FCC-ee software framework

Félix Carlier, A. Abramov, M. Benedikt, R. Bruce, X. Buffat, R. De Maria, A. Faus-Golfe, W. Herr, G. Iadarola, P. Kicsiny, K. Oide, T. Pieloni, M. Rakic, F. Schmidt, D. Schulte, M. Seidel, D. Shatilov, G. Simon, R. Tomás, S. White, Y. Wu, F. Zimmermann

EPFL

CHART

Swiss Accelerator Research and Technology

FCC hh ee he

CERN

The aim of FCCIS workshop is to promote collaboration and inform on activities

Goal of today's presentation:

1. General overview of EPFL project and collaboration
2. Xdeps: Dependencies package for Python
3. Xsequence: Lattice and model manager in Python

A CHART funded project to address key simulation challenges

Establish a modern and maintainable simulation framework to address key limitations for the FCC-ee

- Integrate and merge functionalities of different established simulation tools
- Develop new simulation modules to replace outdated legacy codes
- Perform key simulation campaigns with developed tools: beam-beam, spin dynamics, collimation...
- Collaboration between EPFL & CERN in synergy with current LHC based efforts at CERN.

CERN

EPFL

CHART
Swiss Accelerator Research and Technology

FUTURE CIRCULAR COLLIDER

Current relevant efforts within EPFL project

- Development of a generalized lattice manager to improve model creation and interface between different codes (**F. Carlier**) https://indico.cern.ch/event/1088545/
- Development of self-consistent 6D beam-beam in Xsuite with the goal to perform FCC-ee tuning including beam-beam (**P. Kicsiny**) https://indico.cern.ch/event/1088545/ & FCCIS workshop $\rightarrow$ Wed. 1 December 10:15
- Optimization of optics and tracking for FCC-ee (**G. Simon**)
- Study spin polarization simulations under perturbations of misalignments and magnetic errors for FCC-ee energy calibration studies (**Y. Wu, F. Carlier**), with help from **D. Sagan** (Cornell) and **E. Gianfelice** (Fermilab)

EPFL software framework meetings held regularly involving experts of respective fields
https://indico.cern.ch/category/9606/

Current relevant efforts within EPFL project

- Development of a generalized lattice manager to improve model creation and interface between different codes (**F. Carlier**) https://indico.cern.ch/event/1088545/
- Development of self-consistent 6D beam-beam in Xsuite with the goal to perform FCC-ee tuning including beam-beam (**P. Kicsiny**) https://indico.cern.ch/event/1088545/ & FCCIS workshop → Wed. 1 December 10:15
- Optimization of optics and tracking for FCC-ee (**G. Simon**)
- Study spin polarization simulations under perturbations of misalignments and magnetic errors for FCC-ee energy calibration studies (**Y. Wu, F. Carlier**), with help from **D. Sagan** (Cornell) and **E. Gianfelice** (Fermilab)

EPFL software framework meetings held regularly involving experts of respective fields
https://indico.cern.ch/category/9606/

And much more to come starting from early 2022 (**T. Pieloni**, FCCIS workshop → Thursday 2 December 12:00)

- Ph.D. student on optics, dynamic aperture and lifetime optimization in FCC-ee
- Post-doc on electron cloud studies for FCC-ee
- Post-doc on machine learning applied to dynamic aperture studies for FCC

Collaborations with activities beyond EPFL

- Development of Xsuite, a framework for efficient particle tracking (**G. Iadarola & R. de Maria**)
  https://indico.cern.ch/event/1071856/ & FCCIS workshop → Tuesday 30 November 10:30

- Development of Xdeps, package to describe dependencies and circuits in python (**R. de Maria**)
  https://indico.cern.ch/event/1088545/

- Development of tracking tools for collimation studies in the FCC-ee (**A. Abramov**)
  https://indico.cern.ch/event/995850/ & FCCIS workshop → Wednesday 1 December 09:00

- New optics repository (**G. Roy & M. Hofer**)
  https://indico.cern.ch/event/1014448/ & FCCIS workshop → Tuesday 30 November 09:00

- pyAT developments for FCC-ee functionalities (**S. White ESRF**)
  FCCIS workshop → Wednesday 6 December 09:00

- FCC-ee tuning studies
  https://indico.cern.ch/event/1097453/

Summary of all FCC and code related activities will be given

(**T. Pieloni**, FCCIS workshop → Thursday 2 December 12:00)

Xdeps

Dependencies manager by R. de Maria

https://indico.cern.ch/event/1088545/

https://github.com/xsuite/xdeps

Xdeps as python dependency manager

MADX expressions are often part of the description of machines and are a very convenient tool to describe circuits, layout constraints and, more in general, dependencies between values.

