BNL-221578-2021-TECH

NSLSII-ASD-TN-359

A High-Level Application Tool for Generation and Validation of Insertion Device Orbit Feedforward Tables at NSLS-II

Y. Hidaka

May 2021

Photon Sciences

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC), Basic Energy Sciences (BES) (SC-22)

DISCLAIMER

NSLS-II TECHNICAL NOTE BROOKHAVEN NATIONAL LABORATORY	NUMBER **NSLSII-ASD-TN-359**
AUTHORS: Y. Hidaka, Y. Li	DATE 05/31/2021
A High-Level Application Tool for Generation and Validation of Insertion Device Orbit Feedforward Tables at NSLS-II	

Introduction:

We have developed a Python tool to automate the process of generating orbit feedforward tables for insertion devices (IDs) and their validation at NSLS-II.

Even though the first and second field integrals (I_1 and I_2) of an ID are minimized during the tuning stage, the corrections are never perfect partially because the tuning must correct these field integrals for all the available gap and phase values, as well as correct other higher-order multipole components [1]. These residual I_1 and I_2 of an ID result in an angle error and offset (position) error, respectively, at the end of the ID, as shown schematically in Fig. 1. These errors result in closed orbit distortion (COD) around the ring if not corrected.

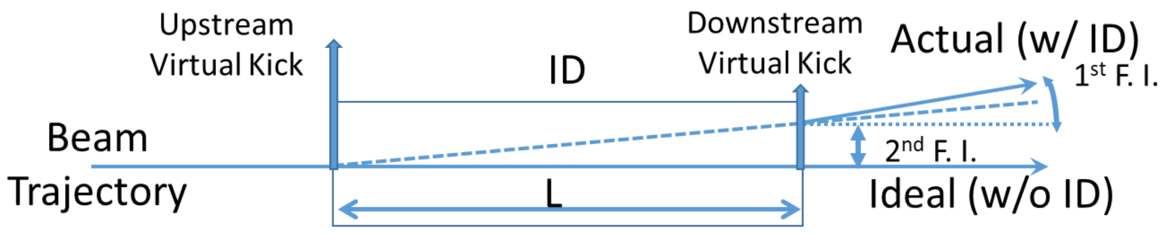

Figure 1: Schematic of the beam trajectory around an ID with length L showing the angle and offset errors caused by the first and second field integrals of the ID.

Since these field integrals are defined as

$$I_{1x,y} = \int B_{x,y}(s)ds,$$
$$I_{2x,y} = \iint B_{x,y}(s')ds'ds,$$

no matter what the actual distribution of magnetic field errors along the s-position is, the error souces can be replaced with a virtual think kick upstream and downstream of the ID (as shown in Fig. 1) with the following relations (only for horizontal kicks shown, but similar relations hold for the vertical kick):

$$\frac{I_{1y}}{B\rho} = \Delta\theta_x = \theta_{x,\mathrm{US}} + \theta_{x,\mathrm{DS}},$$
$$\frac{I_{2y}}{B\rho} = \Delta x = L\theta_{x,\mathrm{US}},$$

where $B\rho$ is the magnetic rigidity (~10 for 3 GeV at NSLS-II), $\theta_{x,\mathrm{US}}$ and $\theta_{x,\mathrm{DS}}$ are the upstream and downstream kick angles, while Δx and $\Delta\theta_x$ are the offset and angle error at the end of the ID.

Therefore, the COD created by non-zero I_1 and I_2 can be eliminated by installing one orbit corrector at or close to the upstream virtual kick and another corrector at or close to the downstream virtual kick, and applying the opposite kicks at these correctors.

When the gap and/or phase of an ID is changed, the residual I_1 and I_2 can change as well. Therefore, we need to create a table for correction values that can minimize the COD for each ID state. The table will be 1-D for gap-only IDs like in-vacuum undulators (IVUs), while it will be 2-D for devices like elliptically polarizing undulators (EPUs) that have gap and phase as their state knobs.

Once these tables are generated, the stand-alone feedforward systems developed by Y. Tian at NSLS-II will monitor the ID gap/phase values and automatically apply the power supply current values based on the feedforward tables to the upstream/downstream correctors. The Python tool discussed in this paper allows you to generate these tables and also confirm that the feedforward system is working properly.

Main Repository & Dependencies:

The main repository of this application is hosted at https://gitlab.nsls2.bnl.gov/hlatools/id-orb-fdfrwrd.

