Science & Technology Facilities Council

Technical Report

RAL-TR-2014-013

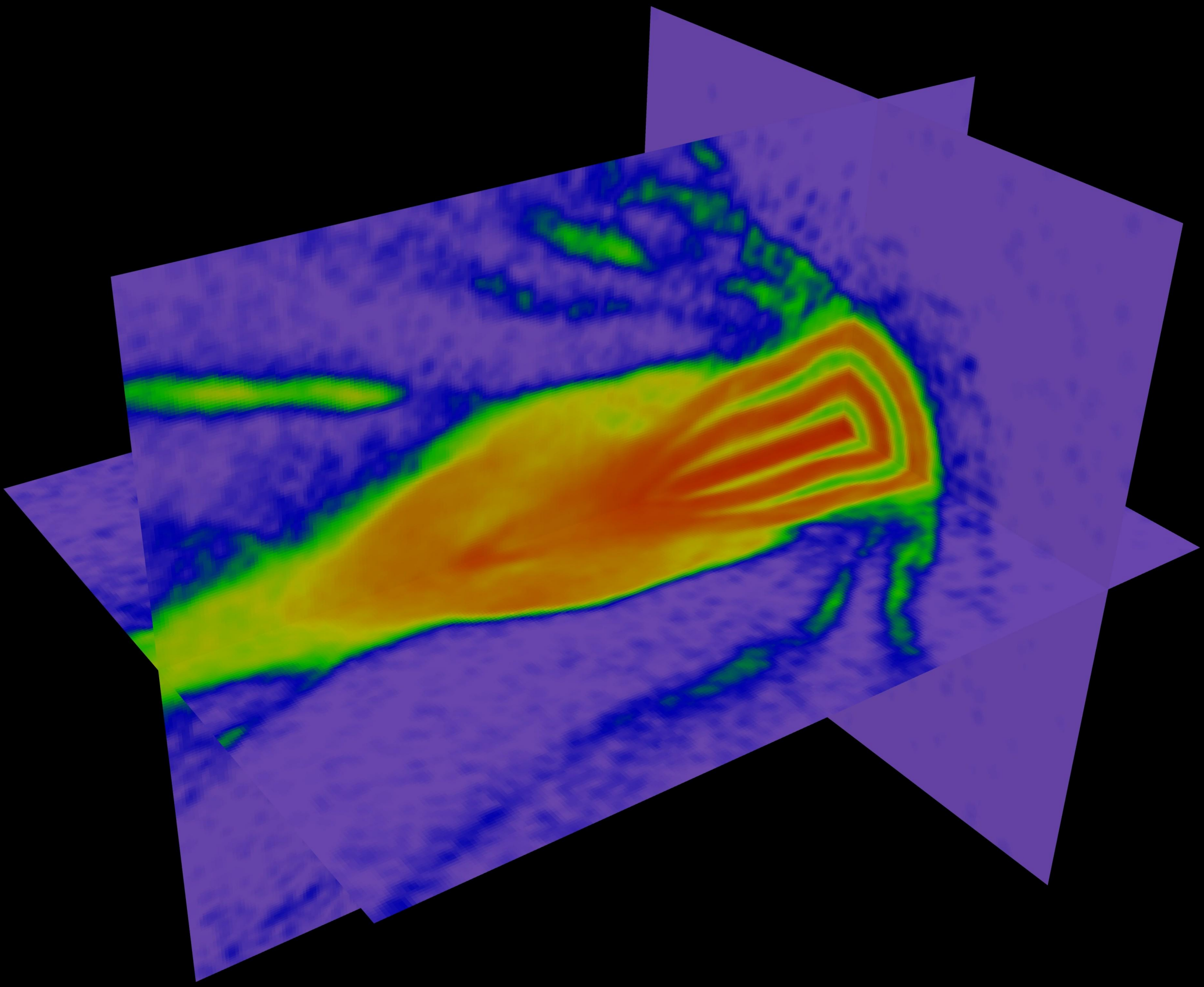
Zephyros User Manual

APL Robinson

October 2014

ISSN 1358-6254

ZEPHYROS User Manual



Dr.A.P.L.Robinson
Central Laser Facility

Contents

1 Introduction **5**
- 1.1 Overview . . . 5
- 1.2 How to Use this Manual . . . 5
- 1.3 Output . . . 6
- 1.4 History . . . 6

2 Building ZEPHYROS **7**
- 2.1 Overview . . . 7
- 2.2 Source . . . 7
- 2.3 Compiling . . . 8
- 2.4 Running . . . 8

3 Input Deck Overview **9**
- 3.1 The Input Deck . . . 9
- 3.2 Units . . . 14
- 3.3 Quick Reference Tables . . . 15

4 General Simulation Parameters **19**
- 4.1 Spatial Parameters . . . 19
- 4.2 Temporal Parameters . . . 19
- 4.3 Modular Physics Switches . . . 19
- 4.4 Functional Switches . . . 20
- 4.5 Boundary Conditions . . . 20
- 4.6 External EM Fields . . . 21
- 4.7 Special Diagnostics . . . 21
  - 4.7.1 Angular Diagnostic . . . 21
  - 4.7.2 K-alpha Diagnostic . . . 21

5 Target Parameters **23**
- 5.1 Homogeneous Cold Targets . . . 24
- 5.2 Resistive Guiding Targets . . . 24
  - 5.2.1 Al-Sn-Al Sandwich Target . . . 24
  - 5.2.2 Fe Wire Target . . . 25
- 5.3 Buried Layer Targets . . . 25

5.3.1 Al-Ti Target 25
5.3.2 CH-Ta-CH Target 25
5.4 Externally Supplied Resistivity Configuration 25
5.5 Configurable Hot Plasma 26
5.6 User Defined Hot Plasma 26
5.7 Control of the Lee-More Resistivity Model 26
5.8 Fixed Resistivity 26

6 Fast Electron and Laser Parameters 27
6.1 Laser Model 27
6.2 Laser Temporal Profile 28
6.3 Laser Spot 28
6.4 Fast Electron Energy Distribution 28
6.5 Fast Electron Angular Distribution 29
6.6 Second Laser Spot 30

7 ZEPHYROS Output 31
7.1 ZEPHYROS Output 31
7.1.1 EM Fields 31
7.1.2 Particle Moments 31
7.1.3 Background 32
7.1.4 Fast Electron Kinetics 32
7.2 Visualization with MATLAB 32
7.3 Visualization with Pytheros Scripts 33
7.3.1 Slicing 34
7.3.2 Auto-plotting 35
7.3.3 Single Slice Scripts 36
7.3.4 Other Pytheros Scripts 37
7.4 Visualization with VisIt 37
7.5 Other Tools 38

8 Example Run 39
8.1 Overview 39
8.2 Example Input Deck 39
8.3 Understanding the Input Deck 41
8.4 Running the Example 41
8.5 Output 42

Chapter 1

Introduction

USER MANUAL VERSION : 0.6

ZEPHYROS VERSION : 1.0 series

DATE = October 23, 2014

AUTHOR = A.P.L.Robinson

1.1 Overview

The ZEPHYROS code is a 3D macroparticle based hybrid code for studying fast electron transport in dense plasmas. The code was developed by Dr.A.P.L.Robinson at the Central Laser Facility from 2009 onwards. The code is heavily based on the classic hybrid model for laser generated fast electron transport in dense plasmas that was originally worked on by J.R.Davies.