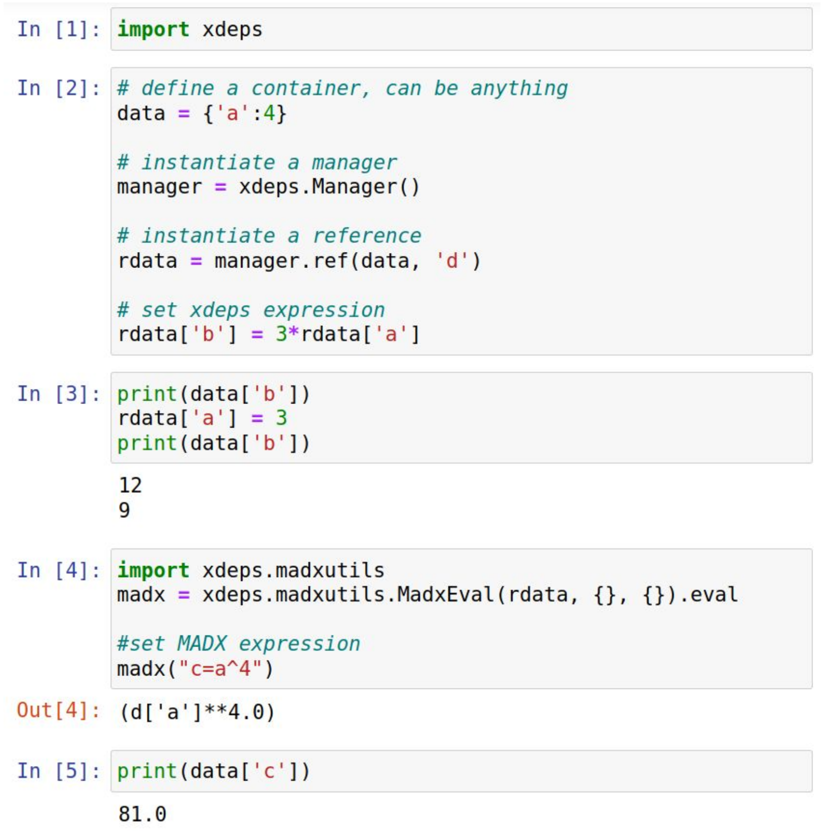
Python does not have built-in capabilities to define deferred expression, however it is possible to implement with a library.

`xdeps` provides a Python implementation of data dependencies management that hinges on:

- dependencies are described as *actions* that modifies output values based on a set (dependent) of input;
- output and input can be any object or part of an object or a container
- actions are triggered by a dependency `manager` (Manager) when one or more input value changes
- actions can be defined using regular python expression on special objects (called `Ref`)

`xdeps` also provides a parser that evaluates MADX syntax expressions that can be used to convert MADX expression to Xdeps actions

Simple example of Xdeps



```
In [1]: import xdeps

In [2]: # define a container, can be anything
        data = {'a':4}

        # instantiate a manager
        manager = xdeps.Manager()

        # instantiate a reference
        rdata = manager.ref(data, 'd')

        # set xdeps expression
        rdata['b'] = 3*rdata['a']

In [3]: print(data['b'])
        rdata['a'] = 3
        print(data['b'])

        12
        9

In [4]: import xdeps.madxutils
        madx = xdeps.madxutils.MadxEval(rdata, {}, {}).eval

        #set MADX expression
        madx("c=a^4")

Out[4]: (d['a']**4.0)

In [5]: print(data['c'])

        81.0
```

1. It already gives expression abilities to python codes like: pyAT and xtrack.

2. Xdeps is still being developed, APIs may change

More examples available:
https://github.com/xsuite/xdeps

Xsequence

Managing lattices and models

https://indico.cern.ch/event/1088545/

https://github.com/fscarlier/xsequence

Codes currently used for FCC-ee optics & tracking studies

Many different codes are currently used for the FCC-ee feasibility studies.

- Conversions and control of lattices between platforms is often complicated
- Managing errors and tuning knobs is complicated between platforms.

SAD	Lattice design
MAD-X	Lattice optimization, optics corrections
Bmad	Spin simulations
Xsuite	Multiparticle tracking simulations, collective effects
pyAT	Tracking and collimation studies, optics
Elegant	Possible use for injection studies

The goal of xsequence

The goal of xsequence is to:

- Simplify lattice conversions to the different codes of interest
- Offer an expandable platform for users to contribute tools for specific conversions
- Simplify the creation of models for the large simulation campaigns by controlling errors and tuning knobs
- Ensure model consistency between platforms for comparative simulations

Xsequence is part of a broader effort in code development

The development of xsequence fits nicely in current efforts of code developments with ABP at CERN in the frame of Xsuite (https://indico.cern.ch/event/1071856/)

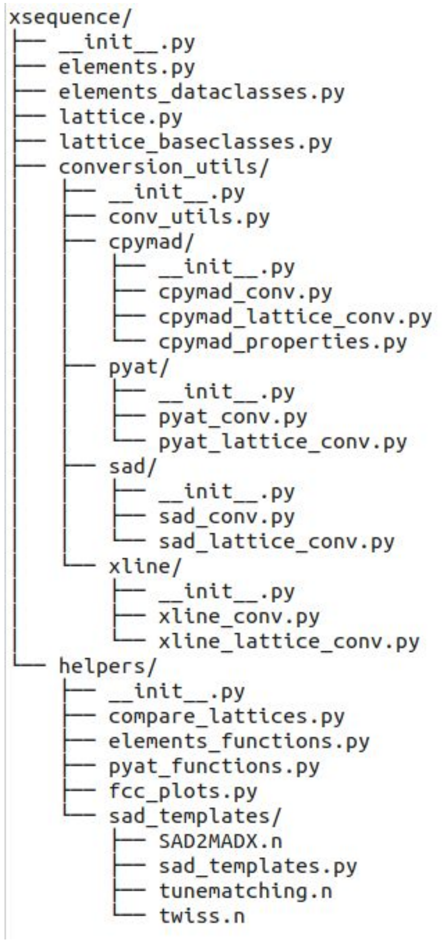
- Allows to bring current code development efforts for the LHC to the FCC-ee community

Xsequence lattice manager

Pypi name has been reserved, and simple pip installation will be available soon.