Requirements of this program are the following:

1. A Python 3 environment with the following packages: `aphla` [2,3], `cothread` [4], `h5py`, `matplotlib`, `NumPy`, `SciPy`. (Optional: `jupyter`)
2. EPICS libraries for `cothread` to work.

Quick Start Guide:

First connect to one of the servers on NSLS-II Control Network:

```
$ ssh -Y physics01
```

If you use one of the official accelerator physics standard `conda` environments, all the dependencies are already installed. This tool is not a package, so you have nothing to install other than the dependencies mentioned above.

To activate the latest official environment, just run

```
$ ap-conda-latest
```

If this command is not available, you can see all the available `conda` environments with

```
$ conda env list
```

Then find an environment whose name is like `ap-2021-1.0a` (latest version as of this publication) and activate the environment with

```
$ conda activate ap-2021-1.0a
```

The environment naming convention is `ap-YYYY-V.v[c]`, where `YYYY` is the release year, `V` and `v` are major and minor versions. Occasionally a single alphabet is appended as well, denoted by `c`. This refers to the bug fix version, which starts from a to z. These environments are available on box64-*, physics0*, and hlaioc0*.

How to Use:

The main repository contains 2 Jupyter notebooks (and their dependent modules) that are used to calculate/refine/upload/validate orbit feedforward tables for IDs in NSLS-II Storage Ring. A third Jupyter notebook is used to assess quickly how effective an existing feedforward table still is.

In the first notebook, you will measure the orbit response matrix for ID orbit correctors and its raw (i.e., uncorrected) gap- and/or phase-dependent orbit disturbance. In the second notebook, an orbit feedforward table is first generated, based on the data measured in the first notebook. Then this table is further refined for better correction. This refined table is finally uploaded to the relevant feedforward system PVs to complete the task of updating the orbit feedforward table for the particular ID. As an optional step, you can also validate how well the loaded table is working to minimize the gap- and/or phase-dependent orbit disturbance. The third notebook will automatically close the gap with and without the feedforward system, while measuring the rms COD of 180 BPMs around the ring. This will help you swiftly determine whether the existing table is still effective or needs update.

Even though no installation is required for this tool, other than the dependencies, you do need to pull the latest files and folders from the repository. If you do not know how, just follow the instructions below. Otherwise, you can skip over the instructions on `git`.

Go to a directory where you want to download this repository.

If you have already set up an SSH key for GitLab, you can issue the following command:

```
$ git clone git@gitlab.cs.nsls2.local:hlatools/id-orb-fdfrwrd.git
```

If not, issue the following command instead:

```
$ git clone https://gitlab.nsls2.bnl.gov/hlatools/id-orb-fdfrwrd.git
```

after which you will be prompted for your username and password for GitLab.

Either way, this command will create a new folder called "id-orb-fdfrwrd" at the current directory. Go into this newly created directory and open Jupyter with the following command, assuming you have already activated the latest accelerator physics official environment:

```
(ap-2021-1.0a) $ jupyter notebook
```

Cloning is only needed for the first time. If you want to update to the latest code, just run the following command in the "id-orb-fdfrwrd" directory (NOT the directory above where you issued the "`git clone`" command):

```
$ git pull
```

1. Measurement Notebook

In Jupyter, click on "`template_simple_orbff_meas`" to open the first of the 3 notebooks. Then follow the instructions contained in the notebook or later in this section.

Once you have measured the ID orbit corrector response matrix and the raw COD data with various gap and phase values in the first notebook, you should open the second notebook "`template_simple_orbff_gen_validation`" to postprocess the raw data and generate a new feedforward table. Transfer the necessary information manually from the first notebook to the second notebook, by following the instructions in the second notebook or in the later section.

Before you start

- Inject beam to Storage Ring (SR) as much as possible (to get better signal-to-noise), but below 2 mA to avoid Active Interlock (AI) beam dump.
- Any fill pattern will do.
- For Lattice MASAR, the latest operation lattice is recommended. But even if you use the bare lattice, the resulting difference is likely very small. So, it is not too critical.
- For Orbit MASAR, the latest orbit file should be loaded.
- Make sure that the local bump setpoints are the latest ones by checking against the latest Local Bump MASAR file ("ID_Localbumps" CID 71). If you see any difference, ask the lead operator, or check the history to make sure the currently loaded setpoints are the values requested by the beamline users, not accidentally changed values.
- **Make sure that you know which ID Orbit Feedforward MASAR file ("Orbit_Feedforward" CID 74) needs to be restored if you are just testing this program and want to restore the original table later.**
- **All feedback systems that could change orbit automatically should be disabled (and, if you forget to do this, they will be disabled automatically by the code later): FOFB, SOFB, RF Freq. Feedback, Bump Feedback (a.k.a. Bump Agents), Tune Feedback**
- Bunch-by-bunch feedback could be left enabled, but will be automatically turned off by the code later.

