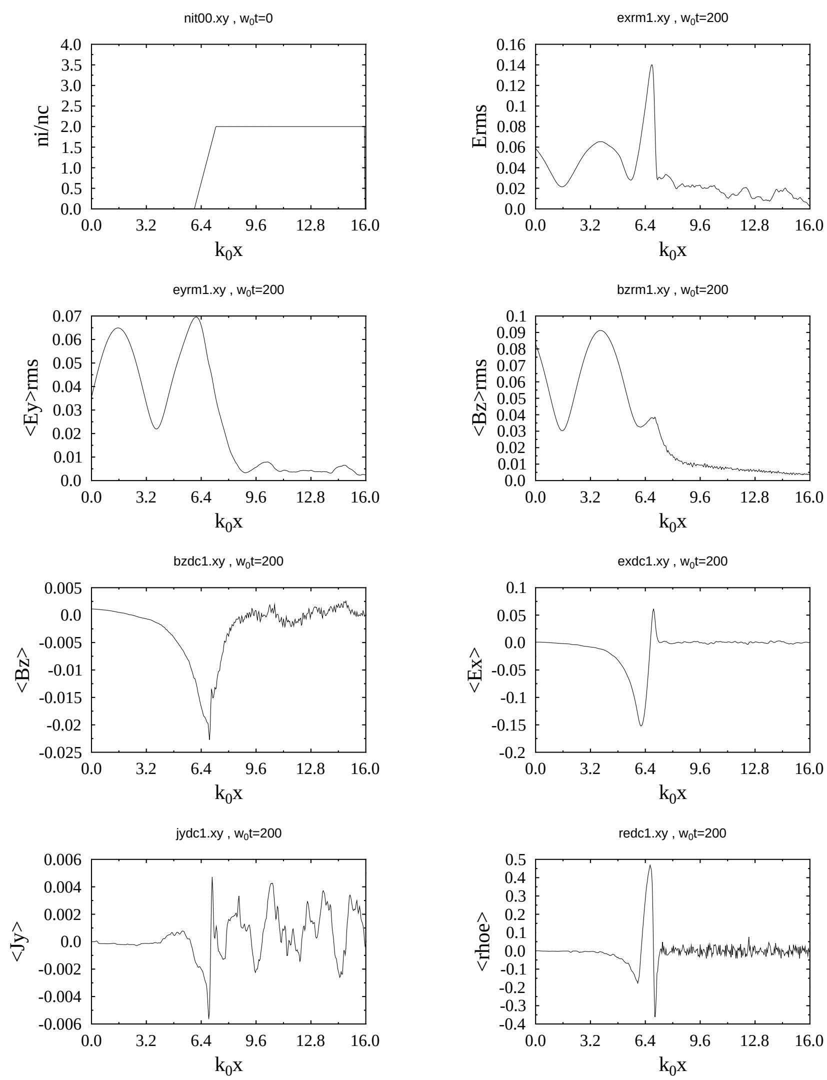

Figure 4: Spatial profiles

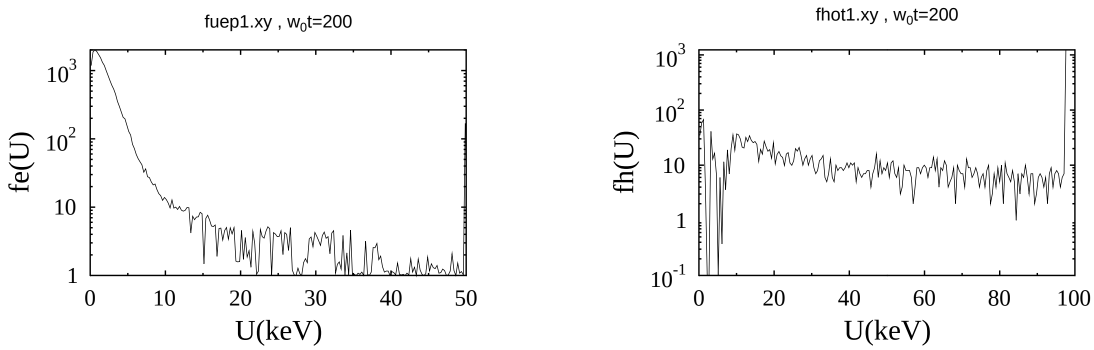

Figure 5: Fast particle spectra

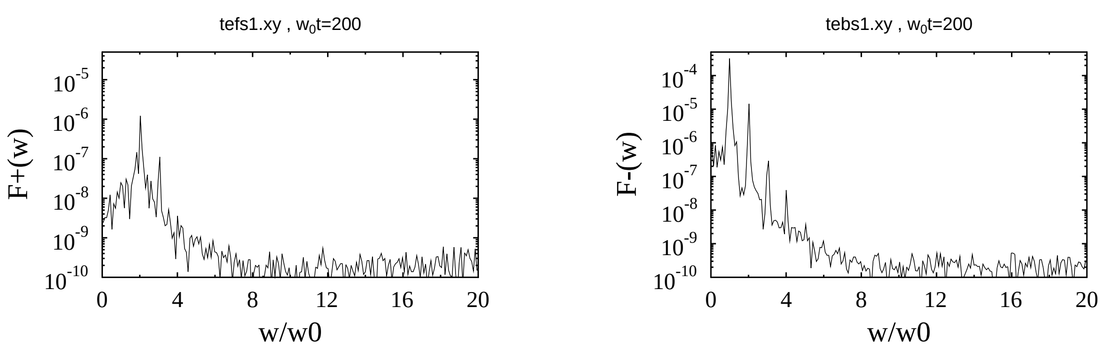

Figure 6: Light (Fourier) spectra

