

# Implementation of BLonD for Booster Simulations

## P. F. Derwent

January 22, 2021

### Abstract

Understanding of the Booster longitudinal dynamics is important for the higher intensity operation expected for NOvA and LBNF/DUNE. This note describes how the CERN longitudinal code BLonD has been implemented for the Booster.

## 1 Introduction

The Fermilab Booster is an interesting machine for understanding of longitudinal dynamics. It is a rapidly cycling synchrotron with a repetition rate of 15 Hz, accelerating up to  $4.8 \times 10^{12}$  protons per pulse from 400 MeV to 8 GeV (kinetic energy). 400 MeV H<sup>-</sup> in a 201.25 MHz bunch structure are injected over up to 18 turns and adiabatically captured in an h=84 RF system. There are 22 RF cavities installed allowing for a maximum of 1.1 MV with a frequency sweep from 37 MHz to 53 MHz. Transition occurs at an energy of 5.45 GeV. To minimize losses in the slip stacking process in the Recycler Ring, there are strict requirements on the longitudinal emittance at extraction.

When the PIP-II linac is commissioned during the next decade, Booster operations will significantly change [1, 2]. The repetition rate will increase to 20 Hz. The injection energy will increase to 800 MeV. The incoming beam will have a 162.5 MHz bunch structure injected over about 300 turns in a bucket to bucket scheme, with extracted intensity of  $6.5 \times 10^{12}$  protons per pulse. The requirements on total beam loss and extraction emittances will stay the same.

To have confidence in the Booster performance at these parameters, a good model of the longitudinal dynamics is required. Studies [3] have been performed using ESME [4].

While ESME is versatile, well understood, and documented, it is not actively maintained. BLonD [5] is a newcomer to the longitudinal dynamics code field. It is a python based code, with intensive computational aspects written in C++. All inputs, models, and definitions are written using python scripts. As a result, it opens up access to all the numerical, scientific, and plotting packages available in python. The class structures make it easily extendable.

## 2 Basic BLonD structure for the Booster

A basic implementation of BLonD requires the user to instantiate an element of the following classes:

- *blond.input\_parameters.ring.Ring* Class
  - contains the general properties of the synchrotron that are independent of the RF systems or the beam
  - Global Parameters:
    - \* Circumference, particle type, number of RF sections, number of turns
  - Turn-by-turn parameters
    - \* Momentum / energy ramp
    - \* Momentum compaction factor
- *blond.beam.beam.Beam* Class
  - contains the general beam properties
  - beam coordinates (dT, dE)
  - needs a defined *Ring* class
  - beam intensity
  - number of macroparticles in the simulation
- *blond.input\_parameters.rf\_parameters.RFStation* Class
  - contains RF parameters for the RF systems in one ring segment or one RF Station
  - Global Parameters

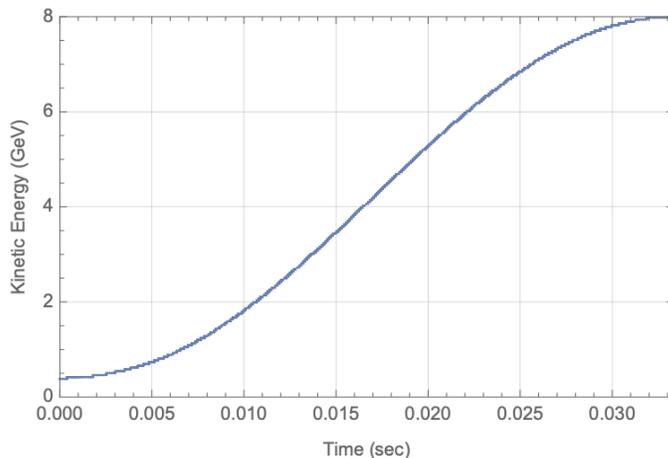


Figure 1: The kinetic energy ramp for the Booster cycle.

- \* *Ring*, Harmonic
- Turn-by-turn Parameters
  - \* Voltage, Synchronous Phase
- *blond.trackers.tracker.RingandRFTracker* Class
  - taking care of basic particle coordinate tracking for a given RF station and the part of the ring until the next station
  - on a turn-by-turn basis

This basic structure can be expanded in many ways. Beam and cavity feedback algorithms can be part of the RF station or the tracker. Additional trackers for impedances (space charge, magnet, other machine impedances), injection and extraction scenarios, plotting and monitoring can be added.