Load the ID feedforward module

```
import idff
```

Running the cell below will print all the available ID names

```
idff.ap.getElements('ID') + idff.ap.getElements('PHASER')
```

Pick one of these names and assign it to `IDName` in the next cell. The following names and mode numbers for EPUs are available as of this publication:

IDName	epu_mode	Notes
epu57g1c02c	1, 2	

ivu20g1c03c	None	
ivu23g1c04u	None	
ivu21g1c05d	None	
ovu42g1c07u	None	
epu60g1c07d	0, 1, 2, 3	Priority order: 2, 3, 0, 1
dw100g1c08ud	None	Not fully implemented
ivu22g1c10c	None	
ivu20g1c11c	None	
ivu23g1c12d	None	
ivu23g1c16c	None	
ivu21g1c17u	None	
ivu21g1c17d	None	
dw100g1c18ud	None	Not fully implemented
ivu18g1c19u	None	
epu57g1c21u	1	
epu105g1c21d	2	
epu49g1c23u	0, 1, 2, 3	
epu49g1c23d	0, 1, 2, 3	
dw100g1c28ud	None	Not fully implemented
_phaserg1c23c	None	

Select an ID for which you want to generate an orbit feedforward table

- For EPUs, you need to also specify the value for "epu_mode". For IVUs, set this variable to `None`.
- As an example, let us pick C05 ID, as shown below.

```
IDName, epu_mode = 'ivu21g1c05d', None
```

(Optional) Relax setpoint/readback difference tolerance if wait time during ORM measurement is long

- Change the `if` clause from `False` to `True` to run this cell if you want.

```
if False:
    diff_tol = 0.1
    #diff_tol = 0.03 # for C07-2 EPU60

    _idobj = idff.ap.getElements(IDName)[0]
    for iCh in range(6):
        _idobj.setEpsilon(f'cch{iCh:d}', diff_tol)
```

Specify whether you are running this notebook to just estimate field integrals (`True`) or to generate an orbit feedforward table (`False`).

- If `False` (default): An orbit response matrix for the ID correctors **will** be measured. **You must select this to generate a table.**
- If `True`: An orbit response matrix for the ID correctors **will NOT** be measured.

```
fld_integ_meas = False
```

Running the cell below will get the pre-defined (recommended) gap/phase values at which table values are generated for the specified ID.

```
params = idff.orbff_select(IDName, fld_integ_meas=fld_integ_meas,
epu_mode=epu_mode)
params['parTable'], params['parList']
```

- If the pre-defined gap/phase values fall outside of the current gap/phase limit values, you would see an error message in the cell above. In this case, you have 2 options.

- The first option is to temporarily change the min/max gap/phase limit values (if allowed, and by asking the lead operator) to make all the pre-defined values valid.

- The other option is to manually adjust the pre-defined gap/phase vectors.

You can see the pre-defined gap and phase vectors with:

```
print(idff.ORBFF_CONFIG[IDName]['gaps'])
print(idff.ORBFF_CONFIG[IDName]['phases'])
```

For example, if the first 2 gap values are outside of the limit, skip the first 2 values:

```
idff.ORBFF_CONFIG[IDName]['gaps'] = idff.ORBFF_CONFIG[IDName]['gaps'][2:]
```

Note that you should NOT remove the fully open gap value from the vector.

After adjusting these vectors, re-run the cell above. This time you should not see any error message.

At this point, you should have a few CSS pages open. First, you need to be able to see the gap/phase setpoint/readback values.