The package is published and accessible on Github:

https://github.com/fscarlier/xsequence



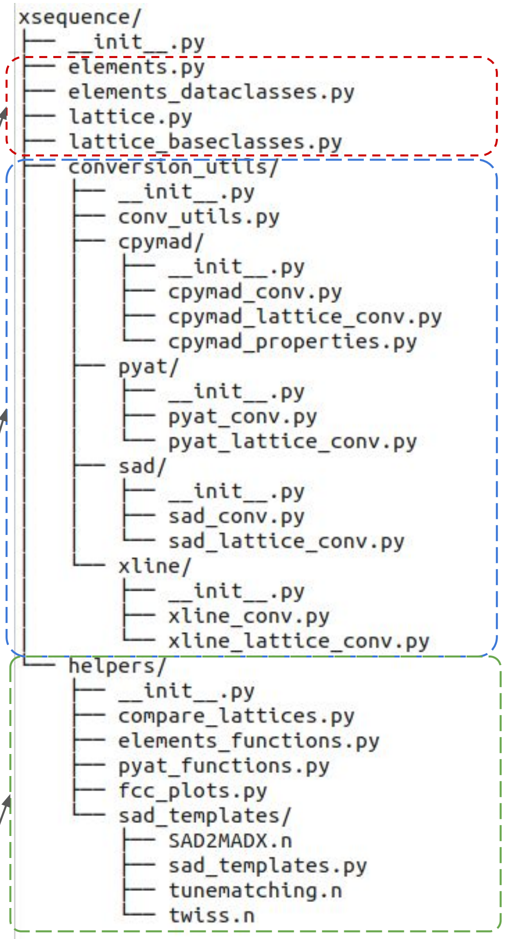
```
xsequence/
├── __init__.py
├── elements.py
├── elements_dataclasses.py
├── lattice.py
├── lattice_baseclasses.py
├── conversion_utils/
│   ├── __init__.py
│   ├── conv_utils.py
│   ├── cpymad/
│   │   ├── __init__.py
│   │   ├── cpymad_conv.py
│   │   ├── cpymad_lattice_conv.py
│   │   └── cpymad_properties.py
│   ├── pyat/
│   │   ├── __init__.py
│   │   ├── pyat_conv.py
│   │   └── pyat_lattice_conv.py
│   ├── sad/
│   │   ├── __init__.py
│   │   ├── sad_conv.py
│   │   └── sad_lattice_conv.py
│   └── xline/
│       ├── __init__.py
│       ├── xline_conv.py
│       └── xline_lattice_conv.py
└── helpers/
    ├── __init__.py
    ├── compare_lattices.py
    ├── elements_functions.py
    ├── pyat_functions.py
    ├── fcc_plots.py
    └── sad_templates/
        ├── SAD2MADX.n
        ├── sad_templates.py
        ├── tunematching.n
        └── twiss.n
```

Xsequence lattice manager

Pypi name has been reserved, and simple pip installation will be available soon.

The package is published and accessible on Github:

https://github.com/fscarlier/xsequence



```
xsequence/
├── __init__.py
├── elements.py
├── elements_dataclasses.py
├── lattice.py
├── lattice_baseclasses.py
├── conversion_utils/
│   ├── __init__.py
│   ├── conv_utils.py
│   ├── cpymad/
│   │   ├── __init__.py
│   │   ├── cpymad_conv.py
│   │   ├── cpymad_lattice_conv.py
│   │   └── cpymad_properties.py
│   ├── pyat/
│   │   ├── __init__.py
│   │   ├── pyat_conv.py
│   │   └── pyat_lattice_conv.py
│   ├── sad/
│   │   ├── __init__.py
│   │   ├── sad_conv.py
│   │   └── sad_lattice_conv.py
│   └── xline/
│       ├── __init__.py
│       ├── xline_conv.py
│       └── xline_lattice_conv.py
└── helpers/
    ├── __init__.py
    ├── compare_lattices.py
    ├── elements_functions.py
    ├── pyat_functions.py
    ├── fcc_plots.py
    └── sad_templates/
        ├── SAD2MADX.n
        ├── sad_templates.py
        ├── tunematching.n
        └── twiss.n
```

Definition of elements and lattice classes and data structures.

Conversion scripts for different codes.

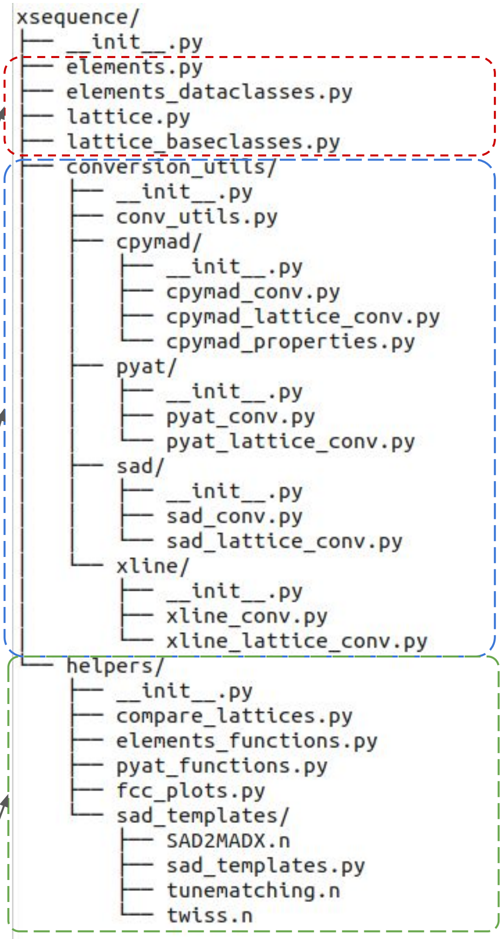
Helper functions for specific tasks.

