Nuclear Instruments and Methods in Physics Research A 775 (2015) 15–26

Contents lists available at ScienceDirect

Nuclear Instruments and Methods in Physics Research A

journal homepage: www.elsevier.com/locate/nima

The PyZgoubi framework and the simulation of dynamic aperture in fixed-field alternating-gradient accelerators

S. Tygier [a,*], R.B. Appleby [a], J.M. Garland [a], K. Hock [b], H. Owen [a], D.J. Kelliher [c], S.L. Sheehy [c]

[a] Cockcroft Accelerator Group, The University of Manchester, UK
[b] University of Liverpool, UK
[c] STFC Rutherford Appleton Laboratory, UK

ARTICLE INFO

Article history:
Received 8 August 2014
Received in revised form
11 November 2014
Accepted 18 November 2014
Available online 26 November 2014

Keywords:
Accelerator physics
FFAG
Dynamic aperture

ABSTRACT

We present PyZgoubi, a framework that has been developed based on the tracking engine Zgoubi to model, optimise and visualise the dynamics in particle accelerators, especially fixed-field alternating-gradient (FFAG) accelerators. We show that PyZgoubi abstracts Zgoubi by wrapping it in an easy-to-use Python framework in order to allow simple construction, parameterisation, visualisation and optimisation of FFAG accelerator lattices. Its object oriented design gives it the flexibility and extensibility required for current novel FFAG design. We apply PyZgoubi to two example FFAGs; this includes determining the dynamic aperture of the PAMELA medical FFAG in the presence of magnet misalignments, and illustrating how PyZgoubi may be used to optimise FFAGs. We also discuss a robust definition of dynamic aperture in an FFAG and show its implementation in PyZgoubi.

1. Introduction

The fixed-field alternating-gradient (FFAG) accelerator, conceived independently by Ohkawa, Kolomensky and Symon in the early 1950s [1–4], is a ring particle accelerator which differs from a cyclotron in that the field gradient alternates in sign during a particle orbit, giving strong focusing in both planes of transverse motion. Similar to cyclotrons, FFAGs allow high-intensity beams to be rapidly accelerated. FFAGs may be advantageous in several applications: accelerating unstable particles such as muons, where the FFAG offers both a much smaller acceleration time and a larger beam acceptance than a synchrotron [5,6]; accelerating intense proton beams to c GeV energies for applications such as spallation neutron production or to drive accelerator-driven subcritical reactors, where cyclotron designs have been found to be difficult [7,8]; production of either protons or light ions for hadron therapy, where the high (c kHz) extraction rate may allow better pulse-by-pulse variation of the delivered energy to a patient [9–12].

Until recently all FFAGs were of the scaling type [13] in which the transverse focusing is kept constant during the acceleration cycle; scaling optics are achieved by using an appropriate variation of magnetic field with radius and hence with energy, since the particles move significantly within the accelerator magnets during acceleration. The close-to-constant number of transverse betatron oscillations per turn (called the ring tune) allows for stable motion of the particles and in particular the avoidance of destructive resonance effects; several test proton FFAGs have been demonstrated in recent years [14]. More recently, non-scaling FFAG designs have been proposed which dispense with the scaling condition to obtain both smaller apertures within the magnets and a much less restrictive magnetic field profile [15]. Such non-scaling designs typically require that several resonances are traversed by a bunch as it accelerates. The smaller and simpler magnet designs have led to several proposals for their use which include the PAMELA (Proton Accelerator for MEdicaL Applications) design study for particle therapy [16]. The first demonstration of a non-scaling FFAG was recently made using the EMMA (Electron Machine with Many Applications) experiment [17–19].

The fact that a particle's radial orbit changes significantly within the accelerator magnets means that their simulation differs from that required in many other accelerator design codes. For example, codes for the simulation of synchrotrons (for example MAD [20,21] or Elegant [22]) make paraxial or other approximations that assume that particles stay close to some reference trajectory. In contrast, particle motion in FFAGs involves both large particle amplitudes and large energy changes; transverse particle amplitudes in modern synchrotrons are typically ~1 mm or less whereas FFAG designs may have particle motion up to ~1 m and the radial orbit shift can be of the order tens of centimetres. Likewise, some FFAG designs (for example those for muon acceleration) aim at accommodating

* Corresponding author. Tel.: +44 161 2754271.
E-mail addresses: sam.tygier@hep.manchester.ac.uk (S. Tygier), robert.appleby@manchester.ac.uk (R.B. Appleby).

http://dx.doi.org/10.1016/j.nima.2014.11.067

energy changes of a factor of two or even more, much larger than the ~1% energy changes typically modelled in synchrotron designs. A suitable code to describe particle motion in an FFAG therefore needs to correctly implement such features as high-order magnetic multipoles, the effect of (perhaps curved) pole face edge angles, and to realistically model the fringe fields at the magnet ends. In particular it is useful to be able to model realistic fields interpolated from magnetostatic modelling data.

Zgoubi is an established particle tracking code originally developed for studying spectrometers [23,24], and which satisfies the particle tracking requirements for a wide range of accelerator simulations including that of FFAGs. Its ray-tracing method is suitable for tracking particles over a wide range of energies within complex magnets, and so it has become popular for studies of FFAGs. Due to the wide range of novel FFAG designs currently being considered for various applications and the large parameter spaces of those designs, a flexible design framework which adds suitable abstraction and functionality to an underlying tracking code is desirable.

In this work we have developed a framework around Zgoubi called PyZgoubi, which gives important capabilities for FFAG design and optimisation as well as functionality which simplifies layout visualisation and data analyses. In Section 2 we introduce the PyZgoubi framework, including its capabilities, structure and input format. The core of PyZgoubi is a set of Python classes that wrap the elements and routines of Zgoubi; this abstraction of the underlying Zgoubi engine enables complex studies to be performed and enables the use of external numerical libraries. For example, we take advantage of well-understood downhill simplex methods in Python to optimise a range of lattice parameters to obtain a desired working point. In Section 3 we describe in detail an algorithm developed in PyZgoubi for defining dynamic aperture (DA) in an FFAG and show its suitability for studying stability in FFAGs. In Section 4 we apply some of the routines we have developed in PyZgoubi to a comprehensive case study of the PAMELA FFAG lattice [16], including tune optimisation, DA studies and the effect of magnet misalignments on the optics and DA.

2. The PyZgoubi framework

2.1. Structure of PyZgoubi

Zgoubi is the most widely used code for simulating FFAGs [25,26] and has been used for several designs, such as EMMA [19,27], PAMELA [16] and RACCAM [28]. It also has applications outside FFAGs: for example it has been used for spin tracking in the AGS Synchrotron [29]. The motion of particles in Zgoubi is determined by direct integration of the Lorentz equation of motion using a Taylor expansion of the local electromagnetic field and its derivatives up to sixth order. The fields are evaluated at the particle location at every step through a magnet, so that dynamics at large amplitudes are correctly modelled. This ray-tracing method is described in detail in the Zgoubi Manual [30].

In common with other accelerator codes, the magnetic and electric fields generated by each accelerator component in a particular design may be obtained either analytically from one of a predefined set of magnetic or accelerating element definitions, or by interpolation of a user-provided field map. In both cases elements may be used repeatedly and placed with respect to a reference trajectory, and the field descriptions may be truncated into a variety of shapes with associated field fall-offs. The analytic magnet model elements include standard accelerator multipoles, with both rectangular and sector shape options. Magnet faces can have additional wedge angles, straight or curved faces, and may include fringe fields.

Zgoubi is controlled using a keyword-based text input file to define the lattice of accelerator elements and which, when run, produces a number of output files. Keywords include both magnetic and other beamline elements, for example `DIPOLE` and `QUADRUPO` for a magnetic dipole or quadrupole, and `CAVITE` for a radio-frequency accelerating cavity. The keyword `REBELOTE` can be used to repeat beamline sections, and can also be used for example to simulate multi-turn injection or to repeat a line with parameters adjusted. The particle generator `OBJET` defines a bunch of particles, where for example `PARTICUL` defines the parameters of a particle (charge and species, etc.). When tracking particles, control commands may be used such as `MATRIX` to calculate a transfer matrix, or `TWISS` to calculate the optical properties of the lattice. However, the format of the results output from using functions such as these can be complicated and difficult to use without post-processing. Fig. 1 shows an example of the input script to make a simple FODO cell and find the transfer matrix for a single energy. It should be noted that all numbers after a given key-word such as `QUADRUPO` must be entered in the correct order so that Zgoubi may interpret them correctly. Fig. 2 shows the results that are printed to the screen after running the script in Fig. 1 in Zgoubi; the transfer matrix for the single energy defined in Fig. 1 is shown. Zgoubi also has some ability to optimise a lattice design, for example the `FIT` routine can be used to adjust certain beta function values within a set of magnets; however we will show that PyZgoubi has a much larger scope for lattice optimisation where a larger range of parameters may be optimised.

PyZgoubi [31] is an accelerator design framework built around Zgoubi and its underlying tracking engine. It simplifies the use of Zgoubi, streamlines the input and output process and adds many high-level tools and capabilities. The use of a higher-level framework abstracts away from the tracking and enables the construction of sophisticated analyses, visualisation and optimisation of accelerator lattices. The core of PyZgoubi is a Python library that provides classes for each element available in Zgoubi. These classes can be combined together to form a line object, representing a beamline, upon which PyZgoubi operations can be performed. PyZgoubi also contains a utility module with a set of routines for common tasks such as finding closed orbits and computing lattice functions.