Version 1.0 of ZEPHYROS is a single processor version of the code written in FORTRAN 90. In this manual we will primarily describe the use of the code via the input deck (readin.in). The following chapters of this manual thus detail the parameters in the input deck and their function.

1.2 How to Use this Manual

In order to set up a ZEPHYROS simulation, it is absolutely necessary to prepare an *input deck* in the form of a file called `readin.in`. This manual concentrates on the preparation of this inupt deck. In terms of the 'code utility' aspect (i.e. how to run or compile the code on certain systems) we only provide instructions for running on a generic linux system in the following chapter. We do not include any details on the internal physical models or algorithms of the code in this manual.

One can copy the input deck from Chapter 3, and use the quick references tables there too. Chapters 4–6 provide more detailed information on what the different parameters do.

1.3 Output

ZEPHYROS writes output in text format to a set of `.dat` files. These files can be viewed by a variety of applications. MATLAB has been found to be a good platform for looking at ZEPHYROS output, however VisIt is also a recommended visualization tool, particularly for producing fully 3D visualizations. The increasing popularity of Python has lead to the CLF PPG producing the PyBUS library of scripts, one division of which are devoted to visualization of ZEPHYROS output. More information on this is given in Chapter 7.

1.4 History

Zephyrus, or just Zephyr, in Latin Favonius, is the Greek god of the west wind. The gentlest of the winds, Zephyrus is known as the fructifying wind, the messenger of spring. It was thought that Zephyrus lived in a cave in Thrace.

from the Wikipedia Entry on Anemoi

2009: Development of ZEPHYROS was started when APLR was asked by Satya Kar and Matt Zepf (both of Queen's University Belfast) to help model fast electron flow through Al-Sn-Al 'structured collimator' targets.

2010: ZEPHYROS capabilities are expanded when Bhuvan Ramakrishna and Matt Zepf ask APLR to help model a further experiment with Fe wires immersed in Al.

2011: APLR expands ZEPHYROS even further in order to model fast electron transport in carbon allotropes and Li in order to support experiments lead by Paul McKenna (University of Strathclyde). APLR also starts using ZEPHYROS for studying the 'magnetic switchyard' concept for Fast Igntion ICF.

2012-2014: ZEPHYROS gradually develops in collaboration with Strathclyde, York, and Oxford.

2014 : 1st major documentation revision (manual version 0.5).

Chapter 2

Building ZEPHYROS

2.1 Overview

In this chapter we provide information on how to build and run ZEPHYROS on a generic linux system. We have been able to build ZEPHYROS on a number of different linux platforms including STFC's SCARF cluster which is based on Red Hat Enterprise Linux, and workstations using Linux Mint. We have also built ZEPHYROS on Windows XP based systems using Silverfrost's Fortran compiler.

2.2 Source

The source code for ZEPHYROS consists of 21 `.f90` files, including a makefile. The source is maintained by the CLF PPG on the CLF Software Repository (SVN server). The files are:

```
zeph_angular.f90
zeph_bkgd.f90
zeph_currents.f90
zeph_diff.f90
zeph_drag.f90
zeph_em.f90
zeph_init.f90
zeph_inject.f90
zeph_kalpha.f90
zeph_main.f90
zeph_move.f90
zeph_psolve.f90
zeph_readin.f90
zeph_rlm.f90
zeph_save.f90
zeph_scatter.f90
zeph_tf.f90
zeph_vars.f90
```

```
zeph_weightparts.f90
zeph_xchang.f90
```

This source is not dependent on any third party libraries.

2.3 Compiling

A makefile is included with the code in the repository. Therefore anyone who obtains the code should also obtain the makefile. The makefile only needs to be altered so that it invokes the fortran compiler that you wish to use. On most linux systems you should be able to install the `gfortran` compiler. Some users may have acquired a commerically available compiler such as the Intel Fortran compiler.

The correct compiler can be invoked by changing the value of `FC` in the makefile. Once this is done doing `make zeph` should result in the code being compiled and built.

2.4 Running

Running ZEPHYROS is straightforward. One needs an executable (which we will call `zeph`), an input deck (`readin.in`), and possibly a few additional input files depending on the problem (`resinp.dat`,`ninp.dat`,`zinp.dat`). Once these files are placed in the run directory the code can be run simply. This will be :
$>$`./zeph`

Chapter 3

Input Deck Overview

3.1 The Input Deck

ZEPHYROS takes a single list of numerical and logical parameters as its input via the FORTRAN NAMELIST facility. Here is an empty input deck:

```
   &inpvars
nfasts =
nx =
ny =
nz =
dx =
dy =
dz =
iconfig =
idiagnostic =
tstop =
dtsave =
timefrac =
temp_in =
loglambda =
loglambdascat =
loglambdadrag =
rspot =
rspotmax =
rspotmin =
Ilaser =
absfrac =
tlaser =
iinject_mode =
ilaser_spot =
iinject_func =
lambda_laser =
div_angle =
control1 =
control2 =
control3 =
control4 =
control5 =
control6 =
control7 =
control8 =
control9 =
control10 =
control11 =
control12 =
icontrol1 =
icontrol2 =
icontrol3 =
ilaser1 =
ilaser2 =
ylaser =
zlaser =
las_control1 =
las_control2 =
las_control3 =
las_control4 =
las_control5 =
Lbrems =
Lxchang =
Lmdiff =
LBevolve =
Lcheck_params =
Lthomasfermi =
LRedLeeMore =
Linjcontrols =
Lkalpha =
LSpecifyLMParam =
LFixedResistivity =
nresinp =
iconfig2 =
iconfig3 =
icontrol4 =
ibcs=
ifields=
lm_mfp_factor =
lm_multiplier =
Lsave_currents=
Lsave_diags =
Lsave_resist=
Lsave_eden =
pxx_x1 =
pxx_x2 =
ipxxmode =
Lsecondspot =
Ilaser_2 =
absfrac2 =
lambda_laser2=
tlaser2=
rspot2=
rspotmax2=
rspotmin2=
ylaser2=
zlaser2=
div_angle2=0/
&end
```

Certain straightforward simulations will not require all these parameters to be used. It is helpful to break the input deck down into the parameters that govern different aspects of the simulation initialization and control. There are really four categories of parameter : GENERAL , CONTROL , FAST ELECTRON , and TARGET :

- GENERAL : These parameters control the over-arching aspects of the simulation, including number of macroparticles, domain size, simulation time, simulation outputs and so on. The functions of these parameters are described in Chapter 4 of this manual.

- CONTROL : These are general inputs that interact with the other input parameters to finely tune an initialization. These parameters only function **in conjunction** with other parameters, i.e. by modifying their effect, and are thus referred to throughout the manual.

- FAST ELECTRON : The parameters control the injection of fast electrons in the simulation. The functions of these parameters are described in Chapter 6.

- TARGET : These parameters control the background or target initialization. The functions of these parameters are described in Chapter 5.