- As an example, we will use C05 ID for the training purpose.
- On CSS, go to: "Main" => "Sub-Systems" => "Insertion Devices" => "ID&FE Main" => "5 SRX" (near the top center)

(CRITICAL) Manually check if the ID corrector power supplies are turned on by putting non-zero setpoint values and see if the readback responds

- You need to open the power supply CSS page by clicking on the "chain"-looking icon button on the CSS page we opened earlier for the gap/phase monitoring purpose.
- If the feedforward mode is "Manual", you can directly modify the "Setpoint" value in each channel.
- If the feedforward mode is "Auto", even if you change the "Setpoint" value, the value may be restored back to the original value due to the feedforward system. In this case, you can switch the mode to "Manual", or, you can change the gap to have the feedforward system change the setpoint value for you.
- In either "Manual" or "Auto" mode, you must confirm that the "Actual" values are also changing with the setpoint values. Some IDs have different units in "Setpoint" and "Actual", so it is OK if the values are very different. The important thing is to check that the "Actual" values are NOT staying near zero. Also, ignore the "Setpoint Readback" column. Also note that some channels are actually disconnected so that "Actual" will always be zero.

Power supplies being used for feedforward (and available modes for EPUs) [Channel indexes here are starting from 0, not 1]:

IDName	epu_mode	Channel Indexes
epu57g1c02c	1, 2	0, 1, 4, 5
ivu20g1c03c	None	0, 1, 4, 5
ivu23g1c04u	None	0, 1, 4, 5
ivu21g1c05d	None	0, 1, 4, 5
ovu42g1c07u	None	0, 1, 2, 3
epu60g1c07d	0, 1, 2, 3	0, 1, 2, 3
dw100g1c08ud	None	Upstream DW ch.0, ch.4; Downstream DW ch.0, ch.4; Upstream vertical slow ring corrector (2nd channel); Downstream vertical slow ring corrector (2nd channel)
ivu22g1c10c	None	0, 1, 2, 3
ivu20g1c11c	None	0, 1, 4, 5
ivu23g1c12d	None	0, 1, 4, 5
ivu23g1c16c	None	0, 1, 4, 5
ivu21g1c17u	None	0, 1, 4, 5
ivu21g1c17d	None	0, 1, 4, 5
dw100g1c18ud	None	Upstream DW ch.0, ch.4; Downstream DW ch.0, ch.4; Upstream vertical slow ring corrector (2nd channel); Downstream vertical slow ring corrector (2nd channel)
ivu18g1c19u	None	0, 1, 3, 4
epu57g1c21u	1	0, 1, 2, 3, 4, 5
epu105g1c21d	1, 2	0, 1, 4, 5
epu49g1c23u	0, 1, 2, 3	0, 1, 2, 3, 4, 5
epu49g1c23d	0, 1, 2, 3	0, 1, 2, 3, 4, 5
dw100g1c28ud	None	Upstream DW ch.0, ch.4; Downstream DW ch.0, ch.4; Upstream vertical slow ring corrector (2nd channel); Downstream vertical slow ring corrector (2nd channel)

`_phaserg1c23c`	`None`	Upstream horizontal/vertical slow ring correctors; Downstream horizontal/vertical slow ring correctors

(CRITICAL: Only for EPUs with current strips in operation) If the feedforward system for the current strips is NOT enabled, enable it now. Also make sure that their power supplies are turned on. (Only ID02, ID21-1, and ID21-2 are currently using current strips, as of this publication).

Running the cell below will perform the following preparation tasks and get a file path to which all the measured data will be saved.

- Open up the ID fully
- Turn off BPM auto gain control (AGC) and adjust BPM attenuation
- Switch feedforward mode to "Manual" (i.e., off)
- Disable all unwanted feedback systems
- Adjust ID local bumps to user setpoints

```
outputFile = idff.prepToMeasureOrbFF(params)
```

(IMPORTANT) This data file path displayed in the cell above is the only information you need to transfer to the second notebook.

Now let the actual measurement begin!

- If this cell stops in the middle (e.g., due to not being able to reach the target gap/phase), simply re-run this cell again. It will automatically resume where it got stopped.
- If this measurement is interrupted and machine conditions have changed substantially, rather than resuming, you may want to start from scratch, i.e., starting over from the top of this notebook.
- If this cell stopped due to a hardware problem, you obviously need to make sure that the hardware problem has been fixed before you attempt to resume or restart from scratch.

```
idff.measure_orbff(params, outputFile)
```

Make sure to restore the ID to the “open” state

- If everything has gone right, the ID should be already fully open for IVUs. But, for EPUs, the phase may not have been brought back to 0. So, it does not hurt to make sure the ID is opened up at the end.

```
idff.openIDs(params)
```

2. Table Generation/Deployment/Validation Notebook

Once you are done with the first notebook, in Jupyter, click on "`template_simple_orbff_gen_validation`"" to open the second notebook. Then follow the instructions contained in the notebook or later in this section.

