FXFEL Python Scripts Manual and Documentation

Piotr Traczykowski

Version 1 : March 7, 2018

Chapter 1

1.1 Introduction

Handshaking Pythons scripts are a set of routines that alter the format of data in well-defined ways, without changing the information content. These routines are included in the UKFEL suite for two separate reasons:

- The suite contains simulator programs from various sources. **ASTRA** [2], **ELEGANT** [3] and **VISIT** [4] were developed externally. **MASP** and **PUFFIN** were developed 'in-house' by the UKFEL team. In general, these programs all use different, non-standard data formats for their input and output, and it would not be appropriate for us to modify them to conform to a common format.

  Some of the hand-shaking routines do the conversions needed to connect the output of one program (say **ASTRA**) to the input of another, such as **MASP**.

- All graphics and visualisations generated by the UKFEL suite are displayed by VISIT, a general-purpose graphics program. To obtain suitable displays, the data supplied to VISIT from the other modules must be conditioned in various ways, depending on the type of display needed. For example, certain types of display may need averages or cumulative totals, and these can be provided by routines that do the calculation and modify the data accordingly, before it is supplied to VISIT.

  The UKFEL suite includes a library of conditioning routines which will serve for most purposes. We also supply instructions and examples to show how to write new routines, using PYTHON.

We have adopted HDF5 as a 'base' format. When representing an electron beam, each record represents for a macro-particle, which is a group of electrons with identical properties. The properties are

- Position in three orthogonal space coordinates X, Y and Z. The principal direction of motion is in the Z axis.
- Momentum in the three directions : Px, Py and Pz. Electrons are usually travelling at near-light speeds, and the momenta take account of the consequent increase in mass.
- The number of electrons in the particle.

.

The values of length are in SI units (meters) but momentum is in scaled units (divided over $(m \cdot c)$).

1.2 Conversion Scripts and SU5 file format – user manual

1.2.1 The purpose of the scripts

The main aim of creating the scripts was to allow converting particle sets between different software packages dealing with FEL and accelerator simulations (i.e. Astra, Elegant, Puffin, Vsim). For maximum portability the scripts are written in Python and don't use system specific commands. The script use SU5 format as 'common format' for exchange. The scripts don't transfer data directly between packages but they convert data either into SU5 or from SU5 format – such attempt reduced number of possible configuration and in future allows easier expanding of format converters library.

1.2.2 Requirements

Scripts are written in **Python 2.7** with use of **numpy** and **table** modules. In order to use them you need to install Python 2.7 (Windows, Linux) and above modules. The other requirement is **SDDSToolkit** which is used by script dealing with Elegant and Puffin data manipulation. The scripts (except SU2CDF) use one core only and sometimes lot of memory – the size of the file you want to convert usually determine the amount of necessary memory.

1.2.3 Installation

The installation is simple – just copy the scripts to directory where you intend to use them or create access path in your system.

1.2.4 SU5 file format

The SU5 file format is HDF5 file which contains particle positions and momentum in common array (the dataset has class array). The table is built of 7 columns: x, px, y, py, z, pz and Ne. Ne column defines number of electrons per each record (macroparticle). All used units are SI (m, kg, s) but momentum is scaled i.e. divided over $(m \cdot c)$ – such approach reduces the problem of having tiny values which may cause problem. There is no reference particle and the coordinates are in absolute space. The class of the array is 'table' and this is due to restrictions from VisIt which is unable to handle properly other classes (fails to open). There is also just one dataset in the file by default. If user wants to pass more datasets then they should have different name and the user is also supposed to take extra care about metadata.

The below table show how the SU5 file should like when opened in 'hdfview' or other similar program:

X	Px	Y	Py	Z	Pz	Ne
1	0.1	2	0.1	1	10	1
....	....	....	....	....	....	....
1	0.1	2	0.1	2	10	1

Please not that Ne column should have values larger than 1 as it means that there is at least one electron it the record – not keeping to this rule may cause some of the script to fail. The file also contains metadata for VisIt (VizSchema) which allows quick visualisation of your SU5 in VisIt – this is to allow users to see what they just converted from other packages or what they are up to convert to other packages. For purpose of history there is also metadata added that contains time and date of conversion into SU5, name of the source file used and origin (i.e. if source file comes from Astra it will show ASTRA). The history/origin metadata is added as follows:

```
ParticleGroup._v_attrs.FXFELConversionTime=now.strftime("%Y-%m-%d_%H:%M:%S")
ParticleGroup._v_attrs.FXFELSourceFileOrigin='ASTRA'
```

```
ParticleGroup._v_attrs.FXFELSourceFileName=file_name_in
```

If the user would like to add his own metadata to pass into SI5 file then the above scheme can be applied as follows:

```
ParticleGroup._v_attrs.MY_METADATA_STRING_NAME=MY_METADATA_VALUE
```

All scripts use only metadata defined in this manual, any other metadata is just ignored.

