Oscar Blanco
Previously at the synchrotron SOLEIL (France)
Now at the synchrotron ALBA (Spain)

Frequency Maps and IDs in AT and pyAT

October the 2nd, 2023

Summarized in one slide

Example in python using the SOLEIL lattice and an Insertion Device from SOLEIL

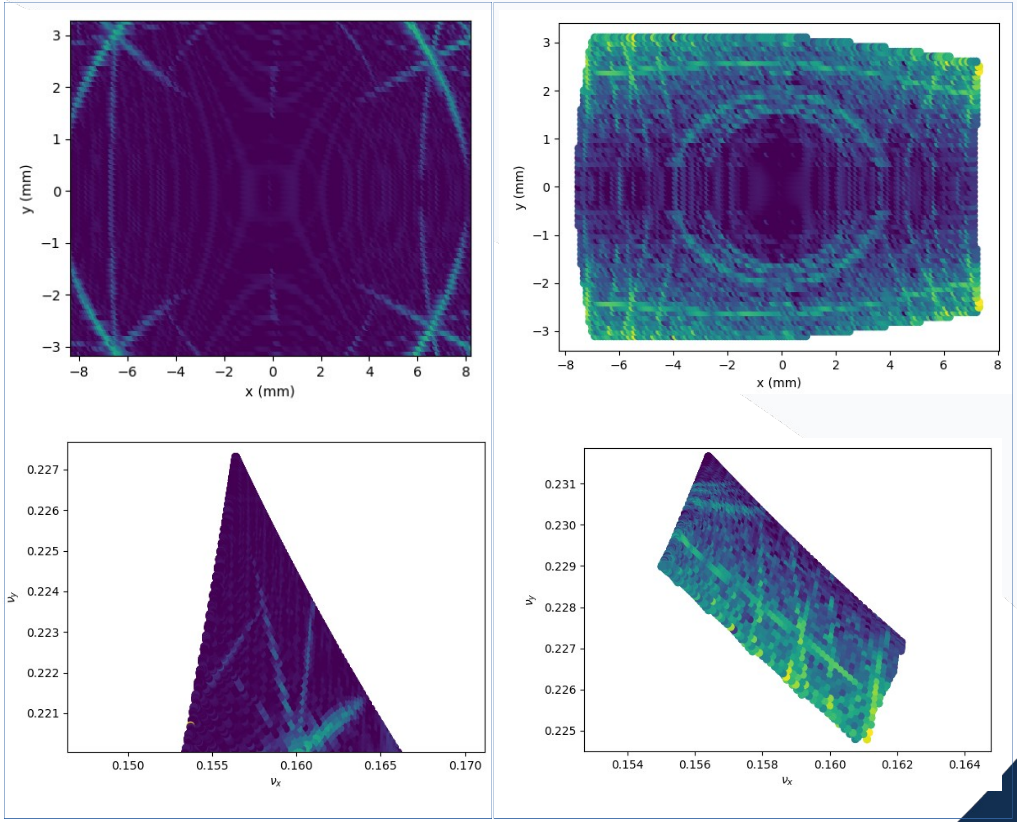
```
# load a ring and insert an Insertion Device
#u20 = at.InsertionDeviceKickMap('u20',10,
#                          "u20_g45mm_kicks_2022nov10.txt",
#                           2.75)
exec(open("example_InsertionDevice.py").read())

#frequency map
newlattice.disable_6d();
fmadata = at.fmap_parallel_track(newlattice,
                    coords=[-10,10,-10,10],
                    steps = [200,200],
                    turns=1024, verbose=False);
fmadata=numpy.array(fmadata[0])

# plot
from matplotlib import pyplot as plt
plt.scatter(fmadata[:,2],fmadata[:,3],c=fmadata[:,6])
plt.xlabel(r'$\nu_x$')
plt.ylabel(r'$\nu_y$')
plt.show()

plt.scatter(fmadata[:,2],fmadata[:,3],c=fmadata[:,6])
plt.xlabel(r'$\nu_x$')
plt.ylabel(r'$\nu_y$')
plt.show()
```

w/o ID | with ID, no correction



Frequency Maps and IDs in AT and pyAT

- **Frequency Maps**
  - Reminder of the mathematical origins
  - History according to github
  - Functions in pyat and AT matlab
  - Parallelization

- **Insertion Devices**
  - Reminder of the analytical modelling
  - History according to github
  - Functions in pyat and AT matlab (create, insert, get beta-beat)
  - Lattice files In/Out possiblities

- **Conclusions**

Frequency Maps

Frequency Maps

The main purpose is to calculate the tune variation Δν along time (a number of particle cycles).
Useful to identify stable, semi-stable, unstable regions in the phase space.
Mostly used to **get resonances**, and **diffusion maps** (Δν vs x-y).

