SARDANA WORKSHOP 2025

# Continuous Energy Scanning at MAX IV:
## Scaling and Adapting Across Beamlines

**Lin Zhu, Áureo Freitas, Benjamin Bertrand | SOFTWARE**

## Continuous Energy Scan

- Beamlines Overview
- Continuous Scan vs Step Scan
- Parametric Trajectory motion

## System Implementation

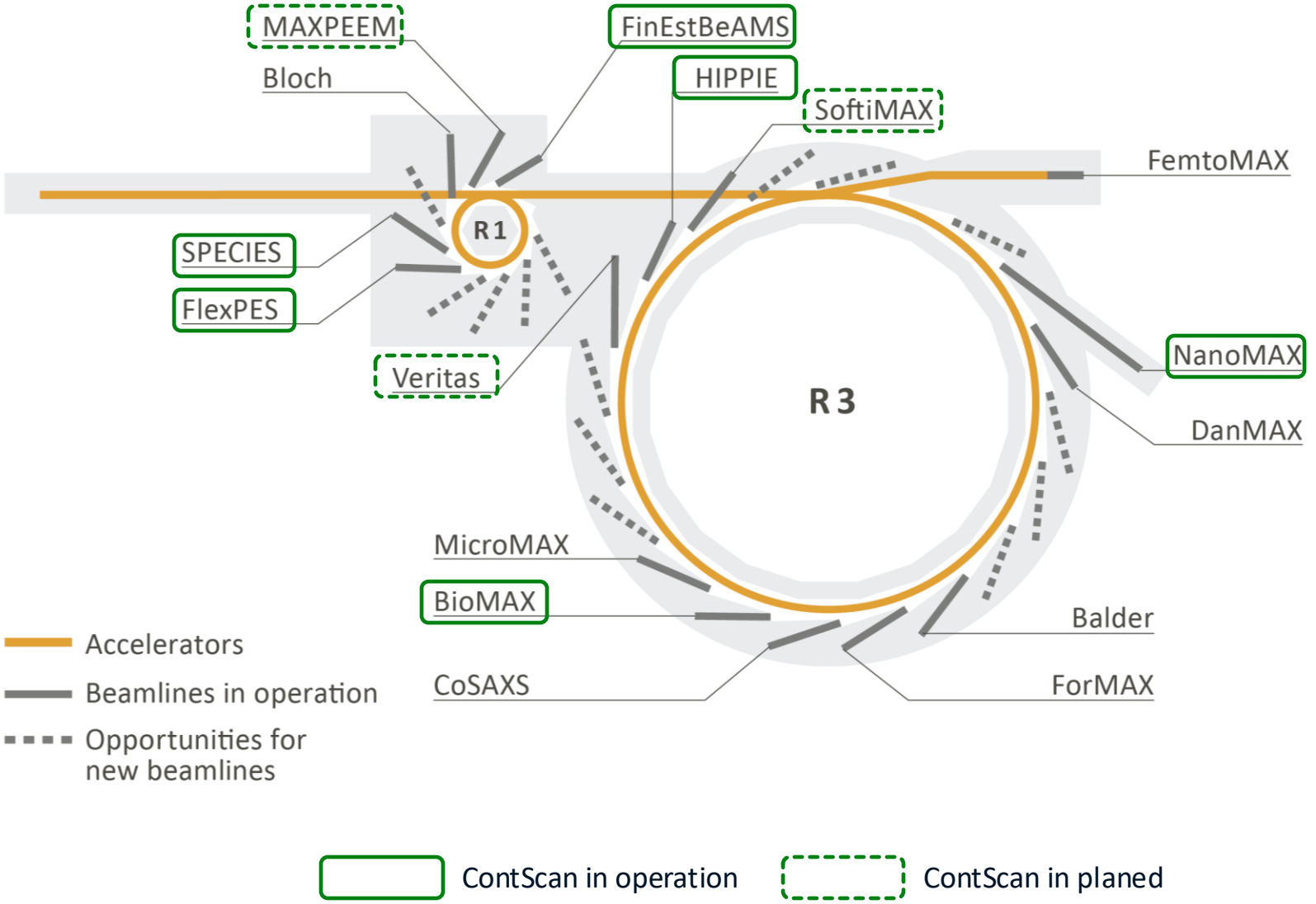
- Synchronization
- Position Based and Time Constant Based Trigger Generation
- Scan procedure

## Beamlines Use Case

- FlexPES, BioMAX, FinESTBeAMS etc.

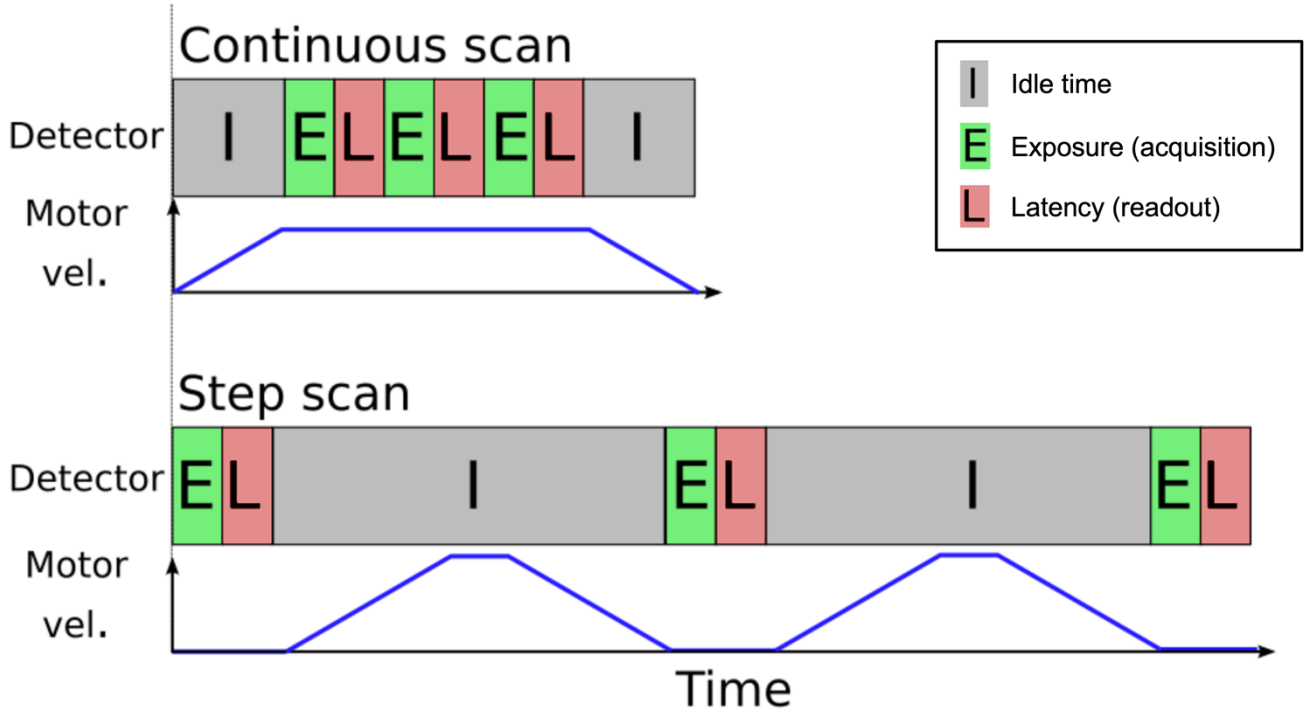
# Beamlines Overview

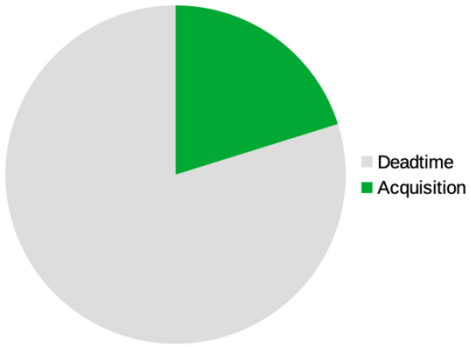
- 9/16 beamlines (6 in operation, 3 planed)
- Both soft and hard X-ray
- R1 (1.5 GeV) and R3 (3 GeV)



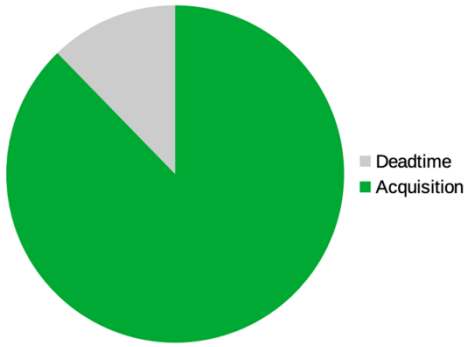
# Continuous Energy Scan

- Reduce the duration of experiments or increase the amount of data collected.
- Compared to step scan, in continuous scans acquire the data while the motors are continuously moving.
- Measurements are triggered as soon as the correct positions are reached and made while the motor is still moving (position-based).
- It is possible with detectors that have short exposure and integration times.



