SIMULATION TOOLS FOR THE DESIGN AND PERFORMANCE EVALUATION OF TRANSVERSE FEEDBACK SYSTEMS

J. Komppula*, W. Hofle, K.S.B. Li, CERN, Geneva, Switzerland

Abstract

Transverse feedback systems are used in synchrotrons and storage rings to damp injection oscillations and suppress transverse instabilities. Especially instabilities driven by high intensity beams in future circular colliders such as the FCC set challenging requirements for transverse feedback systems. In order to develop a transverse feedback system able to meet those requirements, sophisticated simulation tools are required. For this purpose, a new modular framework for modeling a transverse feedback system has been developed in Python. The framework can be used as a transverse feedback module in the macro-particle beam dynamics simulation code PyHEADTAIL or as a separate tool for studying feedback models from a control theory point of view by using a simple signal model for the beam. The main principle of the code is presented and simulation methods used for the conceptual design of the FCC are discussed.

INTRODUCTION

Transverse beam oscillations together with nonlinear imperfections of the accelerator lead to emittance growth, beam quality reduction and potential beam losses. Therefore, these oscillations which are caused by injection errors or impedances, for example, are kept minimal by using active feedback systems, i.e. transverse feedback systems.

Transverse feedback systems have been used for decades especially for correcting injection kick errors, as well as during the energy ramp up and at flattop to ensure beam stability [1]. Recently, mainly due to the progress of digital electronics and signal processing techniques, new possibilities to use the transverse feedback systems have opened, e.g. cleaning the abort gaps [2] and even using those systems as a flexible tool during machine development studies.

High intensity beams in future accelerators set new challenges for the transverse feedback systems. Due to the technological challenges in minimizing impedances in the large future accelerators, operation of those machines relies increasingly on transverse feedback systems [3]. Upgrades of the existing accelerators towards higher brightness beams can be more susceptible to coherent instabilities, which are potentially suppressed by means of transverse feedback systems [4]. In order to meet these challenges, reliable simulation methods are required.

Different approaches have been used for simulating transverse feedback systems. Due to the limited computing resources and the complexity of beam dynamics itself, only simplified and idealistic models of transverse feedback systems are mainly applied to the beam dynamic simulations [5]. However, recently also more detailed models have been included in these simulations [6–9].

From a technological point of view, the challenges arise from the physical realization of the feedback system. Only the beam displacement can be measured, but the correction kick must be applied to the transverse momentum of the beam one turn after the measurements. A large number of studies have focused on optimizing the feedback systems by using rather idealistic models for the beam (e.g. Refs. [1, 10] and references therein). Challenges of these approaches arise from the fact that the beam itself is a dynamic object.

It can be argued that there is a gap between beam dynamic simulations and the studies aiming towards optimization of transverse feedback systems. In other words, the methods allow either a detailed modeling of the beam dynamics or an accurate treatment of the feedback loop, but a self consistent exchange of information between these approaches is challenging. This gap can be filled by using modern modular programming methods.

The recently developed modular macro-particle beam dynamics simulation framework PyHEADTAIL provides a flexible interface to include additional modules to beam dynamics simulations [5, 11]. By using PyHEADTAIL as a reference application, a modular framework for finite length signal processing was developed in Python.

The framework utilizes principles from dataflow programming, i.e. the framework gives tools to generate signals, which flow trough a chain of blocks, "signal processors". Hereby, a "signal processor" is a Python object, which represents an elementary signal processing operation. It can be used with any signal source in any place of the signal processor chain. At the same time it is simple enough to be developed with limited programming skills.

The framework is compatible with the high performance multibunch PyHEADTAIL simulations in an MPI parallel computing environment. More details about the framework and the feedback models developed for the conceptual design of the Future Circular Collider (FCC) are presented in the following sections.

THE FRAMEWORK

The main concept of the framework is presented in Fig 1. The framework can be divided into three levels: the core, signal processors and interfaces. The communication with the beam dynamic code (PyHEADTAIL) or other signal sources (e.g. numerical signal generators or data loaders for external data) are implemented by programming interfaces.