- **Jacques Laskar**, “Frequency Analysis and Particle Accelerators”.
https://accelconf.web.cern.ch/p03/PAPERS/WOAB001.PDF
https://cds.cern.ch/record/301630/files/p183.pdf

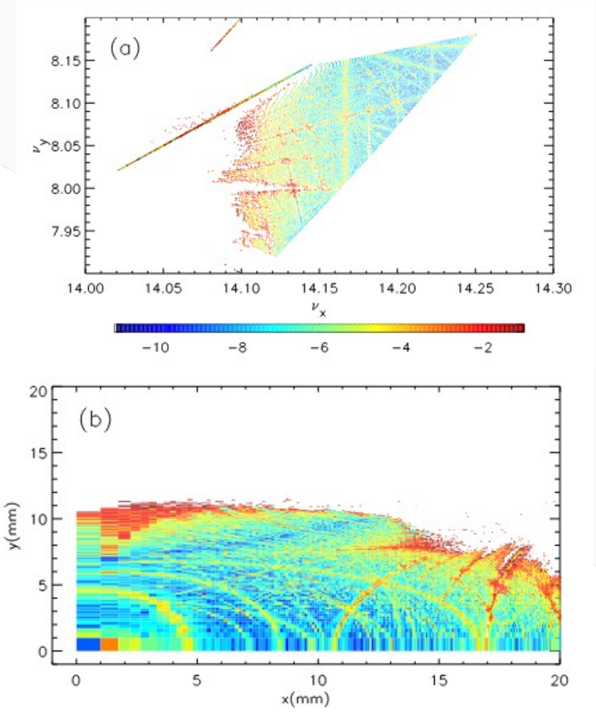
Proceedings of the 2003 Particle Accelerator Conference

FREQUENCY MAP ANALYSIS AND PARTICLE ACCELERATORS

J.Laskar
Astronomie et Systèmes Dynamiques, CNRS-UMR8028, IMCCE-Observatoire de Paris,
77 Av Denfert-Rochereau, 75014 Paris

APPLICATION OF FREQUENCY MAP ANALYSIS TO THE ALS

JACQUES LASKAR[1] and DAVID ROBIN[2]



Frequency Maps history from github

A short history of modifications taken from the github repo :

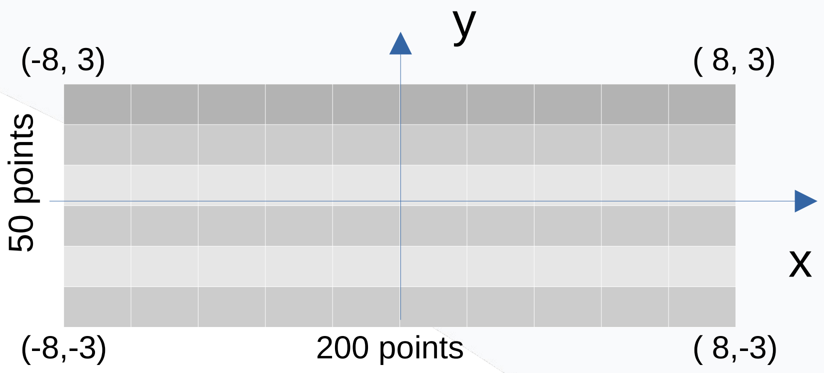
When	Where
2003	at/atmat/pubtools/nafflib/*
2017	at/atmat/atphysics/nafflib/*
2023	at/atmat/atphysics/nafflib/nafflib.c and calcnaff.m
2023	at/pyat/at/physics/frequency_maps.py