In particular, PyZgoubi simplifies the use of Zgoubi by allowing the creation of simpler input files. For example in Zgoubi one must explicitly write every part of the lattice and particle beam, including record points, and control elements in a contiguous text file where the user must know the order of each parameter after a keyword. In PyZgoubi, the user only needs to specify the magnetic lattice using the geometry and magnetic field parameters, etc.; the required particle generators, recording options and Zgoubi input file structure are created automatically. Hence, PyZgoubi requires significantly less code to create a lattice and to find its properties when compared to Zgoubi. This becomes advantageous when, for example, the user wishes to iterate over a given lattice many times under different conditions. For example, scanning the properties of a given lattice over the energy range requires the user to define only the lattice definition and the energy range in PyZgoubi; the lattice is created automatically for each instance of the particle energy to be tested, and the output can be grouped as the user defines, for example the beta function at a given location as a function of energy.

PyZgoubi can also be used merely as an interface to the built-in commands of Zgoubi, so no features of Zgoubi are restricted by use of PyZgoubi. In PyZgoubi the user still has the advantage of programming constructs (e.g. maths and conditionals) and the ability to load output files into arrays for analysis. Interfacing to new elements in Zgoubi requires only a simple text definition to be given, as described in the manual.

```
********************************************************************************
test_line
'OBJET'
458.1553616525
0.001     0.01     0.001       0.01        0.001       0.001
0.0       0.0      0.0   0.0   0.0   1.0
'PARTICUL'
938.27203   1.602176487e-19   1.7928      0.0   0
'FAISCNL'
zgoubi.fai
'QUADRUPO'         q12
20.0      20.0     5.0
0.0       0.0
0         0.0      0.0   0.0   0.0   0.0   0.0
0.0       0.0
0         0.0      0.0   0.0   0.0   0.0   0.0
1.0
1         0.0      0.0   0.0
'DRIFT'
50.0
'QUADRUPO'         q22
20.0      20.0     -5.0
0.0       0.0
0         0.0      0.0   0.0   0.0   0.0   0.0
0.0       0.0
0         0.0      0.0   0.0   0.0   0.0   0.0
1.0
1         0.0      0.0   0.0
'DRIFT'
50.0
'FAISCNL'   end
zgoubi.fai
'MATRIX'
1         11
'END'
********************************************************************************
```

Fig. 1. An example Zgoubi input file, showing the definition of elements and associated parameters for a basic FODO cell. For example, on the 3rd line '458.155' specifies the reference rigidity of the beam in kG cm, this is the fundamental variable for momentum used throughout the tracking. On the 3rd to last line, `MATRIX` specifies the desired type of output. The user must pay attention to the order of the numbers entered after each keyword, as they are read using a FORTRAN formatted read statement, for example `PARTICUL` (the particle species keyword) where the 5 proceeding numbers (separated by white space or a comma) represent the particle mass in MeV/c^2, the particle charge in coulombs, the gyromagnetic factor, the centre-of-mass lifetime and an unused variable. The set of numbers following the keyword `QUADRUPO` specifies the quadrupole parameters and is explained in reference [24].

```
********************************************************************************
     11  Keyword, label(s) :  MATRIX

  Reference, before change of frame (part #    1)   :
   0.0    0.0    0.0    0.0    0.0    1.40000000E+02    3.22408733E-02

           Frame for MATRIX calculation moved by :
            XC =     0.0 cm , YC = 0.0 cm ,    A =  0.0   deg  ( = 0.0rad )

  Reference, after change of frame (part #    1)   :
   0.0    0.0    0.0    0.0    0.0    1.40000000E+02    3.22408733E-02

  Reference particle (#  1), path length : 140.0 cm  relative momentum :1.0


                  TRANSFER  MATRIX  ORDRE  1  (MKSA units)

    -0.187558     1.75305     0.0          0.0          0.0          0.0
    -0.753953     1.71532     0.0          0.0          0.0          0.0
     0.0          0.0         1.33834      0.990104     0.0          0.0
     0.0          0.0        -0.753954     0.189418     0.0          0.0
     0.0          0.0         0.0          0.0          1.0          0.0
     0.0          0.0         0.0          0.0          0.0          1.0

          DetY-1 =      -0.0000004962,    DetZ-1 =      -0.0000004963

          R12=0 at   -1.022     m,         R34=0 at   -5.227     m

      First order symplectic conditions (expected values = 0) :
        -4.9619E-07   -4.9627E-07   0.0     0.0    0.0     0.0

                TWISS  parameters,  periodicity  of   1  is  assumed
                                    - UNCOUPLED -

  Beam matrix (beta/-alpha/-alpha/gamma) and periodic dispersion (MKSA units)

       2.716413   1.474284   0.0         0.0         0.0         0.0
       1.474284   1.168273   0.0         0.0         0.0         0.0
       0.0        0.0        1.534197  -0.890148    0.0         0.0
       0.0        0.0       -0.890148   1.168274    0.0        -0.0
       0.0        0.0        0.0         0.0         0.0         0.0
       0.0        0.0        0.0         0.0         0.0         0.0

                                   Betatron  tunes
                    NU_Y =  0.11164565          NU_Z =  0.11164565
********************************************************************************
```

Fig. 2. An example of the output generated by running the input file in Fig. 1 where the keyword `MATRIX` is used to generate the transfer matrix through the cell or lattice. The transfer matrix for a periodic cell is shown under the heading 'TRANSFER MATRIX ORDRE 1', and the optics matrix expressed in terms of the optics parameters α, β and γ is shown under the heading 'Beam matrix and periodic dispersion'.

2.2. PyZgoubi input, execution and output

PyZgoubi is written in portable Python, using external libraries such as SciPy [32]; the code's structure is shown in Fig. 3. PyZgoubi is free and open source, it is available to download from its webpage [31] and is installed using the standard Python Distutils method. The .tar.gz installer can be downloaded from the website [31] and installed with the command `./setup.py install`. Zgoubi can be installed independently, or automatically using the command `pyzgoubi -install-zgoubi`. PyZgoubi is compatible with Zgoubi 5.1, as well as the current development version from SourceForge, and automatically recognises the changes in output formats between versions. Some PyZgoubi features require features in the development version. A settings file is created at `~/pyzgoubi/settings.ini` which can be used to customise some behaviours of PyZgoubi.

To run PyZgoubi the user creates an input script, written within standard Python syntax. The simplest script is to define a set of magnets, define a line containing those magnets and then call a utility function to calculate properties of the line. To run a script, for example FODO.py, one uses the command `pyzgoubi FODO.py`. This simple script example is shown in Fig. 4. In order to represent an accelerator the user defines a set objects which each represent accelerator components such as magnets, and a `Line` object to contain them in the specified order. A `Line` object can contain repeated elements or even other lines in order to simplify definition of periodic or repeating structures. This `Line` object can then be used as an argument in a function which calculates properties of the cell and returns a results object which can easily be accessed and analysed in Python.

When the PyZgoubi run method is called, a Zgoubi input file is automatically created in the correct format and Zgoubi is run. Zgoubi can be run many times from a single PyZgoubi script for sophisticated analyses and applications. The content of Zgoubi's output files can then be accessed through a results object. This in itself provides a convenient interface to Zgoubi that allows the full features of Python, such as calculations, looping and reading or writing files, which ease lattice creation and data analysis.

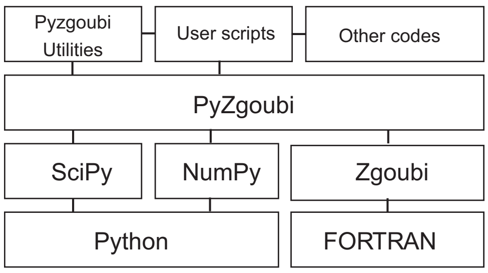

Fig. 3. Relation of PyZgoubi to Zgoubi and the user scripts. PyZgoubi is built on Python, which utilises scientific libraries such as SciPy, and the underlying engine Zgoubi. The user can build scripts on top of PyZgoubi to take advantage of the utilities and other external codes.

PyZgoubi adds much flexibility to Zgoubi. The programmability of the input scripts allows the users to easily extend the built in abilities, without having to modify Zgoubi or PyZgoubi. Other Python libraries can also be included to add additional features, for example custom plotting, data analysis or interfacing. PyZgoubi allowed the implementation of an online modelling system which interfaced to an accelerator controls system [27]. It can also be used to add new physics modelling to a Zgoubi model, such as space charge [33].

PyZgoubi adds some additional overhead in execution time and memory use compared to just using Zgoubi; this is due to using the Python language which is interpreted at runtime, having to read and write files to communicate with Zgoubi and in some cases having to run the Zgoubi program multiple times from with a single PyZgoubi script. The time overhead is typically small except for Zgoubi simulations that themselves only have short run times, for example if only a small number of particles are tracked through a simple accelerator lattice.

To minimise communication time PyZgoubi can use a RAM disk (e.g. `/dev/shm` or `/tmp` on Linux) for files and can read and write Zgoubi's binary file formats.

As a comparison of run times, an iterative closed orbit search for a PAMELA cell was performed both with PyZgoubi (using the `find_closed_orbit` function) and with Zgoubi (using the `FIT2` keyword). The same starting point was used (an initial particle offset about 1.2 cm from the closed orbit to be found), and tolerances in each method (PyZgoubi and Zgoubi) were set to obtain a closed orbit to less than 0.1 μm. PyZgoubi was found to take around twice as long as Zgoubi, but both methods obtain an accurate closed orbit in well under a second; over 5 runs Zgoubi averaged 142.2 ms (best 138 ms) whilst PyZgoubi averaged 278.4 ms (best 275 ms).