In the interfaces, a signal compatible with the framework is generated by utilizing the basic functions from the core. The signal from the interfaces is processed by passing it

* jani.komppula@cern.ch

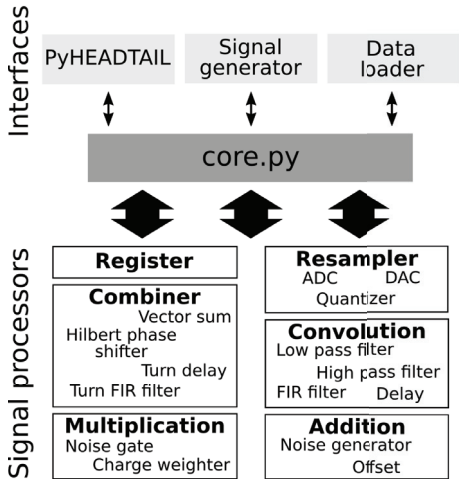

Figure 1: The main concept of the framework for finite length signal processing

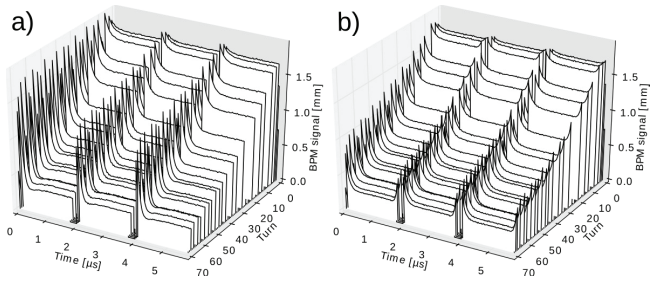

Figure 2: Multibunch PyHEADTAIL simulations of a 3-batch (3x72 bunches) injection in LHC at zero chromaticity with feedback on. In Fig a) the bandwidth is limited by the 1 MHz power amplifier, while in Fig b) the signal is preprocessed before the amplifier by using the digital signal processing and 32 tap FIR filter introduced in Ref. [12].

through a list of signal processors, which represents the model for the feedback system. The signal processors in the list represent elementary analog and digital signal processing steps from the pickup plate to the kicker. The model of the feedback system itself is independent of the interfaces, i.e. it can be used with any signal source. The complex signal processing structures (e.g. parallel signals and internal loops) can be implemented by using specific signal processors for those structures.

The only frozen part of the framework is the core, which includes specifications for the signals and the signal processing units and necessary tools for generating signals and passing them through the signal processors. Both interfaces and signal processors can be developed separately and dynamically without interfering with each other.

In order to simplify the development of signal processors, three different classes for the signals have been specified depending on the underlying assumptions on the signal. No other assumptions are made for Class 0 signals apart from them being representable as a list of numbers. For Class 1 signals it is assumed that the signal consists of a number of equally sampled segments (e.g. equally sampled bunches) and Class 2 signals are assumed to be continuously sampled. Because of the hierarchy of the signal classes the signal processor designed for a specific signal class is also able to process all the higher level signals. The class of the signal can be changed via the signal processors, e.g. by resampling it.

The framework itself contains signal processors for the basic operations. The abstract base classes have been implemented for the basic mathematical operations; addition, multiplication, linear transform and convolution. These base classes allow flexible implementations of, for example, basic filters (RC, Gaussian, Sinc, FIR), ideal amplifiers, noise generators etc. In addition to these, the framework includes special processors for resampling, storing and betatron phase rotating of signals. Supplementary signal processors can be easily programmed by the user by using only a couple of lines of code.

The PyHEADTAIL interface allows for an arbitrary number of pickups and kickers in any location of the PyHEADTAIL one turn map. Signal modifications in the pickups and kickers can be modeled independently by using signal processors. The signal transfer between the pickups and kickers occurs through register and combiner objects, which allow betatron phase rotations by using any algorithm, e.g such as based on vector calculus or FIR filter algorithms (e.g. Hilbert phase shifter). For the simplest simulations separate locations for a pickup and a kicker can be ignored by using OneBoxFeedback, which processes the pickup signal in one location of the accelerator.