When	Who	What
<1998	J. Laskar	Fortran implementation
1998-1999	M. Gastineau	C implementation
2003	L. Nadolski	First implementation at SOLEIL
2012	L. Farvaque	Multiple contributions
2017	L. Farvaque	Moves folder to atphysics
2023	P. Schreiber	Bug fix
2023	O. Blanco	Python implementation

Frequency Maps functions

Basic idea
Each particle on the grid is tracked over 2 x nturns.
- First nturns are analyzed to obtain the frequency.
- Second nturns are analyzed to obtain again a freq.
- We get the difference between the two frequencies

	MATLAB	Python
Parallelization	*parfor using parallel pool	Tracking with patpass
Freq.Library	nafflib in c	harmonic_analysis

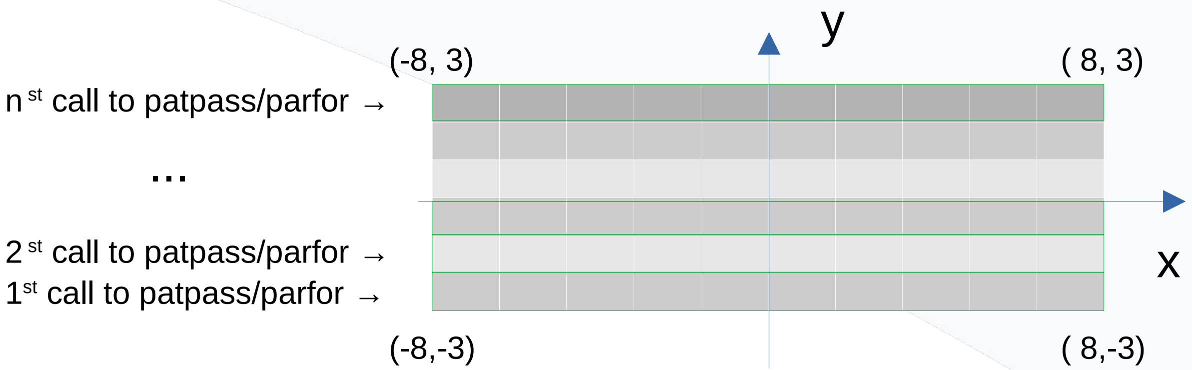



```
# python ( - parallel tracking,
#              - frequency analys IS NOT parallelized)
ring.disable_6d();
[xy_tune_nudiff_array, plosses_array] = \
    at.fmap_parallel_track(ring,
    coords=[-8,8,-3,3],             # [-x,+x,-y,+y] in mm
    steps=[200,50],                 # x by y points grid
    turns= 512,                     # tracking turns
    scale='linear',                 # equally spaced
    lossmap=True,                   # if losses are needed
    verbose=False)                  # print execution percentage
```

```
% matlab ( parfor allows to do:
%              - parallel tracking
%              - parallel frequency analysis)
```

There is no generic function.
Libraries and examples are available in the **nafflib folder**.
I believe every laboratory implements its own routine.
*****I got to know the fmap implementation at soleil (fmap) from which I based the python implementation.

Frequency Maps Parallelization



python
1) grid
2) take one horizontal slice :
   - parallel tracking (**patpass**)
3) freq. Analysis (not in parallel)
4) save
5) repeat from 2) for all slices

% matlab
% using the **Parallel Computing Toolbox**
1) grid, and memory allocation
2) inside **parfor**
   - track one slice
   - do freq. analysis (calcnaff) of one slice
   - save
3) repeat from 2) for all slices

COMMENTS : it works but it could be improved
- Parallelize the python frequency analysis
- Track all particles at the same time with patpass/parallel pool (supposing there is enough memory space)

  Memory ≈ nx.ny.(2.nturns).6D.nbytes + FrequencyAnalysisMemory
- Track custom arrays (diagonals, concentric ellypses, any given array, …)
- Include other grids (radial uniform, radial log, etc.)