If we apply this to the input deck then it looks like:

```
   &inpvars
nfasts =
nx =
ny =
nz =
dx =
dy =
dz =
iconfig  =
idiagnostic  =
tstop  =
dtsave  =
timefrac  =
temp_in  =
loglambda  =
loglambdascat  =
loglambdadrag  =
rspot  =
rspotmax  =
rspotmin  =
Ilaser  =
absfrac  =
tlaser  =
iinject_mode  =
ilaser_spot  =
iinject_func  =
lambda_laser  =
div_angle =
control1  =
control2  =
control3  =
control4  =
control5  =
control6  =
control7  =
control8  =
control9  =
control10  =
control11  =
control12  =
icontrol1  =
icontrol2  =
icontrol3  =
ilaser1  =
ilaser2  =
ylaser  =
zlaser  =
las_control1  =
las_control2  =
las_control3  =
las_control4  =
las_control5  =
Lbrems  =
Lxchang  =
Lmdiff  =
LBevolve  =
Lcheck_params  =
Lthomasfermi  =
LRedLeeMore  =
Linjcontrols  =
Lkalpha  =
LSpecifyLMParam  =
LFixedResistivity  =
nresinp  =
iconfig2  =
iconfig3  =
icontrol4  =
ibcs =
ifields =
Lsave_currents =
Lsave_diags  =
Lsave_resist =
Lsave_eden =
pxx_x1  =
pxx_x2  =
ipxxmode  =
Lsecondspot  =
Ilaser_2  =
absfrac2  =
lambda_laser2 =
tlaser2 =
rspot2 =
rspotmax2 =
rspotmin2 =
ylaser2 =
zlaser2 =
div_angle2 =0/
&end
```

Details of what each parameters controls and its relation to the physics embodied by the ZEPHYROS algorithms is given in the subsequent chapters. Here we will provide tables giving brief descriptions of each parameter which are useful for quick reference.

3.2 Units

ZEPHYROS is an SI based code. Internally the code works on an SI basis. A number of important parameters are not, however, generally expressed in SI in laser-plasma physics. This includes intensity (Wcm^{-2}), laser wavelength (μm), and temperatures (eV). The tables and detailed chapters will indicate where parameters are to be inputted in non-SI units. Otherwise a user should assume SI units.

3.3 Quick Reference Tables

GENERAL

Parameter	**Type**	**Function**
nfasts	INTEGER	Number of macroparticles used by simulation.
nx,ny,nz	INTEGER	Number of spatial cells in each dimension.
dx,dy,dz	REAL	Spatial cell size in each dimension.
idiagnostic	INTEGER	Selects special diagnostic mode.
tstop	REAL	Time at which simulation runs up to.
dtsave	REAL	Time between simulation outputs.
timefrac	REAL	(0–1) $dt/min(\Delta_{x,y,z}/c)$
Lbrems	LOGICAL	Turns bremsstahlung cooling of background on and off.
Lxchang	LOGICAL	Turns electron-ion energy exchange in background on and off.
Lmdiff	LOGICAL	Turns magnetic diffusion term on and off.
LBevolve	LOGICAL	Turns evolution of magnetic fields on and off.
ibcs	INTEGER	Sets spatial boundary condition for fast electrons.
Lsave_currents	LOGICAL	Turns outputting of current densities on and off.
Lsave_diags	LOGICAL	Turns outputting of auxillary diagnostics on and off.
Lsave_resist	LOGICAL	Turns outputting of resistivity on and off
Lsave_eden	LOGICAL	Turns outputting of electron density on and off
pxx_x1	REAL	Parameter for phase space auxillary diagnostic.
pxx_x2	REAL	Parameter for phase space auxillary diagnostic.
ipxxmode	INTEGER	Selects different options for the kinetic diagnostic
Lcheck_params	LOGICAL	Checks for certain errors at start.
ifields	INTEGER	Selects set-ups with externally imposed EM fields.
Lkalpha	LOGICAL	Turns on internal calculation of K_α photon emission rate.

Table 3.1: Summary of General Simulation Parameters

TARGET

Parameter	**Type**	**Function**
iconfig	INTEGER	Selects from a set of target materials and set-ups.
temp_in	REAL	Initial target temperature in eV.
loglambda	REAL	(INACTIVE).
loglambda_scat	REAL	Coulomb Logarithm for fast electron angular scattering.
loglambda_drag	REAL	(INACTIVE).
nresinp	INTEGER	The number of records in `resinp.dat`.
Lthomasfermi	LOGICAL	Thomas-Fermi ionization model and E.O.S.
LRedLeeMore	LOGICAL	Reduced Lee-More model for material resistivity.
lm_mfp_factor	REAL	Multiplied by r_s is min. electron m.f.p.
lm_multiplier	REAL	Gross multiplier to LM resistivity. Should set to 1.0 normally.
LSpecifyLMParam	LOGICAL	Switch to input of LM mfp factor via external file
LFixedResistivity	LOGICAL	Specify a constant resistivity for target.

Table 3.2: Summary of Target Parameters

FAST ELECTRON

Parameter	**Type**	**Function**
rspot	REAL	Characteristic laser spot size.
rspotmax	REAL	Maximum radius of fast electron injection region.
rspotmin	REAL	Minimum (inner) radius of fast electron injection region.
Ilaser	REAL	Peak laser intensity in Wcm^{-2}.
absfrac	REAL	(range 0–1) fraction of laser energy absorbed as fast electrons.
tlaser	REAL	Laser pulse duration.
iinject_mode	INTEGER	Selects angular distribution of fast electrons from presets.
ilaser_spot	INTEGER	Selects transverse laser spot profile from presets.
iinject_func	INTEGER	Selects energy distribution function of fast electrons.
lambda_laser	REAL	Laser wavelength (in μm).
div_angle	REAL	Characteristic fast electron divergence angle.
ilaser1	INTEGER	Controls fast electron temperature model.
ilaser2	INTEGER	Controls laser spot positioning (0:midpoint,1:user defined).
ylaser	REAL	User defined y-position of primary laser spot.
zlaser	REAL	User defined z-position of primary laser spot.
Lsecondspot	LOGICAL	Turns the secondary laser spot on and off.
Ilaser_2	REAL	Peak laser intensity in Wcm^{-2} of secondary pulse.
absfrac2	REAL	(range 0–1) absorption fraction for secondary pulse.
tlaser2	REAL	Laser pulse duration of secondary pulse.
rspot2	REAL	Characteristic laser spot size (secondary).
rspotmax2	REAL	Maximum radius of secondary injection region.
rspotmin2	REAL	Minimum (inner) radius of secondary injection region.
ylaser2	REAL	User defined y-position of secondary laser spot.
zlaser2	REAL	User defined z-position of secondary laser spot.
div_angle2	REAL	Divergence angle for secondary pulse.
Linjcontrols	LOGICAL	Turns on additional injection controls.

Table 3.3: Summary of Fast Electron Parameters

Chapter 4

General Simulation Parameters

The GENERAL parameters control the top-level computational aspects of the code rather than the more physical aspects. These parameters control the domain gridding, simulation time, output times, and logical switches for various solvers.

4.1 Spatial Parameters

ZEPHYROS uses a fully 3D Cartesian grid. The number of spatial cells for each dimension are specified individually by `nx`, `ny`, `nz`. The size of each cell in each dimension is specified individually by `dx`, `dy`, and `dz`. The fully 3D grids can take up a considerable amount of memory (the code uses about 17 such arrays), for example a $200 \times 200 \times 200$ array probably requires 61MB. So a run based on such a grid size will require at least 1GB in memory.

4.2 Temporal Parameters

The time that the simulation will run up to is specified by `tstop`. The time between outputs is defined by `dtsave`. The time step in the code is determined by $\alpha min\,(dx/c, dy/c, dz/c)$ where α is equal to `timefrac`.

4.3 Modular Physics Switches

Various physics modules in the code can be turned on and off by the following switches:

- `Lbrems`:The code has a simple module in which the background cools due to bremsstrahlung under the assumption that the target is optically thin.

