Accelerator Toolbox: fonctionnalités et applications

S. White

The European Synchrotron

A BIT OF HISTORY…

AT was initially introduced by Terebilo (SLAC) as a toolbox to perform beam dynamics simulations in the late 90s:

- the core of AT is a C tracking engine, based on the Pascal version of the Tracy code, that implements the integrators of the accelerator components
- the user interface was developed in MATLAB

Then AT was integrated in MATLAB Middle Layer (MML), a MATLAB interface for control systems:

- provides a framework and graphical interface for beam dynamics measurements and studies (optic, BBA, …) users scripts can be shared

In 2015, it evolved into an international collaboration ATCOLLAB and AT2.0 was released:

- Github repository (2017)
- active development in C and MATLAB
- development of a python interface: pyAT (2017, Will Rogers)
- even though it was made compatible with AT2.0, MML is not integrated in this development effort
- Now widely used in the light source community (and also FCC)

DEVELOPMENT, DISTRIBUTION AND DOCUMENTATION

AT source code (MATLAB, C, Python) is open source and available on Github (Apache 2.0 license):

- Installation: PyPI, MATLAB Central or from source
- Very few dependencies
- New release every ~6 months
- MacOS, Linux and Windows supported

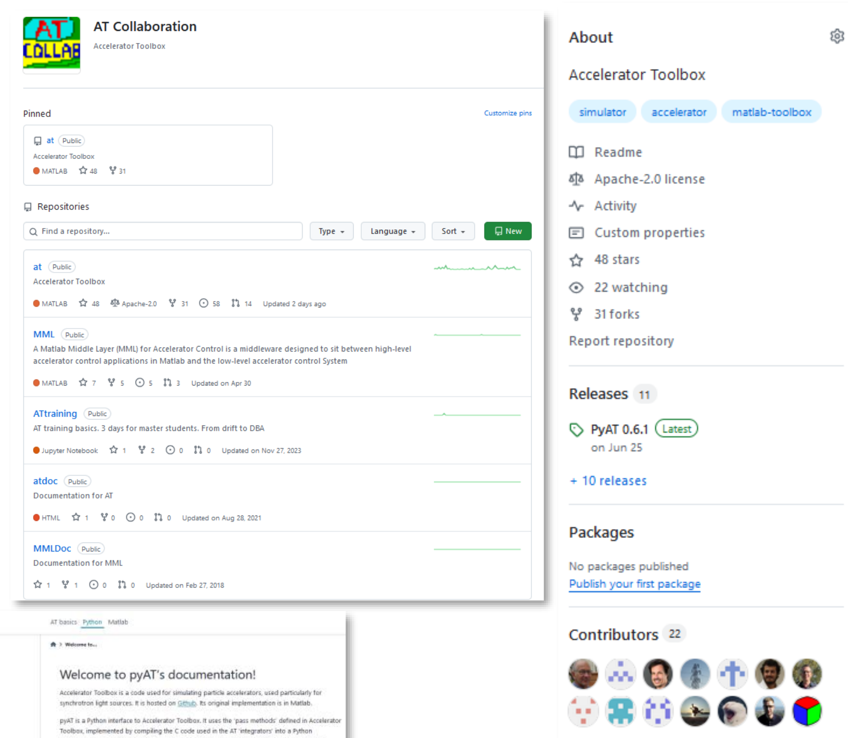
Available documentation and support:

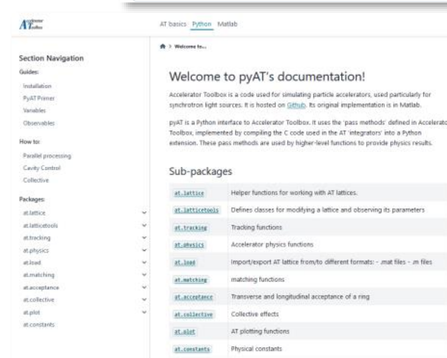
- Python API documentation (sphinx)
- Limited tutorials and examples available
- Github issues and discussion: AT forum for users
- **User's guide needed**