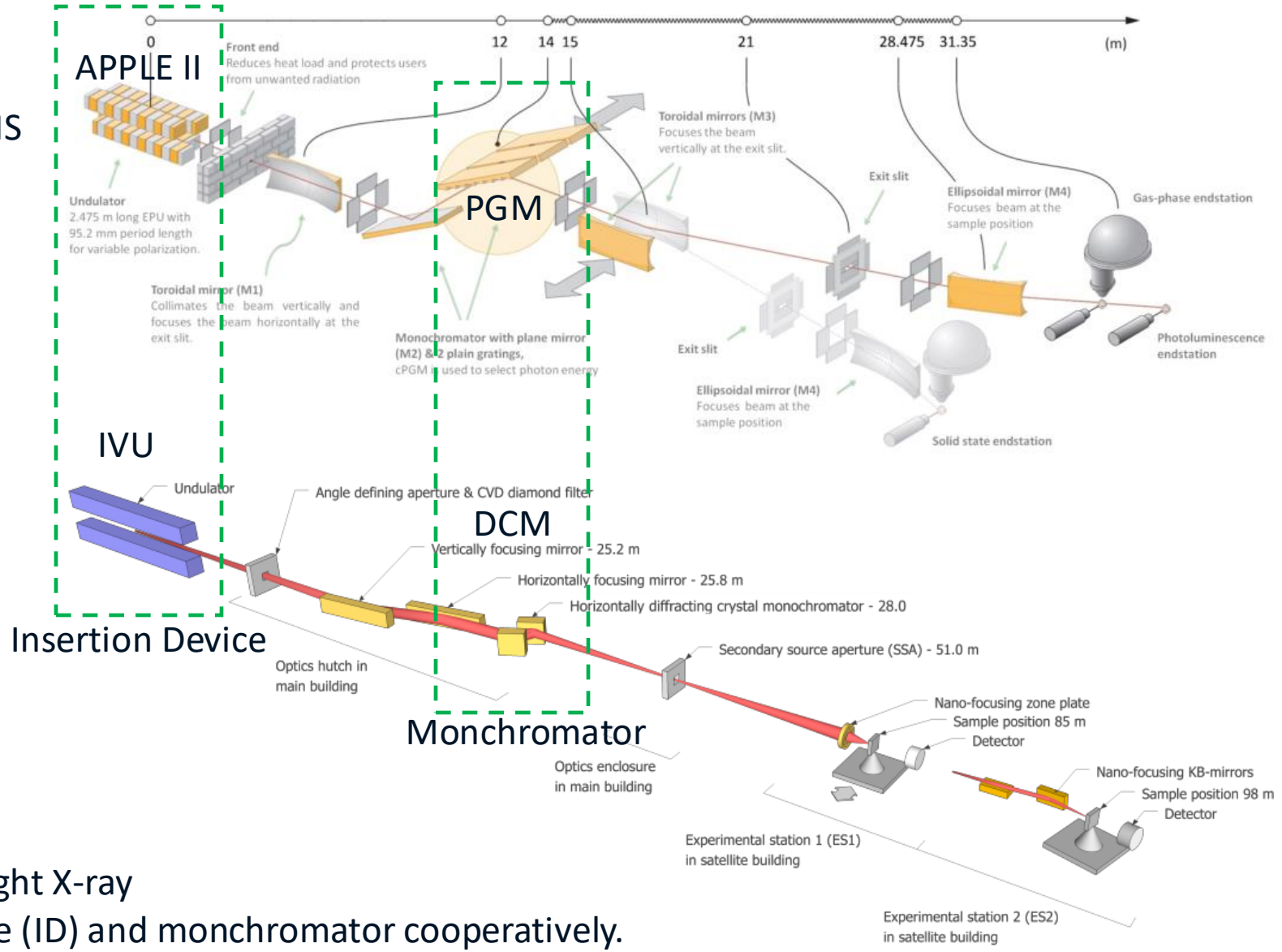

Step scan
248 seconds




Continuous scan
57 seconds

FinESTBEAMS
(R1)

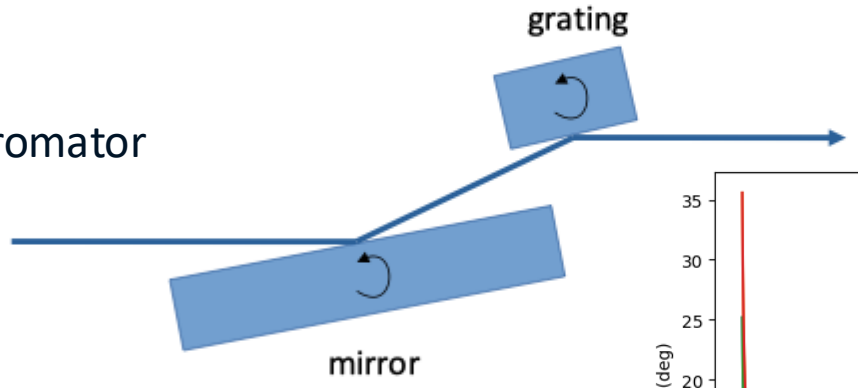
NanoMAX
(R3)



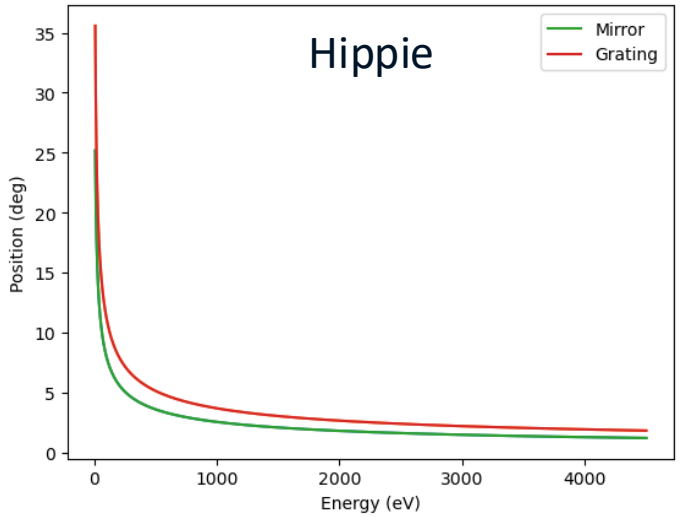

- Tunable and extremely bright X-ray
- Move both insertion device (ID) and monchromator cooperatively.

# Control Energy of X-ray

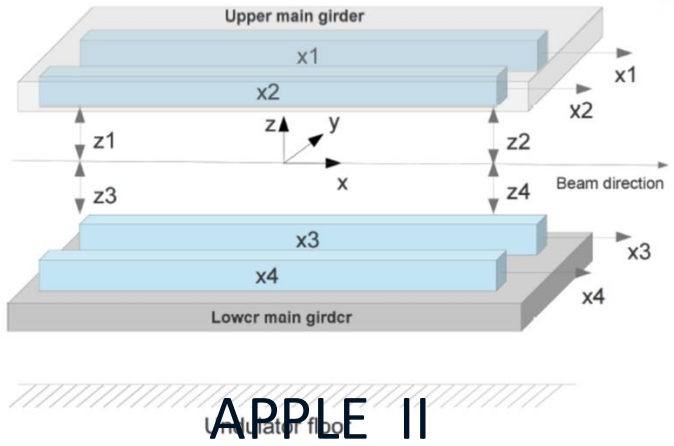
Monchromator



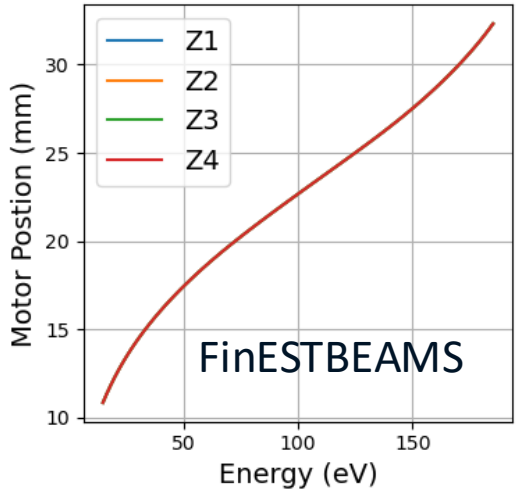

PGM




Insertion Device
(only gap motion)