PyZgoubi also offers a `find_closed_orbit_range` function which allows a search range to be given for cases where an approximate closed orbit is not known. This cannot be done with the `FIT` routine in Zgoubi. PyZgoubi also allows multiple stages of optimisation within a single script, where the closed orbit can be re-found at each iteration; this can be useful, for example, when optimising an accelerator geometry. However, the `FIT` routine can be called from PyZgoubi if its performance is required.

When tracking large numbers of particles PyZgoubi offers a ‘bunch’ module which allows transfer of particles as binary rather than ASCII data to improve performance and memory usage; it also allows the splitting of a set of particles over multiple Zgoubi run instances to make use of multiple CPU cores if they are available, giving shorter run times than using a single instance of Zgoubi. In the case of DA studies – where a particle is typically

```
********************************************************************************
# Define magnets
d1 = DRIFT(XL=50)
q1 = QUADRUPO("q1", XL=20, B_0=5, R_0=20, XPAS=1, KPOS=1)
q2 = QUADRUPO("q2", XL=20, B_0=-5, R_0=20, XPAS=1, KPOS=1)

# Define line
fodo = Line("Fodo")
fodo.add(q1, d1, q2, d1)

# Calculate properties
min_ke, max_ke = 10e6, 20e6
orbits = gcp.get_cell_properties(fodo, min_ke, max_ke, 11, particle='p')

# Print a table of energies and tunes to screen and to a file
print gcp.cell_properties_table(orbits, ['KE', 'NU_Y', 'NU_Z'])
of = open("output.dat", "w")
print >>of, gcp.cell_properties_table(orbits, ['KE', 'NU_Y', 'NU_Z'])

# Create a set of standard plots
gcp.plot_cell_properties(orbits, out_name="fodo")

# Plot lattice and closed orbits
gcp.plot_cell_tracks(fodo, orbits, 'p', "fodo.pdf")
# Plot lattice with fields
gcp.plot_cell_field(fodo, -20, 20, 50, out_file="fodo_mag.pdf")
********************************************************************************
```

Fig. 4. An example PyZgoubi script, showing the definition of elements and the extraction of cell properties for a basic FODO cell.

tracked for many turns – the overhead from PyZgoubi is small compared to the large time spent tracking within Zgoubi.

2.3. Lattice definition, characterisation and optimisation

PyZgoubi has many methods and functions that may define, manipulate, study and optimise an accelerator lattice. Lattice elements such as dipoles and quadrupoles are defined as objects (with default arguments for many parameters) which may be joined into a beamline object, `Line`. These objects, representing entire rings or single cells, can be passed as function arguments and fully parameterised in terms of their lattice properties for subsequent optimisation operations.

One of the core functions of PyZgoubi is the `get_cell_properties()` function in the get cell properties (gcp) module. This function takes a lattice cell, range of energies and a particle type and finds the single particle closed orbits at each energy by tracking a particle over multiple turns at each energy, taking the mean position in transverse phase space and iterating until the cell-to-cell oscillation is minimised below some defined tolerance. The function `get_cell_properties()` then finds parameters such as the transfer matrices, betatron tunes, time of flight, and path length around the closed orbit, calculated at each energy. The majority of the optimisation and visualisation functions in PyZgoubi require an initial call to `get_cell_properties()`. The functions `cell_properties_table()` and `plot_cell_properties()` can then be used to output the parameters as text or plots. An example of the use of the functions `get_cell_properties()` and `plot_cell_properties()` is shown in Fig. 4. Lattice plots (visualisation of the beam dynamics laid over the geometrical layout of magnets) can be created with the `plot_cell_tracks()` and `plot_cell_field()` functions. Magnetic fields may be obtained by tracking test particles with a very high beam rigidity ($B\rho$) though the lattice and then interpolating between the measurement points, where the magnetic field is recorded.

To find a stable lattice that meets desired requirements, PyZgoubi is able to carry out scans of lattice parameters such as the magnet field strengths or their positions, or to vary beam parameters whilst calculating the resultant lattice properties including those related to the beam dynamics. For example, to find the range of possible working points (pairs of ring tunes Q_x and Q_y that have stable focusing properties) in an FFAG like PAMELA, scans may be made of the magnetic field gradient (or index), k, and the ratio of the F to D magnetic field strengths at the reference radius [34]. Key parameters for the PAMELA lattice can be found in Table 1. For each point in this parameter space – i.e. for a given set of magnet strengths – a lattice is created. PyZgoubi can also determine the one-turn transfer matrices M_x and M_y for transverse motion in each plane with respect to each closed orbit, where the occurrence of linear stability of motion in each plane is checked via the relations $|\mathrm{Tr}(M_x)| < 2$ and $|\mathrm{Tr}(M_y)| < 2$.

As previously mentioned, an important abstraction in the PyZgoubi framework is the ability to perform optimisation, which may be used to adjust the lattice to achieve certain goals. This can be used for matching, which is the optimisation task wherein the magnet parameters are changed to obtain specific properties such as (for example) beam sizes at a given location, to obtain more general properties such as minimising the variation of orbit radius with energy, or – as will be discussed below – to maximise the dynamic aperture. As is usual, optimisation proceeds by defining an objective function (penalty function) that quantifies how good a given lattice configuration is compared to some desired goals, weighted according to the importance of each parameter such as the tunes, required magnet apertures, and so forth. Each lattice configuration comprises an input vector of parameters from which a lattice description may be constructed, and from which the penalty function value may then be obtained. Optimisation routines may be then used from the external library SciPy [32] – such as an implementation of the downhill simplex, Powell or basin hopping methods – to minimise the penalty function by optimising the input vector over the configuration space.

Table 1
PAMELA lattice parameters used in simulations.

Kinetic energy (MeV)	30–250
Reference radius r_0 (m)	6.251
RF frequency (MHz) inj., ext.	1.92, 4.56
Rep. rate (kHz)	1
Cell tune ν_x, ν_y	0.72, 0.27
Field index k	38
Max. D field (T)	-4.0
Max. F field (T)	4.25
Orbit excursion (m)	0.176

Compared to the built-in Zgoubi function `FIT`, PyZgoubi offers more flexible use of input parameters and allows a wider range of optimisation criteria. For example, a set of field multipoles could be calculated using an arithmetic expression; the penalty function can be arbitrarily defined, for example it could contain the orbit or tune excursion over a range of energies or expressions to optimise an output to be above or below a value; and there is a wide range of possible minimisation functions. In contrast, to customise the penalty functions available in Zgoubi the source to the `FIT` routine must be edited and recompiled. An example of such an optimisation function in PyZgoubi is given in Fig. 5 where an input vector of parameters is optimised and the output vector is returned.

A particular issue with accelerator optimisation is that small changes in the lattice configuration (e.g. the focusing gradients) can take a lattice from a stable to unstable condition, i.e. such that $|\mathrm{Tr}(M)| < 2$ is no longer true, and thereby lead to a situation where other properties such as the tune or closed orbit become undefined. The penalty function definition must account for this so that the optimisation routine may still find stable configurations; in regions away from the conditions of stable motion, the values of $\mathrm{Tr}(M_x)$ and $\mathrm{Tr}(M_y)$ can be determined in some cases and thereby provide a suitable gradient of the penalty function over the configuration such that the optimisation routine may act. PyZgoubi's flexibility allows different expressions to be used for the penalty function depending on the stability of the lattice.

2.4. Lab frame plotting

Zgoubi tracks particles in the local coordinate systems of the beamline elements. This reference line is usually not a closed orbit. However, it can also be useful to obtain data from the lattice model, in order to view the arrangement of magnets and the tracks of particles, in global lab frame coordinates, in other words Cartesian coordinates.

Zgoubi includes a plotting utility ZPOP [30], which is capable of producing lab frame plots as well as plots of other tracking results, and it also provides some analysis tools. However it is menu driven and therefore difficult to script – for example if multiple plots are required from a single run – and has limited style and output format options. PyZgoubi provides a wider range of plotting options that can be automated into a script. PyZgoubi is able to draw the magnets overlaid with a visualisation of their strength and the resultant trajectories of particle tracks; it takes into account changes to the reference trajectory both due to wedge angles on the magnets and from any `CHANGREF` directives; `CHANGREF` is a pseudo-element in Zgoubi that

```
*******************************************************************************
def func(x):
    b0f, ratio_df, kmag = x
    b0d = -b0f*ratio_df

    # Make a cell using `make_cell' function:
    cell = make_cell(b0f, b0d, kmag)

    # Perform `get cell properties' on the cell
    data = gcp.get_cell_properties(cell, min_ke, max_ke, ke_steps=2,
                                   particle="p")
    # Some parameters to optimise (horizontal injection, extraction
    # orbit and horizontal tune):
    y_inj = data[0]['Y']
    y_ext = data[1]['Y']
    nu_y =  data['NU_Y'].mean()

    # Define the penalty function to minimise:
    pen = 10*(y_inj)**2 + 5*(y_ext-y_inj)**2 + 2*(nu_y-0.31)**2
    return pen

# Final result is minimised using scipy down-hill simplex and
# gives a final optimised vector of parameters:
result = scipy.optimize.fmin(func, [2,0.8,35])
*******************************************************************************
```

Fig. 5. An example of an optimisation function in PyZgoubi using a SciPy down-hill simplex method to optimise a vector of input parameters, `b0f` (the field in the F magnet at the reference radius), `ratio_df` (the ratio of the F/D magnetic fields at the reference radius) and `k` (the magnetic field index).

explicitly defines a change of reference frame. Obtaining a magnet layout in Cartesian coordinates is invaluable for verifying that a particular lattice design is physically realisable (for example, that there are no overlaps of magnets); to produce visualisations PyZgoubi uses the Matplotlib library [35] for graphical output, which allows output to many standard graphics formats as well as providing for on-screen interactive plots. An example of lab-frame plotting of PyZgoubi is shown in Fig. 6, which shows closed orbits through one cell of the 42-cell EMMA lattice.