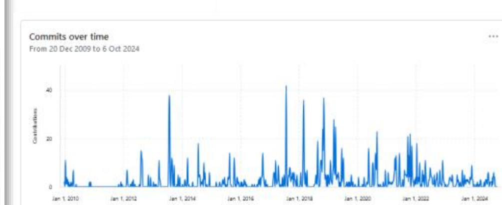
AT workshop held in Grenoble Oct. 2023: 68 participants
https://indico.esrf.fr/event/93/

Very active development and users community: mostly in python
Compatibility with MATLAB interface maintained
Some newer features available only in python

https://github.com/atcollab/at







TRACKING ENGINE

AT is a 6D tracking code and was optimized for that purpose:

- The tracking engine is written in C for best performance
- All the processing (for example linear optics or diffusion matrices) are based on the tracking data
- The engine is modular (elements are treated as individual blocks) and compatible with MATLAB, Octave and python high level interfaces

Tracking through an AT element is done with an integrator or PassMethod:

- An integrator takes as input 6D coordinates, parameters required for the calculation and returns the modified 6D coordinates
- The integrator is an element class attribute: no correlation between element class and integrator, loaded on demand dynamically

Main integrators available:

- Drift (exact and small angle)
- Magnetic elements with and without SR energy loss (rectangular and sector dipoles, multipoles):
    - *4th order symplectic integrators (Forest-Ruth integrator, small angle approximation)*
    - *“Exact” integrators (E. Forest “Beam Dynamics, a new Attitude and Framework”, PTC)*
- RF cavities with and without beam loading (fully self-consistent)
- Wigglers with and without SR energy loss
- Quantum diffusion
- Wake fields
- etc…

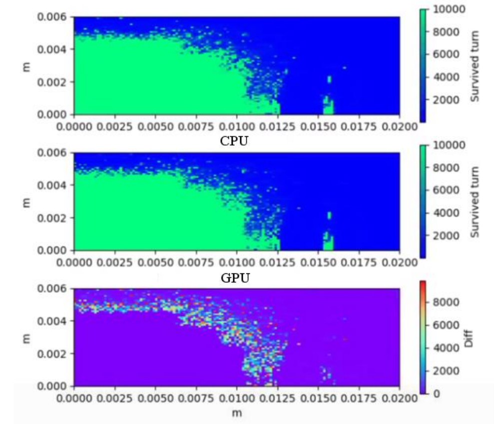
PARALLELIZATION

Several parallelization methods implemented

```
#pragma omp parallel for if (num_particles > OMP_PARTICLE_THRESHOLD*10)
for (c = 0; c<num_particles; c++) { /*Loop over particles  */
```

OpenMP

- Implemented at the integrators level through pragmas on the loop over particles
- Activated with build options
- Available for all interface

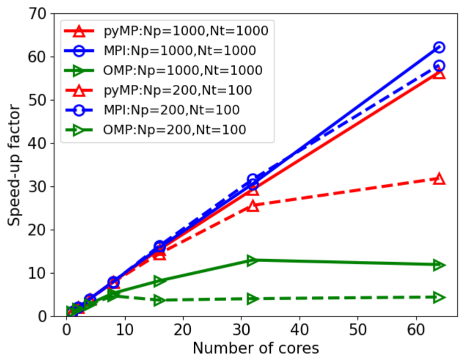


GPU

- GPU implementation of the C engine in CUDA and OpenCL (deprecated for MacOS)
- Compile with all interfaces, fully validated but not deployed yet
- Activated with build options

MPI

- Implemented in collective effects integrators
- Used for massive cluster calculation
- Available only in python, requires mpi4py
- Activated with build options



Standard PassMethod
CPU 14 cores (Xeon@4.1GHz) **199.4 sec**
GPU (RTX A4500) **36.4 sec**
GPU/Single CPU: 77.5

-> 1 RTX A4500 = ~80 cores CPU@4.1GHz

Python multiprocessing

- Implemented in the python interface
- Can be activated with a simple flag

LATTICE DESCRIPTION (PYTHON)

The lattice is described by a list of element objects:

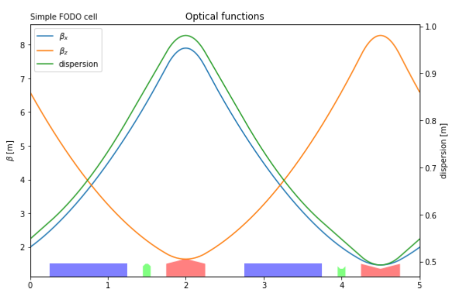
- Inherits from python list: easy manipulation and elements attributes access through indexing of dedicated functions
- The lattice object then gives access to characteristic quantities with simple calls
- Lattice files can be saved and re-loaded in several formats (*.mat, *.m, *.repr, *.json)
- File from other codes can be directly loaded: MADX (*.seq), ELEGANT (*.lte), TRACY (*.lat)

```
Dr = at.Drift('Dr', 0.5)
HalfDr = at.Drift('Dr2', 0.25)
QD = at.Quadrupole('QD', 0.5, -1.2)
Bend = at.Dipole('Bend', 1, 2*pi/40)
```

```
Quadrupole:
    FamName : QF
    Length : 0.5
    PassMethod : StrMPoleSymplectic4Pass
    NumIntSteps : 10
    MaxOrder : 1
    PolynomA : [0. 0.]
    PolynomB : [0.  1.2]
    K : 1.2
```

```
FODOcell = at.Lattice([HalfDr, Bend, Dr, QF, Dr, Bend, Dr, QD, HalfDr],
                      name='Simple FODO cell', energy=1E9)
```

```
FODOcellSext.plot_beta();
```




```
print(at.envelope_parameters(FODOSext))
```

```
  Frac. tunes (6D motion): [0.14995922 0.75000148 0.0184657 ]
                   Energy: 1.000000e+09 eV
          Energy loss / turn: 1.389569e+04 eV
         Mode emittances: [3.47973047e-08 6.71489487e-37 4.03132669e-06]
Damping partition numbers: [0.89298373 0.99999977 2.1070165 ]
           Damping times: [0.05376455 0.04801088 0.02278618] s
           Energy spread: 0.000330959
            Bunch length: 0.0122209 m
        Cavities voltage: 500000.0 V
       Synchrotron phase: 3.1138 rd
   Synchrotron frequency: 55358.8 Hz
```

MAIN FEATURES

Optics calculation and design

- Based on tracking: linear optics are deduced from a one-turn transfer matrix obtained by differentiating the tracking output on a small dimension grid centered on the closed orbit
- Optics calculation available in 4D (un)coupled(1,2,3), 6D(4,5)
- On and off-momentum optics provided for 4D (δ_p offset) or 6D (realistic RF and radiations)
- Optics (and arbitrary other quantities) matching available for all methods

```
ring.disable_6d()
ld0_4d, bd_4d, ld_4d = at.get_optics(ring, refpts=
ring.enable_6d()
ld0_6d, bd_6d, ld_6d = at.get_optics(ring, refpts=
```

Synchrotron radiations

- Fast equilibrium emittance calculation based on Ohmi's envelope formalism(6)
- Energy loss activated with a single function call
- Quantum diffusion element available to get correct emittance from tracking(7)

Single particle effects

- Parallelized (CPU and GPU) calculation of DA and MA
- Parallelized (CPU and GPU) frequency maps
- Collimation and loss maps

```
ring.enable_6d()
bound, surv, track = at.get_acceptance(ring, ('x','xp'), (20,20), (40e-3,1.5e-3))
```

Multi-particle effects

- One turn map lattice description
- Parallelized (MPI) Wake fields and beam loading
- Single and multi-bunch with arbitrary fill pattern
- Rigid bunch of self-consistent algorithms

```
from at.collective import BeamLoadingElement, add_beamloading, BLMode

mode = BLMode.PHASOR
add_beamloading(fring, qfactor, rshunt,
                mode=mode, Nslice=1,
                VoltGain=0.01, PhaseGain=0.01)
```

```
Nbunches = 992
ring.beam_current = 200e-3 #Set total beam current to 200mA
ring.set_fillpattern(Nbunches) #Set uniform filling. Here the harmonic number is equal to 992
```

