

SINGLE PARTICLE DYNAMICS SIMULATION AND CONTROL FOR NSLS-II COMMISSIONING

Lingyun Yang*, Yoshiteru Hidaka, Yongjun Li, BNL, Upton, NY, USA
Guobao Shen, FRIB, East Lansing, MI, USA

Abstract

NSLS-II is the 3 GeV low emittance synchrotron light source recently commissioned and in operation for users. We present some software developed for lattice simulations and machine commissioning. Majority of these tools are callable in a high level programming language Python. These new development and integration at both middle and high level has made our interactive and batch control very efficient.

INTRODUCTION

NSLS-II is the 3 GeV low emittance synchrotron light source recently commissioned and in user operations. From its design to commissioning and operations, we have used various accelerator physics software available in the community and also developed some new tools. Those tools have contributed significantly to the NSLS-II project and many of them are not limited to our own facility, they are general tools that applicable to other accelerator beam dynamics or storage ring commissioning problems.

SINGLE PARTICLE DYNAMICS

At the design stage, many good simulation software are available. They are well understood across the community and a few widely used ones have been used for decades. Many new tools are still being developed for various reasons: some are simple enough to fit one file, some are pursuing more realistic modeling, some are taking advantage of new technologies to improve performance or user experience. At NSLS-II, *elegant* [1], Tracy-2/3 [2] and TESLA [3] are used for lattice simulation and optimization [4, 5]. Some benchmarks were done on single particle trackings for these code [6]. A small first and second order linear lattice code was also developed in Python [7]. It is quick and versatile for any linear lattice matching. An example of FMA (Frequency Map Analysis) calculated with TESLA is shown in Fig. 1.

The local developed Tracy-2/3 [2] and TESLA [3] have also included the momentum dependant kickmap for insertion devices. Both code are symplectic integrator based tracking code and can do TPSA based map generation. Also, since they are developed as a library, the Python interface was implemented to call these C++ routines and analyze/visualize the results in an IPython notebook. This is also an attractive way for interactive real machine control and modeling.

A virtual accelerator was also developed based on Tracy-2/3 for testing high level physics applications. An EPICS access layer is put on top of Tracy-2/3 simulator. The high level scripts can modify and retrieve magnet strength or

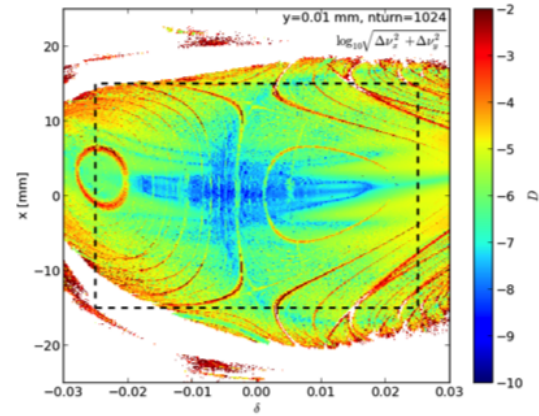


Figure 1: FMA (Frequency Map Analysis) including engineering tolerance such as misalignment, multipole errors.

beam dynamics information via EPICS protocol the exact same way as accessing the hardware. This virtual accelerator provided us an EPICS environment years before the real commissioning.

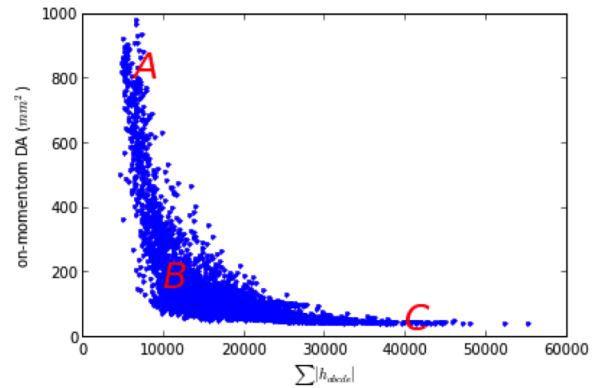


Figure 2: MOGA (Multi-Objective Genetic Algorithm) for NSLS-II lattice optimization. A, B and C are three regions that DA (dynamic aperture) correlates with nonlinear driving terms.

In the linear and nonlinear lattice optimizations, we have used MOGA (Multi-Objective Genetic Algorithm) to optimize the dynamic aperture at different working point and chromaticities [4, 8, 9]. The parallelized optimizer is efficient and has helped us explore many more candidate lattices that were not possible before (Fig. 2). MOGA and its variations with different strategies have been used as a standard tool for nonlinear lattice optimization.