There is one important manual step. You need to copy and paste the path to the measurement file generated in the first notebook into the second notebook. See the detailed instructions in the cell titled "Replace the file path to the path of the HDF5 file you just created in the first notebook" below.

Before you start

- Read and follow the machine setup described in "Before you start" in "`template_simple_orbff_meas.ipynb`" (i.e., the first notebook) or in the previous section "1. Measurement Notebook".

Load the ID feedforward module and matplotlib

```
import idff

import matplotlib.pylab as plt

plt.rcParams['figure.figsize'] = (14, 10)
```

Replace the example file path in the cell to the path of the actual HDF5 file you just created in the first notebook

- Copy the string in the variable "`outputFile`" in "`template_simple_orbff_meas.ipynb`" (i.e., the first notebook) and replace the example file path being used here with this path.
- This file path should have been printed out in the cell right above the heading "(IMPORTANT) This data file path displayed in the cell above is the only information you need to transfer to the second notebook." in the first notebook.

```
orbff_meas_filepath = \
'/epics/aphla/SR/2021_02/ID/_phaserg1c23c_orbitFF_2021_02_25_021724.hdf5'
```

(CRITICAL) Manually check if the ID corrector power supplies are turned on by putting non-zero setpoint values and see if the readback responds

- You can skip this step if you have just completed the first notebook as you must have passed this check in the first notebook.
- If not, see the instructions in the first notebook.

(CRITICAL: Only for EPUs with current strips in operation) If the feedforward system for the current strips is NOT enabled, enable it now. Also make sure that their power supplies are turned on.

Running the cell below will perform data extraction from the measurement performed in the first notebook

- **(IMPORTANT) The path to the HDF5 file (`'*.ffresult.*.hdf5'`) that will be displayed in the output of this cell is the file path you need to manually assign to the variable `"ffresult_filepath"` in a later cell if you skip to the automatic table validation section to just validate a previously created table.**

```
orbffgen = idff.OrbFFGenerator(orbff_meas_filepath)
```

If you want to just perform automatic validation (i.e., if feedforward PVs have already been populated with valid correction currents and you know which HDF5 file is associated with the loaded table value), skip over to the cell titled "Perform an automated validation..." below. Otherwise, do NOT skip.

Calculate the first estimated feedforward table

```
orbffgen.calcOrbFFTable()
```

Flip the safety switch variable "online" to `True`

- If left `False`, no caput() will be performed.

```
online = True
```

Running the cell below will perform the following preparation tasks.

- Open up the ID fully
- Turn off BPM auto gain control (AGC) and adjust BPM attenuation
- Switch feedforward mode to "Manual" (i.e., off)
- Disable all unwanted feedback systems
- Adjust ID local bumps to user setpoints

```
orbffgen.prepToFineTuneTable(online)
```

Now let the iterative table refining process begin!

- If this cell stops in the middle (e.g., due to not being able to reach the target gap/phase), and if you want to resume where it got stopped, simply run this cell again **after** making sure that the variable "`resume`" in this cell is set to **`True`**.
- If this measurement is interrupted and machine conditions may have changed substantially, rather than resuming, you may want to start from scratch. In this case, re-run the cell right above that calls `orbffgen.prepToFineTuneTable()`, and then run this cell **after** making sure that the variable "`resume`" in this cell is set to **`False`**.
- If this cell stopped due to a hardware problem, you obviously need to make sure that the hardware problem has been fixed before you attempt to resume or restart from scratch.

```
resume = False
orbffgen.fineTuneTable(online, resume)
```

Run the cell below after completing all the refinement steps in the previous cell in order to save the collected data to an HDF5 file, visualize the postprocessed results, and save all the plots into a PDF file.

```
orbffgen.summarize()
```

----------- If NOT uploading the new table to PVs, stop here. -----------

Upload the newly computed feedforward table to the relevant PVs

- If you plan to perform a validation step after uploading the new table, set "`will_run_auto_validation`" to **`True`**.
- If you plan to stop after uploading the table to the PVs, without performing the validation step, set "`will_run_auto_validation`" to **`False`**. In this case, the ID will be opened up fully and its feedforward system will be enabled, setting the ID ready for user operation.

```
will_run_auto_validation = True
orbffgen.applyNewTable(online, will_run_auto_validation)
```

----------- If NOT performing the validation step, stop here. -----------

Perform an automated validation (including interpolation) for an ID orbit feedforward table

- If you did NOT skip the sections for table calculation/refining in this notebook, you do NOT have to change anything in the cell below before running the cell.