Insertion Devices (IDs) in AT and pyAT

Insertion Devices (IDs) in AT and pyAT

To first order in α, the undulator has no effect on the beam trajectory. To second order there is a kick.
- Pascal Elleaume, “A new approach to the Electron Beam Dynamics in Undulators and Wigglers”.
Proceedings of the 1992 European Particle Accelerator Conference.
https://accelconf.web.cern.ch/e92/PDF/EPAC1992_0661.PDF

$$\frac{dx}{ds}(\infty) = -\frac{\alpha^2}{2}\int_{-\infty}^{\infty}\frac{\partial}{\partial x}\Phi(x,z,s)\,ds + o(\alpha^3)$$

$$\frac{dz}{ds}(\infty) = -\frac{\alpha^2}{2}\int_{-\infty}^{\infty}\frac{\partial}{\partial z}\Phi(x,z,s)\,ds + o(\alpha^3) \qquad (5)$$

$$\Phi(x,z,s) = \left(\int_{-\infty}^{s} B_x ds\right)^2 + \left(\int_{-\infty}^{s} B_z ds\right)^2$$

where $\alpha = e/\gamma mc$ and $x' = dx/ds$. e is the electron charge, m its mass, c the speed of light and γmc^2 the total energy of the electrons. (2) can be solved by making a power expansion in

A New Approach to the Electron Beam Dynamics in Undulators and Wigglers

Pascal ELLEAUME
European Synchrotron Radiation Facility
BP 220, F-38043 Grenoble
France

Insertion Devices (IDs) history from github

A short history of modifications taken from the github repo :

When	What
2007	at/atmat/pubtools/create_elems/idtable_dat.m
2007	at/at/integrators/IdTablePass.c
2023	at/pyat/at/lattice/idtable_element.py

When	Who	What
2007	Muñoz, Safranek	First implementation
2008	Zeus Marti	Implementation of the IdTable pass method
2012	Boaz Nash	Update
2012-~	L. Farvaque	Multiple contributions
2019	N. Carmignani	Bug fix
2023	O. Blanco	Python element compatible with matlab element

ID file format

A pass method was created in AT to integrate over ‘s’ a field that depends on the particle horizontal ‘x’, and vertical (‘z’ in the article by Pascal Elleaume). This is called **IdTablePass**.

It requires an **input file with two tables** :

- an x-z table for the **horizontal kick (first START)**
- and another x-z table for the **vertical kick (second START)**

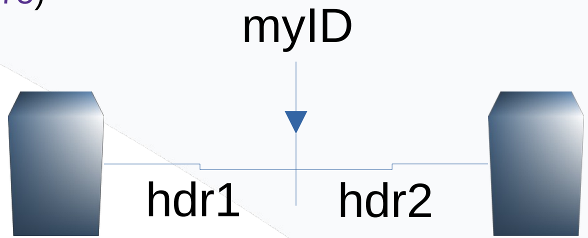
Simplified template of the Insertion Device file format :

```
#comment in line 1
#comment in line 2
Length_in_m
#comment in line 4
Number of points in horizontal plane :nh
#comment in line 6
Number of points in vertical plane :nv
#comment in line 8
START
          pos_point1h pos_point2h ... pos_pointnh
pos_point1v
...                 horizontal kick_map(nv,nh)
pos_pointnv
START
          pos_point1h pos_point2h ... pos_pointnh
pos_point1v
...                 vertical kick_map(nv,nh)
pos_pointnv
(End Of Line)
```

Insertion Devices (IDs) creation and insertion

HOW TO CREATE THE ELEMENT

```
% matlab (name, integration slices along s, inputfile, normalization energy GeV, passmethod)
u20     = atidtable_dat('u20', 10, 'u20_g45mm_kicks_2022nov10.txt', 2.75, 'IdTablePass');
# python (name, integration slices along s, inputfile, normalization energy GeV)
u20     = at.InsertionDeviceKickMap('u20',10,"u20_g45mm_kicks_2022nov10.txt", 2.75)
```



HOW TO INSERT THE ELEMENT INTO A LATTICE (example)

I would like to insert myID between the drifts hdr1 and hdr2 :