Xsequence lattice manager

Pypi name has been reserved, and simple pip installation will be available soon.

The package is published and accessible on Github:

https://github.com/fscarlier/xsequence



```
xsequence/
├── __init__.py
├── elements.py
├── elements_dataclasses.py
├── lattice.py
├── lattice_baseclasses.py
├── conversion_utils/
│   ├── __init__.py
│   ├── conv_utils.py
│   ├── cpymad/
│   │   ├── __init__.py
│   │   ├── cpymad_conv.py
│   │   ├── cpymad_lattice_conv.py
│   │   └── cpymad_properties.py
│   ├── pyat/
│   │   ├── __init__.py
│   │   ├── pyat_conv.py
│   │   └── pyat_lattice_conv.py
│   ├── sad/
│   │   ├── __init__.py
│   │   ├── sad_conv.py
│   │   └── sad_lattice_conv.py
│   └── xline/
│       ├── __init__.py
│       ├── xline_conv.py
│       └── xline_lattice_conv.py
└── helpers/
    ├── __init__.py
    ├── compare_lattices.py
    ├── elements_functions.py
    ├── pyat_functions.py
    ├── fcc_plots.py
    └── sad_templates/
        ├── SAD2MADX.n
        ├── sad_templates.py
        ├── tunematching.n
        └── twiss.n
```

Definition of elements and lattice classes and data structures.

Conversion scripts for different codes.

Not included in this directory tree:
- Tests

Helper functions for specific tasks.

Xsequence lattice manager

Pypi name has been reserved, and simple pip installation will be available soon.

The package is published and accessible on Github:

https://github.com/fscarlier/xsequence

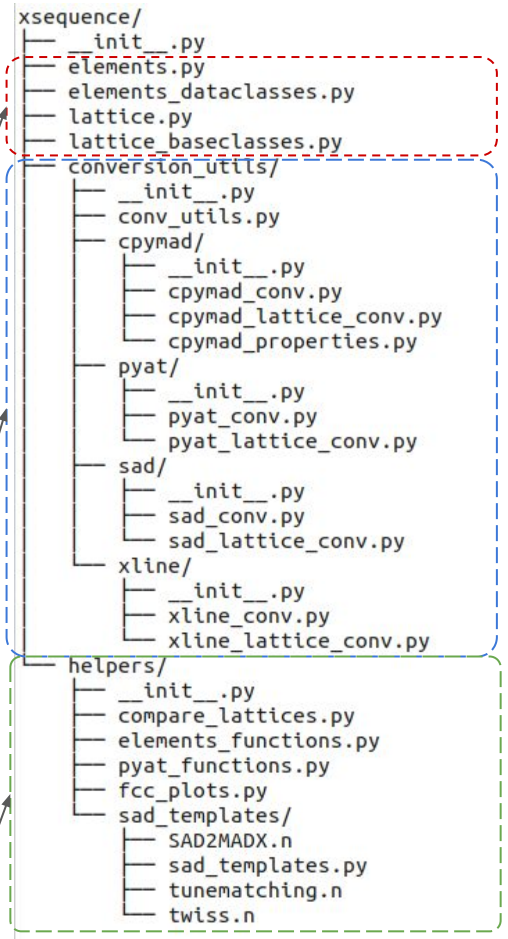
NOTE: The package is still actively developed. Some APIs and aspects may change along the way..

Definition of elements and lattice classes and data structures.

Conversion scripts for different codes.

Helper functions for specific tasks.