* lyyang@bnl.gov

TOOLS FOR OPERATIONS

At the lower level controls, NSLS-II uses EPICS protocol [10]. Lots of general purpose tools, mostly lower and middle level, have been developed by the EPICS collaboration and used in many accelerator facilities. For our facility, we developed some new middle layer services and a set of high level physics tools [11].

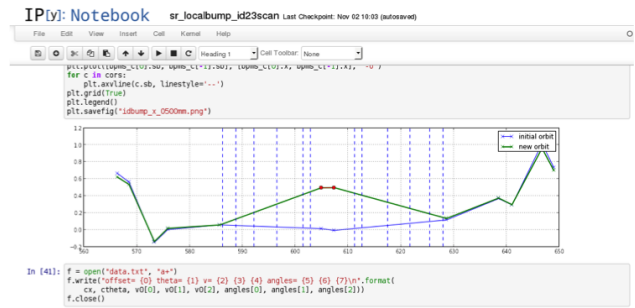
At the lower level we developed MASAR (machine snapshot save and restore) and new features in Control System Studio [12]. They are general tools that applicable to any EPICS based facilities.

MLLT (Matlab Middle Layer Toolkit) has been widely used in lots of accelerator facilities. Each facility can develop its own high level scripts on top. It was also chosen in the NSLS-II project and physicists can use and extend its features. In fact, other tools like SDDS are also installed to support interested users. But given the new infrastructure our controls system provides, developing a new middle layer could take more advantages of them. Python was chosen by one of our developers after careful comparison. It is a general purpose programming language, has good set of scientific and visualization packages, a major player in developing network services, a glue language for FORTRAN and C/C++, a good support on EPICS based controls at both server and client side, open and free. This means we can integrate simulation and controls more close and manipulate data at levels from low to high, server to client, desktop to cluster.

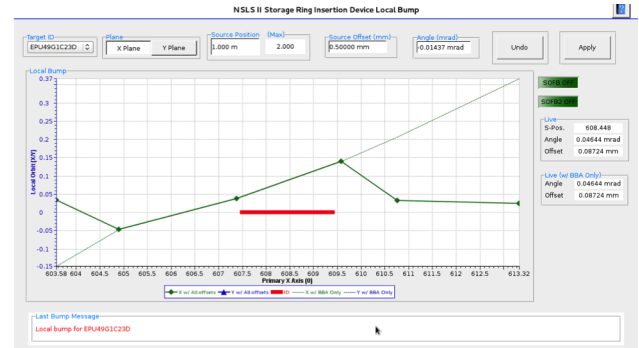
With the new middle layer services, we can map between hardware control channels, i.e. a PV (process variable) in EPICS, and accelerator models dynamically and globally. By querying the server, anyone knows a list of PVs associated to one particular instrument or a family of magnets, and vice versa. It is like a global address book for high level scripts who are looking for the hardware control PVs but only knows the high level element name. It is then easy to hide or disable one non-essential instrument, e.g. corrector or BPM, without disturbing the functioning of client scripts.

An advantage of using general purpose programming language for our high level script is that it is easier to make it a service which runs 24x7 for every client. For some applications, e.g. feedbacks, this single instance service is not optional but required. Programs in Python or C/C++ are trivial to be wrapped for EPICS soft-ioc and share the same code. An example is the ID (insertion device) local bump (Fig. 3). A script which generates orbit local bump was developed in Python is shown in Fig. 3a, and the same library called by the script is also used for an ID local bump IOC used by operators or beamline users is in Fig. 3b. The common part is retrieve a list of available or specified orbit correctors and BPMs, use the corresponding orbit response matrix to correct the orbit to target orbit. This is shared across high level scripts. In fact, even the general orbit correction is using this common routine by using all available BPMs, correctors with target orbit the golden orbit.

Recently Python programming language and its scientific libraries are heard frequently in scientific computing and



(a) IPython notebook on generating and plotting the orbit local bump.



(b) CSS (Control System Studio) [13] Panel for insertion device (ID) radiation source point control, i.e. orbit local bump at the ID.

Figure 3: ID (Insertion Device) local bump control panel.

data science [14]. It has plenty of libraries for mathematical functions, optimization and visualizations. For our applications on NSLS-II commissioning and operations, a good support for both batch and interactive execution are convenient. Before deploying a formal batch mode script, we could easily start interactive control or testing in an IPython notebook where all code and results are in one place. These kind of notebook are easier to be understood when shared with other colleagues.