I have defined the Booster *Ring* to:

- accelerate protons
- have a circumference of 474.20214 m
- have 1 RF section (collapse 22 cavities to 1)

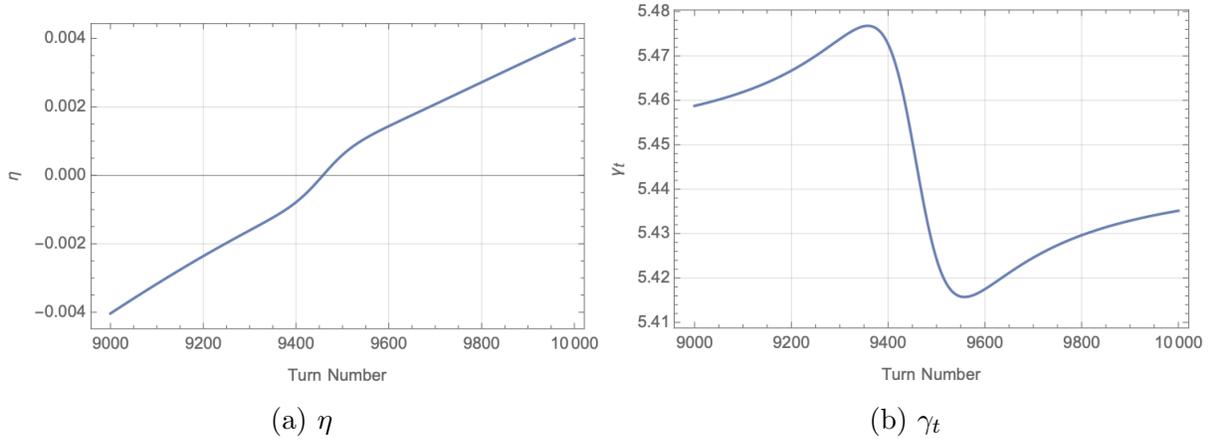


Figure 2: The value of  $\eta$  and  $\gamma_t$  near transition, using the simple phase jump model.

- have 19820 turns in the ramp, this includes 90 turns with constant energy for injection and capture
- follow a kinetic energy ramp from 400 MeV to 8 GeV based on a momentum ramp

$$p(t) = (p_f - p_i)\text{Cos}(2\pi f_{ramp}t) + p_i \quad (1)$$

where  $p_i = 954.26 \text{ MeV}/c$ ,  $p_f = 8888.89 \text{ MeV}/c$ ,  $f_{ramp} = 15 \text{ Hz}$ , and  $t$  is time in the cycle. The kinetic energy ramp is shown in Figure 1.

- momentum compaction factor  $\alpha$  which includes a turn dependent term [6]

$$\alpha = \frac{1}{\gamma_{t,default}^2} + \frac{d\alpha}{dn} \frac{\delta\alpha_n(i - nTr)}{\delta\alpha_n + (i - nTr)(i - nTr)} \quad (2)$$

where  $\gamma_{t,default} = 5.446 \text{ GeV}$ ,  $\frac{d\alpha}{dn} = 7.56\text{e-}6$ ,  $\delta\alpha_n = 10000$ ,  $i$  is turn number, and  $nTr$  is the turn where transition takes place. This mimics a  $\gamma_t$  jump by going through transition faster. See Figure 2 for the value of  $\eta$  and  $\gamma_t$  near transition. In this plot, transition occurs at turn 9458. The jump can be turned on and off as desired.

I have defined the *Beam* to:

- include the default Booster ring as defined above

- Number of macro-particles based on number of Booster bunches to fill and the linac bunch intensity scaled to 333000. 330k is a good number - if filling 81 bunches have enough statistics in each bunch while not having so many macro-particles to slow down the calculations.
- Intensity of 1 linac bunch (6.8e8 H-)

It is important to get the ratio of macro-particles to beam intensity correct at this step, as later calculations use this ratio and the number of surviving macro-particles for current calculations. In a later section, I define an injection method to mimic the Linac injection to Booster.

I have defined the *RFStation* to:

- include the default Booster ring as defined above
- have harmonic number 84
- RF voltage and synchronous phase ( $\phi_S$ ) per turn based on the kinetic energy ramp from one of 4 possibilities:
  - based on ramp Chandra used in ESME simulations [3]
  - based on ramp Valeri used for transition crossing simulations [6]
  - based on ramp calculations done in Mathematica [7]
  - based on the hardware curves in operations in October 2020.

The gap voltage = RF voltage  $\times$  Sin[ $\phi_S$ ] for each turn needs to equal the energy gain per turn, as defined by the kinetic energy ramp in the Booster ring class. See Figure 3.

There are boolean variables in the python scripts to switch between the four ramps. In general, they read information from associated data files and generate the turn-by-turn arrays (using interpolation or other methods).

I have defined the *RingandRFTracker* to:

- include the default Booster *Ring* and *RFStation* as defined above

In addition to the defaults, I have defined monitoring and profiling for use in plotting and calculational methods.