Not included in this directory tree:
- Tests



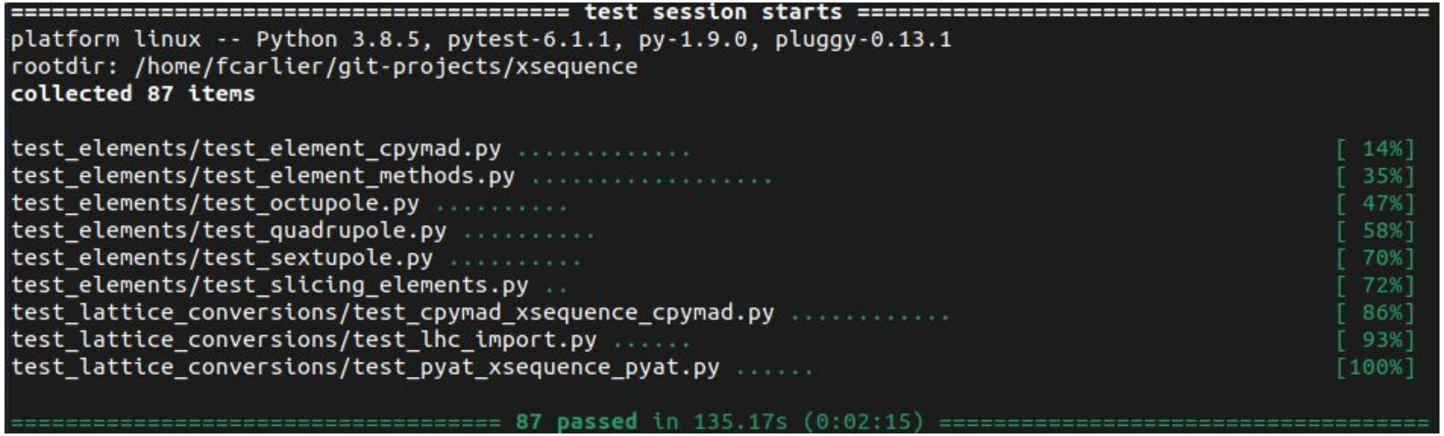
```
xsequence/
├── __init__.py
├── elements.py
├── elements_dataclasses.py
├── lattice.py
├── lattice_baseclasses.py
├── conversion_utils/
│   ├── __init__.py
│   ├── conv_utils.py
│   ├── cpymad/
│   │   ├── __init__.py
│   │   ├── cpymad_conv.py
│   │   ├── cpymad_lattice_conv.py
│   │   └── cpymad_properties.py
│   ├── pyat/
│   │   ├── __init__.py
│   │   ├── pyat_conv.py
│   │   └── pyat_lattice_conv.py
│   ├── sad/
│   │   ├── __init__.py
│   │   ├── sad_conv.py
│   │   └── sad_lattice_conv.py
│   └── xline/
│       ├── __init__.py
│       ├── xline_conv.py
│       └── xline_lattice_conv.py
└── helpers/
    ├── __init__.py
    ├── compare_lattices.py
    ├── elements_functions.py
    ├── pyat_functions.py
    ├── fcc_plots.py
    └── sad_templates/
        ├── SAD2MADX.n
        ├── sad_templates.py
        ├── tunematching.n
        └── twiss.n
```

Test based development of package

Many tests have already been developed to ensure stable element and lattice creation, imports and exports to different platforms.

- Ensures a stable behaviour and easier development moving forward

Further testing will be developed to validate critical functionalities.



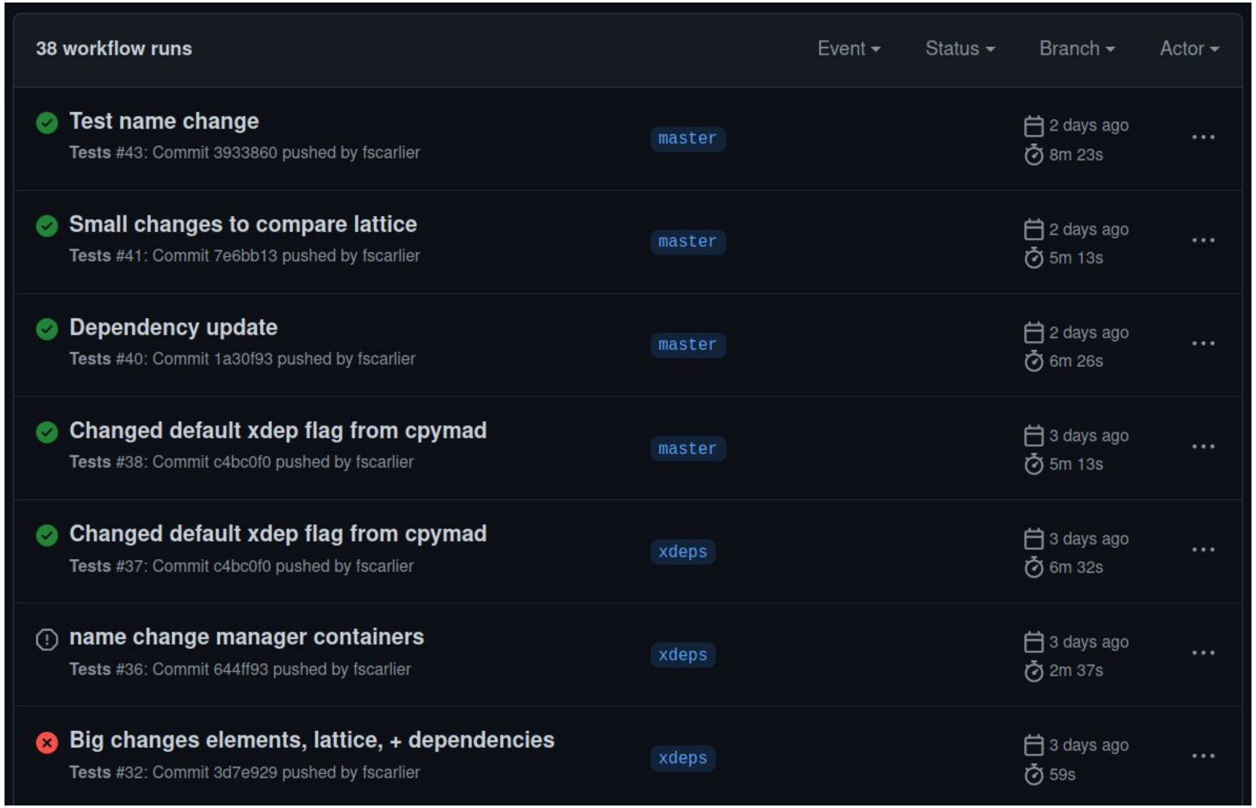
```
========================================= test session starts ==========================================
platform linux -- Python 3.8.5, pytest-6.1.1, py-1.9.0, pluggy-0.13.1
rootdir: /home/fcarlier/git-projects/xsequence
collected 87 items

test_elements/test_element_cpymad.py .............                                               [ 14%]
test_elements/test_element_methods.py ..................                                         [ 35%]
test_elements/test_octupole.py ..........                                                        [ 47%]
test_elements/test_quadrupole.py ..........                                                      [ 58%]
test_elements/test_sextupole.py ..........                                                       [ 70%]
test_elements/test_slicing_elements.py ..                                                        [ 72%]
test_lattice_conversions/test_cpymad_xsequence_cpymad.py ............                            [ 86%]
test_lattice_conversions/test_lhc_import.py ......                                               [ 93%]
test_lattice_conversions/test_pyat_xsequence_pyat.py ......                                      [100%]

==================================== 87 passed in 135.17s (0:02:15) ====================================
```