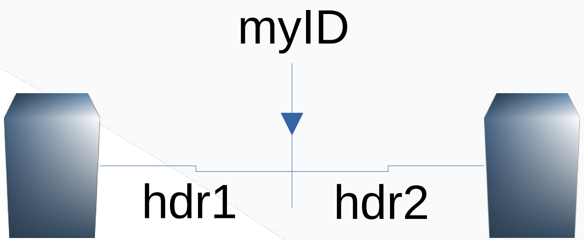
```
% matlab
NEWRING{hdr1_index}.Length = NEWRING{hdr1_index}.Length + NEWRING{hdr2_index}.Length;
NEWRING(hdr2_index) = [];
NEWRING = atinsertelems(NEWRING,hdr1_index,0.5,u20);
IDindex = hdr2_index;
# python
dummyelem = newlattice[hdr2_index].deepcopy()
newlattice.insert(hdr2_index, dummyelem)  # insert a dummy element before hdr2
newlattice[hdr2_index] = u20.deepcopy()   # overwrite dummy element with ID
IDindex = hdr2_index
newlattice[IDindex -  1].Length = newlattice[IDindex -  1].Length - u20.Length/2;
newlattice[IDindex + 1].Length = newlattice[IDindex + 1].Length - u20.Length/2;
```

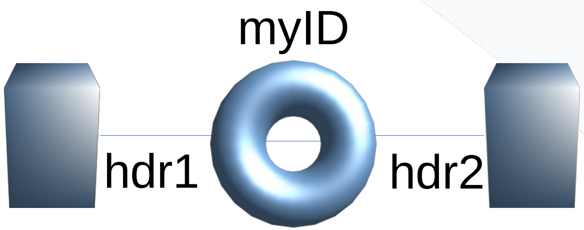
Insertion Devices (IDs) optics perturbation

HOW TO Calculate the Beta-Beat

Remember that your original ring, and the ring with the ID are different, they **have different number of elements and elements' length**. Therefore, comparing the optics of your original lattice and the new one could be misleading.

Here is my workaround:
- calculate the optics with the ID
- **deactivate the ID** by changing the pass method
- calculate again the optics
- substract the two optics as needed





In order to deactivate/activate the ID choose the pass method

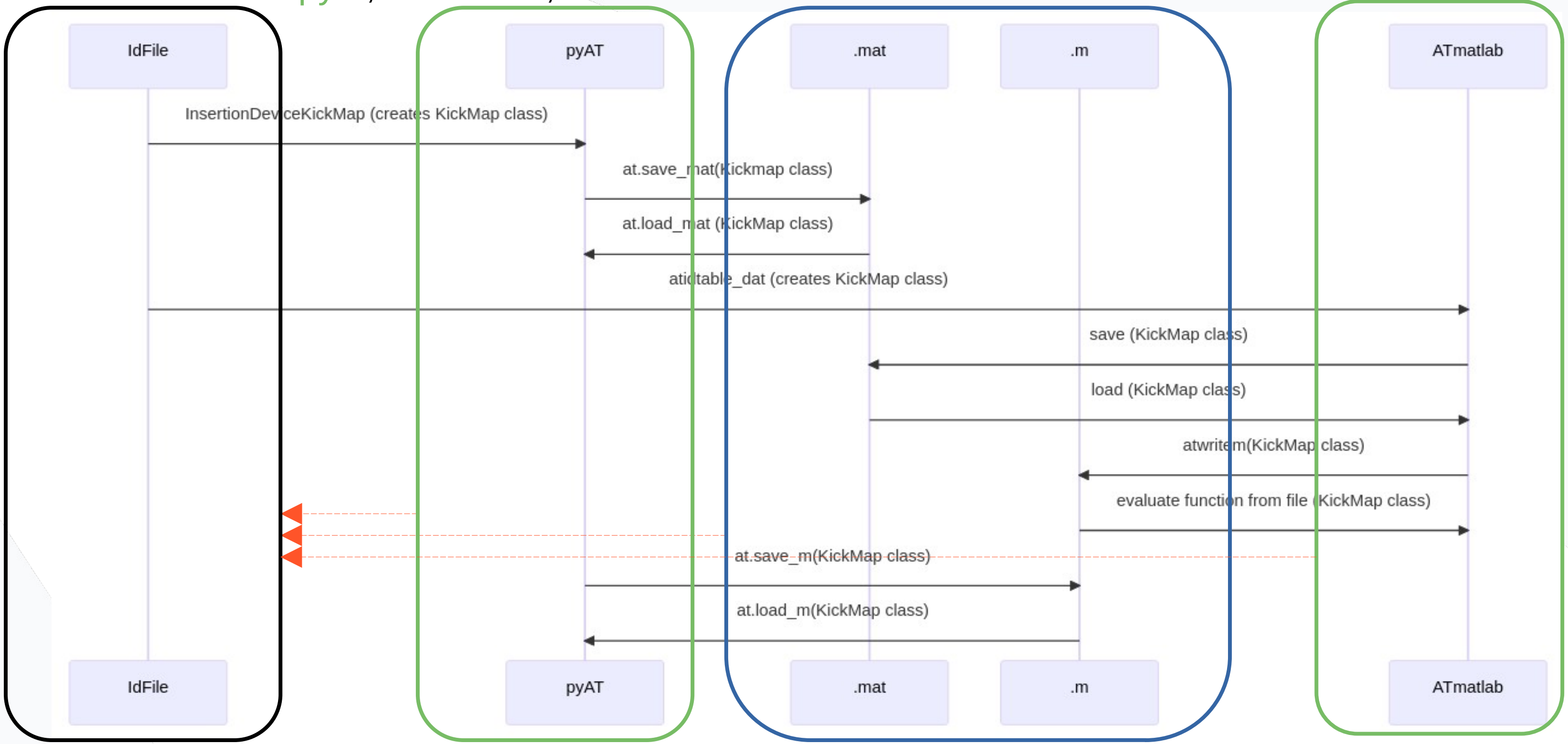
```
# matlab
newlattice{IDindex}.PassMethod = 'DriftPass'
newlattice{IDindex}.PassMethod = 'IdTablePass'
# python
newlattice[IDindex].PassMethod = 'DriftPass'
newlattice[IDindex].PassMethod = 'IdTablePass'
```