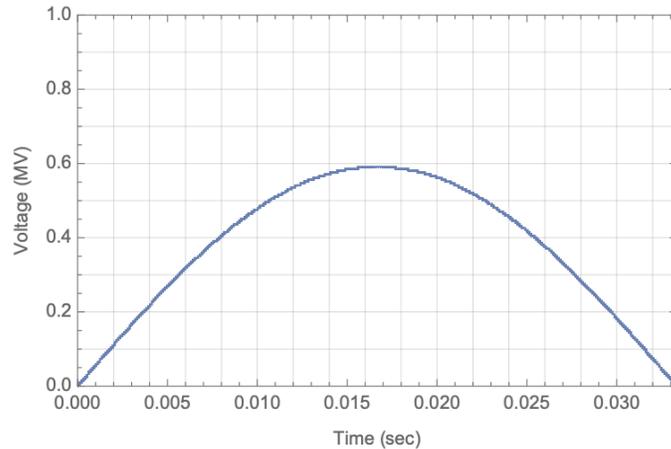


Figure 3: The value of the required accelerating voltage through the Booster cycle to match the energy ramp of Figure 1.

### 3 Extensions to the model

Given BLonD's class structure, it is straightforward to add extensions to the default situations. Some structures, such as those for cavity or beam feedback loops and machine impedances, are part of the class structure for the program. The details of the implementation are then at the control of the user. I have written extensions for the Booster that cover:

- Injection
- Space Charge Impedance
- Magnet Impedance
- Dipole Damper Feedback
- Quadrupole Damper Feedback
- Radial Position Loop

Beam loading compensation corrections are under development. In the following sections, I will define the algorithms for these extensions.

### 3.1 Booster Injection

The class LinacInjection is used for turn-by-turn injection into the Booster from the 201 MHz Linac. It updates the distribution of the beam every turn, up to the number of turns requested. The parameters for the LinacInjection method are:

- *RFStation* : class  
An RFStation type class
- *Beam* : class  
A Beam type class
- *bunchPopulation* : float  
particles per linac bunch 6.824e8 - corresponds to 22 mA
- *EnergyOffset* : float  
energy offset of incoming beam, in eV (default is 0 for on axis)
- *nTurns* : integer  
number of turns to inject, default 15
- *SigmaE* : float  
energy width of incoming pulse, in ev (default 1.6e6/4 eV)
- *SigmaT* : float .. update default value by scaling frequency  
time width of incoming pulse, in seconds (default 530e-12)
- *nBoosterBunch* : integer  
number of bunches to fill (default = 1)
- *eJitter* : float  
jitter / noise on linac - booster energy match, units of eV (default 146,329 eV - 1e-4), not used at this time
- *phiJitter* : float  
jitter / noise on linac - booster phi match, units of seconds (default 0), not used at this time

Injection is implemented in the method 'track' in the LinacInjection class. For turn number (from the RF Station class) less than the number of turns, a biGaussian bunches at 201.25 MHz with  $\sigma_{dE} = \text{SigmaE}$  and  $\sigma_{dt} = \text{SigmaT}$  are placed in the Booster buckets.

For turn 0, I assume the Linac and Booster RF are aligned at  $t=0$ . As the Linac and Booster RF are not frequency matched, the injected Linac Bunches end up in slightly different time locations. During the injection period, the Booster energy is fixed at 400 MeV and there is 1.4 kV of RF voltage with  $\phi_S = 0^\circ$ . In Figure 4, the first 4 turns of injection are shown.

I have also written a PIP-II injection class based on the information presented in [1], which uses a different Booster model also. Details will be discussed in a future note.

### 3.2 Space Charge and Magnet Impedance

BLonD has a class structure to handle impedances and their impact on the beam. For space charge, it is the class *blond.impedances.impedance.InductiveImpedance*, which implements a constant imaginary  $\frac{Z}{n}$  impedance on a turn-by-turn basis. For the Booster, the space charge impedance is defined as:

$$\frac{Z}{n} = -\frac{Z_0[1 + \log(\frac{b}{a})]}{2\beta\gamma^2} \quad (3)$$

where  $Z_0$  is the resistivity of free space (377  $\Omega$ ),  $b$  is the beam pipe radius (2.86 cm),  $a$  is the beam size (1.2 cm), and  $\beta$  and  $\gamma$  are the beam relativistic factors. For the beam pipe radius and beam size, I am using the same values used in ESME simulations [3].

The magnet impedance is more complicated than the Space Charge impedance. I have used the class *blond.impedances.impedance.InducedVoltageFreq* which implements a list of possible impedances per turn. I use the model documented in reference [8] and in presentations to the PXIE technical group [9]. There are different values for F and D gradient magnets, with strong frequency dependent terms. This model compares reasonably to measurements of the machine impedance made in 2001 [10]. The generated voltage is available on a turn-by-turn basis within the code, as an example see Figure ???. On the left is a figure from reference [8] showing the the beam distribution, space charge impedance voltage, and magnet impedance voltage at transition. On the right, the same information from the BLonD simulation at transition. The shapes are quantitatively very similar, while the details are dependent on the beam distribution at the time of the sampling.