APPLE II



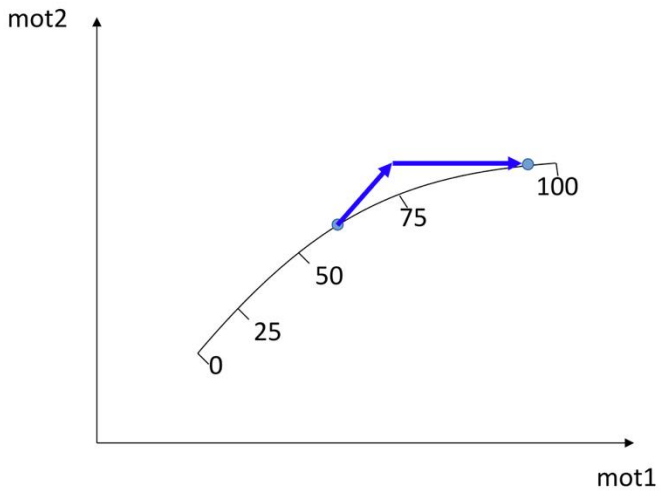

In Low Energy Range

# Conbined Motion Non-linear Path

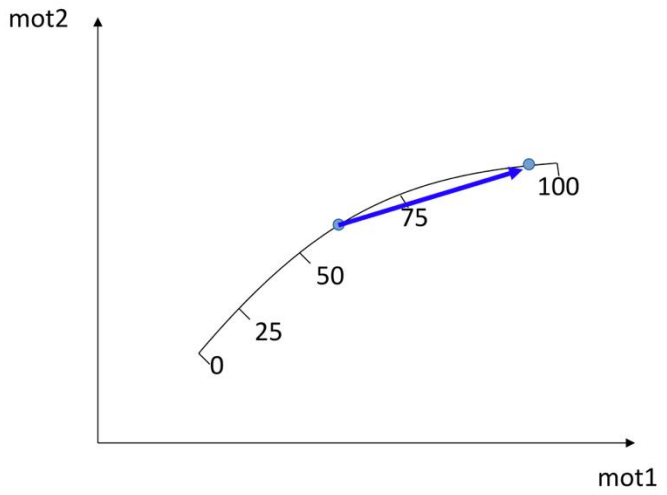
- Energy (scaned axis) is a combined motion
- Each coupled axis can move in non-linear path
- Parametric trajectory mode of the IcePAP

# Icepap Parametric Trajectory Mode

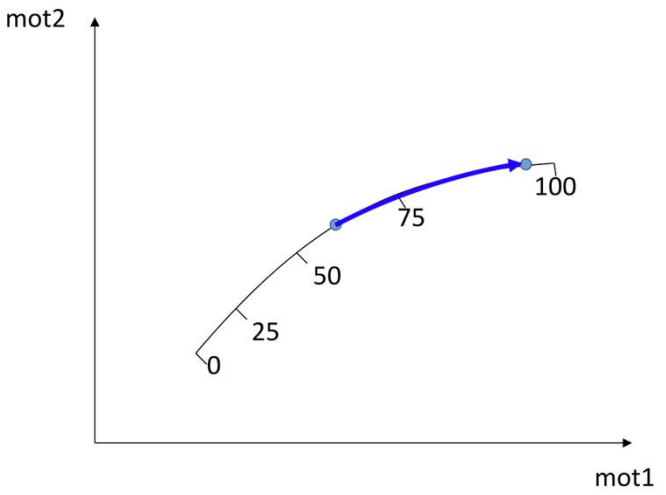
### Nominal Velocities




### Matched Velocities

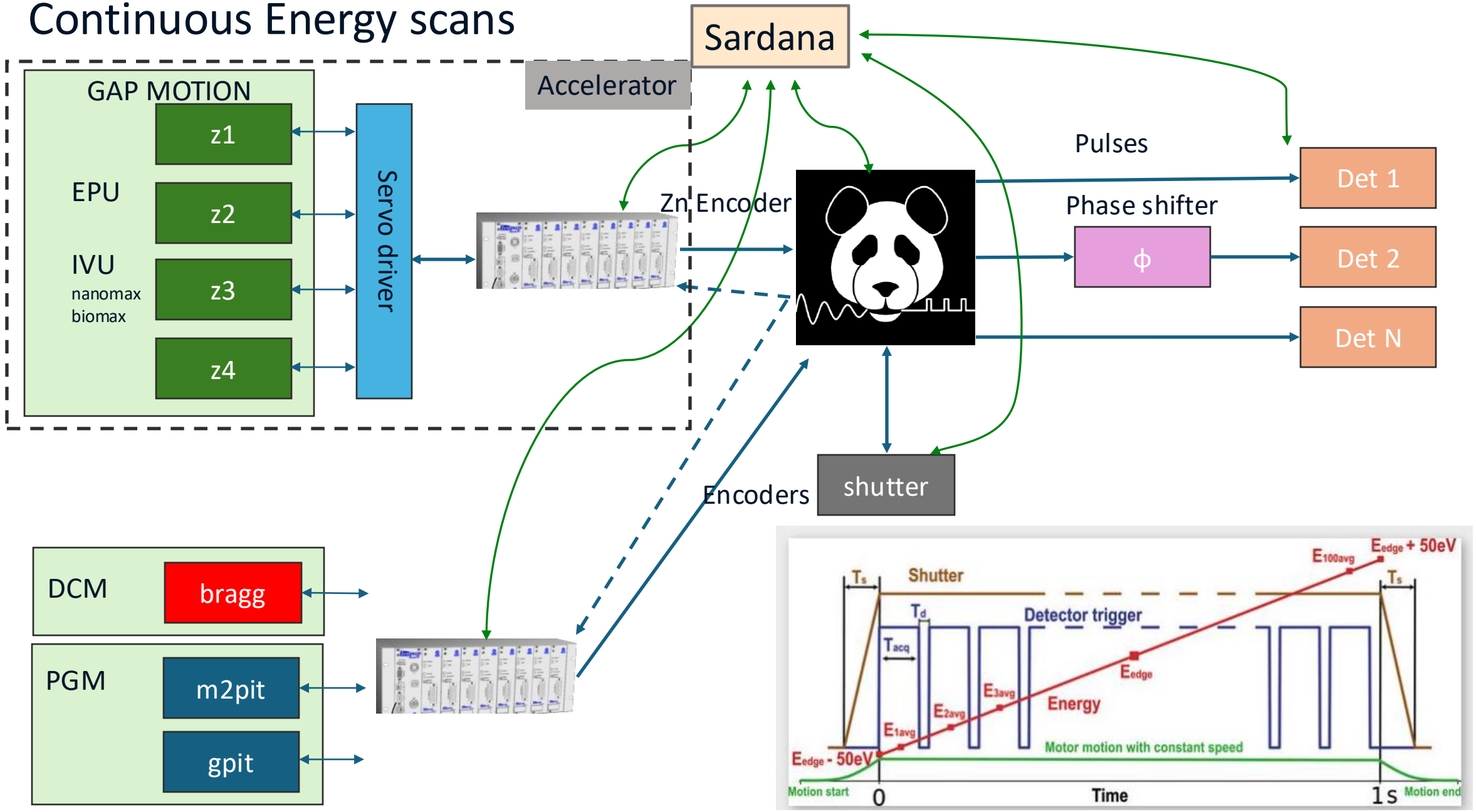



### Parametric Trajectory




- It is possible to make non-linear movements in IcePAP keeping a constant value in eV/s.
- The trajectory tables are calculated by a sardana controller and inserted in the IcePAP driver.
- The IcePAP uses an interpolation method to follow the trajectory.
- Energy range and the number of points in the trajectory table are configurable.

# Continuous Energy scans



# MotorController

Name

**sardana-icepaptrajctrl**

Host

**b-v-veritas-ec-4**
**b-v-species-ec-4**
**b-v-nanomax-ec-0**
**b-v-hippie-ec-3**
**b-v-flexpes-ec-4**
**b-v-finest-ec-2**
**b-v-balder-ec-0**
**g-v-ec-42**
**g-v-ec-39**

Open Conda panel 🔗

Open RPM panel 🔗

Generated by https://nox.apps.okd.maxiv.lu.se

Number of hosts

```
class IPAPTrajCtrl(MotorController):
    """
    This class implements the base functionality of a trajectory motor controller.

    During init, the controller creates the tables for motor positions versus trajectory unit.
    In order to do this, it needs the values of all attributes as well as
    the parameters for the underlying IcePAP motors.
    To ensure that this information is available, the attributes are set as MemorizedNoInit,
    meaning that Tango will not write the memorized values after init.
    Instead the values are read from the database during the init procedure.
```

```
class FinestPGMTrajCtrl(IPAPTrajCtrl):
    axis_attributes = IPAPTrajCtrl.addAttributes(
        {
            "Cff": {
                Type: float,
                DefaultValue: 2.25,
                Access: DataAccess.ReadWrite,
                FGet: "getCff",
                FSet: "setCff",
                Memorize: MemorizedNoInit,
            },
            "LineDensity": {
                Type: float,
                DefaultValue: 300.0,
                Access: DataAccess.ReadWrite,
                FGet: "getLineDensity",
                FSet: "setLineDensity",
                Memorize: MemorizedNoInit,
            },
            "DiffrOrder": {
                Type: float,
                DefaultValue: 1.0,
                Access: DataAccess.ReadWrite,
                FGet: "getDiffrOrder",
                FSet: "setDiffrOrder",
                Memorize: MemorizedNoInit,
            },
            "OffsetGr": {
                Type: float,
                DefaultValue: 0.0,
                Access: DataAccess.ReadWrite,
```