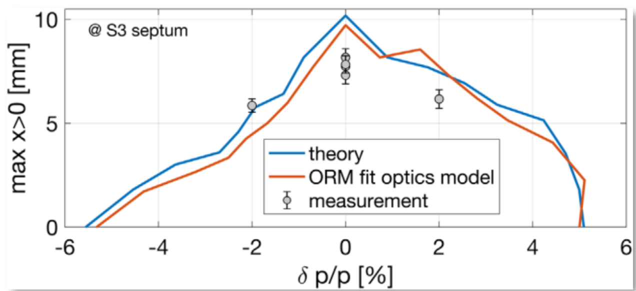

[1] D.Edwards,L.Teng IEEE Trans.Nucl.Sci. NS-20, No.3, p.885-888 , 1973
[2] E.Courant, H.Snyder
[3] D.Sagan, D.Rubin Phys.Rev.Spec.Top.-Accelerators and beams, vol.2 (1999)
[4] Etienne Forest, Phys. Rev. E 58, 2481 – Published 1 August 1998
[5] Andrzej Wolski, Phys. Rev. ST Accel. Beams 9, 024001 – Published 3 February 2006
[6] K.Ohmi et al. Phys.Rev.E. Vol.49. (1994)
[7] B. Nash, N. Carmignani, Quantum Diffusion Element in AT

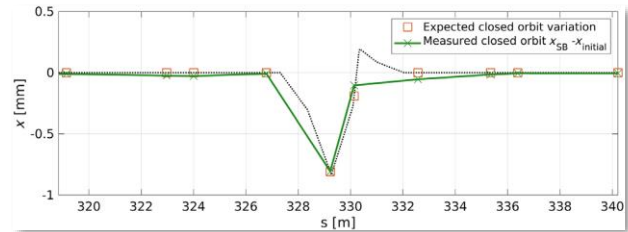
COMPARISON WITH MEASUREMENTS

AT is the main simulation tool at ESRF:

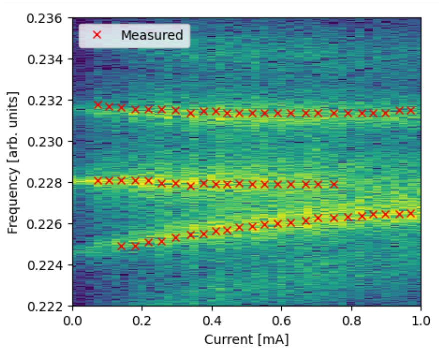
- Lattice design, specification and characterization were done with AT
- We are using theoretical response matrices for orbit and optics corrections

Measured and expected DA for HMBA lattice



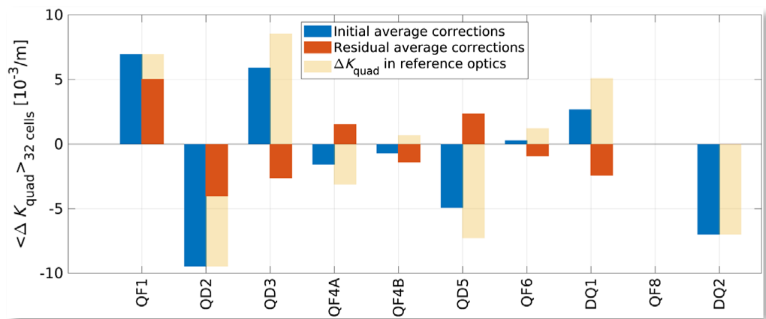


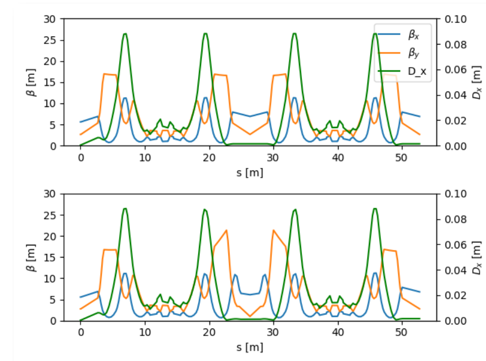
Orbit variation after a SB arc installation



Vertical headtail modes

Offline adaptation of linear optics including magnetic cross talk and DQ correction





Mini-beta optics implementation:

- Expected lifetime: 23.9h
- Observed lifetime: 24.7h

APPLICATION TO ACCELERATOR CONTROL AND OPERATION

AT is used in the operation of many light sources through MML (non exhaustive list):