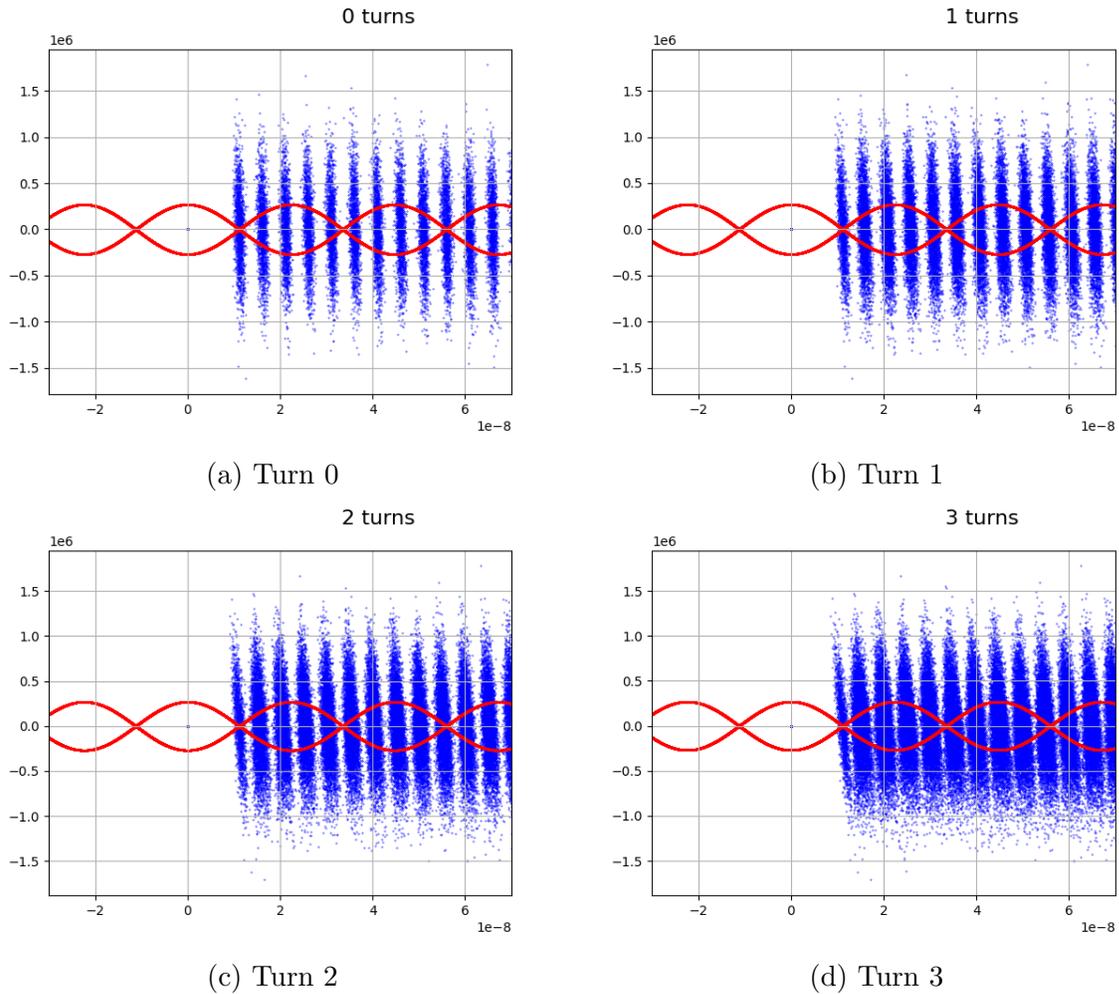


Figure 4: First 4 turns of the Linac injection into Booster, filling 3 Booster buckets. Vertical scale is dE (eV), horizontal scale is dt (seconds). The red trace is the separatrix for 1.4 kV of gap voltage at  $\phi_S = 0^\circ$ .

