FFA23 school - OPAL tutorial

carl.jolly@stfc.ac.uk

11th September 2023

1 Getting started

First, verify that you have OPAL is installed and working.

```
opal --version
```

You should see something like:

```
Using:
Prefix:       /home/carl/Documents/FFA23-school/OPAL-2022.1
Compiler:     gcc
Version:      10.4.0
MPI:          openmpi
Version:      4.1.4
```

If not you may have forgotten to source OPAL, go to the directory where you un-tarred OPAL and enter.

```
source OPAL-2022.1/etc/profile.d/opal.sh
```

2 Jupyter

All the files you will need for the tutorial can be found on this GitHub page. The Jupyter Notebook `FFA23_OPAL_tutorial.ipynb` runs through how to use the `DF_lattice` OPAL input file as well as finding a closed orbit at given energy and calculating the tune of the lattice. You can start a Jupyter Notebook with this command,

```
jupyter notebook
```

Sometimes it is useful to run it as root if you need to move/edit files.

```
sudo -E env "PATH=$PATH" jupyter notebook --allow-root
```

3 Running a simulation

You can parse an input file and run OPAL by entering,

```
opal DF_lattice
```

replacing `DF_lattice` with your input filename if needed, into the terminal. For an FFA simulation the user must also provide an input distribution. This file defines the initial coordinates of each particle in the distribution. The first line of the input distribution file is the initial number of particles in the simulation followed by the coordinates in the local reference frame x p_x y p_y z p_z. The units OPAL uses are explained in the OPAL manual.

```
dist.dat:
1
4.0 0.0 0.0 0.0 0.0 0.0
```

The dist.dat input distribution file will start a particle with the initial momentum specified in the main input file at the coordinates in `dist.dat`.

4 Finding a closed orbit

OPAL does not include a native closed orbit finder, instead the python script `optimiseCO.py` uses the Scipy minimize function to arrive at a closed orbit for a given OPAL input file.

Before running the python script, make sure the `NUM_TUNRS` in the input file is set to 1, the `DO_MAGNET_FIELD_MAPS` is set to false and `SPTDUMPFREQ` is set to 0. This just turns off unnecessary output that is not needed for the closed orbit calculation.

To run the closed orbit finder see thr Juypter notebook or use this example.

```
from optimiseCO import ClosedOrbitFinder
import os

input dist file for Co at 3MeV
distribution_filename = "CO_coords_3MeV.dat"

path = os.getcwd()
input_name = "DF_lattice"

initial_x  = 4.0
initial_px = 0.0
ClosedOrbitFinder(16, path, input_name, distribution_filename).main(initial_x, initial_px)
```

The input distribution file should now contain the closed orbit coordinates for this lattice.

5 Calculating the tune

Now we have initial closed orbit coordinates we can now calculate the tune. An example is given in the Jupyter notebook.

Firstly, run the closed orbit particle for some turns. 50 is more than enough for an accurate tune calculation. OPAL will output the `PROBE.loss` files for this simulation. Save these closed orbit probes to their own directory.

Now we have the closed oribit information we can run a simulation of a slightly perurbed particle which should oscillate around the closed orbit. Give a small pertbation to the x and z coordinates of the particle in the input distribution and run 50 turns of this perturbed simulation.

Now we can call the tune calculating python script.

```
from probes_tune_calc import ProbesTuneCalc
import os

path = os.getcwd() # Path to perturbed simulation probe.loss files
path_to_closed_orbit_probes = os.path.join(path,"co_dir/")

ProbesTuneCalc(16, path, path_to_closed_orbit_probes).main()
```

6 Ensure scaling

In a scaling FFA the tune should be constant with the particle energy. Modify the OPAL input file to verify this.

7 FODO lattice

8 change FD ratio/ K value / tune