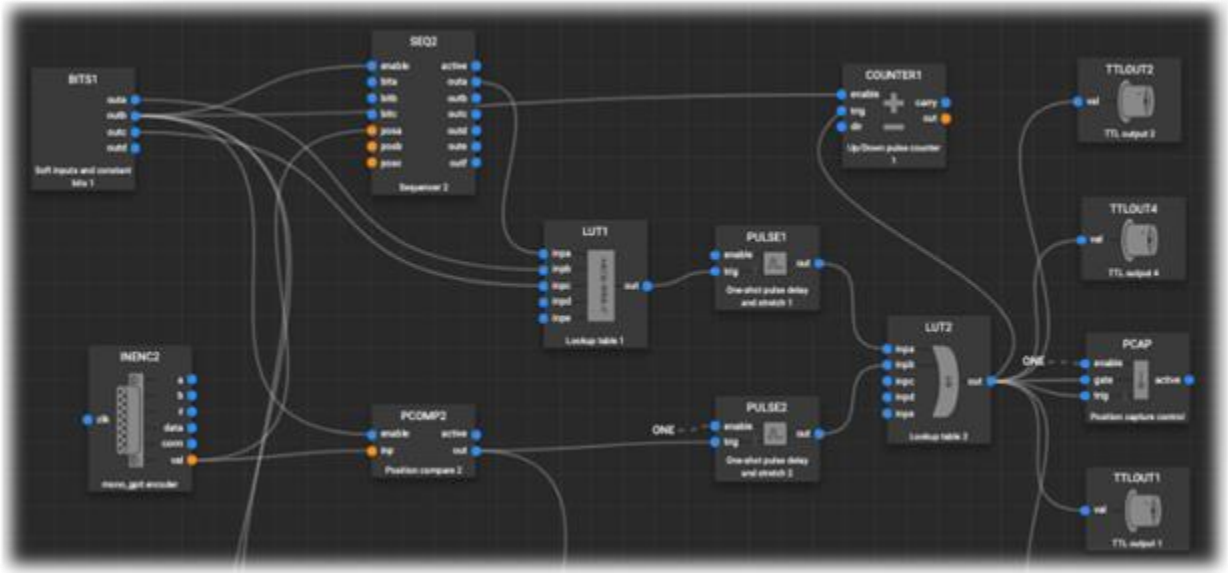
```
class FinestGapTrajCtrl(IPAPTrajCtrl):
    axis_attributes = IPAPTrajCtrl.addAttributes(
        {
            "Harmonic": {
                Type: int,
                DefaultValue: 1,
                Access: DataAccess.ReadWrite,
                FGet: "getHarmonic",
                FSet: "setHarmonic",
                Memorize: MemorizedNoInit,
            },
        }
    )

    extra_properties = {
        "Polynomials": {
            Type: "tango.DevVarStringArray",
            DefaultValue: ["1:1,0"],
            Description: "Polynomials, one harmonic per row and
        },
        "EnergyRanges": {
            Type: "tango.DevVarStringArray",
            DefaultValue: ["1:100,300"],
            Description: "Energy ranges in eV, one harmonic per
        },
        "GapOffsetMotor": {
            Type: str,
```

# TriggerGateController

```python
class FinestPandaBoxTriggerGateCtrl(TriggerGateController):
    """
    TriggerGateController to control Panda Box at FlexPES.
    """

    organization = "MAX IV"
    gender = "TriggerGate"
    model = "Panda Box"
```

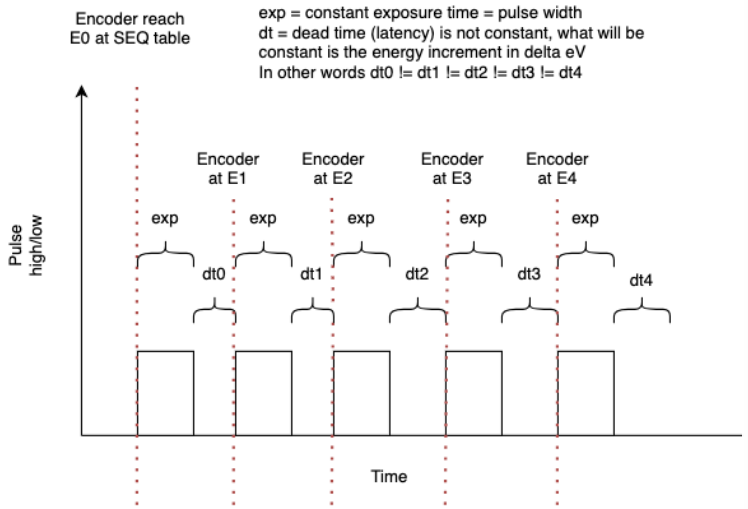



```python
    @debug_it
    @handle_error(msg="Unable to Configure PandaBox")
    def SynchOne(self, axis, configuration):
        group = configuration[0]
        # number of points
        num_points = group[SynchParam.Repeats]
        # integration time in time domain
        int_time = group[SynchParam.Active][SynchDomain.Time]
        # tatal time in time domain: total_time = int_time + latency_time
        total_time = group[SynchParam.Total][SynchDomain.Time]
        if (
            not self.EncInUse
            and self.TriggerDomain.upper() == "TIME"
            and not self._HWStart
        ):
            # configuring PULSES blocks
            self.pandabox.send(Put("PULSE1.PULSES", "{}".format(num_points)))
            self.pandabox.send(Put("PULSE1.STEP", "{}".format(total_time)))
            self.pandabox.send(Put("PULSE1.ENABLE", "ONE"))
            self.pandabox.send(Put("PULSE2.ENABLE", "ZERO"))
        elif (
            not self.EncInUse
            and self.TriggerDomain.upper() == "POSITION"
            and not self._HWStart
        ):
            raise Exception(
                "Encoder not in use and position mode is active \
                        \nOperation not supported!"
            )
        else:
```

- Support postion and time-based synchronism

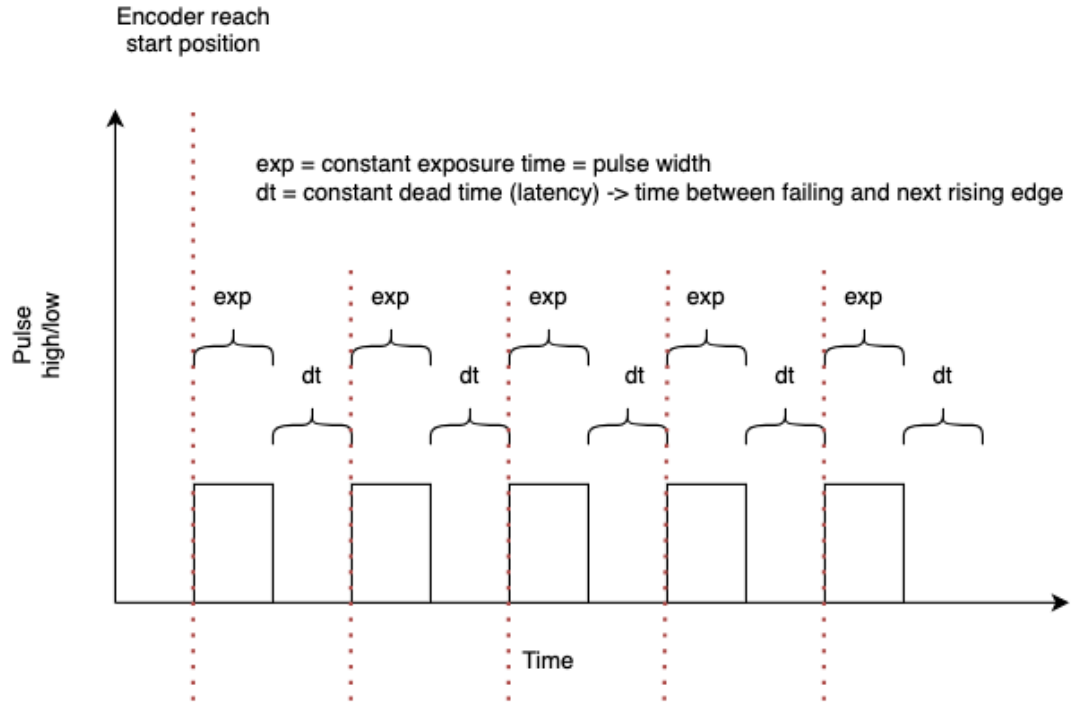
# Position Based Synchronism

- Sequence tables are used to define the positions of the motors in which the detectors should be triggered.
- Sardana calculates the sequence tables and configure them in PandaBox in the beginning of the scan.
- The sequence positions in the table are yielded one after the other by a sequencer.
- The trigger is sent to the detectors when the encoder position reaches the value given by the sequencer.