- If you DID skip over to this cell without running through the table calculation/refining cells in this notebook, you DO need to manually specify a processed HDF5 file.
    - "`ffresult_filepath`" must contain the correct path to a processed HDF5 file ('`*.ffresult.*.hdf5`'), which is automatically generated when `idff.OrbFFGenerator()` is called in an earlier cell.

```
try:
    orbffval = idff.OrbFFValidator(orbffgen.output_hdf5_filepath)
except:
    ffresult_filepath = 
'ivu20g1c03c_orbitFF_2019_02_03_023512.ffresult.2019-02-03T04-20-15.hdf5'

    import os

    try:
        assert os.path.basename(ffresult_filepath).split('.')[0] == \
            os.path.basename(orbff_meas_filepath).split('.')[0]
    except AssertionError:
        print('The name of the underlying measurement file for the 
processed HDF5 file specified by "ffresult_filepath"')
        print('is based must match the name of the measurement HDF5 file 
specified by "orbff_meas_filepath" (defined in')
        print('the 2nd cell of this notebook).')
        raise

    try:
        orbffval = idff.OrbFFValidator(ffresult_filepath)
    except:
        orbffval = idff.OrbFFValidator(ffresult_filepath, 
orbff_meas_filepath=orbff_meas_filepath)
```

Running the cell below will perform the following preparation tasks.

- Open up the ID fully
- Turn off BPM auto gain control (AGC) and adjust BPM attenuation
- Switch feedforward mode to "Auto" (i.e., on)
- Disable all unwanted feedback systems
- Adjust ID local bumps to user setpoints

```
online = True
orbffval.prepToValidate(online)
```

Now let the validation process start!

- If this cell stops in the middle (e.g., due to not being able to reach the target gap/phase), and if you want to resume where it got stopped, simply run this cell again **after** making sure that the variable "`resume`" in this cell is set to **`True`**.

- If this measurement is interrupted and machine conditions have changed substantially, rather than resuming, you may want to start from scratch. In this case, re-run the cell right above that calls `orbffval.prepToValidate()`, and then run this cell **after** making sure that the variable "`resume`" in this cell is set to **`False`**.
- If this cell stopped due to a hardware problem, you obviously need to make sure that the hardware problem has been fixed before you attempt to resume or restart from scratch

```
resume = False
orbffval.validate(online, resume, orbit_stable_wait=3.0,
    nOrbShots=10, include_userBPMs=False)
```

Run the cell below after completing all the validation steps in the previous cell in order to save the collected data to an HDF5 file.

```
orbffval.save()
```

Finally visualize the validation results, and save all the plots into a PDF file.

```
orbffval.summarize()
```

This concludes the whole process of generating, deploying, and verifying the orbit feedforward table for an ID.

3. Quick Table Assessment Notebook

The third notebook “`template_quick_orbff_check`” should be opened in Jupyter and run if you want to quickly check whether the existing table for an ID is still good or not. If not good enough, you should start the process of updating the table by using the first notebook, followed by the second notebook (see “1. Measurement Notebook” and “2. Table Generation/Deployment/Validation Notebook“).

The difference of this notebook from the full validation performed in the second notebook is that this notebook checks the orbit distortion that includes the transient effects of the ID motion and the ID orbit correctors and the lag between them. If the ID is moving too fast or the corrector response is too slow, you may observe a noticeable orbit distortion even if the table is effective. In the second notebook, orbit is measured after the ID stops moving and the corrector current is stabilized. Therefore, it can evaluate purely the effectiveness of the table, excluding the transient effects, at the cost of much longer measurement time.

This notebook will perform the following steps for an ID of your interest:

1. Enable the orbit feedforward system
2. Open up the gap
3. Correct the local bump

4. Set the current orbit as the reference
5. Start monitoring COD from this reference
6. Start closing the gap
7. Wait 10 seconds after reaching the minimum gap
8. Disable the feedforward system and set all ID orbit corrector currents to zero
9. After waiting for 10 seconds, start opening the gap
10. Wait 10 seconds after reaching the maximum gap
11. Enable the feedforward system
12. Plot the rms COD around the ring while moving the gap vs. time.