Automated testing implemented in Github

The tests have been included in automatic workflows on Github.

Pushing new changes will trigger 'github actions' and record the success or failure of tests.

- Improves maintainability
- Ensures stability



38 workflow runs

✓	**Test name change** Tests #43: Commit 3933860 pushed by fscarlier	master	2 days ago 8m 23s
✓	**Small changes to compare lattice** Tests #41: Commit 7e6bb13 pushed by fscarlier	master	2 days ago 5m 13s
✓	**Dependency update** Tests #40: Commit 1a30f93 pushed by fscarlier	master	2 days ago 6m 26s
✓	**Changed default xdep flag from cpymad** Tests #38: Commit c4bc0f0 pushed by fscarlier	master	3 days ago 5m 13s
✓	**Changed default xdep flag from cpymad** Tests #37: Commit c4bc0f0 pushed by fscarlier	xdeps	3 days ago 6m 32s
!	**name change manager containers** Tests #36: Commit 644ff93 pushed by fscarlier	xdeps	3 days ago 2m 37s
✗	**Big changes elements, lattice, + dependencies** Tests #32: Commit 3d7e929 pushed by fscarlier	xdeps	3 days ago 59s

Current imports and exports in xsequence

Interfaces between 'xsequence' and simulations codes currently (partly) exist for

- Cpymad (MADX), pyAT, Xsuite,

  SAD, Bmad, Elegant
- Conversions and further testing are under development

No fixed way to interface with code.

- Directly through python (cpymad, pyat, xline)
- File reader/Writer (SAD, Elegant, Bmad)
- Twiss datatables
- other...

Users can contribute their own converters

- Recently, Elegant was added by **M. Hofer**

Codes	Import	Export
MAD-X (cpymad)	🟩	🟩
SAD	🟩	🟥
pyAT	🟩	🟩
Xsuite	🟥 Trivial	🟩
Bmad	🟥	🟧 testing
Elegant	🟥	🟧 testing

Current imports and exports in xsequence

Interfaces between 'xsequence' and simulations codes currently (partly) exist for

- Cpymad (MADX), pyAT, Xsuite,

  SAD, Bmad, Elegant
- Conversions and further testing are under development

No fixed way to interface with code.

- Directly through python (cpymad, pyat, xline)
- File reader/Writer (SAD, Elegant, Bmad)
- Twiss datatables
- other...

Users can contribute their own converters

- Recently, Elegant was added by **M. Hofer**

Codes	Import	Export
MAD-X (cpymad)	🟩	🟩
SAD	🟩	🟥
pyAT	🟩	🟩
Xsuite	🟥 Trivial	🟩
Bmad	🟥	🟧 testing
Elegant	🟥	🟧 testing

Note: Tilted solenoid treatment is not yet implemented, but different proposals exist

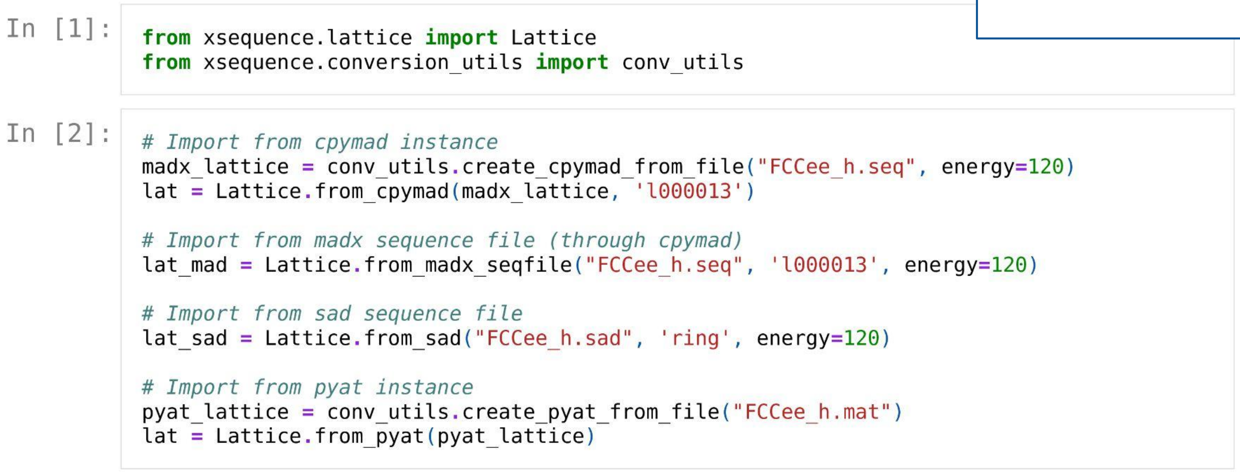
(**H. Burkhart**) https://indico.cern.ch/event/1094574/