In addition to the PyHEADTAIL interface, an interface for numerical signal generators has been developed. It allows studies of the feedback models by using pure sinusoidal signals or a simplified signal model for a beam including only linear betatron motion. This interface allows faster and more flexible testing and benchmarking of the code as well as examination of feedback systems for specific purposes. For example, the LHC injection error damping is simulated with and without the digital FIR filter by utilizing the framework as a PyHEADTAIL module in Fig. 2. These calculations take from minutes of computing time on a desktop computer (1000 macro particles per bunch, 1000 turns without wake fields) to days of computing time on a cluster (800 000 macroparticles per bunch, 600 000 turns with wake fields) in PyHEADTAIL, while the same case can be studied from a technological point of view in terms of seconds on a desktop computer by using the simplified signal models.

FEEDBACK MODELS FOR FCC

In order to study the feedback requirements for the conceptual design of the FCC from the beam dynamics points of view, a simplified model including bandwidth limitations, noise and correction algorithms for betatron motion was developed. In addition, the framework was developed to be able to model all the signal processing steps of the LHC damper system in detail (see details of the system from Refs. [12,13]),

ISBN 978-3-95450-182-3

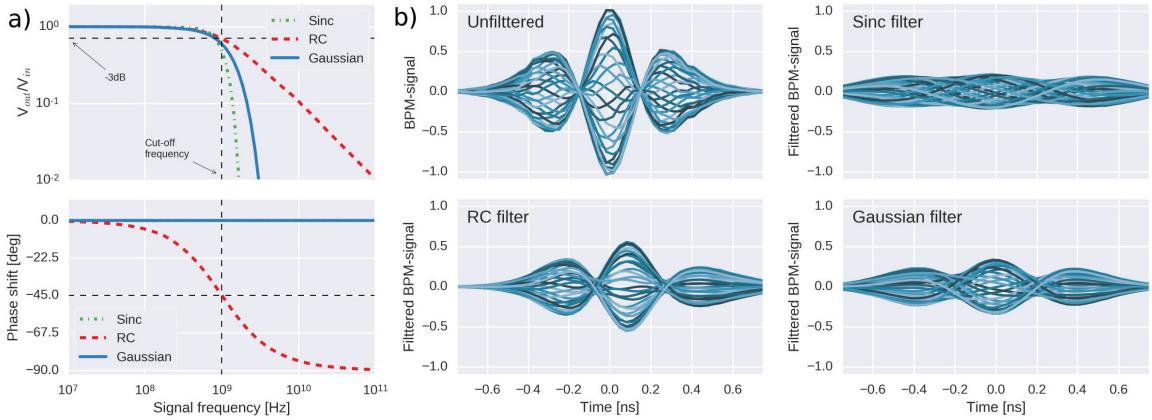

Figure 3: a) Frequency responses of the 1 GHz low pass filters. b) The time domain effects of the filters are demonstrated by passing a BPM signal (simulated k=2 mode instability in HL-LHC when Q'=13) through the filters. All the calculations have been performed by using the presented framework.

in order to allow detailed studies of technological solutions for FCC.

Currently, one of the most important limitation for the transverse feedback systems is the bandwidth, which can be modeled by using a lowpass filter. Different types of filters were studied some of which have been presented in Fig.3. It was found that Gaussian filters work best as a general purpose solution for the simplified model. This is because, the impulse response of the filter must be smooth and symmetric in order to implement realistic bandwidth limitation and avoid unnecessary sensitivity to timing. The very sharp peak in the impulse response of the phase linearized 1st order filter causes unrealistically good damping for intrabunch oscillations and numerical stochastic cooling. Sinc filters are good for the intra bunch feedback, but, however, can get into resonance with the bunch spacing in multibunch simulations.

For the betatron motion correction required by the one turn delay and/or separate pickups and kickers, vector calculus based algorithms (Vector sum) were found to be a good general purpose solution. Gaussian white noise can be added before imposing any bandwidth limitation. It was also found that the direct BPM signal (charge weighted displacements of the simulated slices) is a more stable model for the intra bunch pickup signal than the direct displacement values of the slices (Δ/Σ-hybrid emulation).

