ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB317

ONLINE MODEL DEVELOPMENTS FOR BESSY II AND MLS

P. Schnizer*, J. Bengtsson, T. Birke, J. Li, T. Mertens, M. Ries, A. Schälicke, L. Vera Ramirez
Helmholtz-Zentrum Berlin, BESSY, Berlin, Germany

Abstract

Digital models have been developed over a long time for preparing accelerator commissioning next to benchmarking theory predictions to machine measurements. These digital models are nowadays being realized as digital shadows or digital twins. Accelerator commissioning requires periodic setup and review of the machine status. Furthermore, different measurements are only practical by comparison to the machine model (e.g. beam based alignment). In this paper we describe the architecture chosen for our models, describe the framework Bluesky for measurement orchestration and report on our experience exemplifying on dynamic aperture scans. Furthermore we describe our plans to extend the models applied to BESSY II and MLS to the currently planned machines BESSY III and MLS II.

INTRODUCTION

Modelling is an essential and inseparable part of many scientific disciplines. Also for building and operation of accelerators the existence of appropriate models is a basic requirement. These models were soon realised as computational models which got more and more sophisticated as computational power became readily available.

Other engineering disciplines, in particular space probe control, soon developed analog and digital models of their devices that they controlled. These not solely provide means for modelling it, but based on received telemetry data these models are adjusted to their real counterpart and are used for verification of any commands before applied to the real space probe: this rather conservative approach is motivated by the fact that typically any maloperation could cause a total mission loss.

Accelerator control, however, profits typically from being accessible and maintainable; in particular synchrotron based light sources fall under this case. As these synchrotron light sources serve many users at the same time, maximising machine availability has been more and more on focus next to an continuous overall operation performance improvement. Online models of synchrotron light sources have been used since SLS [1] and are now being ubiquitous e.g. [2–6].

The field of machine learning and data science has emerged over the last decades and got widely spread since some time which improved the usabiltiy of these tools. Furthermore special methods (e.g. Tikhonov regularisation [7] with penalties on the main diagonal [8]) are readily available as dedicated packages (e.g. scikit-learn [9]) typically using the Python programming language. Experiment orchestration packages matured and allow these days to describe complex measurements in a consistent simple fashion, e.g. Bluesky [10].

All these readily available tools convinced us to set out and implement a online model using these new tools.

ARCHITECTURE

Design Requirements Specifications

The online model shall be used as test bed for accelerator measurement tools, to benchmark models to the machine and for studying the impact of new or upgraded accelerator elements. Finally it should allow closing the loop: from model to machine and back to model.

Requirement analysis of the online model showed that the online model should closely reflect the machine. Furthermore it should allow linear and non linear simulations with the actual machine setting and allow forecasting the machine performance. Furthermore "first step" optimisations of the machine should be feasible modelling static and transient behaviour of the machine. It should be possible to use ones "own model" (thus make it convenient to apply the model of choice).

The following design principles are used as guidance: 1) modularity: build the system from modular components, 2) reuse: reuse already existing building blocks, i.e follow the Unix philosophy [11].

These considerations let to a layered design as given in Fig. 1 with the following layers:

Application Layer These applications could be commissioning tools, scripts for measuring or optimising the machine, machine learning (i.e. machine training) applications.

Tools Layer Different applications require different sets of tools. Optimizers are required by optimization applications. Furthermore many applications require that data is gathered in a synchronous manner and stored so that it can be found again.

Front End Layer takes care of making the machine and all its real devices available. (This part does not belong to the online model. The online model has to take care to be compatible with the already existing machine and its exposed interfaces). The online models puts simulators or model to device converters in this layer.

Middle Layer The middle layer is responsible to collect the data from the different devices or transmit or distribute commands to the different devices.

Backend Layer consists of the the simulation engine and the tools that convert from pyhsics to engineering space and vice versa

* pierre.schnizer@helmholtz-berlin.de

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB317

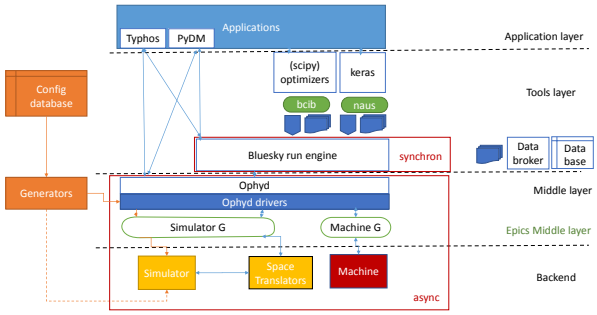


Figure 1: Online model architecture.