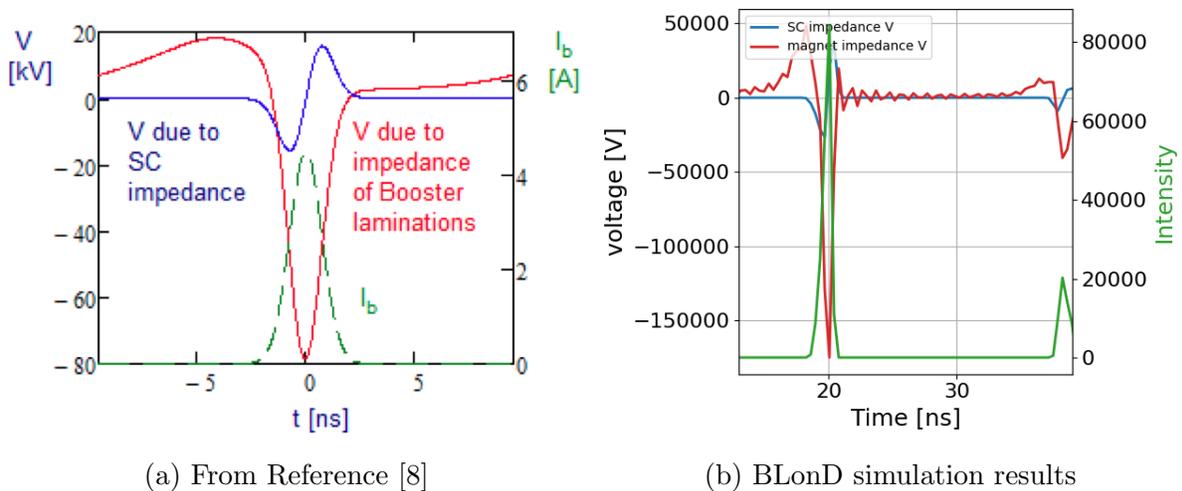


Figure 5: Comparison of beam distribution (green), Space Charge Impedance Voltage (blue), and Magnet Impedance Voltage (red) from the simulation described in references [8] and [9] and the BLonD implementation. Quantitatively the distributions are similar, with differences associated with differences in the beam distribution at the time of the picture.

### 3.3 Dipole and Quadrupole Dampers, RPOS correction

Dampers and a radial position loop have been implemented as cavity feedback objects, as the beam parameters are used to change the cavity parameters (specifically voltage and phase). The class definition is *BoosterCavityDamper* (Note that I have changed the name of the class since the previous version of this documentation) with the following inputs:

- *RFStation* Class  
An RFStation type class
- *Beam* : class  
A Beam type class
- *blond.beam.profile.Profile* : class  
A Profile type class
- *G\_llrf* : float or list  
LLRF Gain [1]; if passed as a float, both dipole and quad damper signals have the same *G\_llrf*; if passed as a list, the first and second elements correspond to the *G\_llrf* of the dipole and quad dampers feedback; default is 0.
- *ROF* : float or list  
Desired radial offset position (mm) for RPOS (next section): if passed as float, then constant through the cycle. If passed as list, turn by turn value of the position
- *RAG* : float of list  
Radial Position Gain (in range [-1:1]) for RPOS (next section) : if passed as float, then constant through the cycle. If passed as list, turn by

The *BoosterCavityDamper* object contains two *BoosterDamperFeedback* objects, one for the dipole and one for the quadrupole. The *BoosterCavityDamper* object is passed to the *RingAndRFTracker* as a feedback method, the *RingAndRFTracker* call the track method of the cavity feedback. The *BoosterDamperFeedback* objects calculate corrections to the synchronous phase  $\phi_S$  (for dipole) and voltage (for quadrupole) which are applied prior to the turn changes from the RF Station settings. I have implemented algorithms based on work done by Valeri Lebedev for previous Booster transition crossing studies [6].

The dipole correction algorithm is:

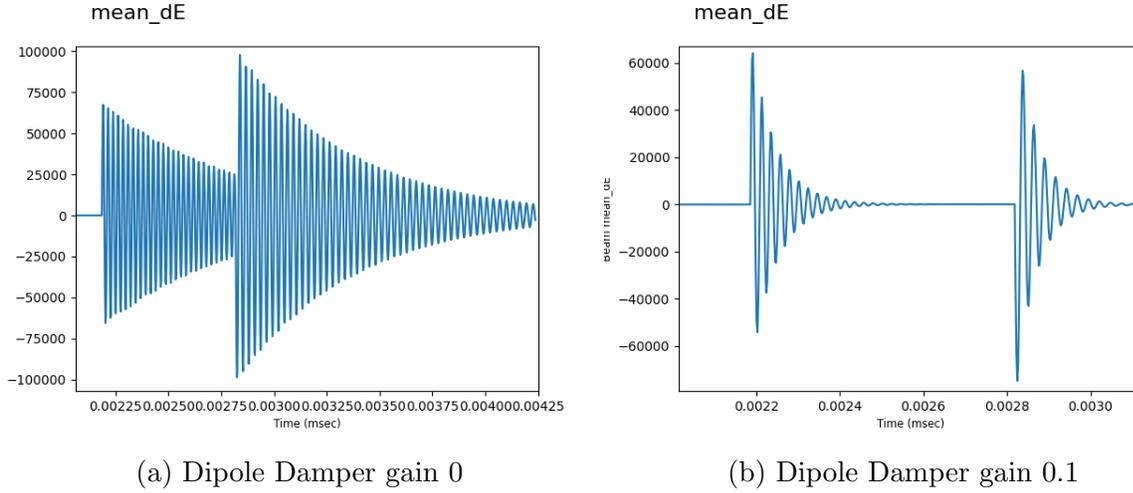


Figure 6: Comparison of beam  $\langle dE \rangle$  for the dipole damper with gain 0 and with gain 0.1. At 0.002 seconds into the cycle, the gap voltage is increased by 30% for 300 turns. The effect of the dipole damper at both steps is clearly visible.