The quick table assessment feature for EPUs is still under development as of this publication.

Before you start

- Inject beam to Storage Ring (SR) as much as possible (to get better signal-to-noise), but below 2 mA to avoid Active Interlock (AI) beam dump.
- Any fill pattern will do.
- For Lattice MASAR, the latest operation lattice is recommended. But even if you use the bare lattice, the resulting difference is likely very small. So, it is not too critical.
- For Orbit MASAR, the latest orbit file should be loaded.
- Make sure that the local bump setpoints are the latest ones by checking against the latest Local Bump MASAR file ("ID_Localbumps" CID 71). If you see any difference, ask the lead operator, or check the history to make sure the currently loaded setpoints are the values requested by the beamline users, not accidentally changed values.
- Make sure that all the ID orbit feedforward tables are the latest ones by checking against the latest ID Orbit Feedforward MASAR file ("Orbit_Feedforward" CID 74). If you see any difference, ask the lead operator for the reason for discrepancy. Note that C23 IDs may well have different tables, as there are 2 modes of operation - canted vs. non-canted.
- **All feedback systems that could change orbit automatically should be disabled (There is no automatic disabling feature for this notebook): FOFB, SOFB, RF Freq. Feedback, Bump Feedback (a.k.a. Bump Agents), Tune Feedback**
- Bunch-by-bunch feedback could be left enabled.

Load the ID feedforward module and matplotlib

```
from orbff_quick_check import start_quick_check_v2 as start_quick_check
from orbff_quick_check import plot_result_v2 as plot_result

from orbff_quick_check import plt

plt.rcParams['figure.figsize'] = (14, 10)
```

Flip the safety switch variable "online" to `True`

- If left `False`, no caput() will be performed.

```
online = True
```

Select an ID for which you want to assess the feedforward table

- As an example, let us pick C03 ID, as shown below.

```
idname = 'ivu20g1c03c'; opts = dict(max_gap_mm=None, min_gap_mm=None)
```

The following table shows the currently available valid names and options:

idname	Opts
ivu20g1c03c	dict(max_gap_mm=None, min_gap_mm=None)
ivu23g1c04u	dict(max_gap_mm=None, min_gap_mm=None)
ivu21g1c05d	dict(max_gap_mm=None, min_gap_mm=None)
ovu42g1c07u	dict(max_gap_mm=None, min_gap_mm=None)
ivu22g1c10c	dict(max_gap_mm=None, min_gap_mm=None)
ivu20g1c11c	dict(max_gap_mm=None, min_gap_mm=None)
ivu23g1c12d	dict(max_gap_mm=None, min_gap_mm=None)
ivu23g1c16c	dict(max_gap_mm=None, min_gap_mm=None)
ivu21g1c17u	dict(max_gap_mm=None, min_gap_mm=None)
ivu21g1c17d	dict(max_gap_mm=None, min_gap_mm=None)
ivu18g1c19u	dict(max_gap_mm=None, min_gap_mm=None)
_phaserg1c23c	dict(max_gap_mm=123.0, min_gap_mm=None)

By specifying `None` for max_gap_mm and `min_gap_mm`, the minimum and maximum values are automatically retrieved from the relevant PVs. If you want to customize the scan range, use these options.

Start the measurement!

```
$ output_filepath = start_quick_check(idname, online=online, **opts)
```

Plot the measurement result

```
plot_result(output_filepath)
```

References:

[1] T. Tanabe, Y. Hidaka, C. Kitegi, D. Hidas, M. Musardo, D. A. Harder, J. Rank, P. Cappadoro, H. Fernandes, and T. Corwin, “Latest experiences and future plans on NSLS-II insertion devices”, AIP Conf. Proc. 1741, 020004 (2016).

[2] L. Yang, J. Choi, Y. Hidaka, G. Shen, G. Wang, “Development Progress of NSLS-II Accelerator Physics High Level Applications” in Proc. IPAC2012, New Orleans, Louisiana, USA (2012), THPPR018.

[3] https://github.com/NSLS-II/aphla

[4] http://controls.diamond.ac.uk/downloads/python/cothread/; https://github.com/dls-controls/cothread