- `Lxchang`:There is an electron-ion energy exchange module in the code based on the classical rate of e-i equilibriation in a plasma.
- `Lmdiff`:There is a magnetic diffusion module in the code that effectively solves for the resistive magnetic diffusion term in the induction equation.
- `LBevolve`:The evolution of the magnetic fields can be turned on and off using this switch.
- `Lthomasfermi` : Turns on the general Thomas-Fermi model for determining ionization state. See Ch.5.
- `LRedLeeMore` : Switches resistivity model to a 'reduced' Lee-More model. See Ch. 5.
- `Lkalpha` : Turns on internal calculation of K_α photon emission rate due to fast-electron-induced K-shell excitation. Uses the algorithm developed in A.G.R.Thomas et al., *New J.Phys.*, **15**, 015017 (2013).

4.4 Functional Switches

There are a number of switches that affect the general aspects of the code function. These include:

- `Lcheck_params`:The code checks the input deck for a certain number of hard-coded errors. This does *not* guarantee that every possible error has been checked for. The code will abort with an error message if an error is found.
- `Lsave_currents`:Turns the saving of the current density outputs on and off.
- `Lsave_diags`: Turns the saving of any special diagnostics on and off.
- `Lsave_resist`: Turns the saving of the target resistivity on and off.
- `Lsave_eden`: Turns the saving of the background electron density on and off.

4.5 Boundary Conditions

The boundary conditions for the fast electrons are controlled by `ibcs`. This selects from the following options:

0	All boundaries are reflective.
1	$x = 0$ boundary reflective. $x = x_{max}$ and side (y and z) are open.
2	x-boundaries are reflective. Side (y and z) boundaries are open.

Table 4.1: Spatial boundary conditions for fast electrons.

4.6 External EM Fields

The `ifields` parameter can be used to select set-ups with externally imposed EM fields. This will be of particular interest to those who are modelling scenarios in which a large scale magentic field has been imposed (e.g. by external coils and a pulsed-power system). The current options are :

`0`	All fields are zero (default case).
`1`	B_x is uniform and set equal to `control_param5`.

Table 4.2: EM initialization options.

4.7 Special Diagnostics

The code has a certain number of 'special diagnostics' that are hard-coded. Special diagnostics will *only* be saved if `Lsave_diags` is set to TRUE.

`0`	No special diagnostics.
`1`	Angular diagnostic.
`2`	K_α diagnostic.

Table 4.3: Special Diagnostic configurations available

4.7.1 Angular Diagnostic

This returns an analysis of the fast electrons in each spatial cell. The `dg1` file will contain the average fast electron propagation angle with respect to the x-axis. The `dg2` file will return the standard deviation of the fast electron propagation angle with respect to the x-axis in each spatial cell.

4.7.2 K-alpha Diagnostic

This will output the K_α photon emission rate as determined by the code's internal K_α algorithm. This algorithm is largely identical to the algorithm described in A.G.R.Thomas et al., *New J.Phys.*, **15**, 015017 (2013). In order for this to work properly, that component must be activated, i.e. `Lkalpha` must be set to TRUE. The photon emission rate in each spatial cell is written to `dg1`.

Chapter 5

Target Parameters

The TARGET parameters control the target or the background plasma. This is done through the `iconfig` parameter which chooses from the following:

- **Configurable Hot Plasma**: One can choose a homogeneous hot plasma (Spitzer resistivity and Ideal E.O.S. for electrons) with specified ion density, Z, and initial temperature.

- **Arbitrary Hot Plasma**: One can supply a 3D map of Z and n_i to the code via additional files (`ninp.dat` and zinp.dat) in order to obtain a multi-material hot plasma with an arbitrary distribution of Z and n_i (uniform initial temperature).

- **Set Cold Target Configurations**: There are a number of hard-coded fixed configurations that can be chosen. This includes structured and multi-material configurations as well as homogeneous targets.

Additionally one can choose any of the above, and then request the use of a reduced Lee-More model for the resistivity curves by setting `LRedLeeMore` to `.true.`. The atomic model that is used for the effective Z of the background ions is, by default, a very simple analytic fit to the Thomas-Fermi model used by Davies. The analytic fit used by More can be selected by setting `Lthomasfermi` to `.true.`. The reduced Lee-More model automatically makes use of this atomic model for determining the resistivity curve. The ionization state of the ions affects not only the resistivity curves, but the rate of angular scattering from the background ions and the level of drag on the fast electrons due to collisions with the background electrons.

The settings that `iconfig` chooses from are summarized in the table below. These are then described in more detail in the following sections.

ICONFIG Value	Description
1	Ideal Hot Plasma
2	Al-Sn-Al 'Sandwich' target
3	Homogeneous cold Al target
4	Homogeneous cold CH target
5	Fe-Al embedded wire target
6	Carbon Allotrope with external resistivity
7	CH foam target
8	Sulphur foil target
9	CH-Ta buried layer target
10	Gold foil target
11	Ti-Al buried layer target

Table 5.1: `iconfig` settings

5.1 Homogeneous Cold Targets

There are many situations where one wants to investigate fast electron propagation through homogeneous targets at low temperatures (where they may only be heated to a few eV). The `iconfig` presets 3,4,8, and 10 provide these for Al, CH, S, and gold respectively. The initial temperature of the target is then set by the `temp_in` parameter. These settings are not otherwise configurable in general.

5.2 Resistive Guiding Targets

The configurations selected by `iconfig` =2 and 5 are two 'resistive guiding' configurations. `iconfig = 2` selects an Al-Sn-Al sandwich configuration (layers alternating in the y-direction), and `iconfig = 5` selects a configuration with an Fe wire immersed in an Al substrate.

5.2.1 Al-Sn-Al Sandwich Target

In the `iconfig` =2 target configuration, the default profile of the sandwich layer `icontrol1` =0 is a tanh profile,

$$\psi = \frac{1}{2}\left(1 - \tanh(\frac{y - y_m - \alpha}{\beta})\right), \text{for} y \geq y_m, \tag{5.1}$$

where ψ is the volume fraction of Sn, y_m is the y-midpoint, α is the boundary of the Sn layer, and β is the smoothing length. The parameter α is set by control3, and the parameter β is set by control4.

5.2.2 Fe Wire Target

In the `iconfig` =5 target configuration, the default profile of the wire (`icontrol1 = 0`
is defined by,

$$\psi = \frac{1}{2}\left(1 - \tanh(\frac{\rho - \alpha}{\beta})\right), \tag{5.2}$$

where ψ is the volume fraction of Fe, $\alpha =$ `control3`, and $\beta =$ `control4`. Setting `iconfig` =1 will eliminate the Al wire (for testing).

5.3 Buried Layer Targets

The configurations selected by `iconfig` =9 and 11 are 'traditional' buried layer targets (layers alternating in x-direction). `iconfig = 9` is a Ta buried layer in CH, and `iconfig = 11` is a Ti buried layer in Al.

5.3.1 Al-Ti Target

In the `iconfig = 11` configuration, the only configurable aspect is the position of the front edge of the Ti layer and the position of its back edge. These positions are set by `control1` and `control2` respectively. The Al component uses Davies' resisitivity curve. The default resistivity curve for Ti is user defined through the equation,

$$\eta = \frac{T}{\alpha + \beta T + \gamma T^{5/2}}, \quad \Omega\mathrm{m} \tag{5.3}$$

,where T is in eV, α is set by `control3`, and β is set by `control4`.