$$\delta\phi_S = \pm \frac{G_{llrf} \times \langle dE \rangle}{V_{gap}} \quad (4)$$

where  $\pm$  represents the sign of  $\eta$ ,  $dE$  is the mean energy of the beam distribution with respect to the synchronous particle, and  $V_{gap}$  is the gap voltage. In Figure 6 I show the effect of the dipole damper on the beam. For 300 turns starting around time of 0.002 seconds, I increased the voltage applied to the beam by 30%. In the figure on the left, there is no dipole damper, while on the right the dipole damper has a gain of 0.1. The difference in the mean energy offset with and without the damper is clearly visible.

The quadrupole correction algorithm is more complicated (and I am not yet convinced I have implemented it properly). A general correction is calculated as:

$$\delta V = G_{llrf} (\sigma_{d\phi}^n - \sigma_{d\phi}^{n-1}) - \frac{\sum_1^{n-1} \delta V}{\tau_Q} \quad (5)$$

where  $\sigma_{d\phi}^k$  is the width for the  $k$ th turn and  $\tau_Q = 2000$ . If the  $\delta V > 0.05$  (5%), it is fixed at 0.05. The gap voltage is then scaled as  $V_{gap}^* = (1 + \delta V)$  and  $\phi_{S+} = \delta V \tan(\phi_S)$ . I am still working on the validation of the quadrupole damper implementation.

The Radial Position loop has been implemented as *BoosterCavityDamper* object

called *BoosterRPOS*. The *BoosterRPOS* objects calculate corrections to the synchronous phase  $\phi_S$  based on an energy offset, as measured in a BPM at a dispersive location. It is modeled on the hardware RPOS corrections in the existing LLRF system (as of October 2020).

The existing hardware uses the S16 BPM, where the dispersion has been measured to be on average 2.985 m with 6% variation through the cycle and uncertainty of 10% [12]. I have chosen to use a constant value of 2.985 m through the cycle, as the uncertainty is larger than the variation. To convert from measured position to a change in  $\phi_S$ , I follow the hardware.

The LLRF hardware devices use a common transform from volts (in the analog circuitry) to degrees (for calculations) of  $9 \frac{\circ}{\text{volt}}$ . The hardware uses the B:PSDRV as the synchronous phase. B:PSDRV is the sum of 2 sources, a provided BDOT signal (a sinusoid) and the B:RPERR signal. From measurements with and without beam, I have reconstructed B:PSDRV as follows:

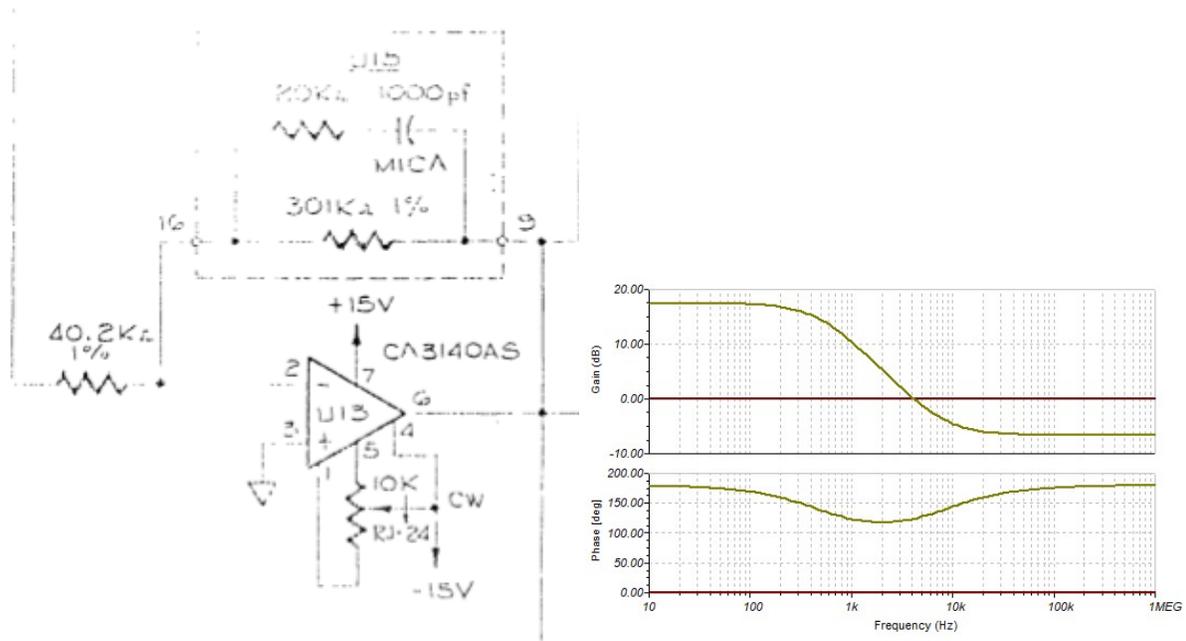
$$PSDRV(\text{volts}) = 4.175 \sin[2\pi f_{\text{ramp}} t] + 3.5 \times RPERR(\text{volts}) \times RAG(t) \quad (6)$$