- **North America:** ALS, SSRL (SPEAR3), Duke FEL, NSLS2, (VUV or X-Ray rings), CLS
- **Europe:** SOLEIL, LAL/THOMX (France), DIAMOND (UK), ALBA (Spain), KIT/ANKA (Germany), ILSF (Iran), MAX IV (Sweden), SOLARIS (Poland), IJCLAB (France), BESSY and HZB (Germany), PETRA-IV (Germany)
- **Asia:** PLS2 (Korea), SLS (Thailand), SSRF (China), NSRRC/TPS (Taiwan)
- **Middle East:** SESAME (Jordan)
- **Australia:** ANSO

It provides ready to use software and operation script (Magnet control, BBA, LOCO, …) through an abstraction of the control system and lattice model

Some facility heavily rely on this software for daily operation but:

- Relies on MATLAB interface
- Many users, very little developers

→ An alternative is strongly desirable

→ The simulation is not necessarily restricted to AT, any proposal welcome!

PYAML EU COST ACTION

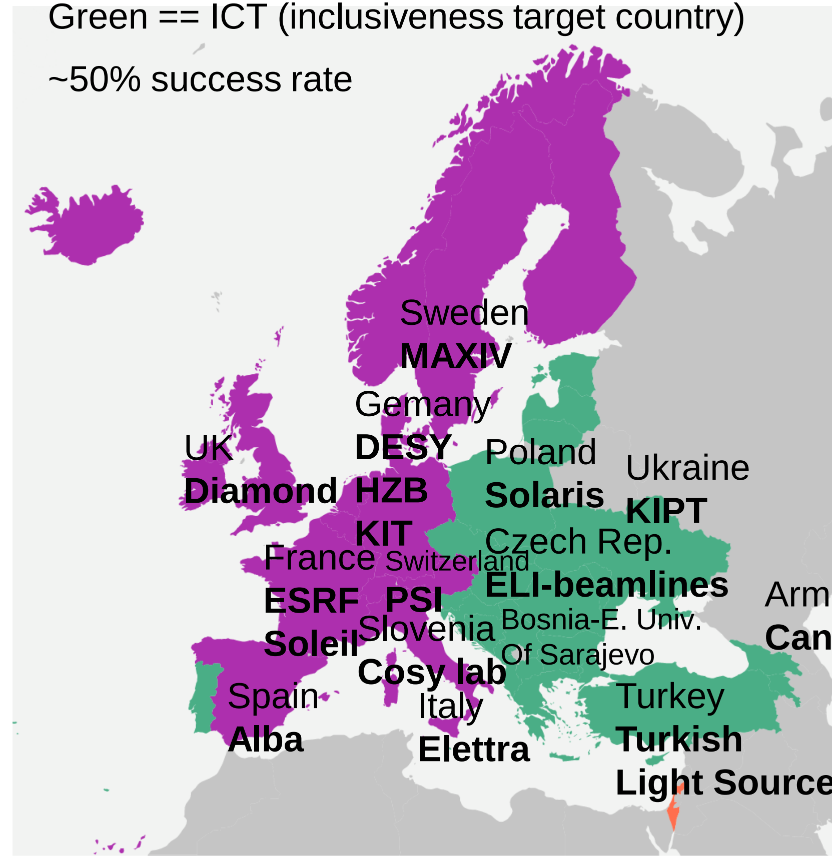
The AT/MML community is setting up a collaboration to produce a python successor to MML:

First workshop help in DESY June 2024:
https://indico.desy.de/event/43233/

Applied for EU funds through COST:

- ~ 150kEuro / year for 4 years for "networking"
- Must involve 50% ITC countries
- All country/lab gets payed trips to conferences, trainings, exchanges, summer schools, coding workshops.
- Non-EU countries collaborators can be paid as expert giving a presentation or a training.