5.3.2 CH-Ta-CH Target

In the `iconfig = 9` configuration the only configurable aspects are the positions of the front and back edges of the Ta layer. As with `iconfig = 11`, these positions are set by `control1` and `control2` respectively. The CH component uses Davies' resisitivity curve. The default resistivity curve for Ta is user defined through the equation,

$$\eta = \frac{T}{\alpha + \beta T + \gamma T^{5/2}}, \quad \Omega\mathrm{m} \tag{5.4}$$

,where T is in eV, α is set by `control3`, and β is set by `control4`.

5.4 Externally Supplied Resistivity Configuration

The configuration selected by `iconfig = 6` allows one to specify the ion density and initial temperature, *as well* as the resistivity curve. The resistivity curve must be supplied in a file named `resinp.dat` which should be a two column text file with the first column being a list of temperatures (in eV) and the second column being a list of of corresponding resistivities (in Ωm). The ion density n_i is taken from `control2` and the ion charge, Z, from `control1`.

5.5 Configurable Hot Plasma

By selecting `iconfig = 1`, one is choosing a homogeneous hot plasma (ideal E.O.S. and Spitzer resistivity). The standard operation of this configuration is to take the ion density n_i from `control2` and the ion charge, Z, from `control1`. The initial temperature is taken from `temp_in` as usual.

5.6 User Defined Hot Plasma

By selecting `iconfig = 1` *and* `iconfig2 = 3`, one selects a configuration in which both the Z and n_i are read in via the files `zinp.dat` and `ninp.dat`. These need to be prepared in advance and placed in the code's run directory. This setting is highly flexible and allows the user to investigate a very wide range of hot plasma target configurations. There is also the `iconfig = 1` *and* `iconfig2 = 5` setting, where one can also specify the initial electron and ion temperature by supplying the file `tcinp.dat` as well.

5.7 Control of the Lee-More Resistivity Model

There are number of ways in which the Lee-More model can be 'fine tuned'. The minimum mean free path can be set as a multiple of the interatomic spacing via `lm_mfp_factor`. The resistivity curve can have a global multiplier applied to it via `lm_multipler`. Finally one can specify a mean free path multiplier for each cell individually by setting `LSpecifyLMParam` to `.true.` and supplying an external file containing the values.

5.8 Fixed Resistivity

If the user would like to specify a universal, constant resistivity for the target then this can be done by setting `LFixedResistivity` to `.true.`. The value of the resistivity is then set via control9.

Chapter 6

Fast Electron and Laser Parameters

The FAST ELECTRON parameters control the injection of fast electrons. This is meant to model fast electrons that are generated by an ultraintense laser. Hence this entire aspect of the code is based around a 'laser model'.

6.1 Laser Model

Fast electrons are injected into the computational domain by assuming that they are promoted from the background. Fast electron generation is not done self-consistently, but is done as an energy dump over a 'laser spot' profile in the transverse direction and a few cells depth in the x-direction. Effectively the number of fast electron injected into these 'generative' cells every time step is,

$$N_{fast} = \frac{\eta_{abs} I(r) \Delta y \Delta z dt}{\bar{\varepsilon}_{fast}}, \tag{6.1}$$

where η_{abs} is the `absfrac` (absorption fraction) parameter, $I(r)$ is the intensity as a function of radius (in the transverse direction), and $\bar{\varepsilon}_{fast}$ is the average fast electron energy.

The laser model parameters of immediate importance are : `Ilaser`, `absfrac`, and `tlaser`. The parameter `Ilaser` determines the peak laser intensity and is given in units of Wcm^{-2}. The parameter `tlaser` determines the laser pulse duration and is given in units of seconds. The parameter `absfrac` should be in the range 0–1 and this simply defines η_{abs}. The user needs to determine this depending on what he or she believe the conversion efficiency to be in the problem that is being simulated.

The primary laser spot is always located adjacent to the $x = 0$ plane, however the central position of this laser spot can be defined by the user. If `ilaser2` is set to zero then the spot will be centred on the midpoint. If however `ilaser2` is set to 1 then the spot will be centred by the values given to `ylaser` and `zlaser`.

A second laser spot on the $x = x_{max}$ plane can be used by setting `Lsecondspot` to TRUE. Details of this feature are given later on.

6.2 Laser Temporal Profile

The only temporal profile that is currently implemented is a top hat profile from $t = 0$ up to $t =$`tlaser`

6.3 Laser Spot

The transverse profile of the 'laser', i.e. $I(r)$, is controlled by the `ilaser_spot` parameter in the input deck. This provides a multiplier for `Ilaser` to determine the 'laser intensity' at a given radial point. The following options are available for `ilaser_spot`:

`default (0)`	Gaussian; $\propto \exp\left[-\frac{r^2}{2r_L^2}\right]$
1	JRD's q-Gaussian;$\propto \frac{1}{(1+\left(\frac{r}{r_L}\right)^2)^{1.4748}}$
2	4th order super-Gaussian; $\propto \exp\left[-\frac{r^4}{r_L^4}\right]$
3	8th order super-Gaussian; $\propto \exp\left[-\frac{r^8}{r_L^8}\right]$
4	Top-Hat; I = I_0
5	Exponential; $\propto \exp\left[-\frac{r}{r_L}\right]$

Table 6.1: The angular distribution models available and corresponding `ilaser_spot` values.

In these relations the term r_L is defined by the input parameter `rspot`. Use the relations provided in table 6.2 to determine the relation between r_L and the HWHM of the spot. The injection region extends up to the radius defined by `rspotmax`. A ring-shaped injection profile can be specified by giving the parameter `rspotmin` a value.

6.4 Fast Electron Energy Distribution

The distribution function of the fast electron is selected via `iinject_func`. The various options are detailed in what follows. Nearly all of these are based on a single energy parameter. This is determined by default via a Wilks-like scaling relation :

$$T_f = 2m_e c^2 \left(\sqrt{1 + \frac{I\lambda_L^2}{\texttt{1.38e18}}} - 1 \right), \tag{6.2}$$

where λ_L is determined by the input parameter `lambda_laser` (which is given in units of microns). This can be circumvented by setting `ilaser1` to 1 (it should be set to 0 otherwise). Once this is done the user can specify an intensity independent fast electron temperature via `control5` in units of electron volts.

The options for the distribution function are :

`iinject_func = 0`: The fast electrons have an exponential energy distribution of the form,

$$f(E) = \exp\left[-\frac{E}{T_f}\right]. \tag{6.3}$$

`iinject_func = 1`: The fast electrons have a monoenergetic energy distribution.

`iinject_func = 2`: The fast electrons have a top-hat or waterbag distribution. This is a uniform distribution from 0.5 to 1.5 times the characteristic energy.

6.5 Fast Electron Angular Distribution

The angular distribution is set by the `iinject_mode` parameter. This selects from a number of hard-coded models that are included in the code. Below is a list of these models.

0	Uniform distribution over a solid angle subtended by a half-angle of `div_angle`
1	$\propto \cos^2$
2	$\propto \cos^4$
4	$\propto \cos^8$
5	$\propto \cos^{10}$
6	$\propto \cos^{12}$
7	$\propto \cos^{20}$
8	”Moore” model (Energy Dependent)
9	“Sheng” model (Energy Dependent)

Table 6.2: The angular distribution models available and corresponding `iinject_mode` values.

