1.2 Raytracing with `Zgoubi`- Solving the Exercises

`Zgoubi` is a stand alone series of Fortran files, compiling does not require any specific library. Running `zgoubi` requires no interface (various interfaces have been developed over the years though, and made available, see Sect. 1.3).

A beam optics problem in `zgoubi` consists in an ASCII input data file, its default name is zgoubi.dat. That ASCII file may actually be split, in as many ancillary files as desired, for instance according to a modular structure of an optical sequence.

Executing zgoubi.dat is as simple as this:

[pathTo]/zgoubi-code/zgoubi/zgoubi

i.e. typing the address of the executable file. The execution produces an output ASCII listing, zgoubi.res, always. `Zgoubi` may produce various additional output files during execution, according to user's requests.

One has to bear in mind that the only thing `zgoubi` knows to do is pushing particles: starting from an initial position and velocity, it computes particle coordinates along an optical sequence, by stepwise integration of the Lorentz force differential equations of motion. The input data file describes that optical sequence; it also includes diverse commands aimed at delivering ancillary results, the latter anyway derived from particle coordinates. As aforementioned a few things may actually happen while particles are pushed: spin motion, decay in flight, synchrotron radiation, space charge perturbation, etc.

An optical sequence in `zgoubi` is a sequence of keywords, most of them followed by one or more lines of numerical data (*e.g.*, in the case of optical elements: length, field value, integration step size, fringe field parameters possibly, etc.), like so:

```
Title: this is my optical sequence. Particles will be
! pushed through, all the way to 'END'
'OBJET'
a few lines of data define initial particle coordinates (initial
conditions are needed to solve the differential equation of motion!)
'DIPOLE'
a few lines of parameters: field, fringe fields, etc.
                                            ! this is a comment line
'FAISCEAU'                    ! print out local particle coordinates
'QUADRUPO'
a few lines of parameters: field, fringe fields, etc.
'DIPOLE'
a second dipole
                                        an empty line, not a problem
'BEND'
another type of dipole, with its own parameters and subtleties
'DRIFT'
drfit length
'FAISCEAU'                    ! print out local particle coordinates
'SYSTEM'
2                                                 ! 2 commands follow
echo 'this is a system call'
gnuplot < ./gnuplot_ellipses.gnu                ! some gnuplot script
'END'                                          ! execution stops here
trash                           ! whatever follows is trash, ignored
more trash
```

An optical sequence begins with a title line. And then:

OBJET: most of the time the first keyword, it defines the coordinates of particles making up the object to be transported; this is mandatory as initial conditions are needed in order to solve the Lorentz force equation.

Optical elements and commands follow, for instance

- DIPOLE: define a dipole magnet;
- EBMULT: a combined **E**, **B** multipole;
- ELCYLDEF: a cylindrical electrostatic deflector; MULTIPOL: lenses; CAVITE to accelerate; TOSCA to handle field maps; WIENFILTER; etc.

Zgoubi offers a total of about 50 magnetic and/or electrostatic optical elements [1, pp. 9, 10 and 13, 14].

Commands - which are keywords as well - are added wherever desired along the optical sequence, they include such procedures as

- FAISCEAU, FAISTORE: log local particle coordinates, respectively in zgoubi.res or in an ancillary output file;
- IMAGE[S]: compute local image density and size, etc.;
- GOTO: move the execution pointer to some arbitrary location along the sequence (useful for instance for managing beam transport amongst recirculating linacs spreader and combiner sections);
- TWISS, MATRIX: compute paraxial quantities from rays; SYSTEM: a system call;

- INCLUDE: to include ancillary input data files, a recursive command.

Keywords include switches, for instance to request

- spin tracking: SPNTRK, whose numerical data include initial spins, a necessary ingredient as initial conditions are needed in order to solve the Thomas-BMT equation;
- space charge perturbations: SPACECHARGE;
- in-flight decay: MCDESINT, synchrotron radiation: SRLOSS, etc.

Launching matching procedures resorts to FIT, FIT2 keywords, two different matching methods.