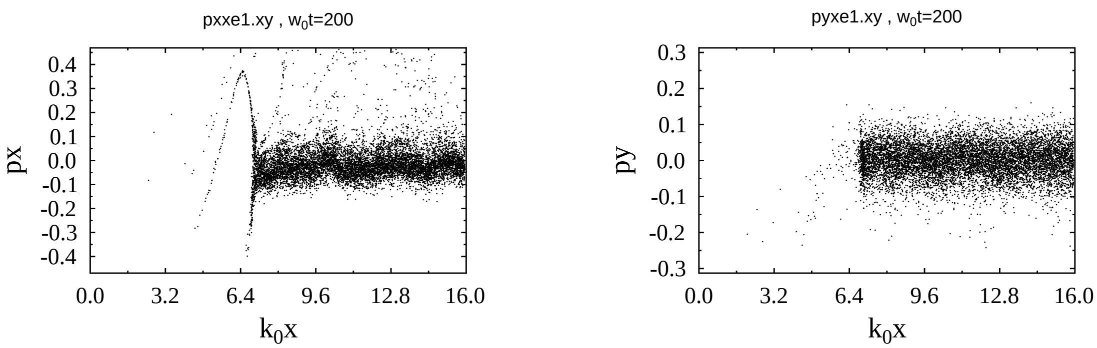

Figure 7: Phase space

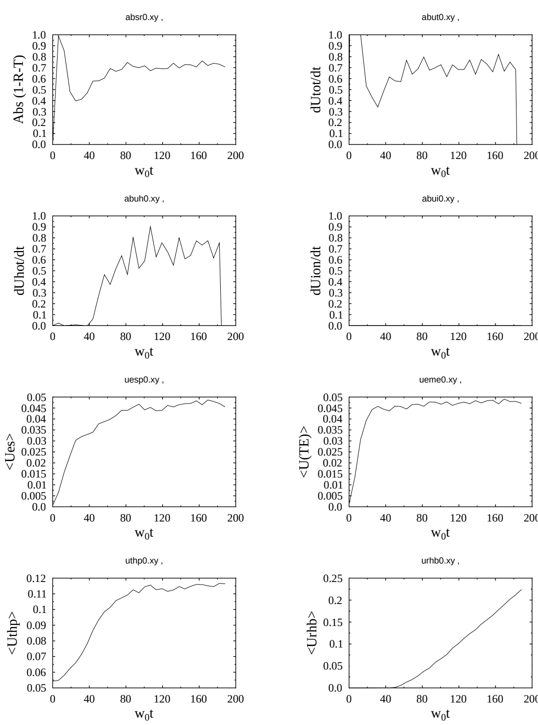

Figure 8: Time histories

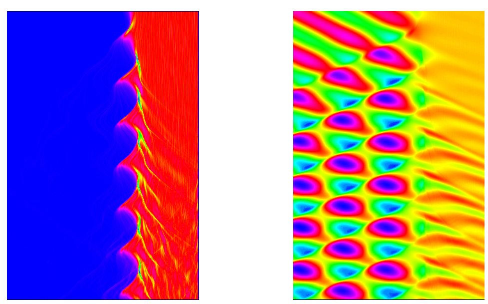

Figure 9: 2D plots: electron density (left); magnetic field (right)