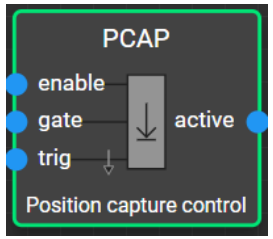



# Time Constant Based Synchronism

- Only the initial position value of the scan is configured in PandaBox.
- When the encoders reach that initial position, PandaBox generates a burst of N triggers with *exp* length and *dt* spacing between them.
- This approach trusts that the IcePAP motors will have a constant velocity in eV/s



# CountTimerController




| NAME | VALUE | UNITS | CAPTURE |
|---|---|---|---|
| COUNTER2.OUT | | | Value |
| COUNTER3.OUT | | | Value |
| COUNTER4.OUT | | | No |
| COUNTER5.OUT | | | No |
| COUNTER6.OUT | | | No |
| COUNTER7.OUT | | | No |
| COUNTER8.OUT | | | No |
| FILTER1.OUT | | | No |
| FILTER2.OUT | | | No |
| INENC1.VAL | | | Min Max Mean |
| INENC2.VAL | | | Min Max Mean |
| INENC3.VAL | | | Min Max Mean |
| INENC4.VAL | | | No |
| PGEN1.OUT | | | No |
| PGEN2.OUT | 1.00000000 | 0.00000000 | No |

```
class FinestPGMPCAPCoTiCtrl(CounterTimerController):
    """
    This class is a Tango Sardana CounterTimerController fo
    PandaBox PCAP block.
    """

    organization = "MAX IV"
    gender = "CounterTimerController"
    model = "Panda Box"

    ctrl_properties = {
        "pandabox_hostname": {
            Type: str,
            Description: "PandaBox Hostname",
            DefaultValue: "b-finest-pandabox-0",
        },
        "mirror_motor_alias": {
            Type: str,
            Description: "mirror motor",
            DefaultValue: "mono_m2pit",
        },
        "grating_motor_alias": {
            Type: str,
            Description: "grating motor",
            DefaultValue: "mono_gpit",
        },
        "energy_motor_alias": {
            Type: str,
            Description: "mono energy motor",
            DefaultValue: "mono_energy_traj",
        },
        "energy_ctrl_alias": {
            Type: str,
            Description: "mono energy controller",
            DefaultValue: "mono_traj_ctrl",
        },
```

```
    def ReadOne(self, axis):
        if self.new_data is None:
            return []
        if axis == 1:
            return self.data_energy_min
        elif axis == 2:
            return self.data_energy_max
        elif axis == 3:
            return self.data_energy_av
        elif axis == 4:
            return self.data_cff_min
        elif axis == 5:
            return self.data_cff_max
        elif axis == 6:
            return self.data_cff_av
        elif axis == 7:
            return self.data_idenc_min
        elif axis == 8:
            return self.data_idenc_max
        elif axis == 9:
            return self.data_idenc_av
        elif axis == 10:
            return self.data_idmm_min
        elif axis == 11:
            return self.data_idmm_max
        elif axis == 12:
            return self.data_idmm_av
        elif axis == 13:
            return self.data_genc_min
        elif axis == 14:
            return self.data_genc_max
        elif axis == 15:
            return self.data_genc_av
        elif axis == 16:
            return self.data_m2enc_min
        elif axis == 17:
            return self.data_m2enc_max
        elif axis == 18:
            return self.data_m2enc_av
        elif axis == 19:
            return self.data_idenergy_av
        elif axis == 20:
            return self.data_diff_enrgy
        elif axis == 21:
            return self.data_counter2
        elif axis == 22:
            return self.data_counter3
        elif axis == 23:
            return self.data_gapmm_av
        else:
            return []
```

# Macro Set

- Pre-scan hooks:

```
Door_FinEst [1]: %lsgh
   Hook place                                          Hook(s)
   Hook place                                          Hook(s)
 ------------ ------------------------------------------------
    post-scan                        hook_newton b112a/ea/spc-cam-01
     pre-scan   set_moveable_timeout energy_circular_fa 12000
                        set_moveable_timeout energy_fa 12000
                        set_moveable_timeout id_energy 12000
                         configure_triggergate pandabox ctrl
                                              check_panda_schema
```

- Supportive macros:

```
7   class configure_panda_layout(Macro):
8       param_def = [
9           [
10              "schema_filename",
```

```
class sync_id_poly(BlAccMixin, Macro):
    """
    Sardana macro to sync the undulator Polynomials and EnergyRanges properties
    between accelerator and beamline sardana environments.
    """
```

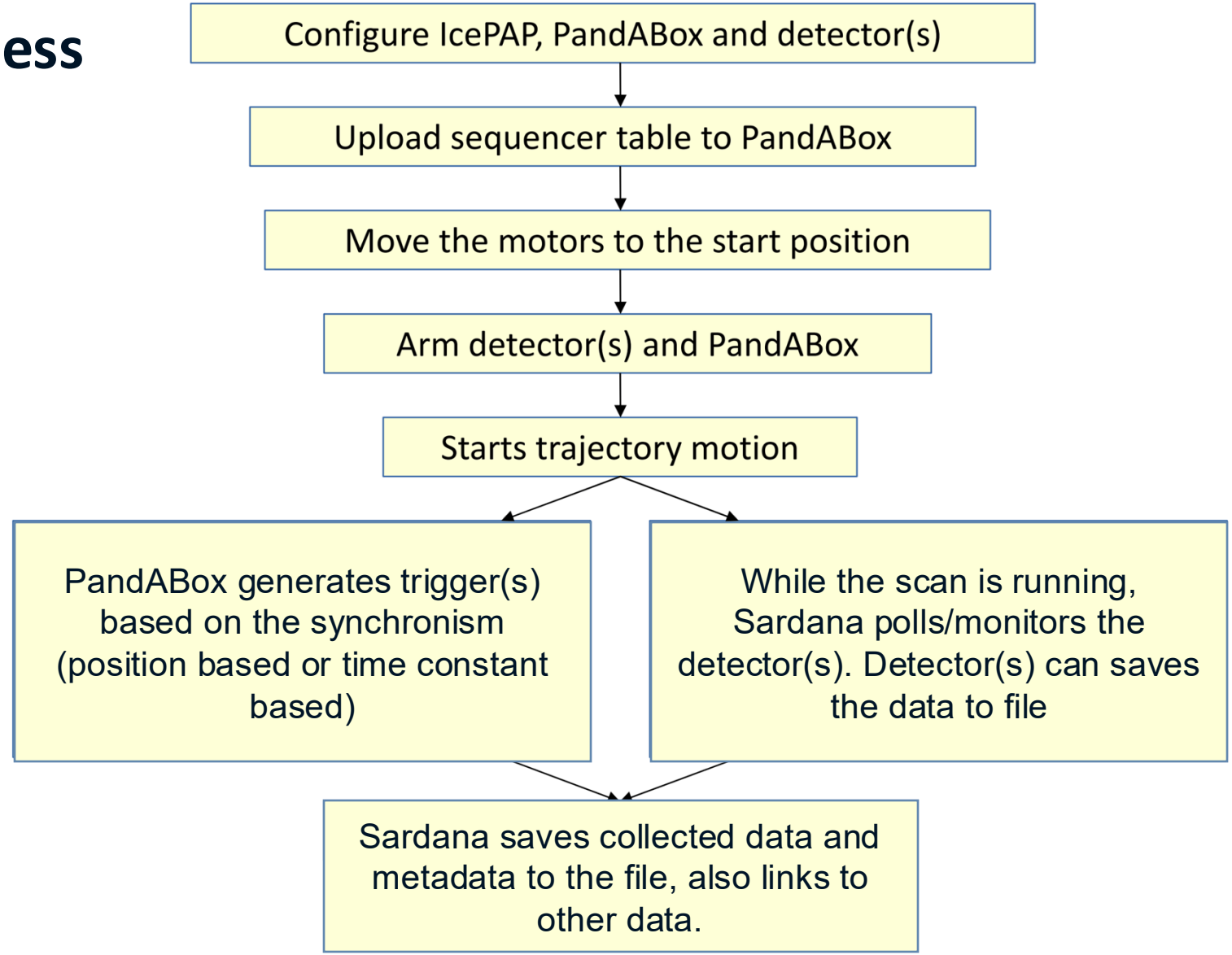
```
class check_id_poly(BlAccMixin, Macro):
    """
    Sardana macro to check if the polynomial properties are synced between beamline and a
    """
```

- Continuous energy scan macros:

```
512 class FinestCtScan_mono(CtScanMixin, Macro):
513     """A continuous scan for the mono energy only"""
514
515     def run(self, start_pos, final_pos, nr_interv, integ_time, latency_time, **
516         self.check_trigger_domain(nr_interv)
```