DISCUSSION

A modular framework for finite length signal processing has been developed. Together with PyHEADTAIL, it produces a dynamic environment for a variety of purposes, e.g. studying requirements for the feedback systems from the beam dynamics points of view, optimizing current feedback systems and scaling them towards future requirements.

The development of the framework was started in spring 2016. The single bunch version of the framework has been used since summer 2016 for the feedback studies of HL-LHC [14] and FCC. The next version compatible with multi-bunch simulations is currently under commissioning. So far the framework has fulfilled all the expectations for performance, usability and flexibility.

The framework has opened new possibilities for the detailed studies of the feedback systems. However, the tools themselves are only the first step towards reliable simulations. At the end, neither simplified nor very detailed models for feedback systems can be fully established without benchmarking to experimental observations and subsequently improving of the models.

Due to the previous challenges in the modeling of transverse feedback systems, experimental studies have not focused on providing data for benchmarking. However, this data would be increasingly important for reliable simulations required in near future.

With the flexibility provided by the modular codes, on the other hand they can be often applied for other purposes than originally designed. For example, beam oscillations and their diagnostics are of general interest in accelerator technology, for which the presented tools might be applied.

ACKNOWLEDGEMENT

The authors would like to thank members of the CERN BE/RF/FB and especially Gerd Kotzian for fruitful discussions.

REFERENCES

[1] W. Höfle, "Progress in transverse feedbacks and related diagnostics for hadron machines," *in Proc. IPAC'13, Shanghai, China, May 2013*, pp. 3990–3994.

[2] A. Koschik et al., "Abort gap cleaning using the transverse feedback system: Simulation and measurements in the SPS for the LHC beam dump system," *in Proc. EPAC'08, Genoa, Italy, June 2008*, pp. 2656–2658.

[3] F. Zimmermann et al., "Beam dynamics issues in the FCC," *in Proc. HB2016, Malmö, Sweden, July 2016.*

[4] G. Arduini et al., "High Luminosity LHC: challenges and plans," *J. Instrum.*, vol. 11, no. 12, p. C12081, 2016.

[5] E. Métral et al., "Beam instabilities in hadron synchrotrons," *IEEE T. Nucl. Sci.*, vol. 63, no. 2, pp. 1001–1050, April 2016.

[6] G. Kotzian, W. Höfle, and E. Vogel, "LHC transverse feedback damping efficiency," *in Proc. EPAC'08, Genoa, Italy, June 2008*, pp. 3632–3634.

[7] J. R. Thompson, J. M. Byrd, W. Höfle, and G. Rumolo, "Initial results of simulation of a damping system for electron cloud-driven instabilities in the CERN SPS," *in Proc. PAC'09, Vancouver, Canada, May 2009*, pp. 4713–4715.

[8] J.-L. Vay et al., "Simulation of e-cloud! driven instability and its attenuation using a feedback system in the CERN SPS," *in Proc. IPAC'10, Kyoto, Japan, May 2010*, pp. 2438–2440.

[9] K. Li et al., "Modelling and studies for a wideband feedback system for mitigation of transverse single bunch instabilities," *in Proc. IPAC 2013, Shanghai, China, May 2013*, pp. 3019–3021.

[10] J. Fox and E. Kikutani, "Bunch feedback systems and signal processing," *in Proc. Joint US–CERN–Japan–Russia School on Particle Accelerators, Montreux, Switzerland, May 1998*, pp. 579–620.

[11] K. Li et al., "Code development for collective effects," *in Proc. HB2016, Malmö, Sweden, July 2016*.

[12] P. Baudrenghien, W. Höfle, G. Kotzian, and V. Rossi, "Digital signal processing for the multi-bunch LHC transverse feedback system," *in Proc. EPAC'08, Genoa, Italy, June 2008*, pp. 3269–3271.

[13] D. Valuch and P. Baudrenghien, "Beam phase measurement and transverse position measurement module for the LHC," *LLRF'07, Knoxville, TN, USA, 2007*, https://edms.cern.ch/file/929563/1/llrf07poster.pdf

[14] K. Li et al., "Do we need a wide-band transverse feedback in the LHC/HL-LHC," *75th HiLumi WP2 meeting*, August 23 2016.

ISBN 978-3-95450-182-3