(**L. van Riesen-Haupt**) FCCIS workshop → Tuesday 30 November 09:00

Quick example for xsequence

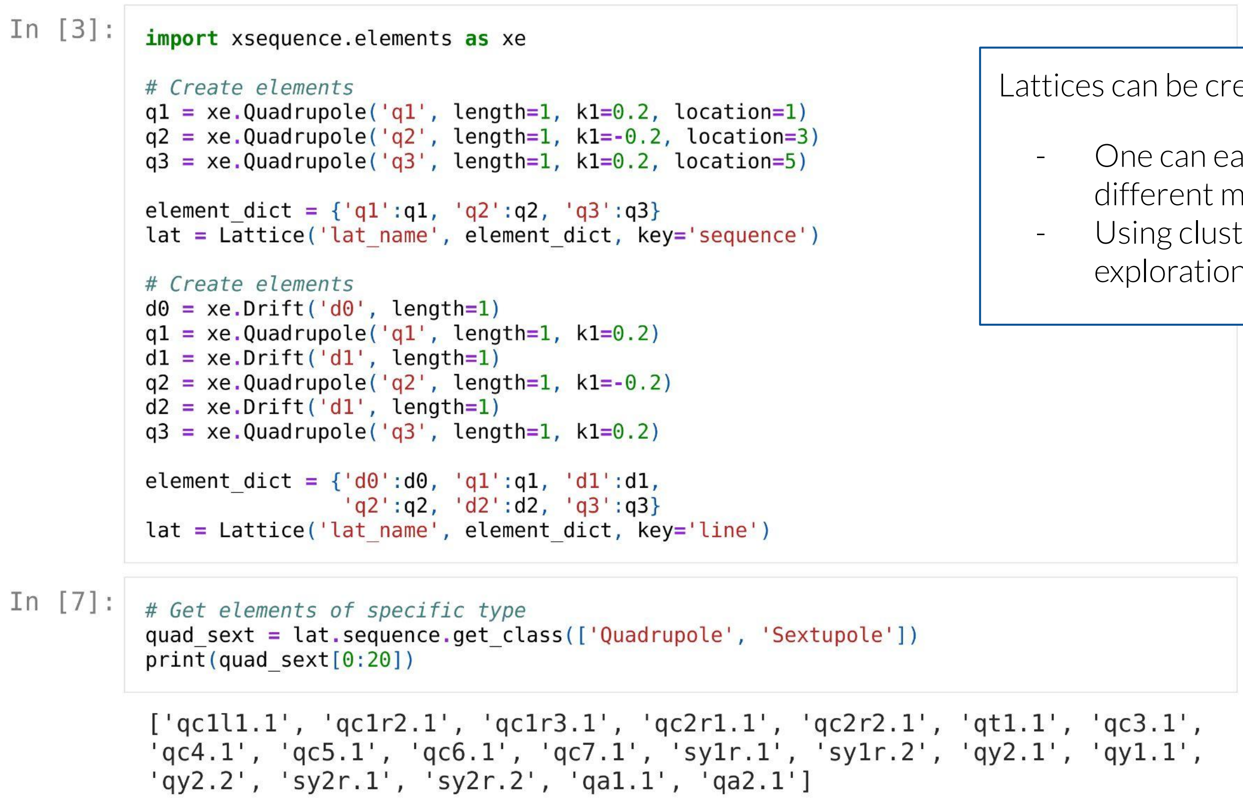
More detailed example is available on Github, or as slides ⟶ https://indico.cern.ch/event/1088545/

Lattice imports from various sources



```
In [1]:  from xsequence.lattice import Lattice
         from xsequence.conversion_utils import conv_utils

In [2]:  # Import from cpymad instance
         madx_lattice = conv_utils.create_cpymad_from_file("FCCee_h.seq", energy=120)
         lat = Lattice.from_cpymad(madx_lattice, 'l000013')

         # Import from madx sequence file (through cpymad)
         lat_mad = Lattice.from_madx_seqfile("FCCee_h.seq", 'l000013', energy=120)

         # Import from sad sequence file
         lat_sad = Lattice.from_sad("FCCee_h.sad", 'ring', energy=120)

         # Import from pyat instance
         pyat_lattice = conv_utils.create_pyat_from_file("FCCee_h.mat")
         lat = Lattice.from_pyat(pyat_lattice)
```