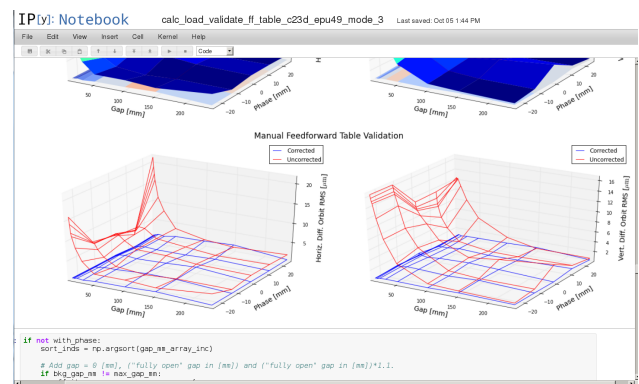


Figure 4: IPython notebook for 2D ID (Insertion Device) feedforward table.

The feedforward table for insertion devices are generated and analyzed in an IPython notebook (Fig. 4). In Python, accessing database like MySQL or SQLite is simple and part

of its standard library. Together with the hardware/model mapping [12], we can retrieve information from magnetic field measurement in the lab to temperature reading in the tunnel. Although there are 1D and 2D types of feedforward tables depending on ID types, they can share a good amount of code and the high level script can query for the information easily.

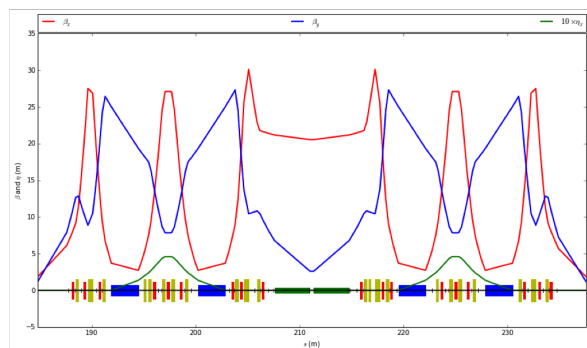


Figure 5: NSLS-II lattice from a simple Python linear lattice code [7].

For the design we would prefer a code modeling both physics dynamics, linear and nonlinear, and the engineering tolerance. But for the commissioning and operations, it might be more helpful to have a simple, fast and flexible code. The PyLat [7] we developed is a single file Python script (Fig. 5), but it only depends on standard library for first and second order linear lattice computing and lattice optimization (matching). Loading online machine data and convert unit with measured magnet field table to models and lattice properties are an integrated process. After tuning the model, its settings can be put back to the real machine again.

ACKNOWLEDGEMENT

We want to thank Dr. Samuel Krinsky (1945–2014) for his guidance and invaluable discussions, Bob Dalesio for

his support of developing controls related toolset and his vision on system architecture. We are grateful for support from the NSLS-II. This work is supported in part by the U.S. Department of Energy (DOE) under contract No. DE-AC02-98CH1-886.

REFERENCES

- [1] M. Borland, “Elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation”, LS-287, Advanced Photon Source, 2000.
- [2] J. Bengtsson, “Tracy-2/3”, unpublished.
- [3] L. Yang, “Tracking Code Development for Beam Dynamics Optimization”, PAC’11, New York, March 2011.
- [4] L. Yang, Y. Li, W. Guo, S. Krinsky, “Multiobjective optimization of dynamic aperture”, Phys. Rev. ST Accel. Beams 14 (2011) 5.
- [5] W. Guo, S. Krinsky, L. Yang, “NSLS-II Lattice Optimization with Non-zero Chromaticity”, IPAC’10, Kyoto, May 2010.
- [6] J. Choi, unpublished.
- [7] S. Krinsky, Y. Li, unpublished.
- [8] Y. Li, L. Yang, Y. Hidaka, “Efficient MOGA for NSLS-II Ring Dynamic Aperture Optimization”, Low Emittance Workshop 2014, INFN-LNF, Frascati, Italy, Sep. 2014.
- [9] L. Yang, D. Robin, F. Sannibale, C. Steier, W. Wan, “Global optimization of an accelerator lattice using multiobjective genetic algorithms”, NIM-A vol. 609 (2009) 1, 50–57.
- [10] EPICS, <http://www.aps.anl.gov/epics/>.
- [11] G.M. Wang etc., “Tools for NSLS-II Commissioning”, IPAC’15, Richmond, May 2015.
- [12] G. Shen, Y. Hu, M.R. Kraimer, K. Shroff, “NSLS II Middlelayer Services”, ICALEPCS’13, San Francisco, October 2013.
- [13] Control System Studio, www.cs-studio.org
- [14] J. M. Perkel, “Programming: Pick up Python”, Nature, Vol 518, Issue 7537, p 125, Feb, 2015