1.2.5 Usage

The use of the script is very simple. Assuming that there is file called *input.txt* it is enough to launch the script with the name of the file as parameter e.g.:

```
python Astra2SU.py input.txt
```

The only exception is VSim2SU which needs to arguments. The output will be new file with similar name – only the end of the name of the file will differ depending on which script was used.

1.2.6 List of the scripts

Current release has 10 scripts but the number of scripts can go higher as new packages (e.g. Genesis) will be included in FXFEL project.

Scripts that convert INTO SU5 format

- Astra2SU.py

  The scripts convert particle output from ASTRA to MASP. The conversion is changing the coordinate system from ASTRA system to absolute Cartesian one. The reference particle is still left as first particle in the set and the rest of coordinates and momentum are written in normal Cartesian system. The charge is converted to number of electrons per macroparticle (Ne).

- Elegant2SU.py

  This script converts Elegant particle set to MASP compatible file. The script uses SDDSToolkit ('sdds2stream') and the output file has at the end of file name.

- Puffin2SU

  The script loads selected step from Puffin saved files (e.g. step number 0) in SDDS format and converts the data into SU file format. The data is 'unscaled' and SU units are applied.

- VSim2SU.py

  This script converts HDF5 file which contains particle assembly into SU5 file. In case of this file user is supposed to pass TWO argument not just one. The first argument is the name of the file to process while second is the name of the array witihn the file which contains the necessary data.

- MASP2SU.py

  As current release of MASP saves files in ASCII there was need to write simple script that will just rewrite this data in SU5 format. There is no manipulation on coordinates or charge/number of particles as generally MASP uses very similar data as input/output.

Scripts that convert FROM SU5 format

- SU2ASTRA.py

  The scripts reads SU5 file and then converts the particle set into ASTRA format (generates the reference particle, scales rest of particles, converts charge units).

- SU2Elegant.py

  This scripts converts SU5 into Elegant but assumes that charge is equal for all particles – this requires first equalizing particles charge running some other software.

- SU2Genesis.py The script converts SU5 file into particle input for Genesis. As Genesis requires particles to be equally charged (each macroparticle has same number of electrons) the input data must also have equally charged macroparticles.

- SU2MASP.py

  As current release of MASP needs files in ASCII format there was need to write simple script converting SU5 into ASCII. There is no manipulation on coordinates or charge/number of particles as generally MASP uses very similar data as input/output.

- SU2Puffin.py

  This scripts converts data into Puffin. The Puffin input file needs rescaling the units and calculating some parameters. Therefore the script creates Puffin input file but also some parameters are displayed on the screen after the file is created $(\gamma, \rho, \lambda_u, \lambda_r)$ and these parameters should be next used in the proper Puffin input files. The script creates HDF5 (VizSchema) file that can be also used in VisIt for plotting the data before launching your jobs (data validation). The user is obliged to change parameters a_u and k_u according to FEL design the conversion is aimed at – both parameters are in the script lines 75 and 76.

Other scripts

- SU2CDF.py

  This script is used to prepare sparse data distribution to data suitable for Puffin. The principle of this script is basing of Cumulative Distribution Function. The script is rather a complex one and therefore has its own manual. The most important parameters are multiplier of particles and number of slices per scaled wavelength. The first parameter is responsible for global increase of particles in the distribution (e.g. $20x$) while the second one inform the script that it has to generate certain number of slices per scaled wavelenght – the number of particles in each slice is a constant value. However, there are much more parameters needed as the data is being analysed according to the target FEL undulators design.

- Emmitance.py The script analyses the input file and provides current, energy, energy spread (sliced) and emittance curves. The input data is sliced to 100 slices – if user wishes to have more or less slices a change in the script is necessary. The script is parallelized.

1.3 Detailed description

This paragraph will describe in more details how the scripts are converting the data.