2.5. *Example of PyZgoubi FFAG analysis*

To demonstrate the capabilities of the PyZgoubi framework we model and optimise the medical accelerator design PAMELA [16]. The PAMELA design proposes two concentric FFAGs, the smaller of which is designed to accelerate protons from a kinetic energy of 30–250 MeV. This FFAG has 12 repeating cells, each comprising an F–D–F triplet of rectangular superconducting magnets. Each magnet is composed of a novel arrangement wherein multiple field components (dipole, quadrupole, etc. up to decapole) are provided each by dedicated superconducting electromagnet coils, whose relative strength is used to obtain a variation of tune with energy that is sufficiently small to avoid damaging resonance conditions during an acceleration cycle [36]. The PAMELA magnets thereby approximately provide a scaling condition [4,13], but where the orbit radius change is still small. Fig. 7 shows the PAMELA cell closed orbit for proton energies between 30 MeV and 250 MeV, as obtained with PyZgoubi and using the configuration defined in Table 1 where the magnets in the model had an Enge-type fringe-field fall off [37,38] which was used to approximate the real fringe field from a magnet.

The stability and orbit excursion of PAMELA depend on the strengths and field index of the magnets as previously discussed. In PAMELA the working point is chosen such that the horizontal tune is above 0.5 per cell due to the relatively strong focussing, which leads to a smaller orbit excursion [11,34]. PyZgoubi was used to scan over a parameter space consisting of the magnet strengths, their gradients and their strength-ratio in order to measure the stability and lattice properties for each configuration. Fig. 8 shows the resulting stability scan for PAMELA with colours used to indicate the tune. A suitable working point can then be selected and used as the initial condition for further optimisation, for example to flatten the tune variation with energy by varying higher-order magnetic multipole terms. More extensive results can be found in reference [16,34].

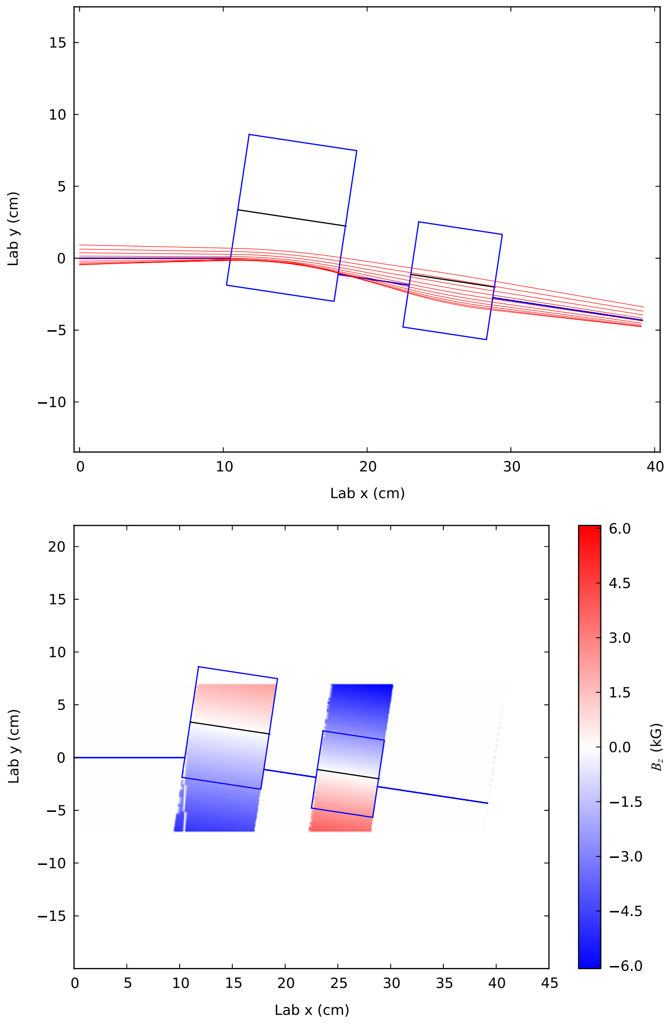

Fig. 6. Closed orbit as a function of energy from 10 MeV to 20 MeV (top) and magnetic fields obtained from PyZgoubi tracking data (bottom) in one cell of the 42-cell EMMA F–D doublet non-scaling FFAG lattice design [17]. The transverse extent of the magnets is illustrative as there are no transverse boundaries in the model, the extent of the field displayed depends on the range of test particles used and can be controlled by the user.

3. Dynamic aperture algorithms for FFAGs.

We now consider the dynamic aperture of FFAGs and present an algorithm, implemented in PyZgoubi, for its calculation. In most

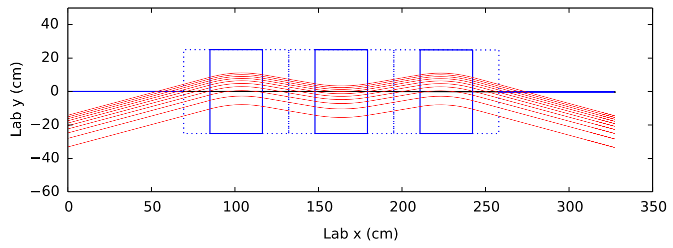

Fig. 7. PAMELA lattice cell showing closed orbits over the proton acceleration range of 30–250 MeV. The solid boxes show the effective field boundaries of the magnets and the dotted boxes show the integration region around the magnets.

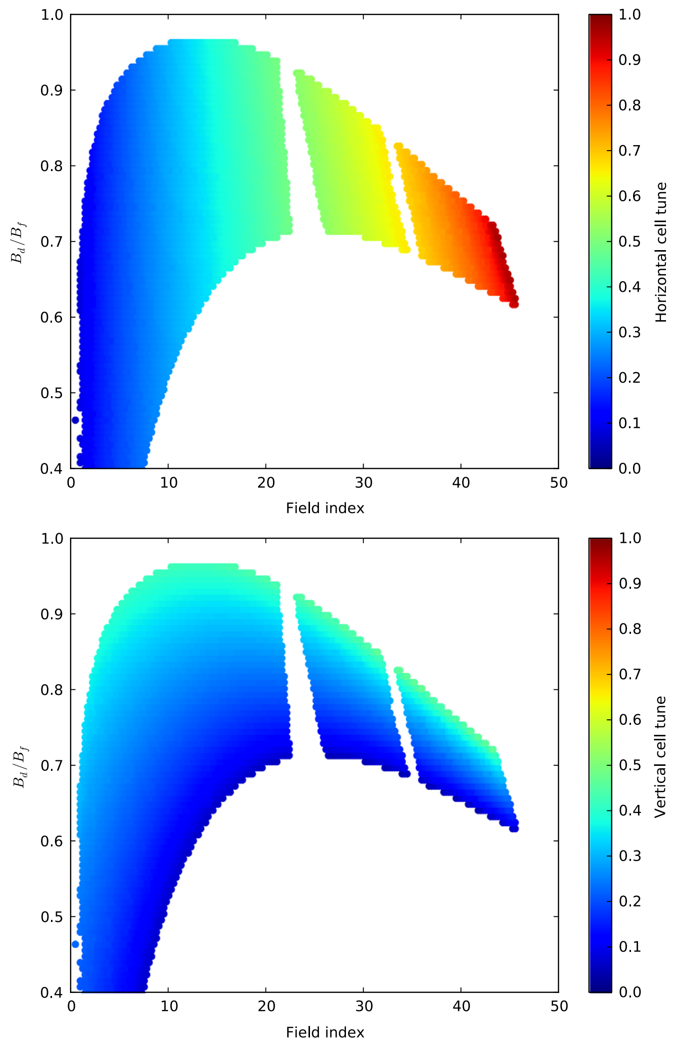

Fig. 8. Stability scan for PAMELA when varying field index k and F–D ratio of the magnetic field strengths, showing the horizontal and vertical tunes of configurations which are stable in both planes. Tunes in each plane are indicated by colour. The gaps are the regions not stable at all energies, around horizontal tunes of $\frac{1}{2}$ and $\frac{2}{3}$. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

simulations of particle accelerators the effective transverse restoring force with respect to the closed orbit at some energy – which arises from the alternating-gradient focusing – is expressed as an expansion about that orbit since typically the magnets are of a discrete nature, i.e. a single magnet predominantly provides a single restoring force term over some range of the particle path s along the design trajectory. This range over s is taken to be the effective magnet length. For example, quadrupoles produce predominantly a linear restoring force for a transverse offset x from the design trajectory, sextupoles provide restoring forces quadratic in x, and so on. However, real magnets produce unwanted higher-order terms in addition to their primary field component, and also induce other focusing and nonlinear effects due to their finite length and their use of entrance and exit magnet faces which may lie at an angle to the plane normal to s.