pyAML-Net COST Action Proposal
submitted

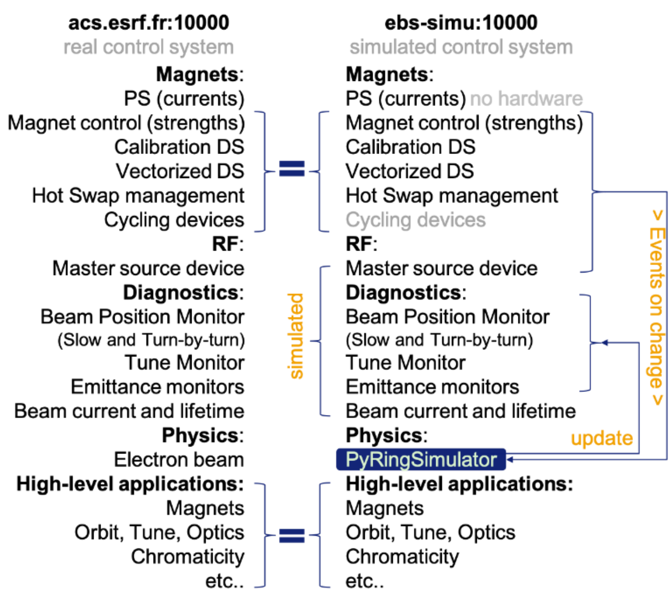


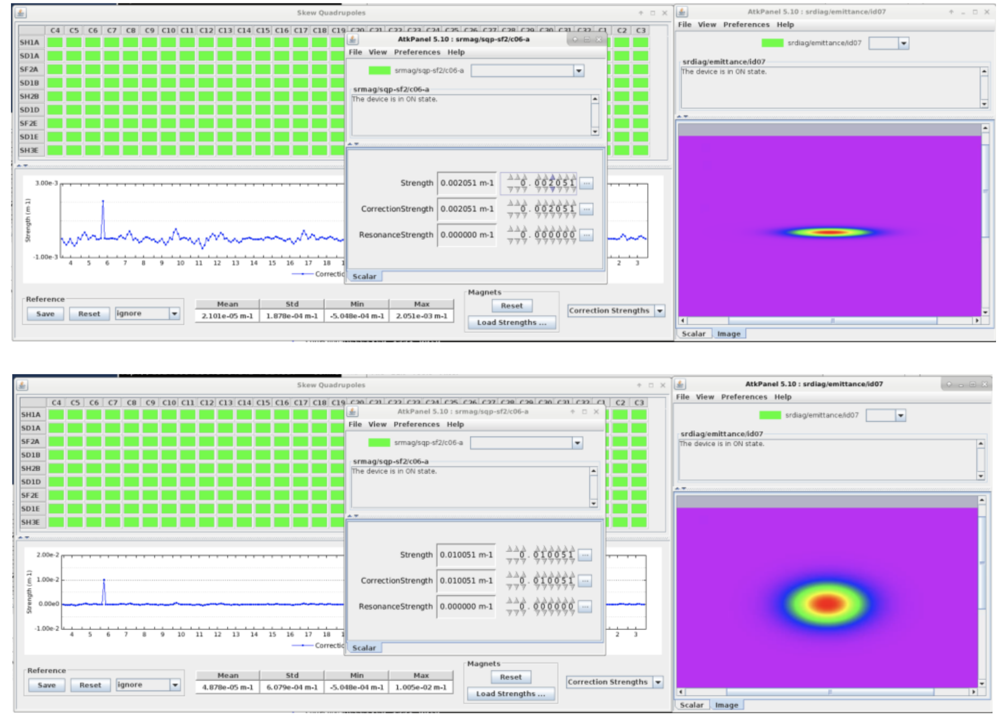
www.cost.eu

ESRF CONTROL SYSTEM SIMULATOR

Developed an ESRF-EBS simulator to reproduce control system in a simulated environment:

- Same software application and controls
- AT is the simulation engine used for this simulator
- Use mostly to develop and test new application





Example of the magnet control application and emittance monitor from the simulator: changing the strength of a skew quadrupole modifies the emittance as expected in the real machine

SUMMARY

Accelerator toolbox started ~25 years ago

It is now widely used in the light source community and is still actively developed

Validated against several other beam dynamics codes and experimental data

It is used in accelerator operation through MML or dedicate application

A wide international collaboration just started to develop the successor for MML in python

MANY THANKS FOR YOUR ATTENTION