The design depicted in Fig. 1 can be executed in a parallel fashion. The only part that is purely sequential by design is the part of the Bluesky [10] run engine that executes plan commands.

The authors of this document are aware of the impact of Amdahl's law [12]. This bottle neck is considered acceptable as: 1) the Bluesky run engine is a sequencer: it has to ensure that these commands are executed one after the other. 2) the required processing power is rather limited. Thus this limitation seems acceptable.

Converters are typically implemented as EPICS records; some are implemented in Python using pydevice [13, 14].

Status

At the current state the authors are experimenting with the available components while the available software is already used by the authors for developing and testing commissioning scripts based on Ophyd / Bluesky.

Simulator The simulator uses TRACY-3.5 implemented as an EPICS input output controller (IOC). This IOC is based on the code base used for NSLS-II [5] with its functionality extended. Now devices can be set by their names as found in their lattice files, given that these names are unique. Lattice information – names of components, their position next to their Twiss parameters – are accessible as EPICS database records.

Translator and Glue Transport and glue are provided by EPICS' database records generated by appropriate templates. The parameters are currently loaded executing scripts offline. An automatisation of this step is looked into. First discussions among the authors show that proper device management requires tracking device position and version; an approach currently not implemented yet at the centre. It looks like that a rather practical approach will be chosen restricting to simply tracing it to the coefficients linked to the device's position names.

The translators between magnets and machine values have been implemented as linear functions for the quadrupoles, sextupoles and steerers. Furthermore misalignment can be set or changed during run time. The last layer in the glue layer pushes the value in physical units to the device using the device name and functionality as token.

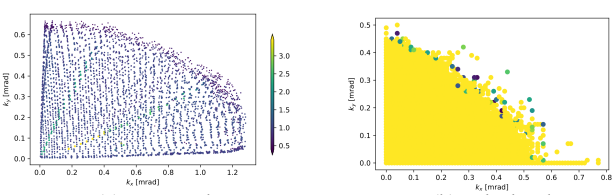
Data Access These database records are gathered in dedicated ophyd devices, thus these can be retrieved together with any other device information (see Fig. 2). Datastorage and retrieval is then handed over to the databroker. xarrays [15] then simplify further processing in the chain. Inspection and view of the betatron function is then only a single method evaluation away.

```
import xarray as xr
import matplotlib.pyplot as plt
from bact2.ophyd.devices.dt import tracy, beam
from bluesky import RunEngine
import bluesky.plans as bp
from databroker import Broker

bm = beam.Beam('Pierre:DT:beam', name='bm')

RE= RunEngine({})
db = Broker.named('temp')
RE.subscribe(db.insert);

uid, = RE(bp.count([bm], 1))

header = db[uid]

# Old style ..
conf = header['descriptors'][0]['configuration']
d = conf['bm']['data']
bm_names = list(d['bm_names'])
bm_ds = d['bm_ds']

bd_ = header.xarray_dask()
items = bd_.dims.items()
replace_dims = {name : 'name' for name, dim in items
                if dim == len(bm_names)}
beam_data = bd_.rename(replace_dims).assign_coords(name=bm_ds)

beam_data.bm_twiss_beta_x_vec.plot()
```

Figure 2: Accessing device information.

USAGE EXAMPLE: DYNAMIC APERTURE SCAN

The dynamic aperture was scanned with a kicker on the real machine and on the online model (see Fig. 3). The kicker strength was increased until the single bunch was lost. The excitation was measured using a Libera electron from Instrumentation Technologies [16], i.e. a turn by turn beam position monitor. This access to the device was implemented



(a) measured (b) calculated

Figure 3: Dynamic aperture scan.

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB317

as Ophyd devices and the measurement plan, including reinjection of the single bunch, was implemented as bluesky plans.

This plan was also used on the online model calculating turn-by-turn data. The angle of the kicker was estimated calibrating the measured particle oscillations to the ones forecast by the online model.

The online model has not yet thoroughly been benchmarked or optimised, thus the dynamic aperture plots (see Fig. 3) were solely produced to check if the analysis pipeline is functional.