The presence of nonlinearities in the restoring force can give rise to unstable motion, and over a very large number of turns through the accelerator a particle's motion may become unbounded regardless of its initial amplitude. However, for most practical purposes it is found [39,40] that an approximate boundary exists between stable and unstable motion, and the particle amplitude at which this boundary occurs is termed the dynamic aperture by analogy with the real physical machine aperture.

The period of time over which particles must be followed in simulation to predict the dynamic aperture depends upon the application of that accelerator. For example, in electron storage rings the electron motion is damped due to synchrotron radiation emission in a time equivalent to a few thousand turns; hence the particles need only to be followed in simulation for a similar period, and the resulting estimate of dynamic aperture predicts whether particles will remain bounded for that time. In contrast, hadron storage rings circulate particles for many millions of turns without significant damping, which is both impractical to calculate and whose accuracy suffers from numerical limitations; a smaller number of turns are instead followed, with some assessment as to how that determined dynamic aperture converges to the true case. FFAGs typically accelerate over a limited number of turns; for example, medical accelerator designs such as PAMELA assume a time equivalent to ~ 1000 turns. This limited acceleration time allows a direct estimate of the dynamic aperture over a full acceleration cycle, which we discuss next.

In the algorithm implemented here, we consider an initial point in phase space to be stable if a particle tracked from there is not lost during the number of turns in a full acceleration cycle; simulation over a longer time can be carried out to obtain a more precise determination of whether that particle is truly stable.

The tracking engine Zgoubi models the non-linear dynamics within the ring. PyZgoubi uses several criteria to determine if particle motion has become unbounded: firstly, a check is made whether anomalous motion is occurring, for example if at large amplitudes in the magnets particles are deflected more than 90° from the reference path through each element or if too many particle tracking steps are attempted in a magnet, i.e. the trajectory is not taking it to the exit face of the magnet; secondly, unbounded motion can be detected as an increase in particle amplitude beyond some chosen limit. Observation of both these phenomena in simulation indicates that particular particle is not stable and will be lost in reality. These stopping conditions are inherited from the Zgoubi tracking engine.

The dynamic aperture is a measure of the largest stable particle oscillation in the accelerator. It is more convenient to define the dynamic aperture in terms of the amplitude of the oscillation rather than the physical coordinates of the particle, as the amplitude is independent of location around the ring. This amplitude can also be directly related to emittance requirements for the injection system. We are concerned with the ability of an accelerator to transport an injected beam matched to the linear Courant–Snyder functions at the injection point. Hence we define the amplitude by calculating the linear action-angle variables J_x and its conjugate ϕ_x using the relation

$$x = \sqrt{2\beta_x J_x}\cos\phi_x$$
$$x' = \sqrt{\frac{2J_x}{\beta_x}}(\sin\phi_x + \alpha_x\cos\phi_x). \quad (1)$$

where x and x' are the transverse phase space coordinates and β_x and α_x are the Courant–Snyder parameters. From here we use A_x to represent the amplitude instead of the invariant of motion J_x which is only valid in the linear case. Because we are only concerned with injecting a linearly matched beam, this representation is considered to be valid.

At large amplitudes, the non-linearities in an accelerator give the transverse phase space a non-elliptical shape. Because we use the linear definition of amplitude, it then becomes important to consider in which direction we increase the single particle amplitude in the transverse phase space when searching for the largest stable oscillation. We then want to know the size of the largest matched ellipse at injection, within which all the injected particles are transported for a given number of turns around the accelerator. It is necessary therefore, to track a number of particles at different phase space angles, as a single particle could give an over-estimate depending on the shape of the distorted phase space. Fig. 9 illustrates the horizontal dynamic aperture (zero vertical amplitude) in PAMELA for 1000 turns at the injection energy. If we were to increase the single particle orbit along the positive x-axis alone, the dynamic aperture calculated would be an over-estimate because a matched ellipse of that amplitude would not be stably transported. Whilst it may be theoretically possible to match an injected beam to higher order in order to increase injection efficiency, in practice this is not done.

We sample the phase space angle at four points along x and x' axes in the positive and negative directions from the closed orbit. Additionally, in order to be as close to reality as possible, the vertical coordinates of the particle should be considered simultaneously. Thus the amplitude in the orthogonal plane should also be increased in a similar manner to the horizontal case. Hence there are 16 combinations of single particle coordinates to test for stability. For example, one combination might be to increase the amplitude A_x by increasing the single particle coordinates $(+x, 0)$ in the horizontal direction and increasing A_y by increasing the single particle coordinates along $(-y, 0)$ in the vertical direction, and so on. In order for an amplitude to be considered stable, we require that all 16 combinations are stable at that amplitude.

Additionally, due to non-linearities in an FFAG it is clearly important to consider the coupling between the horizontal and vertical planes, i.e. where there is amplitude in the horizontal and vertical planes. The DA can be parameterised in terms of amplitude in both the horizontal and vertical planes by defining an angle θ_{DA} between them, as shown in Fig. 10 where the left-hand plot shows that when $A_x = A_y$, $\theta_{DA} = 45^\circ$. In this picture, when $\theta_{DA} = 0^\circ$ and $\theta_{DA} = 90^\circ$, these give what is commonly referred to as the horizontal DA and vertical DA respectively. We calculate the DA by taking the magnitude of the vector described by increasing A_x and A_y along a line with angle θ_{DA} as shown in Fig. 10. Hence, for a given θ_{DA} the amplitude $A_{x,y}$ in the respective transverse planes is given by

$$A_x = A_{DA}\cos(\theta_{DA})$$
$$A_y = A_{DA}\sin(\theta_{DA}). \quad (2)$$

It should be noted that when comparing the DA measured using this method to the DA measured using other methods, it may be necessary to include a normalisation factor of $1/\sqrt{2}$, as shown in the central plot of Fig. 10. In order to investigate the effect of coupling in the transverse phase space, the angle θ_{DA} may be scanned over a given range as illustrated in the right-hand plot of Fig. 10. When scanning θ_{DA} through 0–90°, the minimum DA value could be taken as a reasonable assumption of the realistic dynamic aperture for a matched injected ellipse. It should be noted that the DA defined by an angle of precisely 0 or 90° is unrealistic as they contain only horizontal or vertical emittance respectively.

It is common to normalise the DA to the energy using the factor $\beta\gamma$. This is done for all the results presented in this paper.

In PyZgoubi we have implemented the above 16-single-particle method, and for θ_{DA} to be scanned in order to obtain detailed information about the DA using the `get_dynamic_aperture()` routine which is part of the `get_cell_properties()` function set. The user may specify the number of turns over which to measure the DA and the set of angles in θ_{DA}. The user can also specify whether to use just one single particle orbit as opposed to all 16. The DA is acquired by tracking each single particle orbit first at the smallest amplitude (on the closed orbit) and stepping out in user-defined steps until the particle is lost within the turn limit. When this unstable amplitude is found, a smaller step size is then used to bisect between the stable and unstable amplitudes until a defined boundary is reached, within a user-defined resolution. Single particle tracking from a range of amplitudes can then be plotted to visualise the boundary between stable and unstable motion and the deformation of phase space using the `plot_dynamic_aperture()` routine. Note we explicitly search for and reject small islands of instability with a size below some user-defined threshold.

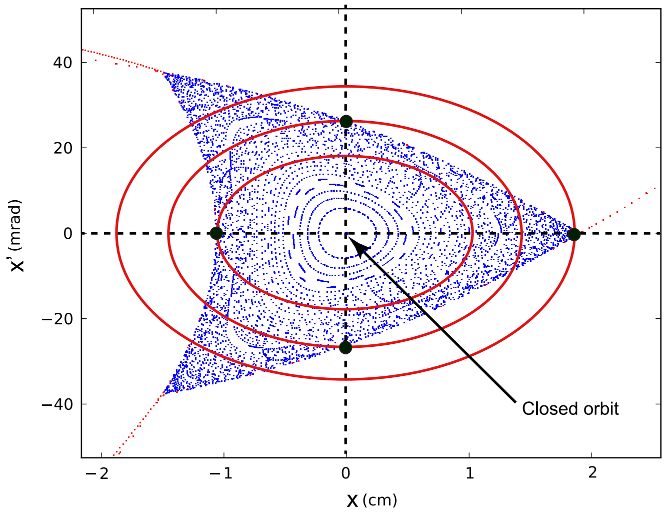

Fig. 9. Horizontal DA in PAMELA acquired by single particle tracking in Zgoubi at the injection energy of 30 MeV. The phase space is distorted by the sextupole non-linearly, giving the triangular shape. Blue points represent single particle tracks which survived 1000 turns in the simulation and red points outside the acceptance region represent the points on orbits which did not survive 1000 turns. Red ellipses illustrate matched elliptical beams at injection for different amplitudes. The black points where the red ellipses intersect the axes represent stable single-particle trajectories at these amplitudes. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

4. Case study: PAMELA

4.1. Tune optimisation

We now apply PyZgoubi to a case study of the PAMELA FFAG, exploiting the flexible framework to optimise the tune properties of the lattice and study the dynamic aperture in the presence of magnetic misalignments.