where  $f_{\text{ramp}}$  is 15 Hz and  $RAG(t)$  is the radial gain curve (B:RAG). The factor of 3.5 will be explained shortly. B:RPERR can be constructed from B:RPOS and B:ROF

$$RPERR(\text{volts}) = 2.4 \times (RPOS(\text{volts}) + ROF(\text{volts})) \quad (7)$$

as documented in the Booster LLRF phase controller drawing (drawing #63397) [13]. The RPERR signal goes through an amplifier where the gain varies as a function of frequency, to filter out the turn by turn variation. The section of the drawing and response are shown in figure 7. The factor of 3.5 (11 dB) comes from a fit of the change in PSDRV from a no-beam measurement (only BDOT) to a beam measurement including RPERR (see figure 8).

There are multiple conversion steps also, to represent the energy deviation (which is from the synchronous particle) to a phase correction. First, I calculate  $\frac{\delta p}{p}$  from the beam  $\langle dE \rangle$  and beam energy. Using the dispersion (2.985m), the radial offset is calculated, then converted to volts (0.184 mm/Volt) and finally to degrees ( $9 \frac{\circ}{\text{volt}}$ ). This phase correction is then added to the value of  $\phi_S$  for the next turn. The sign of the correction is flipped at transition. For modeling purposes, I am going to consider the circuit in figure 7 as a low-pass filter and do exponential averaging of the turn by turn value. With a -3 dB point of 500 Hz, I am using a time constant ( $RC$ ) of 2 msec. The net change at turn  $n$  is therefore:



(a) RPERR amplification section

(b) Gain and Phase as function of frequency

Figure 7: On the left, the amplification section in drawing #63397 for RPERR. On the right, the gain and phase as a function of frequency. This section acts as a low pass filter on the RPERR signal, with -3 dB point at approximately 500 Hz.