In python you could also use :

```
newlattice[IDindex].set_DriftPass()
newlattice[IDindex].set_IdTablePass()
```

Insertion Devices (IDs) files In/Out

From an IDfile to pyat, AT matlab, .m and .mat files :



Warning : currently there is no way to recover the IDfile from the model

CONCLUSIONS

The work here presented wrt frequency maps and Insertion Devices is an adaptation to python from already known models in Fortran, C and matlab with more than 10 years history.

These two tools (ID modelling, and frequency analysis) complement the optics/beam dynamics tools already existing in AT. They might open the road for further use of pyat.

BEYOND THIS PRESENTATION

There is room for further improvements, and among the multiple possibilities, I could mention :

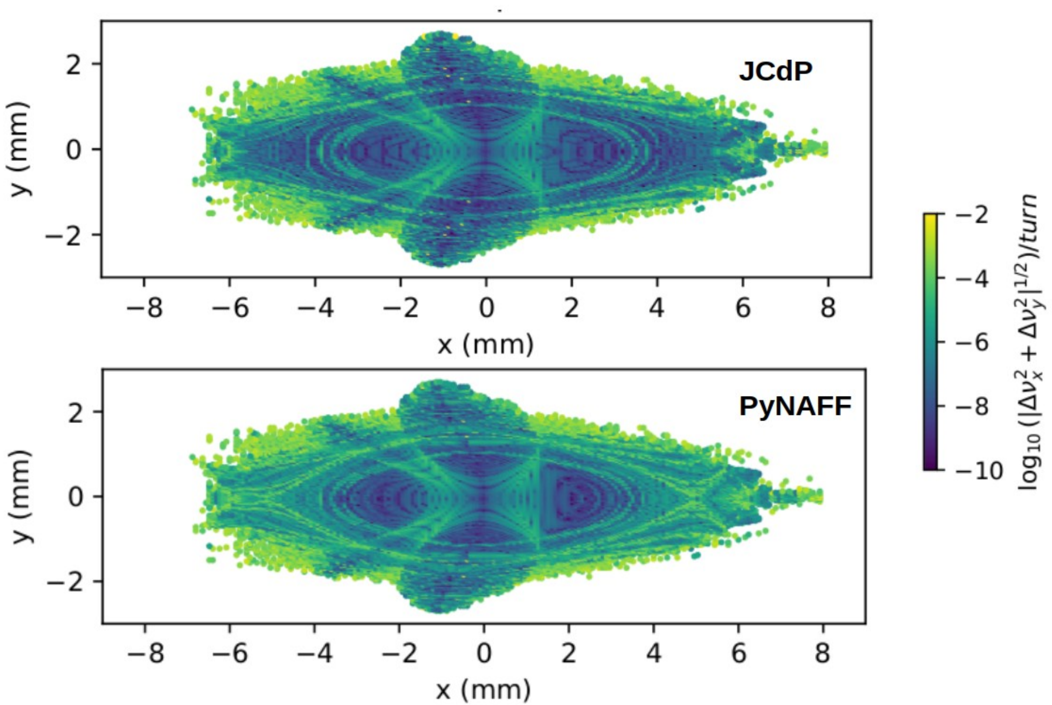
- frequency analysis full parallelization
- frequency analysis flexible/custom grids
- Insertion Devices 3D (x,y,s) tracking ?
- Insertion Devices including radiation effects ?
- Insertion Devices Additional tables for errors ?
- S.I. units always ?