The basic model of PAMELA was established in Section 2. We begin the case study by using the PyZgoubi optimisation framework to reduce the tune variation over the energy range of the machine. To avoid resonance crossings an FFAG must have a small tune variation with energy, and PAMELA achieves this by setting multipole magnet strengths to approximately follow the FFAG scaling law, but attempts to achieve zero chromaticity (zero tune variation); in practise there is a small non-zero tune variation [11]. PyZgoubi can be used to optimise this solution by adjusting the multipole parameters. This is achieved with a penalty function

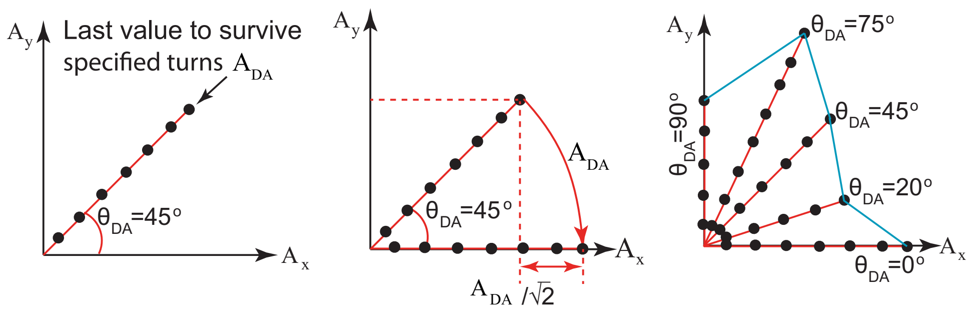

Fig. 10. DA is illustrated by increasing A_x and A_y until the particle is lost within a turn limit, and is represented in these 2D plots by black points which create a vector (joined by a red line) with angle θ_{DA}. The middle plot shows the importance of normalising the DA vector by $1/\sqrt{2}$ when comparing it with other methods, as is often the convention. The right-hand plot illustrates how the DA may vary with θ_{DA}. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

that minimises the tune variation over the machine energy range, while keeping the mean tune constant and following the PyZgoubi optimisation functionality described in Section 2.

The result of this procedure is shown in Fig. 11, which shows the tune variation as a function of energy before and after optimisation and shows that the optimised lattice has a smaller spread in tune than the standard lattice. The required adjustments to the multipole coefficients were up to around 5% for the octupole case and 1% for decapole, and provide flatter tunes or a potentially simplified magnet design for a constant tune variation with energy. This optimised lattice and any other user-defined optimisation are readily obtained with the PyZgoubi framework.

4.2. Dynamic aperture calculation

We calculated the dynamic aperture using the PyZgoubi interface and the methods described in Section 3. The dynamic aperture was scanned as a function of horizontal and vertical tune with the tune values acquired by varying the dipole component of the defocussing magnet and the ratio of the defocusing magnet strength to the focussing magnet strength (F/D ratio). The results can be seen in Fig. 12, which shows the dynamic aperture for different values of horizontal and vertical tune respectively using different combinations of starting conditions. In the upper plot, 'prtcl $(+x, 0)$' shows the DA calculated by increasing the single particle coordinates along only the positive x axis from the closed orbit shown in Fig. 9 and when $\theta_{DA} = 45^\circ$.

The results in Fig. 12 are consistent with those in references [11,16,41] and while the results in these references were also calculated with PyZgoubi, they used a different DA calculation method and routine. The same applies to the vertical direction in the lower plot. The data labelled 'prtcl $(0, +x')$' and 'prtcl $(0, +y')$' show that the dynamic aperture is lower when we increase the single particle coordinates along a different axis in transverse phase space. We can also see that the method of using all 16 combinations described in Section 3 gives us the minimum DA. The minimum DA is important if we want to be confident that all particles within a defined DA limit will survive the specified number of turns rather than just some of them. This comprehensive algorithm in PyZgoubi allows us to be confident of the true minimum of the quantity.

One lattice configuration was chosen to illustrate the contributions from vertical and horizontal DA; this lattice had a horizontal and vertical tune of 0.725 and 0.293 respectively. In Fig. 13 the contribution to the DA in each transverse plane is described by the vector in the 2D space of the plot and the angle between the two components, θ_{DA}; the red point shows the DA when $\theta_{DA} = 45^\circ$. Fig. 13 shows the potential importance of coupling resonances in the transverse planes when calculating the DA; in this case the minimum DA is realised when $\theta_{DA} = 45^\circ$. If there was no coupling

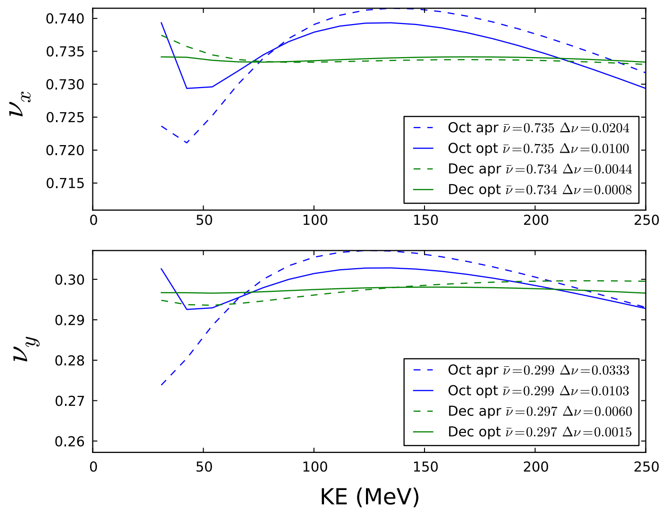

Fig. 11. Horizontal and vertical cell tunes as a function of energy for PAMELA, using approximate results from the literature [11,16] (dashed) and PyZgoubi optimised (solid) using multipole components up to octupole (blue) and decapole (green). Using PyZgoubi to optimise the multipole components of the magnetic fields it is clear to see that the tune variation has been reduced. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

then the DA vector would not depend on θ_{DA}, however for the case in Fig. 13 it clearly does. It is reasonable in the case of PAMELA to assume that the injection emittance is a similar magnitude in the horizontal and vertical planes and therefore the 45° DA is a good estimate to use for a machine design. For other machines with a non-round beam, other angles may need to be considered.

4.3. Dynamic aperture and optics studies with magnet misalignments

We next use PyZgoubi to go beyond the established DA calculations of PAMELA and study the impact of the dynamic aperture in the presence of magnet misalignments. These unavoidable errors will limit the available dynamic aperture if a machine is built and commissioned. Fifty sets of random horizontal magnet misalignments were introduced to all 36 magnets in the PAMELA lattice with a Gaussian distribution of width σ_H truncated at $3\sigma_H$. The width σ_H of the distribution was varied from 10 to 100 μm in steps of 10 μm, corresponding to typical alignment inaccuracies obtained in an accelerator. The optics and the dynamic aperture as a function of the error width were computed in the PyZgoubi framework.

The horizontal tune, vertical tune and the closed orbit distortion for the injection energy of 30 MeV were obtained as a function of σ_H and can be seen in Figs. 14 and 15 respectively.

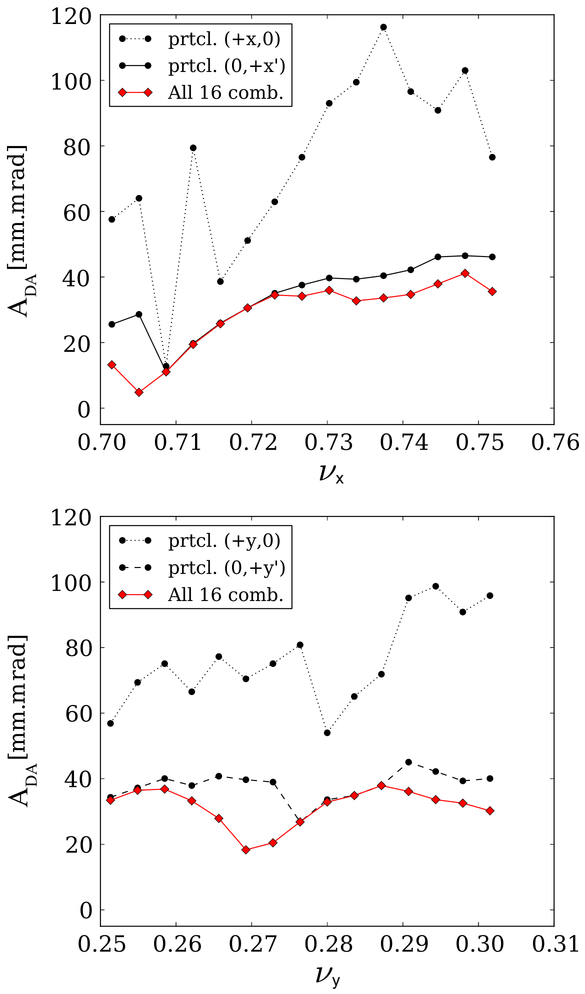

Fig. 12. The dynamic aperture for different values of horizontal and vertical cell tune using different combinations of starting conditions. In the top and bottom plots, 'prtcl $(+x,0)$' and 'prtcl $(+y,0)$' indicate the dynamic aperture calculated by increasing the single particle coordinates along the axis shown in Fig. 9. The data labelled 'All 16 comb.' indicates the 16 combinations used to acquire the minimum dynamic aperture estimate. The dynamic aperture values are scaled by $1/\sqrt{2}$ in order to compare with the results in references [11,16,41].