1.3.1 Astra2SU

This conversion is relatively straight forward. SU5 needs data in absolute coordinates system while Astra uses first particle in data set as reference particle for Z axis and P_z column. First thing done by the script is to convert the space coordinates to absolute ones. First row is the reference particle, therefore we use this row to convert the rest of the particles – note that we only convert the Z axis (Z and P_z, third and sixth column in dataset) ! The same loop in the script is also responsible for converting Astra units to SI units – in case of Astra momentum is eV/c and we need to scale it to $kg * m/s$ (multiply by 5.36^{-28} and then divide over $(m \cdot c)$. Last operation is to put the number of particles in last column of our output. To do this we scale charge of single record to Coulombs and then divide over single electron charge. Next, the data is saved in SU5 format (ordered as follows: $X, P_x, Y, P_y, Z, P_z, Ne$).

1.3.2 Elegant2SU

Elegant to SU5 **conversion requires SDDSToolkit installed** as SDDS commands are used for accessing the SDDS file used by Elegant as native format. Elegant uses its own style format to store data. The columns we can retrieve from dataset are: x, xp, y, yp, t and P. Please note that xp and yp are not straight momentum data – this is 'projected on Z axis' momentum. The way to get absolute momentum is to find first p_z

$$p_z = \frac{P \cdot ERM \cdot 10^6}{\sqrt{xp^2 + yp^2 + 1}}$$

where ERM is Electron Rest Mass and 10^6 term scales MeV to eV – this is all done to scale momentum units from eV/C to SI units. While we calculated p_z we can now find p_x and p_y as:

$$p_x = p_z \cdot xp$$

$$p_y = p_z \cdot yp$$

The one more lacking data is Z positions of particles. To calculate this we use following equation:

$$z = \frac{t \cdot c \cdot p_z}{\sqrt{p_x^2 + p_y^2 + p_z^2}}$$

The only one more thing to do is to calculate proper charge of particles. Elegant doesn't distinguish particles relating to charge – all particles have same charge and this value is stored in source SDDS file under name 'Charge'. We need to retrieve this value, divide this over number of records (particles) and then divide over charge of single electron to get number of particles in single record. Next the data is stored in SU5 file.

1.3.3 VSim2SU

The Vsim software stores data in HDF format files. First we need to load such file. The structure is relatively simple as it contains spatial positions of particles, momentum (scaled) and their charge (in weighted format). There is also one issue that Puffin requires particles to travel along z-axis while this is not necessary in Vsim, to avoid problems the user needs to check what is the 'travel axis' in source (Vsim) data. In example we had particles were travelling along x-axis and thus axis z and x are switched (so is momentum p_x and p_z). To obtain SI units the momentum in Vsim needs to be divided over mass of single electron – as we want to go to Scaled Units we divide the momentum only by c. Little more complicated issue occurs when it comes to charge as Vsim stores scaling parameter in metadata – the metadata variable is called 'numPtclsInMacro' and its value needs to be read from the file and then we can calculate the numbers of particles by:

```
n=len(Electrons)
RecordsNumber=n
ParticlePerRecord=ParticleNumber/RecordsNumber
```

The value of 'n' is calculated as length of the table (this gives us the total number of records in file). Finally when saving the the data the source column containing the scaled charge needs to be multiplied by 'ParticlePerRecord' value. The output file is saved as HDF5 data in SU input format.

1.3.4 MASP2SU

The conversion is reduced to just rescale the momentum to SU units (p/mc) and reordering columns to SU friendly format x, px, y, py, z, pz, N, adding proper metadata (VizSchema) and saving the data in HDF5 file.

1.3.5 SU2Astra

The script works in a reverse way to Astra2SU script. Data is unscaled to SI units and one more particle (reference particle) is created to fulfill Astra requirements. Reference particle is created as geometrical centre of the beam (not mass centre as the average $x/y/z$ values are not weighted by mass). Finally the ouptut is saved to Astra format and ASCII file.

1.3.6 SU2Elegant

Again, the script works on reveres principle – in this case reverse to Elegant2SU script. First the data is loaded and some additional variables are calculated to allow creating output array in Elegant format x, xp, y, yp, t and P. Please note that Elegant requires all macroparticles having same charge/weight – if the input data doesn't fulfill this requirement then the output data will be rather useless for Elegant run. Finally the data is converted and saved into SDDS file to further use with Elegant/SDDS tools.

1.3.7 SU2Genesis

The Genesis particle input file in fact is very similar to SU format except it is ASCII file and requires a header describing some beam info: charge of the beam, number of records in the file (macroparticles) and format (in our case X, PX, Y, PY, Z, P is used). Similarly to Elegant, Genesis requires all macroparticles having same charge/weight – if the input data doesn't fulfill this requirement then the output data will be rather useless for Genesis run (the data will be wrong).