In the exercises, optical elements and procedures are most of the time referred to by their corresponding keyword, with little additional explanation: further information regarding their use and functioning is to be found in the indispensable companion to the resolution of the exercises, Zgoubi Users' Guide [1]:

- PART A of the guide describes what keywords do and how, and the physics content of the code, optical elements in particular.
- PART B details the formatting of the input data which follow most keywords (a few keywords do not require any data, for instance YMY, FAISCEAU, MARKER).
- A complete list of the available keywords can be found in the "Glossary of Keywords" sections at the beginning of both PART A and PART B.
- A quick overview of what optical elements can be simulated using `zgoubi`, and what keywords can be used for that, is given in the "Optical elements versus keywords" sections which follow the "Glossary of Keywords" sections. Note in passing, there are most of the time various ways to simulate one particular optical element, either for historical reasons, or to allow for actual and/or real life subtleties (for instance, between a gradient dipole and an offset quadrupole; between the various modes of operation of an accelerating radio-frequency system).
- The Index at the end of Zgoubi Users' Guide is a convenient tool to navigate keywords.

A concise notation KEYWORD[ARGUMENT1, OPTION, ...] is used in the exercises and solutions: it is believed that the reader will get promptly familiarized with these shortcuts, of which the main goal is to alleviate the text. The nomenclature KEYWORD[ARGUMENT1, OPTION, ...] follows the nomenclature of the Users' Guide, Part B. Three examples:

- OBJET[KOBJ=1] stands for keyword OBJET (generating particle coordinates), and KOBJ=1 option retained here;
- DIPOLE[IL=2,XPAS=2.5] stands for keyword DIPOLE (magnetic dipole); print out stepwise particle data to zgoubi.plt (this is what "IL=2" stands for!); integration step size XPAS=2.5 cm;
- OPTIONS[CONSTY ON, WRITE OFF] stands for keyword OPTIONS (gives access to various options), and two options retained here, (i) CONSTY (maintain constant transverse coordinates during stepwise integration through optical elements), switched ON; (ii) switch off most print outs to zgoubi.res.

- INCLUDE[NBF=N,FNAME=fileName,LBL_1A=from_A,LBL_1B=to_B] inserts locally, N times, a piece of a sequence copied from 'fileName' file, comprised between LABEL1-type MARKERS 'from_A' and 'to_B'.

Coordinate nomenclature

In the theoretical reminders, *i.e.* Sect. 3 in the following chapters, conventional notations are used for particle coordinates, namely,

$$\underbrace{\overbrace{x,\ x'}^{\text{radial}},\ \overbrace{y,\ y'}^{\text{axial}}}_{\text{transverse coordinates}}\ ,\ \underbrace{\delta s,\ \delta p/p}_{\text{longitudinal}}$$

with δp the momentum offset and δs the distance to a reference particle. These coordinates are defined in the Serret-Frénet frame, or moving frame, Fig. 1.2.

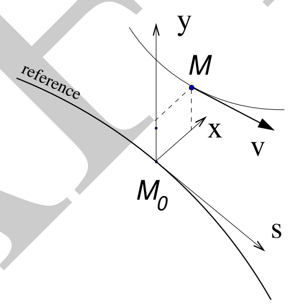

Fig. 1.2 Moving frame $(M_0; s, x, y)$ along a reference line. M_0, at path distance s from some origin, is the projection of particle location $M(x, y, s)$ on the reference