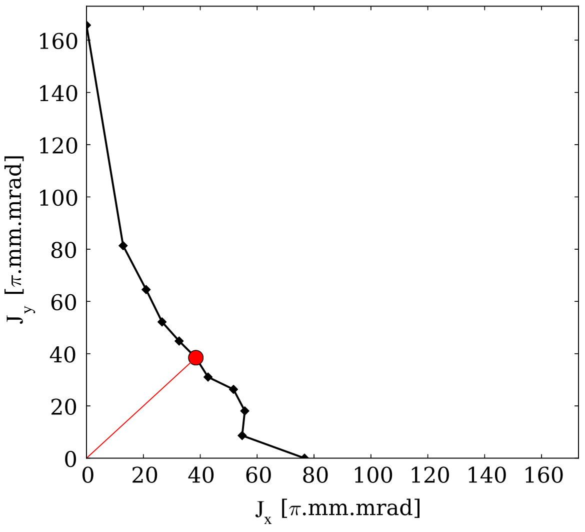

Fig. 13. The dynamic aperture acquired by tracking particles with different values of the horizontal and vertical amplitudes; each point represents a different θ_{DA}. The red point shows the "45°" DA value where $\theta_{DA}=45°$. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

Fig. 14 shows the horizontal and vertical tune as a function of the horizontal misalignment and indicates that as alignment errors become larger, the tune can wander further from the target value and hence encounter undesired resonances. The origin of this tune shift is due to the closed orbit distortion in the non-linear fields of the PAMELA cell. This is important when designing an FFAG, so we are able to understand what errors can be tolerated given a tune shift limit.

The closed orbit distortion (COD) shown in Fig. 15 in the presence of the misalignment errors is calculated by finding the difference between the closed orbit in an error-free lattice and the standard deviation of the orbit in the lattice containing errors. The figure shows a linear increase in the closed orbit distortion on average over 50 seeds, as a function of σ_H. The amplification factor A is calculated as the gradient of the linear fit (as in reference [16]) i.e.

$$A=\frac{\delta COD}{\delta\sigma} \tag{3}$$

where COD and σ are the closed orbit distortion and the width of the error alignments respectively. The value obtained for the amplification factor is 4 in the case shown in Fig. 15 where the simulated injection energy of 30 MeV was used. In reference [16] the amplification factor is 6 for the PAMELA lattice shown, however this value is taken over the acceleration range from 30 to 250 MeV and so we expect it to be larger due to the tune change (although small) during acceleration

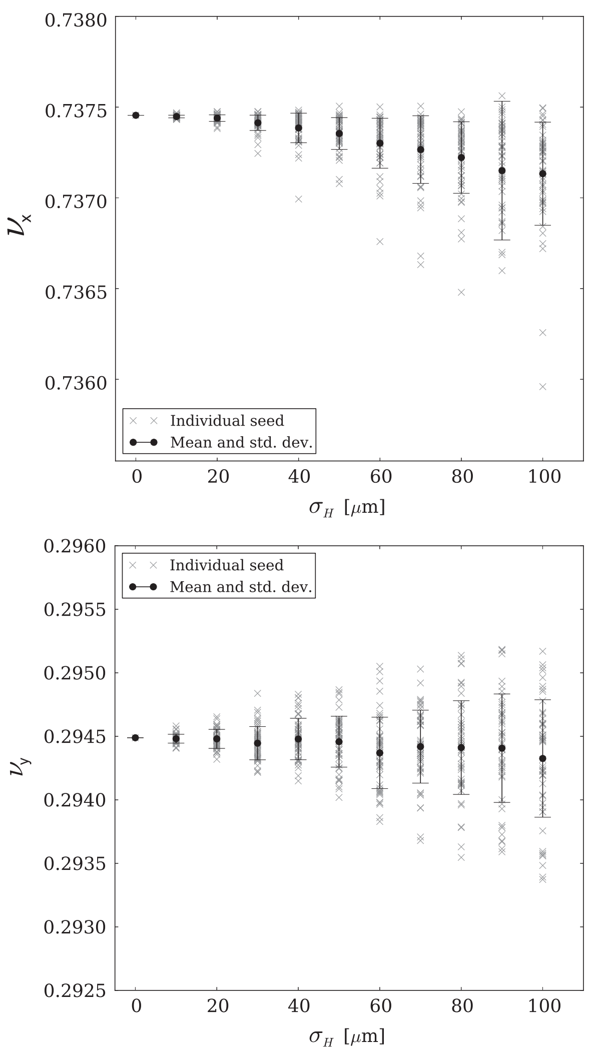

Fig. 14. The horizontal and vertical cell tune as a function of the horizontal magnet misalignment errors with a distribution of width σ_H. Grey crosses show the individual seed value and the black points show the mean.

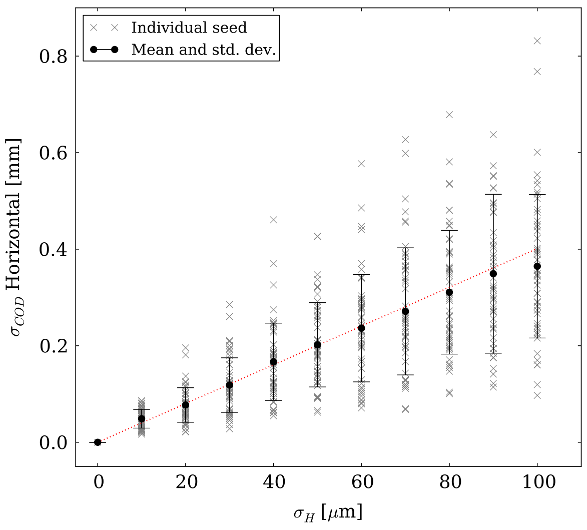

Fig. 15. The standard deviation of the closed orbit distortion as a function of the horizontal magnet misalignment errors with a distribution of width σ_H. A linear fit is shown in red and is used to obtain the amplification factor, A. Grey crosses show the individual seed value and the black points show the mean. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

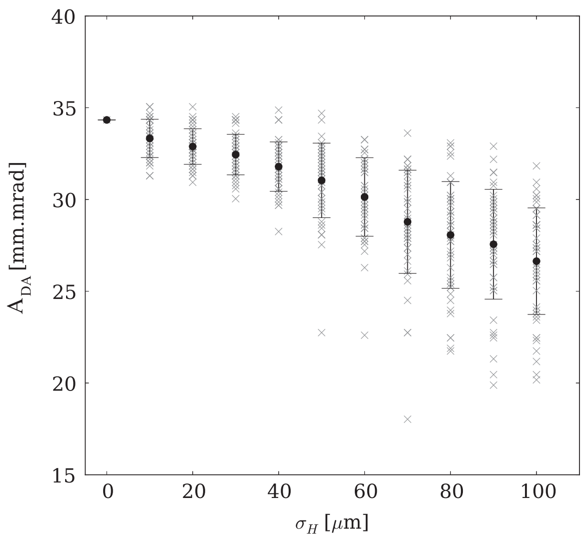

Fig. 16. The dynamic aperture as a function of σ_H is shown. Grey crosses show the individual distribution and the black points show the mean.

and the resonances crossed or approached in the PAMELA tune range. This orbit amplification factor A can be used to fix a maximum magnet alignment tolerance for some required maximum COD.

In Fig. 16 the DA as a function of σ_H is shown. The analysis shows that as the error width σ_H increases, the dynamic aperture reduces and the reduction in the mean over the seeds (black points) is broadly linear in this case. The reduction of the dynamic aperture in the presence of misalignment errors is an important lattice property in the design and optimisation of the FFAG and can be used to define a tolerance on the magnet misalignment for some required machine DA.

In summary, the dynamic aperture of an accelerator like the PAMELA FFAG can be conveniently analysed with the PyZgoubi framework and the tolerance of the machine to magnet misalignments can be well studied using comprehensive algorithms. Furthermore, lattice optimisation was demonstrated using the example of optimising the tune shift as a function of energy by changing the multipole components of the magnets.

5. Concluding remarks

In this paper we presented and described the PyZgoubi framework and discussed its suitability for tracking, designing and optimising novel FFAGs. We showed the simpler input and output format and how this allows for easier analysis and visualisation of FFAG lattices, as well as the potential for comprehensive optimisation of a wide range of lattice parameters may be carried out. We discussed a robust method for defining and measuring dynamic aperture in an FFAG and showed its implementation in PyZgoubi when applied to the PAMELA lattice. The focus of this work was the transverse dynamic aperture, giving a rigorous algorithm, and the development of the tools. Longitudinal effects are a topic for future work. We also showed how scripts can be used in PyZgoubi to optimise the tune variation and dynamic aperture in a lattice. The modular, object-oriented framework of PyZgoubi allows for further evolution of the code where users may define new functions and routines for lattice definition and optimisation; this allows PyZgoubi to be tailored to suit many accelerator design needs.

PyZgoubi aims to be a general purpose interface to the Zgoubi tracking code. New features have been added in order to facilitate the work presented in this paper. The authors will continue to develop PyZgoubi for future applications, for example to extend the DA routines to include longitudinal effects. A test suite is included in PyZgoubi, which can be used to check that changes in Zgoubi do not effect the communication between the codes.

Acknowledgements

We wish to thank François Méot for all his help with understanding the Zgoubi code. We would also like to thank Ken Peach for his help in checking the PAMELA results. Additionally we would like to thank Scott Berg for his great help in understanding many theoretical concepts in beam dynamics and lattice design. Research supported by STFC Grant number ST/K002503/1 “Racetrack FFAGs for medical, PRISM and energy applications”.