1.3.8 SU2MASP

The conversion is reduced to just rescale the momentum to SI units and reordering columns to MASP friendly format x, px, y, py, z, pz, N and saving the data in ASCII (text) file.

1.3.9 SU2Puffin

This script converts data from SU5 into Puffin accepted format. The detailed description of Puffin data is published in Puffin papers – this section will focus here only on equations used for direct conversion of data. First of all please note that this is script that is most time consuming of all described here scripts – running time depend on source data size, can run even few minutes and if you suppose that the script is 'frozen' check your system monitor if CPU is busy (if yes – script is running), the end of the script will also perform some intense hard disk operations. The script calculates number of records by counting the lines and then initiates empty floating point data tables. Script calculates weights (N_p) using 3 dimensional bins command available

in python – the maximum value is then divided over size of one bin and that gives us value of maximum concentration of electrons. This value might be depending on the number (and of course) size of bins and if the user feels that default value is far from expected it may be easily changed in proper line inside script (see the comments inside the code). The total momentum of particles is calculated as p_{tot}:

$$p_{tot} = \sqrt{p_x^2 + p_y^2 + p_z^2}$$

next γ:

$$\gamma = \sqrt{1 + (\frac{p_{tot}}{m \cdot c})^2}$$

ω_p:

$$\omega_p = \sqrt{\frac{e_{ch}^2 \cdot N_p}{e_0 \cdot m}}$$

where e_0 is vacuum permitivity and m is mass of single electron. γ_0 is calculated as mean value of γ. The above allows to calculate ρ, λ_u and λ_r:

$$\rho = \frac{1}{\gamma_0} \cdot (\frac{a_u \cdot \omega_p}{4 \cdot c \cdot k_u})^{\frac{2}{3}}$$

$$\lambda_u = \frac{2 \cdot \pi}{k_u}$$

$$\lambda_r = (\frac{\lambda_u}{(2 \cdot \gamma_0^2)}) \cdot (1 + \frac{a_u^2}{2})$$

Please note the part responsible for planar type undulator ($\frac{a_u^2}{2}$).
Next step is to calculate L_c and L_g:

$$L_c = \frac{\lambda_r}{4 \cdot \pi \cdot \rho}$$

$$L_g = \frac{\lambda_r}{4 \cdot \pi \cdot \rho}$$

Hence we calculated most necessary variables we can start to rescale the

$$z_2 = \frac{z}{L_C}$$

$$\acute{x} = \frac{x}{\sqrt{L_g \cdot L_c}}$$

$$\acute{y} = \frac{y}{\sqrt{L_g \cdot L_c}}$$

$$\acute{p_x} = \frac{p_x}{\sqrt{m \cdot c \cdot a_u}}$$

$$\acute{p_y} = -\frac{p_y}{\sqrt{m \cdot c \cdot a_u}}$$

$$N_e = \frac{NumberOfParticles}{N_p \cdot L_g \cdot L_c^2}$$

After all data is processed we need to combine it in Puffin accepted format which is currently HDF table of 7 columns ($\acute{x}, \acute{y}, z_{2,}, \acute{p_x}, \acute{p_y}, \gamma, N_e$).

The data is saved as HDF5 with VizSchema metadata applied. The HDF file can be loaded into VisIt and user can visualise the input data before launching time consuming calculations in Puffin.

Bibliography

[1] B.J.W. McNeil, M.W. Poole and G.R.M.Robb
Unified model of electron beam shot noise and coherent spontaneous emission in the helical wiggler free electron laser
Physical Review Special Topics - Accelerators and Beams, Vol 6, 070701 (2003)

[2] Klaus Floettmann
ASTRA : A Space Charge TrackingAlgorithm
www.desy.de/~mpyflo/Astra-manual/Astra_Manual/_V3.1.pdf

[3] Michael Borland
User's Manual for Elegant
www.aps.anl.gov/Accelerator_Systems_Division
/Accelerator_Operations_Physics/manuals/elegant_latest/elegant.html

[4] VIS1T User manuals
https://wci.llnl.gov/simulation/computer-codes/visit/manuals

[5] L.T. Campbell and B.W.J. McNeil, Physics of Plasmas **19**, 093119 (2012)

[6] L T Campbell, B.W.J. McNeil and S. Reiche, New J. Phys. **16** (2014) 103019