The “Moore” model uses a uniform distribution at each electron energy up to $\tan^{-1}\left(\frac{2}{\sqrt{\gamma-1}}\right)$ or `div_angle` (whichever is smaller). The “Sheng” model is similar but includes the angle of incidence (equal to `div_angle`;no high angle clamping).

6.6 Second Laser Spot

A second laser spot is available by setting `Lsecondspot` to TRUE. This spot is controlled as follows:

- The intensity, pulse duration, and spot characteristics are independently controlled by `Ilaser_2`, `absfrac2`, `lambda_laser2`, `tlaser2`, etc. in the same way as these control the primary laser spot.
- Currently the fast electron temperature relation cannot be set independently.
- Currently the fast electron angular distribution cannot be set independently, but a different characteristic angle (`div_angle2`) can be specified.
- The central position of this spot can be set independently when `ilaser2` is set to 1 via `ylaser2` and `zlaser2`.

The macroparticles are divided equally between each pulse when `Lsecondspot` is TRUE. This means that the PPC of each beam is degraded compared to a single beam. To maintain PPC the user must therefore double the total number of macroparticles.

Chapter 7

ZEPHYROS Output

7.1 ZEPHYROS Output

In this chapter we will look at visualizing ZEPHYROS output using common scientific software. Firstly let us look at the various files that are produced:

7.1.1 EM Fields

`ex__**.dat`
`ey__**.dat`
`ez__**.dat`
`bx__**.dat`
`by__**.dat`
`bz__**.dat`
These are the electric and magnetic fields (in SI) at the grid cell centres. They are written into a single data column of total length equal to `nx*ny*nz`. This is written with the x loop innermost, and the z loop outermost.

7.1.2 Particle Moments

`nf__**.dat`
`jx__**.dat`
`jy__**.dat`
`jz__**.dat`
These are the fast electron density and the components of the fast electron current density (both in SI) at the grid cell centres. They are written into a single data column of total length equal to `nx*ny*nz`. This is written with the x loop innermost, and the z loop outermost. The fast electron current densities will only be saved if `Lsave_currents` is `.true.`.

7.1.3 Background

```
ni__00.dat
Z___**.dat
tb__**.dat
ti__**.dat
eta_**.dat
```

These are the background ion density, average ion charge, background electron temperature, the background ion temperature, and the background resistivity in the aforementioned format. The background electron and ion temperatures are not in SI and are in units of eV instead. The switch `Lsave_resist` must be set to `.true.` in order for the resistivity files to be saved.

7.1.4 Fast Electron Kinetics

The parameters `pxx_x1` and `pxx_x2` define a lower and upper y-z plane in the domain. At each save point the code will record output about the macroparticles within this 'sample window' depending on the value of `ipxxmode`:

`ipxxmode = default`: y,z,p_x,p_y,p_z (momenta are *per electron* not *per macroparticle*) of every macroparticle between these two planes.

`ipxxmode = 1`: p_x,p_y,p_z (momenta are *per electron* not *per macroparticle*) of every macroparticle between these two planes, *and* the number of real electrons per macroparticle as a fourth number.

This output is saved in the `pxx_**.dat` files.

7.2 Visualization with MATLAB

MATLAB is a common commercial mathematical and data analysis package. Visualizing ZEPHYROS output in MATLAB is fairly straightforward, however a full explanation of all that can be done in MATLAB goes well beyond the scope of this manual. ZEPHYROS users who will regulary used MATLAB are advised to seek MATLAB specific training! First set the MATLAB path to point to the directory. The files can then be immediately loaded in via MATLAB's `load` facility:

```
y = load('nf__01.dat');
```

This line for example will load the first data output of the fast electron density into the vector `y`. This can then be transformed into a 3D array as follows:

```
nx = 100;
ny = 100;
nz = 100;
for iz = 1:nz
for iy = 1:ny
for ix = 1:nx
n = ix + nx*(iy-1) + nx*ny*(iz-1);
nf(ix,iy,iz) = y(n);
end;
end;
end;
```

Now that the full 3D array has been created, any slice or lineout that one wishes can be created. A typical approach might be to do a 'pseudocolor' plot the x-y midplane:

```
pcolor(log10(max(1e24,f(:,:,50).')));
```

If one only intends to look at the midplane plots, then the procedure can be simplified as follows:

```
y = load('nf__01.dat');
nx = 100;
ny = 100;
nz = 100;
iz = 50;
for iy = 1:ny
for ix = 1:nx
n = ix + nx*(iy-1) + nx*ny*(iz-1);
nf(ix,iy) = y(n);
end;
end;
pcolor(log10(max(1e24,f(:,:).')));
```

7.3 Visualization with Pytheros Scripts

The Pytheros suite is a set of Python scripts that were build to complement the ZEPHYROS code by providing handy utilities for post-processing the ZEPHYROS data. The main ZEPHYROS output files are usually fairly bulky (> 50MB), and ZEPHYROS users often only want to study small portions of the data, at least initially. Currently Pytheros contains the following:

1. `pyth_zslicer.py`: This will produce an x-y slice at a specified z-plane. It will write the output in a format ready for use by `pyth_autoplot.py`. This operation will be performed on all files of a specific prefix (e.g. 'nf') in the directory that it is called from.

2. `pyth_autoplot.py`: This script will look for files of a certain prefix, and then prepare, write and execute a gnuplot script to produce pseudocolor plots of all these files.

3. `pyth_zsliceplot.py`: Extracts slice for a specified value of z, i.e. an $x - y$ slice. Plots using matplotlib library.

4. `pyth_xsliceplot.py`: Extracts slice for a specified value of x, i.e. an $y - z$ slice. Plots using matplotlib library.
5. `pyth_ysliceplot.py`: Extracts slice for a specified value of y, i.e. an $x - z$ slice. Plots using matplotlib library.

So to automate the graphing up of the fast electron density at the x-y midplane for all outputs can be done by simply making two calls to each of these scripts. You will then have a set of PNG files that can be quickly downloaded for viewing, with axes already labelled!

7.3.1 Slicing

Before auto-plotting, one must first slice the data. For this, use `pyth_zslicer.py`. The first step is to edit the script. On opening the script, you will see that there is a clear 'header' region, and a 'code' region, with a comment asking you not to edit past a certain point. The header should look like this:

```
#
# z−slicer script for ZEPHYROS post−proc
# Currently written assuming Python 2
#
# Formats sliced file
# for GnuPlot plotting
#
import os
import math

#Set file set to be sliced:
toslicename = 'nf__'
#Set output file prefix:
outprefix = 'nfzs'
outsuffix = '.dat'
#Define grid sizes:
nx = 100
ny = 400
nz = 400
#Define cell sizes
delx = 0.5
dely = 0.5
#Set slice layer:
nzslice = 200

# NO NEED TO DEAL WITH ANYTHING BELOW!!
```

So one needs to check that all of these parameters are set correctly. The first parameter 'toslicename', is the leading string that the script will search for in the files that you want to slice. In this case it will search for the fast electron density files, but one can set it to any of the other outputs (e.g. 'ex__','bz__','tb__', etc etc).

Next one must make sure that you are happy with the 'outprefix' and 'outsuffix' which along with a number between them, will define the name of the sliced file that is produced. The script must also know the grid lengths ('nx','ny', and 'nz') and also the z-index that you want to slice at ('nzslice'). The cell sizes should also be specified, as the sliced file contains x and y coordinates (for plotting in gnuplot). In the example above, I have specified the cell size in microns.