MACHINE LEARNING EXAMPLES

Along with the twin developments, first attempts of machine learning have been made [17]. A rather practical one focuses on searching a harmonic orbit distortion at frequency of 2.5 Hz and an amplitude of 0.8 µm. Beam position monitor data were acquired using the fast orbit feedback system (FOFB, [18]), with its feedback switched off. The acquired data were sorted in 10 seconds long packages, and filtered so that only contributions in the frequency range of 1 to 3 Hz were left. After this filter the 2.5 Hz vibration could be clearly seen. Analysing the different packages it was obvious that its phase was not stable. Therefore it was decided at a start to try to identify the location of the source generation using the absolute value of the individual kicker responses Δ_x, Δ_y

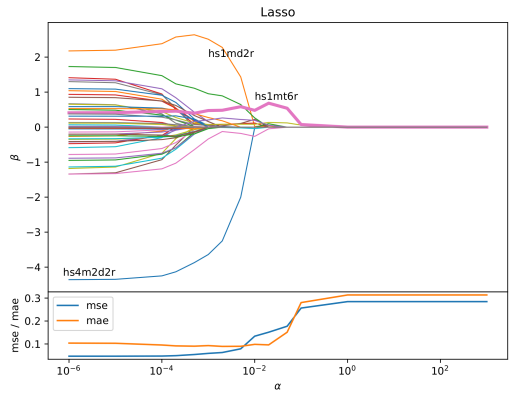
$$\Delta_{x,y} = \frac{\sqrt{\beta}}{2\sin(\pi Q_{x,y})}\sqrt{\beta_i}\vartheta_i \cos\left(\pi Q_{x,y} - |\phi_i - \phi(s)|\right), \tag{1}$$

using the working points Q_x, Q_y, phase advance ϕ and betatron function β_i from the model. β_i, ϕ_i denote the Betatron function and phase advance at the location of the steerer. The kick angles ϑ_i were all set to 10 µrad. These were used as independent values and fit to the absolute value of the data. The kicks Eq. (1) are highly correlated, thus regularisation was considered to be required, in particular as linear regression produced many large contributions.

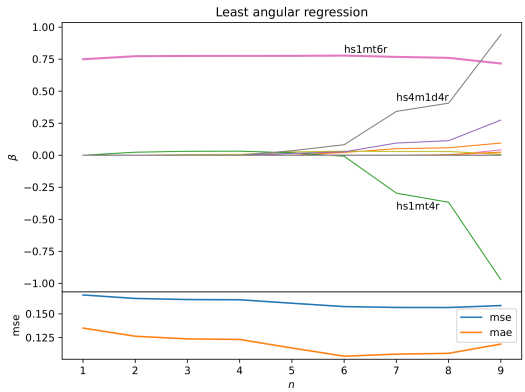
As a single distortion is searched the absolute value of the contribution was penalised (L_1-loss or mean absolute error) using the Lasso algorithm [19, 20], in particular its implementation in the scikit-learn package [9]. One can see in Fig. 4(a) that at a penalsing parameter of $\alpha \approx 10^{-2}$ only one distortion source is dominating. For the least angular regression algorithm [20, 21] this source is constantly identified. One can, however, see that the loss functions, only change for the Lasso methods, and only by a factor of 3. This indicates that the signal to noise ratio has to be improved before one of these methods can be used.

CONCLUSION

HZB has set out implementing a online model for the synchrotron light sources it operates: BESSY II and MLS. These developments shall be adaptable to the projected BESSY III or MLS II machines.



(a) Lasso



(b) Least angular regression

Figure 4: Machine learning: search for a harmonic vibration, The names of the individual kickers are together with their line. α…tuning parameter, mse…mean square error mse…mean absolute error.

The online model is based on a layered concept with EPICS as the transport layer, TRACY as core simulation engine as an EPICS IOC. Conversion from engineering to physics space is implemented as EPICS records; data transfer and pipeline tests were made. Now the BESSY II machine model requires to be bench marked. Then the online model will be transfered and tested on MLS.

Machine learning gets being applied at HZB, e.g. for searching harmonics disturbances of the electron beam.

ACKNOWLEDGEMENT

The author's like to thank Roland Müller for introducing us to "Bluesky", Guobao Shen for sharing the NSLS-II code, and Günther Rehm for introduction to the Diamond/Libera software.