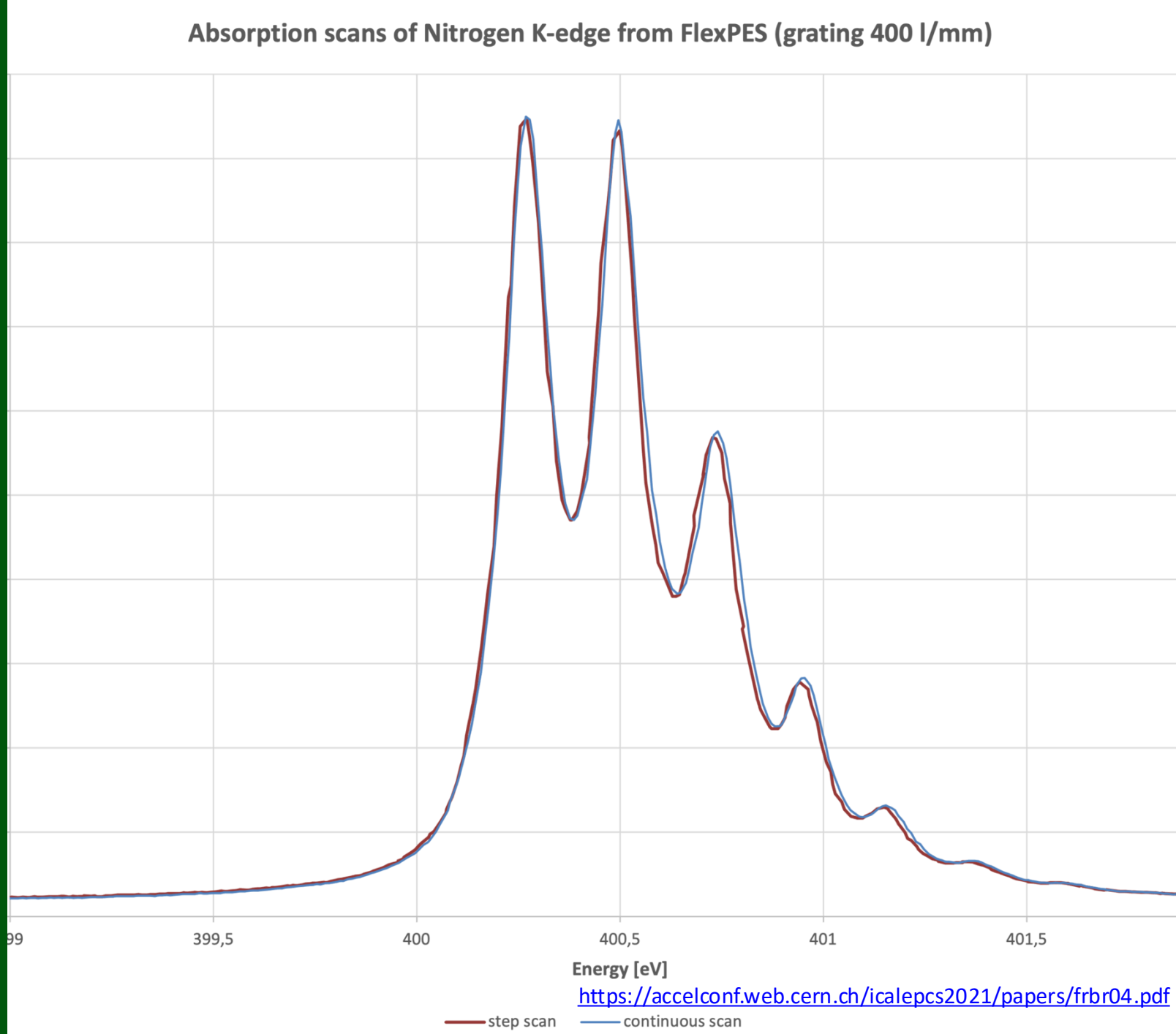
```
366 class FinestCtTrajScan(CtScanMixin, Macro):
367     """A continuous scan for the mono and id energy trajectories"""
368
369     def delete_motor_group(self, motors):...
376
377     def prepare_run(self, start_pos, final_pos, nr_interv, integ_time, latency_time):...
423
424     def run(self, start_pos, final_pos, nr_interv, integ_time, latency_time, **opts):
425         self.prepare_run(start_pos, final_pos, nr_interv, integ_time, latency_time)
426         try:
427             self.a2scanct(
428                 MONO_ENERGY_TRAJ_MOTOR,
429                 start_pos,
430                 final_pos,
431                 ID_ENERGY_TRAJ_MOTOR,
432                 start_pos,
433                 final_pos,
434                 nr_interv,
435                 integ_time,
436                 latency_time,
437             )
438         finally:
439             # Always restore the ID velocity
440             self.info(f"Restoring the ID velocity to {ID_NOMINAL_VELOCITY}")
441             self.undulator_gap_proxy.Velocity = ID_NOMINAL_VELOCITY
442             # Delete motor groups
443             # Bug in 3.4.0: the first motor group might stay in MOVING
444             # if the state of the mono_energy is updated due to a client read
445             # when mono_energy_traj is moving
446             # See https://gitlab.com/sardana-org/sardana/-/issues/1873
447             self.delete_motor_group((MONO_ENERGY_PSEUDO_MOTOR, ID_ENERGY_PSEUDO_MOTOR))
448             self.delete_motor_group((MONO_ENERGY_TRAJ_MOTOR, ID_ENERGY_TRAJ_MOTOR))
```

# Scanning Process



# FlexPES

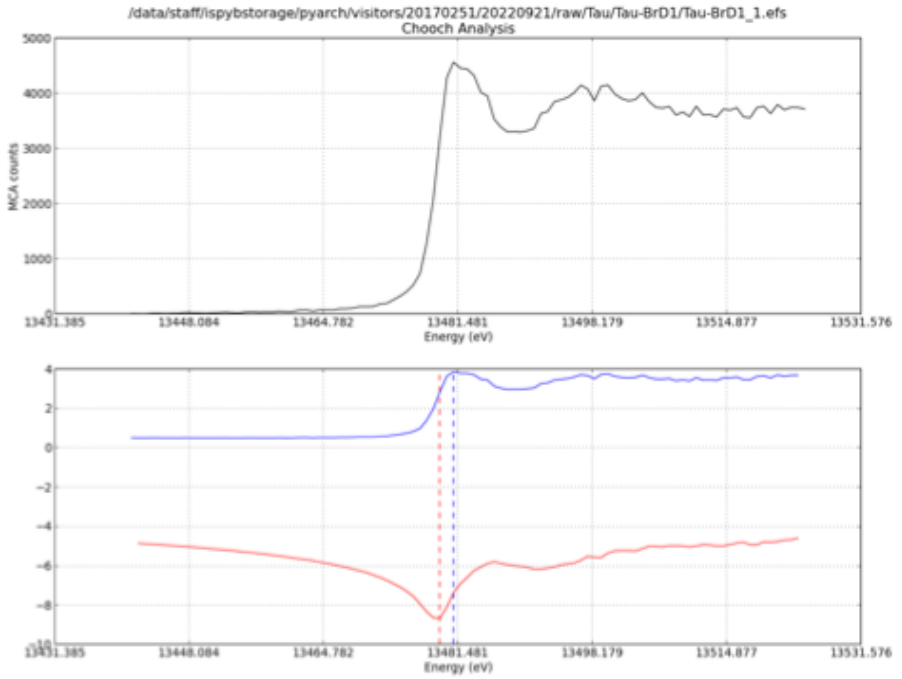
- The FlexPES beamline can do continuous scans of the beamline energy.  Each scan is kept less than 50 eV.
- Planar Undulator and PGM.
- Trajectory controller simultaneously moves the monochromator mirror and grating.
- The speed gain depends on the signal intensity.  Typically, from 5x to 10x
- Reduces jitter associated with inaccuracies in the step scanning