Once these are set. Call the script from the command line via:

```
>python pyth_zslicer.py
```

and check that a set of sliced files appears in the directory. Note that you must do this from the directory in which the files sit. The sliced files can now be used on their own, or as an input to `pyth_autoplot.py`.

7.3.2 Auto-plotting

The `pyth_autoplot.py` script will automatically generate a gnuplot based script that includes a plotting call for every sliced file that it finds in its local directory. It requires the sliced files produced by `pyth_zslicer`. The script is currently configured for use on SCARF (where RMGMT and I have installed gnuplot in the CLF work area, and where Derek Ross (e-Science) has kindly the MS TrueType fonts package), although with a little re-jigging it can work on other systems as well. To use this script, first edit the 'header':

```
# Python script to do automatic plotting
# of sliced data via GnuPlot
#

import os

# Prefix of slices to be auto—plotted:
prefix = 'nfzs'
# Plotting details:
plotxlabel = 'x (Microns)'
plotylabel = 'y (Microns)'
plotxrange = '[0:50]'
plotyrange = '[0:200]'
colrange = '[1e24:1e27]'
xformat = '%3.0f'
yformat = '%3.0f'
reqfont = 'comic'

# Please don't change these unless
# you REALLY have to
#Specify location of gnuplot:
gplotloc = '/work/clf/gplot_public/gnuplot—4.6.0/src/'
#Specify location of TrueType Fonts:
fontpath = '/usr/share/fonts/msttcorefonts/'
```

The most important first step to to set the prefix of the sliced files that you want to plot, which is done by setting the string **prefix**. So here I am setting it to look for the z-sliced files of the fast electron density that I produced using the z-slicer script.

The remaining bits are cosmetic/aethestic really. You can write x and y labels using the **plotxlabel** and **plotylabel** strings. The spatial ranges over which you plot is then controlled by setting **plotxrange** and **plotyrange**, and the color range is set by **colrange**. The precision of the axis labels is set by the **xformat** and **yformat** strings. The font is set by the **reqfont** string (SCARF has arial, times, verdana, courier, and georgia available too).

Currently the script is set to do log plots. In later versions this will be controlled via the header. To switch back to linear, one will need to go into the code section and comment out the following line with a '#':

```
gplotfile.write('set log cb' + "\n")
```

7.3.3 Single Slice Scripts

The `pyth_zsliceplot.py`, `pyth_xsliceplot.py`, and `pyth_ysliceplot.py` scripts are all useful for where one wants to get single slices at particular times rather than process an entire data set. Use of these scripts will require the matplotlib library to be installed. To use these scripts start by editing the 'header' material:

```
# PYTH_ZSLICEPLOT.PY:
#-------------------
# by A.P.L.Robinson, CLF, 2013
# Copyright : Science and Facilities Council 2013
# email : alex.robinson@stfc.ac.uk
#
#--------------------------------------------------
# Dependencies : None
#--------------------------------------------------
# Release History : ver 1 January, 2013
#--------------------------------------------------
# Single Plot script for ZEPHYROS post-proc
# Currently written assuming Python 2
# makes use of the matplot lib library
#
#--------------------------------------------------


import os
import math
import numpy as np
from matplotlib import pyplot as plt

#Set file set to be sliced:
toslicename = 'nf__10.dat'
#Set output strings:
title = "Z Map"
xlab = "x / microns"
ylab = "y / microns"
```

```
figfile = "nf10.png"
#Define grid sizes:
nx = 250
ny = 200
nz = 200
#Define cell sizes
delx = 1.0
dely = 1.0
#Set slice layer:
nzslice = 100

# NO NEED TO DEAL WITH ANYTHING BELOW!!
#
#CODE *****************
```

Set the cell numbers and cell sizes to conform to your simulation data. Set the desired slice index via `nzslice`, and that `toslicename` refers to the correct file. You may have to check in the code section for anything to do with extra data processing, e.g. for log plotting.

7.3.4 Other Pytheros Scripts

Other Pytheros scripts that may be of interest include:

- `pyth_makeb.py`:Will produce files containing the magnitude of **B** in each cell.
- `pyth_makej.py`:Will produce files containing the magnitude of $\mathbf{j_f}$ in each cell.

7.4 Visualization with VisIt

The VisIt package (https://wci.llnl.gov/codes/visit/) can be used to perform a range of fully 3D visualizations of ZEPHYROS output. There is a Windows version of VisIt available.

The first step is to convert the ZEPHYROS output files into .vtk files. This can be done with the clfVtkWriter script that is available. This linux script should be run as follows :

```
>./clfVtkWriter <input filename> <nx> <ny> <nz> <output filename>
```

So a typical usage might be something like this :

```
>./clfVtkWriter nf__01.dat 200 200 200 thickness_scan_1.vtk
```

The .vtk files will most likely be somewhat larger (a factor of a few) bigger than the .dat files. If data storage is an issue then please be aware of this. The .vtk files can then be visualized in VisIt. Like MATLAB, the entireity of the possiblities in VisIt is well beyond the scope of this manual. However a basic procedure would look something like this:

1. Choose ‘Open File’ and select your .vtk file. Tell VisIt to open it as a VTK file.
2. Add a pseudocolor plot, selecting ‘scalars’
3. Add a three-slice operator, choosing the slice planes as 1, `ny`/2, `nz`/2 for x, y, and z respectively.
4. Select Draw to produce the plot.

7.5 Other Tools

MATLAB and VisIt are not the only tools that can be used to visualize ZEPHYROS output. ZEPHYROS output can be visualized using Golden Software’s *Voxler 3* package with a simple post-processing of the .dat files in MATLAB, and in the GNU-licensed *Octave* package.

Chapter 8

Example Run

8.1 Overview

In this chapter we will present an example input deck and the output that this produces. New users may wish to try running this input deck and checking that they can obtain the same or at least similar output.

8.2 Example Input Deck

The input deck used for this example is as follows:

```
!##
!##  Example Input Deck for Zephyros v1.1
!##  A.P.L.Robinson July 2014
!##

&inpvars

nfasts = 200000000
nx = 200
ny = 200
nz = 200
dx = 1.0d−6
dy = 1.0d−6
dz = 1.0d−6
iconfig = 3
idiagnostic = 0
tstop = 2.0d−12
dtsave = 0.1d−12
timefrac = 0.75d0
temp_in = 1.0d0
loglambda = 5.0d0
loglambdascat = 5.0d0
loglambdadrag = 5.0d0
rspot = 5.0d−6
```

```
rspotmax = 15.0d-6
rspotmin = 0.0d-6
Ilaser = 3.0d20
absfrac = 0.3d0
tlaser = 5.0d-13
iinject_mode = 0
ilaser_spot = 0
iinject_func = 0
lambda_laser = 1.053d0
div_angle = 0.8727d0
control1 = 0.0
control2 = 0.0
control3 = 0.0
control4 = 0.0
control5 = 7.04d6
control6 = 0.0
control7 = 0.0
control8 = 0.0
control9 = 0.0
control10 = 0.0
control11 = 0.0
control12 = 0.0
icontrol1 = 0
icontrol2 = 0
icontrol3 = 0
ilaser1 = 1
ilaser2 = 0
ylaser = 100e-6
zlaser = 100e-6
las_control1 = 0.0
las_control2 = 0.0
las_control3 = 0.0
las_control4 = 0.0
las_control5 = 0.0
Lbrems = .false.
Lxchang = .true.
Lmdiff = .true.
LBevolve = .true.
Lcheck_params = .true.
Lthomasfermi = .true.
LRedLeeMore = .true.
Linjcontrols = .false.
Lkalpha = .false.
LSpecifyLMParam = .false.
LFixedResistivity = .false.
nresinp = 401
iconfig2 = 0
iconfig3 = 0
icontrol4 = 0
ibcs=2
ifields = 0
lm_mfp_factor = 8.0d0
lm_multiplier = 1.0d0
Lsave_currents=.false.
Lsave_diags = .false.
Lsave_resist = .true.
Lsave_eden = .false.
```

```
Lsave_zstar = .true.
pxx_x1 = 0.0d0
pxx_x2 = 1.0d-6
ipxxmode = 0
Lsecondspot = .false.
Ilaser_2 =1.0d20
absfrac2 =0.3d0
lambda_laser2=1.0d0
tlaser2=1.0d-12
rspot2=5.0d-6
rspotmax2= 20.0d-6
rspotmin2=0.0d0
ylaser2= 120.0d-6
zlaser2= 100.0d-6
div_angle2=0.6d0/
&end
```