REFERENCES

[1] M. Böge and J. Chrin, "A CORBA based client-server modelfor beam dynamics applications at the SLS", in *Int. Conf.*

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-WEPAB317

Accelerator and Large Experimental Physics Control Systems, Trieste, Italy, Oct. 1999, pp. 555–557.

[2] A. Terbilo, "Accelerator toolbox for MATLAB", Stanford Linear Accelerator Center, Stanford University, Stanford, CA, USA, Tech. Rep. SLAC-PUB-8732, 2001.

[3] G. J. Portmann, W. J. Corbett, and A. Terebilo, "An accelerator control middle layer using matlab", in *Proc. 21st Particle Accelerator Conf. (PAC'05)*, Knoxville, TN, USA, May 2005, paper FPAT077, pp. 4009–4011.

[4] M. T. Heron *et al.*, "The diamond light source control system", in *Proc. 10th European Particle Accelerator Conf. (EPAC'06)*, Edinburgh, UK, Jun. 2006, paper THPCH113, pp. 3068–3070.

[5] G. Shen, K. Shroff, and L. Yang, "NSLS-II high level application infrastructure and client API design", in *Proc. 24th Particle Accelerator Conf. (PAC'11)*, New York, NY, USA, Mar. 2011, paper MOP250, pp. 582–584.

[6] P. P. Goryl, A. I. Wawrzyniak, M. Sjöström, and T. Szymocha, "An implementation of the virtual accelerator in the tango control system", in *Proc. 11th Int. Computational Accelerator Physics Conf. (ICAP'12)*, Rostock-Warnemunde, Germany, Aug. 2012, paper MOSBC3, pp. 23–25.

[7] A. N. Tikhonov, "On the stability of inverse problems", *Doklady Akademii Nauk SSSR*, vol. 39, no. 5, pp. 195–198, 1943.

[8] A. E. Hoerl and R. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems", *Technometrics*, vol. 12, pp. 55–67, 1970.

[9] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] D. Allan, T. Caswell, S. Campbell, and M. Rakitin, "Bluesky's ahead: A multi-facility collaboration for an a la carte software project for data acquisition and management", *Synchrotron Radiation News*, vol. 32, no. 3, pp. 19–22, 2019. doi:10.1080/08940886.2019.1608121

[11] D. McIlroy, E. N. Pinson, and B. A. Tague, "Unix time-sharing system: Foreword", *The Bell System Technical Journal. Bell Laboratories*, vol. 57, pp. 1899–1904, 1978. doi:10.1002/j.1538-7305.1978.tb02135.x

[12] G. M. Amdahl, "Validity of the single processor approach toachieving large scale computing capabilities", in *Proc. AFIPS'67*, Atlantic City, NJ, USA, Apr. 1967, pp. 483–485.

[13] K. Vodopivec, "Easy integration of Python into EPICS IOCs", presented at EPICS Collaboration Fall Meeting 2020, Berlin, Germany, Sep. 2020, unpublished.

[14] K. Vodopivec *et al.*, PyDevice, https://github.com/klemenv/PyDevice.

[15] S. Hoyer and J. Hamman, "xarray: N-D labeled arrays and datasets in Python", *Journal of Open Research Software*, vol. 5, no. 1, 2017. doi:10.5334/jors.148

[16] M. G. Abbott, G. Rehm, and I. Uzun, "The Diamond light source control system interface to the Libera beam position monitors", in *Proc. 12th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'09)*, Kobe, Japan, Oct. 2009, paper THP011, pp. 694–696.

[17] L. Vera Ramierz, T. Mertens, R. Müller, J. Viefhaus, and G. Hartmann, "Adding Machine Learning to the Analysis and Optimization Toolsets at the Light Source BESSY II", in *Proc. 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19)*, New York, NY, USA, Oct. 2019, pp. 754–760. doi:10.18429/JACoW-ICALEPCS2019-TUCPL01

[18] R. Müller *et al.*, "Fast orbit feedback at BESSY-II: Performance and operational experiences", in *Proc. IPAC'13*, Shanghai, China, May 2013, paper WEPME002, pp. 2920–2922.

[19] S. S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit", *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998. doi:10.1137/S1064827596304010

[20] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*, 2nd ed. New York, NY, USA: Springer, 2009.

[21] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression", *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004. doi:10.1214/009053604000000067