There is no defined path, I have developed these functionalities as the need rised.

EXTRA SLIDES

An example of a frequency map

Different frequency analysis libraries lead to sligthly better/worse speed/precision. In pyat we adopted the library harmonic_analysis derived from Jaime Coelho de Portugal (JcdP) to gain in speed.
Further discussion is available at https://github.com/atcollab/at/pull/556
Similar effect is visible wrt matlab calcnaff library



Recent development info :

IDs
- Create element, insert and calculate the beta-beat https://github.com/atcollab/at/pull/558
- Read and Write https://github.com/atcollab/at/pull/597

Frequency Maps
- Discussion on the speed and precision : https://github.com/atcollab/at/pull/556

example_InsertionDevice.py

```
$ cat example_InsertionDevice.py
### example: insert ID in the ring
# orblancog
# 2023jul15 Including read/write tests
# 2023may24 Revisiting the file saving
# 2023feb05 Initial release

print('\n\nExample to insert an Insertion Device in a ring')
print('    and get the tune variation\n\n')

import at
# load ID file
u20 = at.InsertionDeviceKickMap('u20',
                                10,
                                "u20_g45mm_kicks_2022nov10.txt",
                                2.75
                               )

# load lattice
lattice = at.load_m('lat_soleil.m')
_, beamdata_lat, _ = at.get_optics(lattice,get_chrom=True)
print(f'Parameters of the lattice without ID')
print(f'tune  : {beamdata_lat.tune}')
print(f'orbit : {at.find_orbit(lattice)}')

# choose a drift and get the index
dr_name = 'SDAC1'
refDRIFT = at.get_cells(lattice, 'FamName', dr_name)
# get index of drift
dr_index = at.get_refpts(lattice,dr_name)
# choose next element downstream the desired location of the ID
insertIDindex = dr_index[5]

# create a new lattice where we will insert the InsertionDevice
newlattice = lattice.deepcopy()
### 2023feb05: insert function does not work, it removes the Energy property
### newlattice.insert(insertIDindex+1, u20)
### work around because at.lattice.insert does not copy the Energy property
dummyelem = newlattice[insertIDindex].deepcopy()
newlattice.insert(insertIDindex, dummyelem)
newlattice[insertIDindex] = u20.deepcopy()
# substract half length on each side
newlattice[insertIDindex - 1].Length = newlattice[insertIDindex - 1].Length - u20.Length/2;
newlattice[insertIDindex + 1].Length = newlattice[insertIDindex + 1].Length - u20.Length/2;

## choose pass method, for a quick test
#newlattice[insertIDindex].PassMethod = 'DriftPass'
#newlattice[insertIDindex].PassMethod = 'IdTablePass'
```

```
### … continue


[_,beamdata_newlat,_] = at.get_optics(newlattice, get_chrom=True)
print(f'Parameters of the lattice with ID')
print(f'tune  w ID: {beamdata_newlat.tune}')
print(f'orbit w ID: {at.find_orbit(newlattice)}')

## expected difference : 0.0000 0.0044 -0.0000 (from matlab)
print(f' ... Tune variation ...')
print(f'expected difference  : 0.0000 0.0044 -0.0000 (from matlab)')
print(f'calculated difference:{beamdata_newlat.tune - beamdata_lat.tune} (from python)')

print(u20)
```