References

[1] M. Craddock, The Rebirth of the FFAG, ⟨http://cerncourier.com/cws/article/cern/29119⟩, CERN Courier, 2004.
[2] C. Ohkawa, in: Proceedings of Annual Meeting of JPS, 1953.
[3] A.A. Kolomensky, ZhETF 33 (1957) 298.
[4] K. Symon, The FFAG Synchrotron Mark I, Technical Report, MURA, MURA-043 MURA-KRS-6 ⟨http://lss.fnal.gov/archive/other/mura/MURA-043.pdf⟩, 1954.
[5] J. Scott Berg, S. Kahn, R. Palmer, D. Trbojevic, C. Johnstone, E. Keil, M. Aiba, S. Machida, Y. Mori, T. Ogitsu, et al., FFAGs for muon acceleration, in: Proceedings of PAC, vol. 5, IEEE, Portland, OR, 2003, pp. 3413–3415.
[6] C. Johnstone, S. Koscielniak, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 503 (3) (2003) 445. http://dx.doi.org/10.1016/S0168-9002(03)00997-5.
[7] C. Rubbia, F. Carminati, R. Klapisch, J.P. Revol, C. Roche, J.A. Rubio, An Energy Amplifier for Cleaner and Inexhaustible Nuclear Energy Production Driven by a Particle Accelerator, Technical Report, CERN, CERN/AT/93-47 (ET) ⟨http://cdsweb.cern.ch/record/256520⟩, 1993.
[8] M. Tanigaki, K. Mishima, S. Shiroya, Y. Ishi, S. Fukumoto, Y. Mori, S. Machida, M. Inoue, Construction of FFAG accelerators in kurri for ADS study, in: Proceedings of PAC05, 2005, pp. 350–352. ⟨http://dx.doi.org/10.1109/PAC.2005.1590431⟩.
[9] M. Aiba, K. Koba, S. Machida, Y. Mori, R. Muramatsu, C. Ohmori, I. Sakai, Y. Sato, A. Takagi, R. Ueno, et al., Development of a FFAG proton synchrotron, in: Proceedings of EPAC, vol. 581, 2000. ⟨http://accelconf.web.cern.ch/AccelConf/e00/PAPERS/MOP1B21.pdf⟩.

[10] H. Owen, R. MacKay, K. Peach, S. Smith, Contemporary Physics 55 (2) (2014) 55. http://dx.doi.org/10.1080/00107514.2014.891313.
[11] S.L. Sheehy, K.J. Peach, H. Witte, D.J. Kelliher, S. Machida, Physical Review Special Topics—Accelerators and Beams 13 (2010) 040101. http://dx.doi.org/10.1103/PhysRevSTAB.13.040101.
[12] F. Meot, B. Autin, J. Collot, J. Fourrier, E. Froidefond, J.-L. Lancelot, F. Martinache, D. Neuveglise, The FFAG R&D and medical application project RACCAM, in: Proceedings of EPAC, 2006, p. 2308, WEPCH161.
[13] K.R. Symon, D.W. Kerst, L.W. Jones, L.J. Laslett, K.M. Terwilliger, Physical Review 103 (6) (1956) 1837–1859. http://dx.doi.org/10.1103/PhysRev.103.1837.
[14] S. Machida, Y. Mori, A. Muto, J. Nakano, C. Ohmori, I. Sakai, Y. Sato, A. Takagi, T. Uesugi, A. Yamazaki, et al., Status of 150 MeV FFAG synchrotron, in: Proceedings of Particle Accelerator Conference, 2003, vol. 5, IEEE, Portland, OR, 2003, pp. 3452–3454. http://dx.doi.org/10.1109/PAC.2003.1289945.
[15] C. Johnstone, W. Wan, A. Garren, Fixed field circular accelerator designs, in: Proceedings of PAC99, vol. 5, IEEE, New York, USA, 1999, pp. 3068–3070. http://dx.doi.org/10.1109/PAC.1999.792155.
[16] K.J. Peach, M. Aslaninejad, R.J. Barlow, C.D. Beard, N. Bliss, J.H. Cobb, M.J. Easton, T.R. Edgecock, R. Fenning, I.S.K. Gardner, M.A. Hill, H.L. Owen, C.J. Johnstone, B. Jones, T. Jones, D.J. Kelliher, A. Khan, S. Machida, P.A. McIntosh, S. Pattalwar, J. Pasternak, J. Pozimski, C.R. Prior, J. Rochford, C.T. Rogers, R. Seviour, S.L. Sheehy, S. L. Smith, J. Strachan, S. Tygier, B. Vojnovic, P. Wilson, H. Witte, T. Yokoi, Physical Review Special Topics—Accelerators and Beams 16 (2013) 030101. http://dx.doi.org/10.1103/PhysRevSTAB.16.030101.
[17] J.S. Berg, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 596 (3) (2008) 276. http://dx.doi.org/10.1016/j.nima.2008.08.068.
[18] R. Barlow, J. Berg, C. Beard, N. Bliss, J. Clarke, M. Craddock, J. Crisp, R. Edgecock, Y. Giboudot, P. Goudket, et al., Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 624 (1) (2010) 1. http://dx.doi.org/10.1016/j.nima.2010.08.109.
[19] S. Machida, R. Barlow, J. Berg, N. Bliss, R. Buckley, J. Clarke, M. Craddock, R. D'Arcy, R. Edgecock, J. Garland, et al., Nature Physics 8 (3) (2012) 243–247. http://dx.doi.org/10.1038/nphys2179.
[20] F. Schmidt, H. Grote, MAD-X User's guide, ⟨http://madx.web.cern.ch/⟩, 2005.
[21] F.C. Iselin, The MAD Program Physical Methods Manual, CERN, Geneva, September. ⟨http://cern.ch/Hans.Grote/mad/mad8/doc/phys_guide.ps.gz⟩.
[22] M. Borland, ELEGANT: A Flexible SDDS-Compliant Code for Accelerator Simulation http://dx.doi.org/10.2172/761286, 2000.
[23] F. Méot, The ray-tracing code Zgoubi—status, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, (2014) http://dx.doi.org/10.1016/j.nima.2014.07.022.
[24] F. Méot, J.S. Berg, Zgoubi, ⟨http://sourceforge.net/projects/zgoubi/⟩, 2014.
[25] F. Méot 6-D beam dynamics simulations in FFAGs using the ray-tracing code Zgoubi, Beam Dynamics Newsletter (ICFA), vol. 43, 2007, pp. 44–50.
[26] F. Lemuet, F. Méot, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 547 (2) (2005) 638.
[27] J. Berg, Y. Giboudot, D. Kelliher, S. Machida, F. Méot, B. Shepherd, S. Tygier, et al., Recent developments on the EMMA on-line commissioning software, in: Proceedings of IPAC, 2010, pp. 4235–4327, THPD024. ⟨http://accelconf.web.cern.ch/AccelConf/IPAC10/papers/thpd024.pdf⟩.
[28] J. Fourrier, F. Martinache, F. Méot, J. Pasternak, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 589 (2) (2008) 133. http://dx.doi.org/10.1016/j.nima.2008.01.082 ⟨http://www.sciencedirect.com/science/article/pii/S0168900208001228⟩.
[29] F. Meot, L. Ahrens, K. Brown, Y. Dutheil, J. Glenn, C. Harper, H. Huang, V. Ranjbar, T. Roser, V. Schoefer, et al., AGS model in Zgoubi. RHIC run 13 polarization modeling. status, in: Conference Proceedings of IPAC13, 2013.
[30] F. Méot, Zgoubi Users' Guide, Technical Report, Brookhaven National Laboratory, BNL-98726-2012-IR ⟨https://www.bnl.gov/isd/documents/79375.pdf⟩, 2012.
[31] S. Tygier, D. Kelliher, Pyzgoubi, ⟨http://sourceforge.net/projects/pyzgoubi/⟩, 2014.
[32] SciPy Developers, SciPy—Scientific Libraries for Python, ⟨http://scipy.org/⟩, 2014.
[33] S. Tygier, High current proton fixed-field alternating-gradient accelerator designs (Ph.D. thesis), University of Manchester ⟨http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.551353⟩, 2011.
[34] S.L. Sheehy, Design of a non-scaling fixed field alternating gradient accelerator for charged particle therapy (Ph.D. thesis), University of Oxford ⟨http://ora.ox.ac.uk/objects/ora:4672⟩, 2010.
[35] matplotlib Developers, Matplotlib—Scientific Libraries for Python, ⟨http://matplotlib.org/⟩, 2014.
[36] H. Witte, T. Yokoi, S.L. Sheehy, K. Peach, S. Pattalwar, T. Jones, J. Strachan, N. Bliss, IEEE Transactions on Applied Superconductivity 22 (2) (2012) 4100110. http://dx.doi.org/10.1109/TASC.2012.2186135.
[37] H.A. Enge, Review of Scientific Instruments 35 (3) (1964) 278–287.
[38] B. Muratori, J. Jones, A. Wolski, Analytical Expressions for Fringe Fields in Multipole Magnets, arXiv preprint arxiv:1404.1762.
[39] M. Giovannozzi, W. Scandale, E. Todesco, Physical Review E 57 (3) (1998) 3432. http://dx.doi.org/10.1103/PhysRevE.57.3432.
[40] M. Giovannozzi, W. Scandale, E. Todesco, Prediction of long-term stability in large hadron colliders, Particle Accelerators 56 (LHC-Project-Report-45) (1996) 195–225.
[41] S. Sheehy, K. Peach, H. Witte, T. Yokoi, D. Kelliher, S. Machida, PAMELA lattice design and performance, in: Proceedings of PAC09, 2009, FR5PFP001.