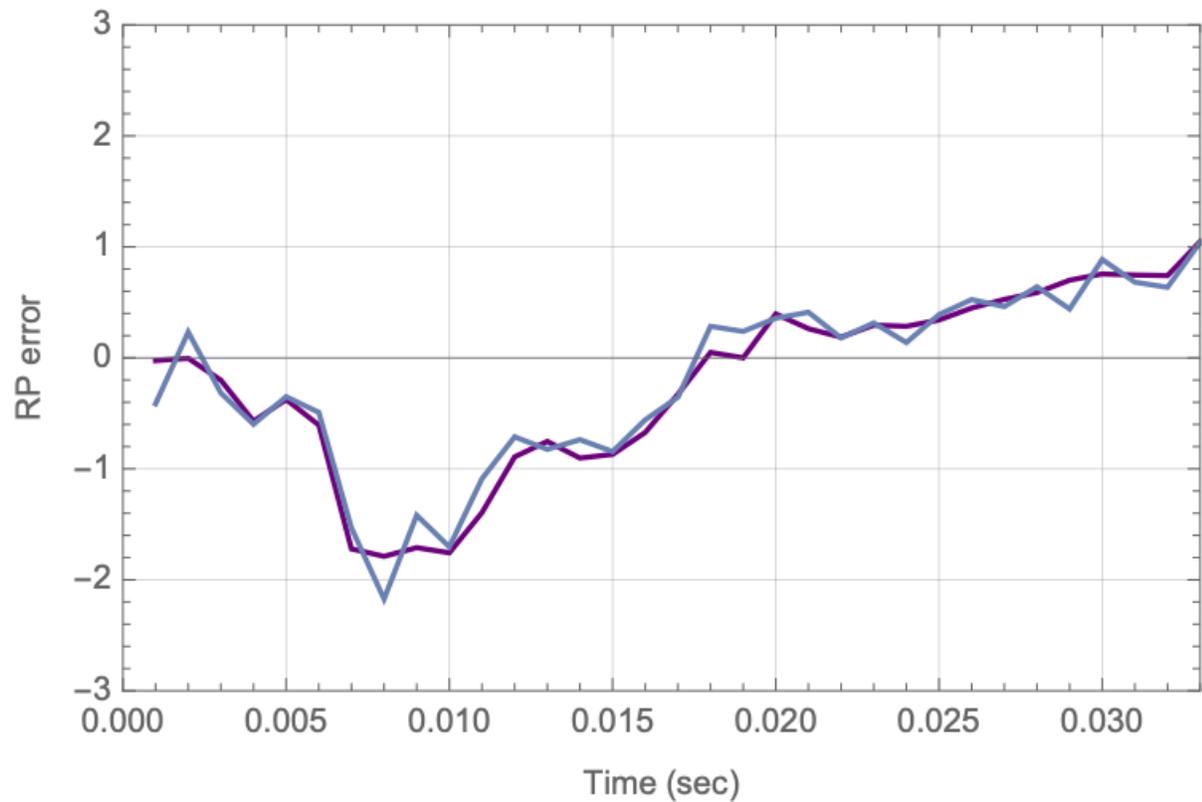


Figure 8: A comparison of scaled B:RPERR (blue) to change in B:PSDRV from a no-beam situation to a beam situation (purple). On average, the amplification from figure 7 is a factor of 3.5.

$$\delta\phi_{S,n} = \pm \frac{t_{rev,n}}{RC + t_{rev,n}} \times 2985mm \times \frac{\text{Volts}}{0.184mm} \times 9 \frac{\circ}{\text{Volt}} \frac{dp}{p} - \sum_{i=1}^{n-1} \delta\phi_{S,i} \quad (8)$$

where  $t_{rev,n}$  is the period for the nth turn and the other terms cover the dispersion and appropriate scaling and the  $\pm$  represents above/below transition.

## 4 Summary

A description of the implementation of BLonD for Booster simulations has been described. The basic structure for a BLonD simulations includes a Ring, Beam, RFStation, and Tracker. Additional elements specific to the Booster, including an injection model, dipole damper, quadrupole damper, radial loop, magnet and space charge impedance terms have been described. Beam loading compensation are under development. This structure forms a basis for simulation studies of existing Booster operation and future operation with the PIP-II Linac. These studies will be described in a separate note.

## References

- [1] M. Ball, et al., "The PIP-II Conceptual Design Report", Sept 2017. [http://pip2-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=113&filename=PIP-II\\_CDR\\_v.0.1.pdf&version=8](http://pip2-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=113&filename=PIP-II_CDR_v.0.1.pdf&version=8)
- [2] J. Adetunji, et al., "Proton Improvement Plan-II Preliminary Design Report", January 2020. <http://pip2-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=2261&filename=PIP-II%20Preliminary%20Design%20Report%20Version%201-17-2020.pdf&version=33>
- [3] See the following presentations the PSP Task Force Meeting:  
<https://beamdocs.fnal.gov/AD/DocDB/0083/008365/001/20200507TaskforceTalk.pptx>  
<https://beamdocs.fnal.gov/AD/DocDB/0085/008539/001/FB-20200702-TaskforceTalk.pptx>  
<https://beamdocs.fnal.gov/AD/DocDB/0085/008576/001/FB-20200716-TX-TaskforceTalk.pptx>
- [4] J. MacLachlan, "Particle Tracking in E-Phi Space for Synchrotron Design & Diagnosis", Fermilab-CONF-92/333 (Nov. 92), presented at 12-th Int'l Conf. on Appl. of

Acc. in Res. and Ind., Denton TX, 4 Nov 1992.  
see also <https://esme.fnal.gov/esmref.html>

- [5] CERN Beam Longitudinal Dynamics code BLonD <http://blond.web.cern.ch>
- [6] V. Lebedev, private communication. Used in simulation documented in reference [8].
- [7] P.F. Derwent, "Understanding the shape of the Booster RF Curve", January 2020  
<https://beamdocs.fnal.gov/AD-private/DocDB/ShowDocument?docid=7965>
- [8] J. F. Ostiguy, C. Bhat and V. Lebedev, "Modeling Longitudinal Dynamics in the Fermilab Booster Synchrotron," Proceedings of the 7th International Particle Accelerator Conference (IPAC2016) Busan South Korea, doi:10.18429/JACoW-IPAC2016-MOPOY013, FERMILAB-CONF-16-162 (2016).
- [9] Presentation February 10 2015, Project X Document Database #1352  
<https://projectx-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=1352&filename=BoosterLongImpedance.pdf&version=1>.
- [10] J. Crisp and B. Fellenz, "Measured Longitudinal Beam Impedance of Booster Gradient Magnets", Fermilab-TM-2145, March 22, 2001 [http://lss.fnal.gov/cgi-bin/find\\_paper.pl?tm-2145](http://lss.fnal.gov/cgi-bin/find_paper.pl?tm-2145).
- [11] J. Edelen and B. Chase, "RF Transient Analysis and Stabilization of the Phase and Energy of the Proposed PIP-II Linac", IEEE Transactions on Nuclear Science, Vol 65, No. 8 (August 2018). The code is available upon request.
- [12] Jeff Eldred, private communication. Results from lattice measurements in fall 2020.
- [13] Booster LLRF Phase Controller Drawing, January 1980 <https://www-bd.fnal.gov/proton/booster/lowlevel/63397/63397.pdf>