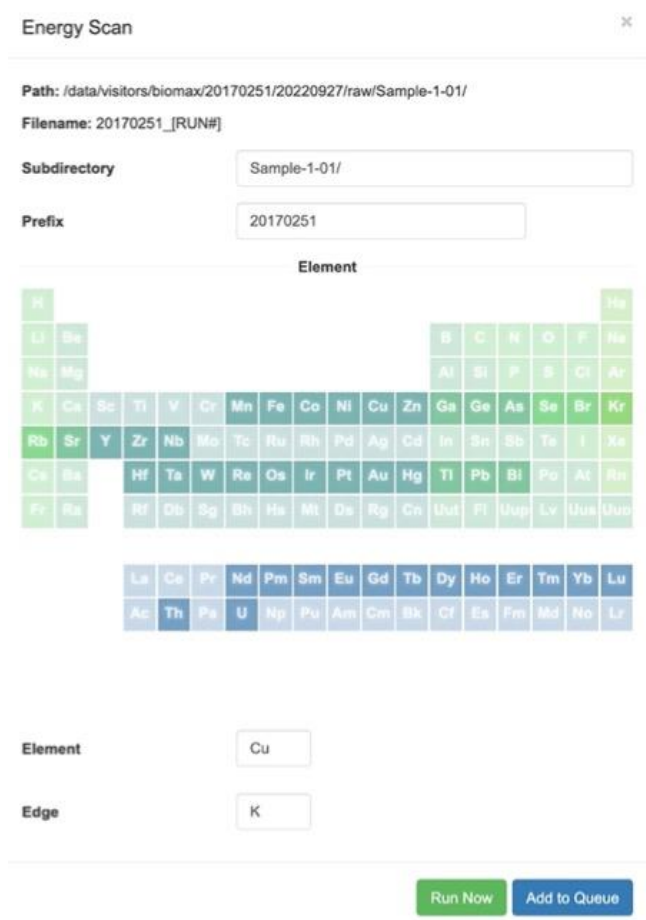



https://accelconf.web.cern.ch/icalepcs2021/papers/frbr04.pdf

# BioMAX

- The energy range is between 6 keV and 24 keV.
- Each scan has a range of 100 eV with 1 eV per step and a total duration of 1 s.
- IVU and DCM
- We can synchronize by position or by time constant.
- The continuous scan is available from the MXCuBE user interface. Using a periodic table, energy scans can be launched for a particular element's K-edge



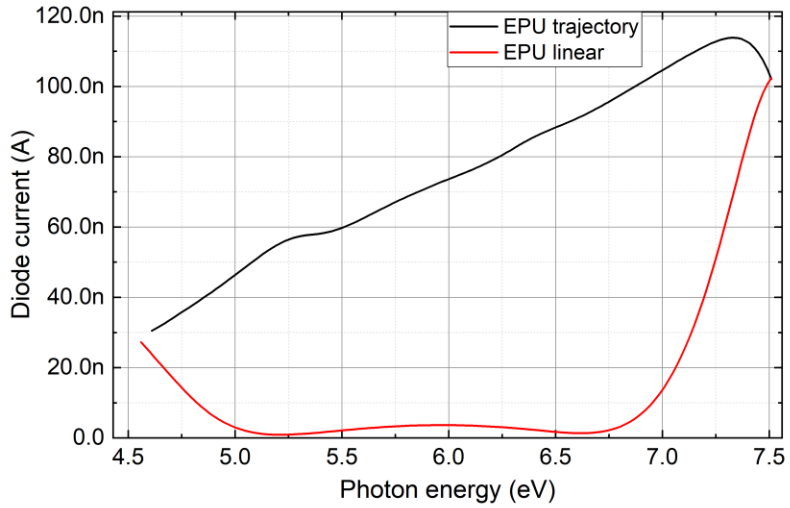



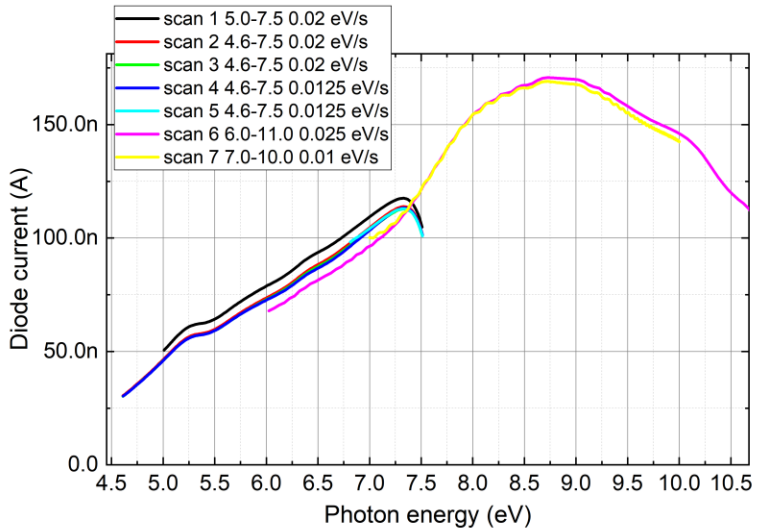


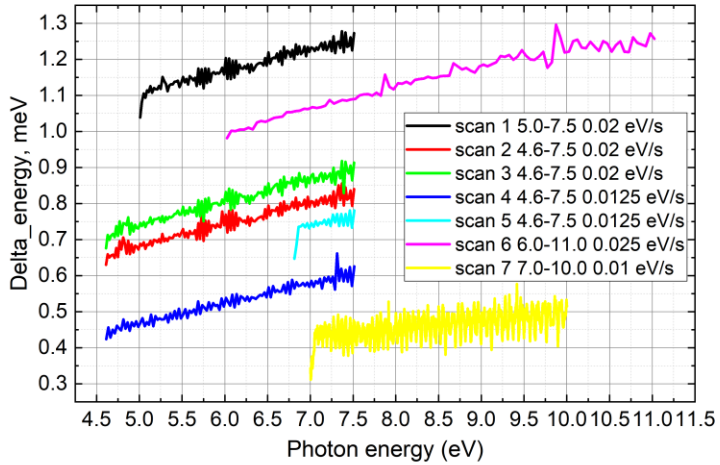

https://doi.org/10.1107/S1600577523005738

# FinEstBeAMS

- The scanned energy down to 4.5 eV
- Uses parametric trajectory for insertion device and monochromator
- synchronize by position or by time constant
- Pandabox triggers the motion