Lattice creation in python



```
In [3]:
import xsequence.elements as xe

# Create elements
q1 = xe.Quadrupole('q1', length=1, k1=0.2, location=1)
q2 = xe.Quadrupole('q2', length=1, k1=-0.2, location=3)
q3 = xe.Quadrupole('q3', length=1, k1=0.2, location=5)

element_dict = {'q1':q1, 'q2':q2, 'q3':q3}
lat = Lattice('lat_name', element_dict, key='sequence')

# Create elements
d0 = xe.Drift('d0', length=1)
q1 = xe.Quadrupole('q1', length=1, k1=0.2)
d1 = xe.Drift('d1', length=1)
q2 = xe.Quadrupole('q2', length=1, k1=-0.2)
d2 = xe.Drift('d1', length=1)
q3 = xe.Quadrupole('q3', length=1, k1=0.2)

element_dict = {'d0':d0, 'q1':q1, 'd1':d1,
                'q2':q2, 'd2':d2, 'q3':q3}
lat = Lattice('lat_name', element_dict, key='line')
```

```
In [7]:
# Get elements of specific type
quad_sext = lat.sequence.get_class(['Quadrupole', 'Sextupole'])
print(quad_sext[0:20])
```

```
['qc1l1.1', 'qc1r2.1', 'qc1r3.1', 'qc2r1.1', 'qc2r2.1', 'qt1.1', 'qc3.1',
'qc4.1', 'qc5.1', 'qc6.1', 'qc7.1', 'sy1r.1', 'sy1r.2', 'qy2.1', 'qy1.1',
'qy2.2', 'sy2r.1', 'sy2r.2', 'qa1.1', 'qa2.1']
```

Lattices can be created in python too.

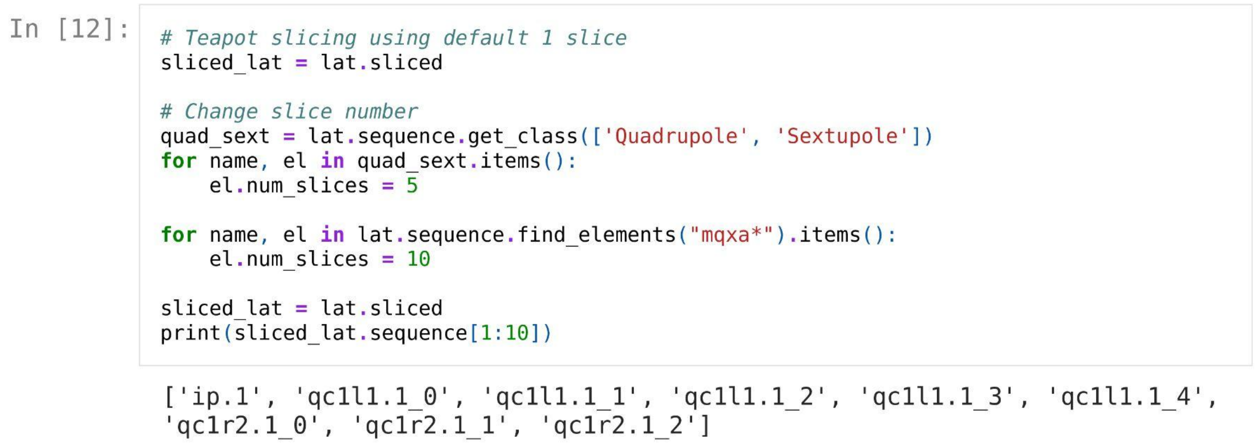
- One can easily change sections and adapt different models
- Using clusters like HTCondor could speed up exploration of different models

Lattice slicing available

Slicing algorithms have been implemented. This is necessary for certain codes, i.e. xtrack.

- Conversions of complicated elements to unsupported codes can be done as slices

Currently contains 'teapot' and 'uniform' slicing



```
In [12]:
# Teapot slicing using default 1 slice
sliced_lat = lat.sliced

# Change slice number
quad_sext = lat.sequence.get_class(['Quadrupole', 'Sextupole'])
for name, el in quad_sext.items():
    el.num_slices = 5

for name, el in lat.sequence.find_elements("mqxa*").items():
    el.num_slices = 10

sliced_lat = lat.sliced
print(sliced_lat.sequence[1:10])
```

```
['ip.1', 'qc1l1.1_0', 'qc1l1.1_1', 'qc1l1.1_2', 'qc1l1.1_3', 'qc1l1.1_4',
'qc1r2.1_0', 'qc1r2.1_1', 'qc1r2.1_2']
```

Exports to different platforms

Lattice exports are currently limited to:

cpymad	Obtain a cpymad instance of the model
pyAT	Create a pyat Lattice() instance of the model
Xsuite	Create a python xtrack Line() instance
Bmad	Create sequence file (Under testing)
Elegant	Create sequence file (Under testing)

Further efforts will be made on integrating already available tools to expand access

In [17]:

```
madx = lat.to_cpymad()
pyat = lat.to_pyat()
line = lat.to_xline()
```

Conclusions

- Xsequence offers a first step to a solid platform to interface different simulation codes and centralize lattice conversions.
- In combination with ‘xdeps’ dependency management it is a powerful tool to manipulate models and simplify simulation campaigns.
- Xsequence is built to be user extendible. Any user can bring new functionalities or conversions to the platform
- Actively being developed, so changes to APIs can still be expected

Next steps

- Include magnetic error and misalignment definitions and conversions to codes.
- Integrate existing tools for further lattice conversions.
- Explore use of xsequence for lattice version control
- Start first uses and community testing.

Feel free!

Ideas, discussion, suggestions?

Also time for further discussion this afternoon in 6/R-012