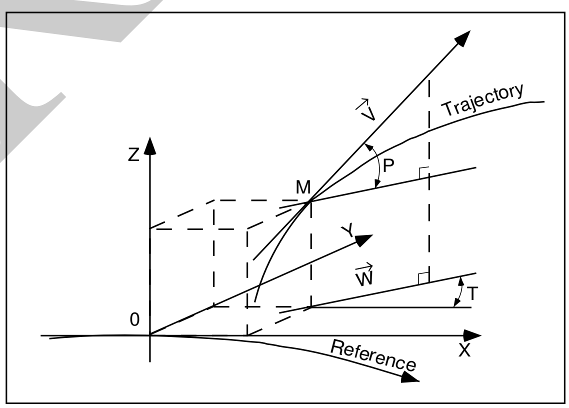

Fig. 1.3 Coordinates Y, T, Z, P in zgoubi [1, Sect. 1.1]. Reference curve: a straight axis in optical elements defined in a Cartesian frame; an arc of a circle in those defined in a cylindrical frame. OX: in the direction of motion, tangent to the reference; OY: normal to OX; OZ: orthogonal to the (X, Y) plane; **W**: projection of the velocity, **v**, in the (X, Y) plane; T: angle between **W** and the X-axis; P: angle between **W** and **v**

In the exercises instead, zgoubi coordinates are used, namely

$$\underbrace{\overbrace{Y,\ T}^{\text{radial}},\ \overbrace{Z,\ P}^{\text{axial}}}_{\text{transverse coordinates}}\ ,\quad \underbrace{S, D}_{\text{longitudinal}}$$

The transverse coordinates are explicited in Fig. 1.3. S is the path length, D is the relative rigidity of the particle, relative to a reference rigidity specified as part of the initial object definition in `zgoubi` input data file. As a matter of fact, an initial object, *i.e.* the set of initial coordinates of particles to be raytraced, and possibly their spins, always has to be defined, for `zgoubi` to solve the differential equations of particle and spin motion.

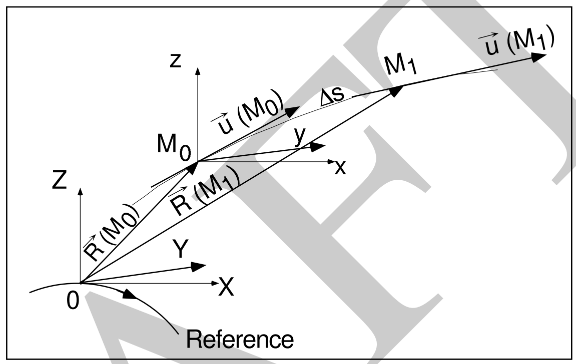

Fig. 1.4 Position vector **R** and normalized velocity vector ($\mathbf{u} = \mathbf{v}/v$) of a particle in `zgoubi` frame. A Δs push takes the particle from position M_0 to position M_1

An important additional parameter is the integration step. Figure 1.4 displays the position and velocity vectors of a particle in `zgoubi` frame, and a Δs push from position M_0 to position M_1. That push is performed using a Taylor expansion in Δs [1, Sect. 1.2]. The integration step size is one of the available controls on the accuracy of the integrator, when applied to the Lorentz force equation, or to the Thomas-BMT spin equation. It also controls the accuracy of the simulation of events, such as photon emission, in-flight decay, etc.

Conventional and `zgoubi` coordinate notations may sometimes be used concurrently, for instance when equations from the main text are referred to, or resorted to, in the exercises. This is presumably in contexts exempt of ambiguity.

Reference frames of optical elements

Optical elements in `zgoubi` define fields in a Cartesian reference frame: this is the case for instance for MULTIPOL, BEND, EBMULT; or in a cylindrical reference frame: case of *e.g.*, DIPOLE, ELCYLDEF. And similarly for field map handling keywords: CARTEMES, TOSCA[MOD≤19], BREVOL use a Cartesian meshing, whereas POLARMES, TOSCA[MOD≥20] use polar or cylindrical meshing. Referring to Fig. 1.5: let a particle location M(X,Y,Z) project at m(X,Y) (the dashed