8.3 Understanding the Input Deck

The fortran comments ('! blah blah') at the top are indeed just comments. You can use this to insert notes into each input deck. It is a good idea to make use of this to label each input deck, and in the long run it can be invaluable for record-keeping.

The first seven lines should be clear. Twenty million macroparticles will be used on a 200×200×200 grid. Assuming 8 bytes per double precision number, you should make sure that your system memory can comfortably accommodate this. Next we notice that `iconfig` is set to three. This means that the target will be a homogeneous Al target. Further down we see that the Thomas-Fermi and Lee-More logical switches are set to `.true.` as well. From `tstop` and `dtsave` we see that the simulation is set to run up to 2 ps, with output data saved every 0.1 ps.

In terms of the fast injection model (or 'laser') we see that it consists of a 5 μm Gaussian spot, an intensity of 3×10^{20}Wcm^{-2}, a laser to fast electron conversion efficiency of 30%, and a 0.5 ps pulse duration. As `ilaser1` is set to 1, the fast electron temperature is taken from `control5` (7.04 MeV in this case). The fast electrons are injected with a uniform distribution over a solid angle subtended by a half-angle of 50°.

These are the essential aspects of the simulation. In summary we have a 240 TW, 0.5 ps, 120J system. In reality since only 30% of the laser energy will be in the central laser spot, an actual laser system that delivered this would have to be a 360 J, 720 TW system. This is comparable to the CLF's VULCAN facility or LLNL's TITAN. This is irradiating a thick Al foil.

8.4 Running the Example

This input deck does not require any files other than the executable and the input deck. Here are a simple set of steps for running the code on a linux system, assuming that you have the source files and makefile.

1. Ensure that the fortran compiler is the correct one for your system. Most linux distributions will include **gfortran** in the packages. So you should be able to install this, and then set `FC` equal to `gfortran`. Do `make zeph` at shell prompt (in the directory containing both the .f90 files and the makefile) to build the code. You should end up with a `zeph` executable.

2. Create a 'run' directory in a suitable location using `mkdir`. Copy the `zeph` executable to this location.

3. In your 'run' directory create a text file. Copy the input deck (above) into this file and save it as `readin.in`.

4. Now do `./zeph` at the shell prompt in your run directory. The code should execute.

8.5 Output

The output of this simulation can be visualized using the "pytheros" python scripts mentioned in Section 7.3 of Chapter 7. We will use 'pyth_zslicer' and 'pyth_autoplot'.

Starting with the electron density we check that, in 'pyth_zslicer.py', the target prefix is set to 'nf' and the correct cell numbers and cell sizes are inserted. Then run the python script from the shell prompt via `python pyth_zslicer.py`. Once this has run you can run 'pyth_autoplot.py'. Make sure that it is targetting the correct prefix, which should be 'nfzs', and the the x, y, and colorbar ranges are set appropriately. For this run they should be 0 to 200 for x and y, and 1e24 to 1e27 for the colorbar range. On running 'pyth_autoplot.py' via `python pyth_autoplot.py`, you should get a series of .png files.

Here are the fifth (500 fs) outputs for fast electron density, temperature and the B_z component of the magnetic field:

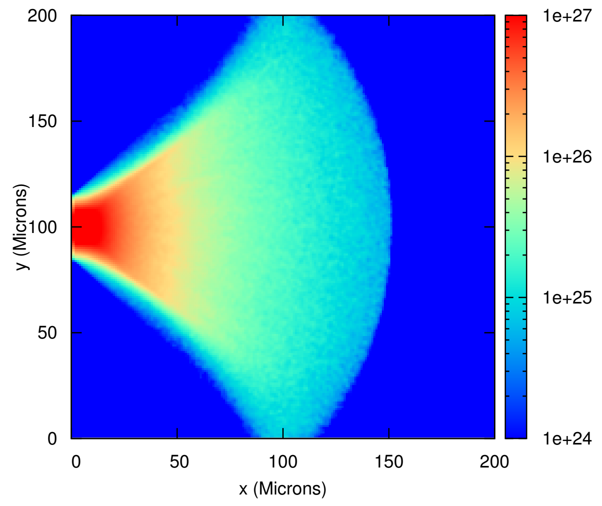


Figure 8.1: Fast Electron Density (m^{-3}) at 500 fs in example run.

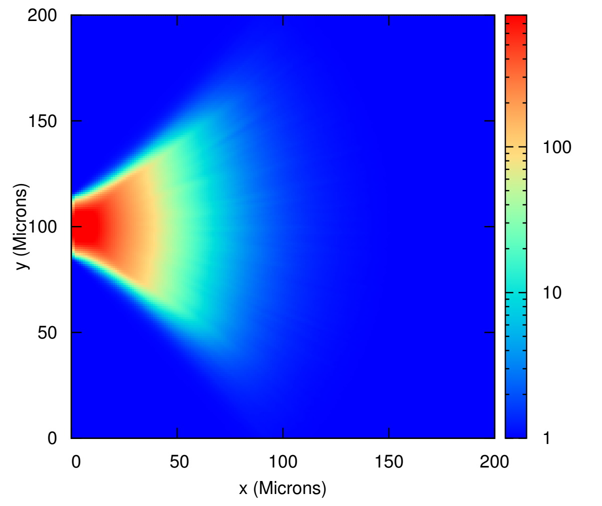


Figure 8.2: Background Electron Temperature (eV) at 500 fs in example run.

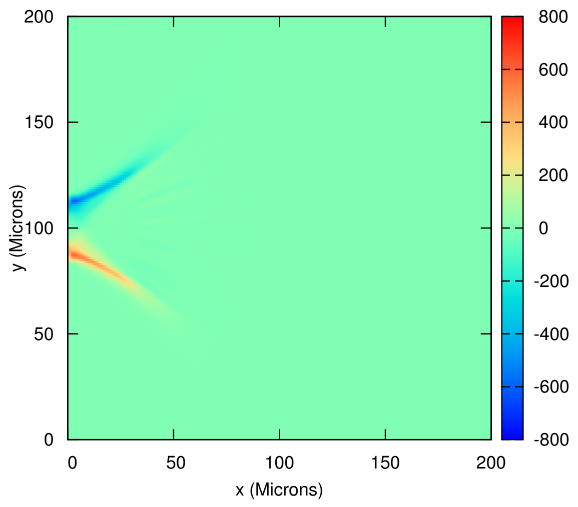


Figure 8.3: B_z (T) at 500 fs in example run.