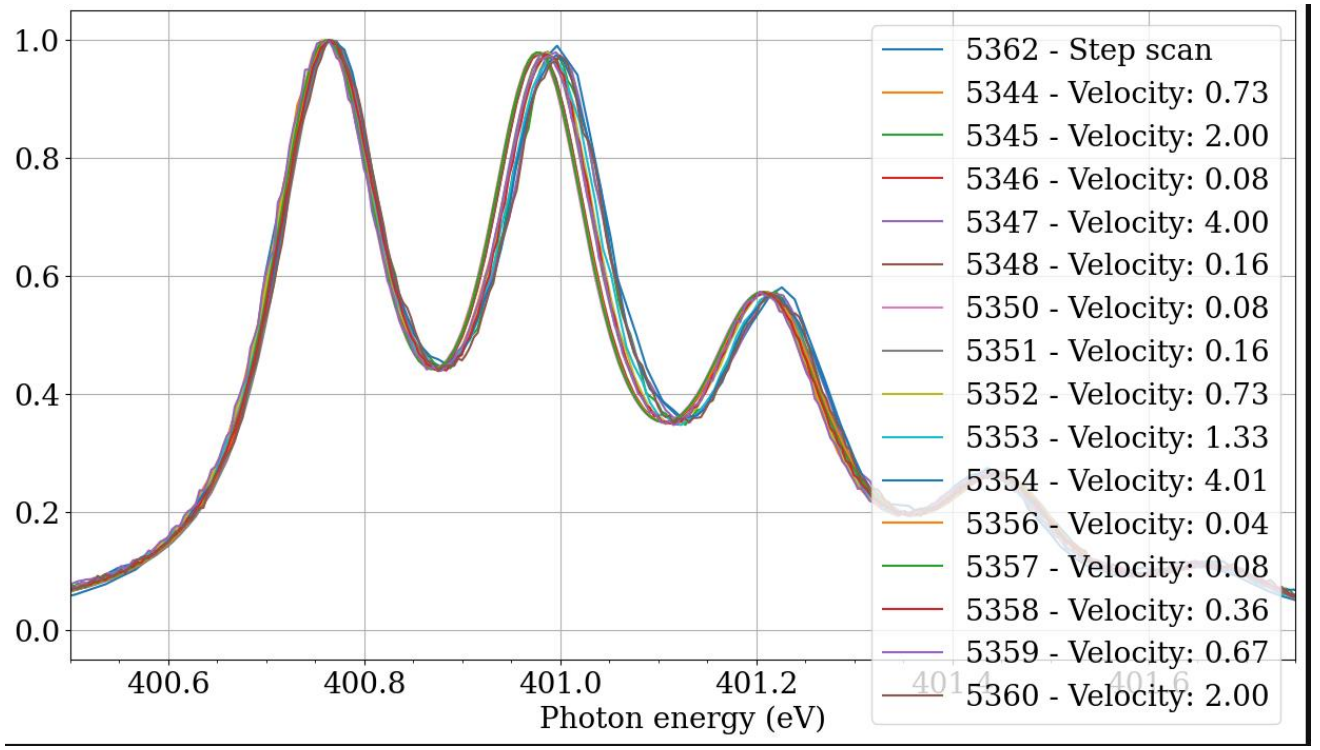




https://accelconf.web.cern.ch/icalepcs2023/papers/tupdp145.pdf

# SPECIES

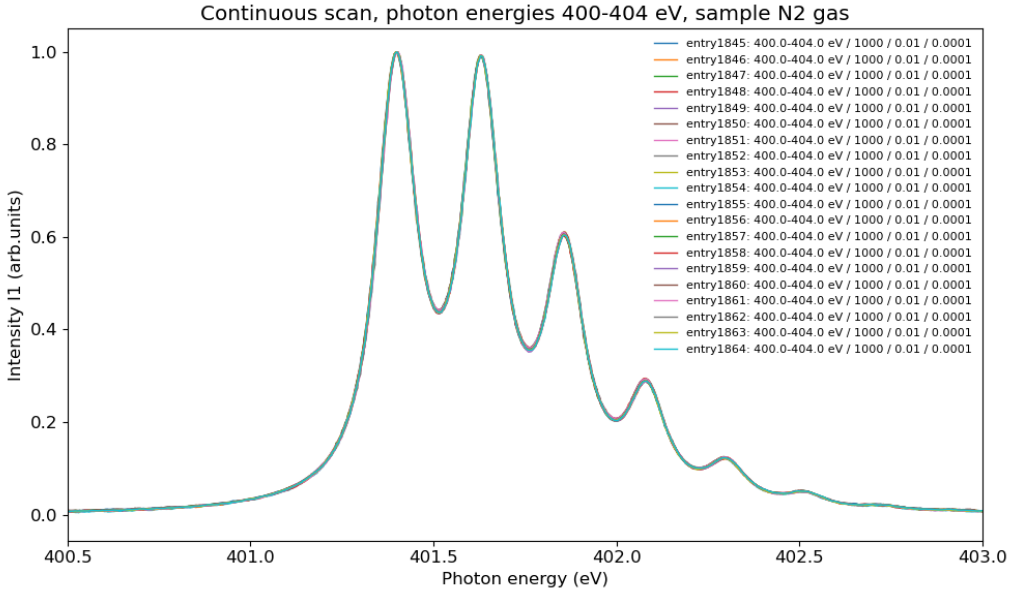
- The scan subset of the full energy range of HIPPIE beamline
- PGM only (so far)
- We can synchronize by position or by time constant.
- The scans in continuous mode are faster

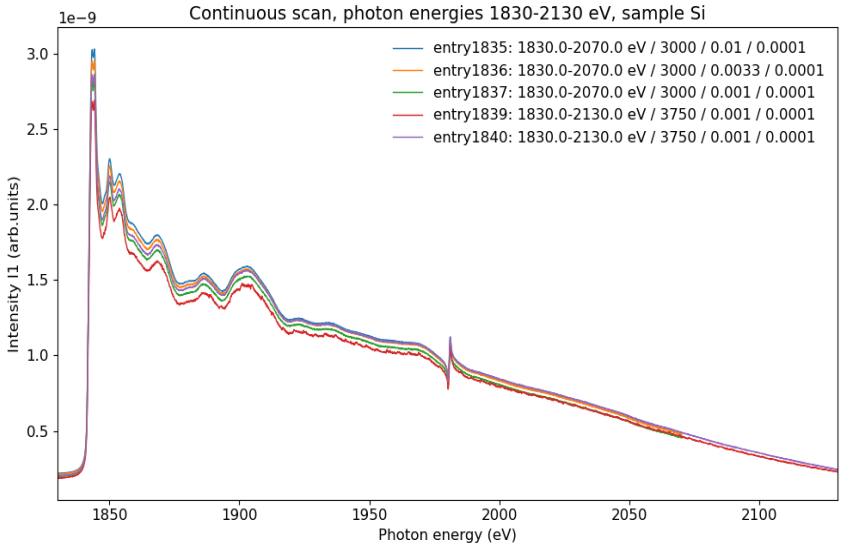


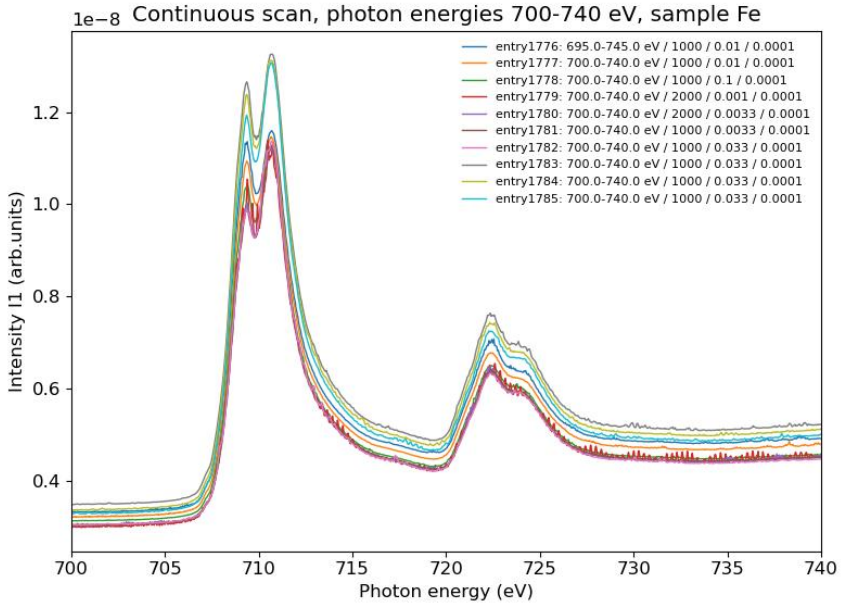

https://elogy.maxiv.lu.se/logbooks/175/entries/42564/
https://elogy.maxiv.lu.se/logbooks/175/entries/40564
https://elogy.maxiv.lu.se/logbooks/175/entries/40766
https://elogy.maxiv.lu.se/logbooks/175/entries/41189

# HIPPIE

- The scan subset of the full energy range of HIPPIE beamline
- PGM only (so far)
- We can synchronize by position or by time constant.
- The scans in continuous mode are fast and of a good quality




https://elogy.maxiv.lu.se/logbooks/636/entries/41757

# OUTLOOK

- Include phase motors of EPU, enable continuous scan energy in helical, inclinded, even universal modes
- Implement ID gap trajectory for beamlines if needed (no-linear ID trajectory)
- Hardware starts the trajectory motion

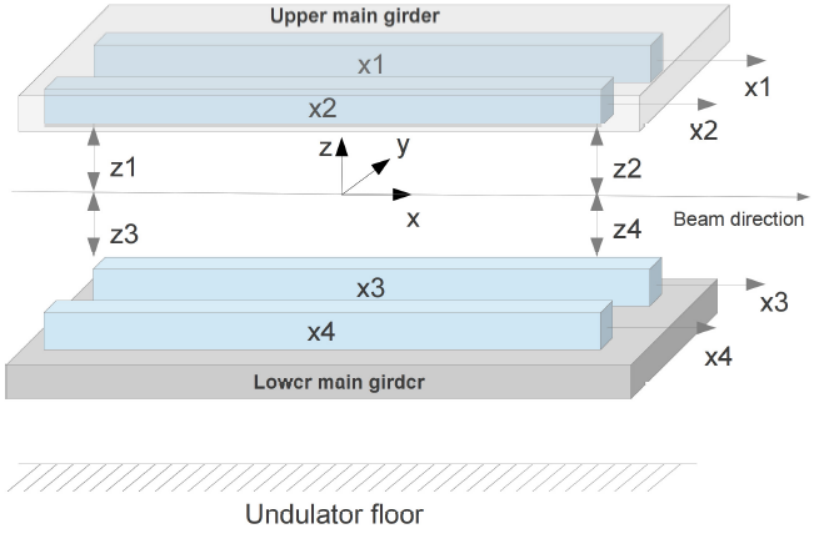
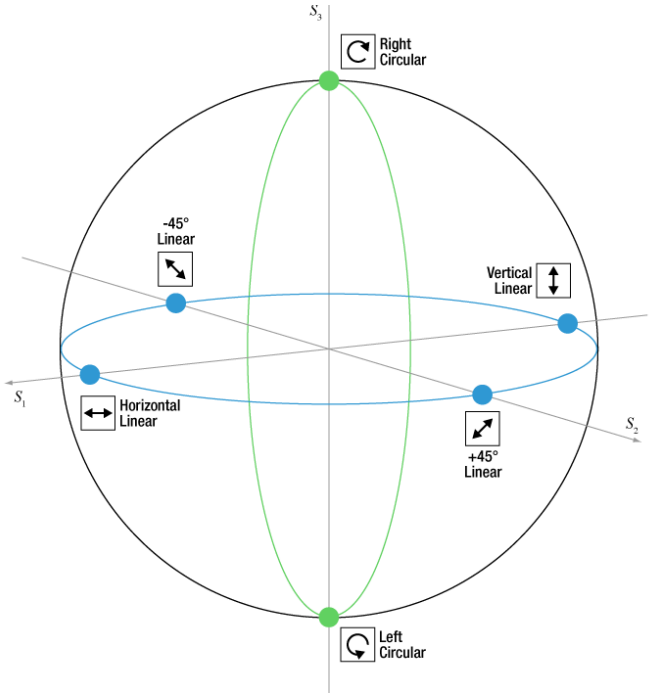



Figure 1 Undulator Layout



# Thank You!
# Questions?