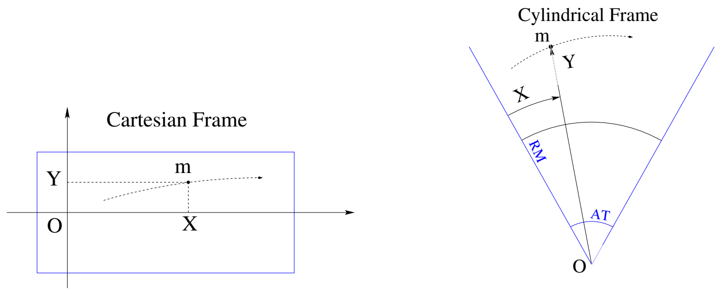

Fig. 1.5 Cartesian and cylindrical reference frames in optical elements

curve figures the projected trajectory). In the case of an optical element (figured as a rectangular box) defined in Cartesian coordinates, X and Y in zgoubi.plt (columns respectively 22 and 10 [1, Sect. 8.3]) denote the coordinates taken along the fixed reference frame axes. In the case of an optical element (figured as an angular sector AT with some reference radius RM) defined in a cylindrical coordinate frame (Y, X, Z), Y is the radius, X is the polar angle, counted positive clockwise, Z is the vertical coordinate (column 12 [1, Sect. 8.3]).

1.3 Graphics, Data Treatment: `zpop`, gnuplot, awk, python

An execution of a beam optics problem in `zgoubi` produces a listing, zgoubi.res, always. However, when running a problem the user often requests logging of execution data in zgoubi.fai (produced by FAISTORE[FNAME=zgoubi.fai, or else]) and/or zgoubi.plt (produced as a result of IL=2 flag, *e.g.* as in DIPOLE[IL=2]).

The output file zgoubi.fai is a record of more than 50 particle data (coordinates, spin, etc.) [1, Sect. 8.2], at the location(s) where the keyword is inserted in the optical sequence.

The output file zgoubi.plt is a record of more than 50 particle data, step-by-step (coordinates, fields, step sie, etc,) [1, Sect. 8.3] while integration proceeds through an optical element.

Beyond, a PRINT command available in several keywords allows specific printouts during raytracing. For instance, CAVITE[PRINT] will cause particle acceleration data to be logged in zgoubi.CAVITE.Out, which can then be accessed from gnuplot scripts, to produce graphs, data treatment, or provide debugging help. In the same line, one would get zgoubi.HISTO.out from HISTO[PRINT], zgoubi.OPTICS.out from OPTICS[PRINT], zgoubi.PICKUP.out from PICKUPS[PRINT], zgoubi.SPNPRT.Out from SPNPRT[PRINT], etc. [1, Sect. 8].

`Zpop` [12], an old companion postprocessor of `zgoubi`'s, allows handling zgoubi.fai ad zgoubi.plt. It also allows brute reading of and plotting from any of the other files mentioned above. `Zpop` is part of the sourceforge package, portable on any linux and Mac OS. Quick to launch (in an xterm window), quick to operate. After years of development and utilization `zpop` allows all sorts of graphs, and various post-processing, reading particle coordinates and other data from zgoubi.plt or zgoubi.fai records.

`Zpop` menu 7, for instance allows plotting any variable entering the process of pushing particles step by step and element by element, against any other. There are of the order of 60 of them: particle coordinates, **E** and **B** field components, spin components, RF phase, step size, optical eleemnt number, turn number, etc., as well as derivatives or combinations [1, PART D, Sect. 1.3]. By experience, menu 7 answers most of the needs of lattice studies and beam dynamics simulations.

`Zpop` menu 8, allows further treatment of data read from these output files from a run, for instance drawing of synoptics with trajectories superimposed, Fourier analysis of periodic motion, matching of Enge's fringe field coefficients, etc.

Although this book is not a guide to the use of `zpop`, graphs found in the solutions of simulation exercises (Chap. 15) often use the latter.

When they are not produced using `zpop`, data analyzis and graphic in the solutions use gnuplot, an incredibly simple yet powerful tool, even more so when added awk. By experience, gnuplot is quite suited as a graphic interface to `zgoubi` output data files, awk adds a powerful data analyzis and treatment tool, both combined answer about any needs.

There is more, about python, following section.

1.4 Interface to `Zgoubi`?

`Zgoubi` can be run without an interfacing software, there is no need for that. Again, all that is needed is (i) an input data file, zgoubi.dat, which starts with the definition of initial coordinates, followed by a linear description of the optical sequence to be raytraced through, and with a few commands sprinkled around, and (ii) the following command:

[pathTo]/zgoubi

which is the address of the executable. Execution results are logged in output files, of which zgoubi.res *a minima*. Whatever is needed to handle the code is found in Zgoubi Users' Guide, which is part of the sourceforge package [1].

python [13]

A `Zgoubi` user quick startup has been written by beginners a few years ago [14]. This startup introduces to pyZgoubi, a python based interface to zgoubi developed by Sam Tygier, which has its own web site [15] and at present maintained at RAL and BNL.

is another python interface, developed by a group from Brussels unuversity, available on internet as well [16].

Not strictly speaking python, but based on anyway, Sirepo accelerator simulation package by Radiasoft company also offers an interface to `zgoubi` [17].

References

1. Méot, F.: Zgoubi Users' Guide. https://www.osti.gov/biblio/1062013-zgoubi-users-guide The latest version of the guide, on Sourceforge: https://sourceforge.net/p/zgoubi/code/HEAD/tree/trunk/guide/Zgoubi.pdf
2. Kowalski, S. and Enge, H.A.: RAYTRACE. Laboratory For Nuclear Science, MIT, Cambridge, MA, USA, May 16, 1986. http://aea.web.psi.ch/Urs_Rohrer/MyFtp/RAYTRACE/raytrace1.pdf
3. Agostinelli, S., et al.: GEANT4 - A simulation toolkit. NIM A, Volume 506, Issue 3, 1 July 2003, Pages 250-303 https://geant4.web.cern.ch/
4. Thirion, J. et Birien, P.: LE SPECTROMETRE II. Rapport Interne DPhN/ME, CEA Saclay, 23 Dec. 1975
5. Catz, H.: LE SPECTROMETRE SPES II. Rapport Interne DPhN/ME, CEA Saclay, 1980
6. Paul Birien, Saby Valéro: Projet de spectromìre magnétique à haute résolution pour ions lourds. Note CEA-N-2215, CEA-Saclay, Mai 1981.
7. A. Tkatchenko et F. Méot: Calculs optiques pour le spectromètre à kaons de GSI. Rapport Interne CEA/LNS/GT/88-07, Saclay (1988)
8. Zgoubi downloads on sourceforge: https://sourceforge.net/projects/zgoubi/files/stats/map?dates=2007-09-01%20to%202023-01-01&period=monthly
9. Méot, F. and Valéro, S.: *Manuel d'utilisation de Zgoubi*, Rapport IRF/LNS/88-13, CEA Saclay, 1988.
10. Méot, F. and Valéro, S., in collaboration with Doornbos, J. and Stewart, P.: *Zgoubi users' guide*, Int. Rep. CEA/DSM/LNS/GT/90-05, CEA Saclay (1990) & TRIUMF report TRI/CD/90-02 (1990).
11. Méot, F.: A numerical method for combined spin tracking and raytracing of charged particles. NIM A313 (1992) 492, and proc. EPAC (1992) p.747.
12. Zgoubi's data treatment software `zpop` comes, and compiles independently, as part of the `zgoubi` package [8], when downloaded from sourceforge. It is available, including source files, at https://sourceforge.net/p/zgoubi/code/HEAD/tree/trunk/zpop/
13. https://www.python.org/
14. Pressman, Annette, and Hock, Kai: Zgoubi. A startup guide for the complete beginner. May 21, 2014. https://sourceforge.net/p/zgoubi/code/HEAD/tree/trunk/guide/aGuide4Beginner/arxiv.org_abs_1405.4921.pdf
15. Tygier, S., Kelliher, D. developers: pyZgoubi. https://github.com/pyZgoubi/pyZgoubi
16. Hernalsteens, C., Tesse, R., and Vanwelde, M.: Zgoubidoo. https://ulb-metronu.github.io/zgoubidoo/
17. https://www.sirepo.com/en/apps/particle-accelerators/