FERMILAB-TM-2409-CD

LFSC – Linac Feedback Simulation Code

Valentin Ivanov

Version 1.0 – January 2008

Fermilab, IL 60510, U.S.A.

ABSTRACT

The computer program LFSC (« Linac Feedback Simulation Code ») is a numerical tool for simulation beam based feedback in high performance linacs. The code LFSC is based on the earlier version developed by a collective of authors at SLAC (L.Hendrickson, R. McEwen, T. Himel, H. Shoaee, S. Shah, P. Emma, P. Schultz) during 1990-2005. That code was successively used in simulation of SLC, TESLA, CLIC and NLC projects[1,2,3]. It can simulate as pulse-to-pulse feedback on timescale corresponding to 5-100 Hz, as slower feedbacks, operating in the 0.1-1 Hz range in the Main Linac and Beam Delivery System.

The code LFSC is running under Matlab for MS Windows operating system. It contains about 30,000 lines of source code in more than 260 subroutines. The code uses the LIAR[4] ("Linear Accelerator Research code") for particle tracking under ground motion and technical noise perturbations. It uses the Guinea Pig code to simulate the luminosity performance.

A set of input files includes the lattice description (XSIF format), and plane text files with numerical parameters, wake fields, ground motion data etc. The Matlab environment provides a flexible system for graphical output.

[1] P.N.Burrows et al., Nanosecond-timescale intra-bunch-train feedback for the linear collider, SLAC-PUB-11185, 2004

[2] L.I.Hendrickson et al., Beam-based Feedback for the NLC Linac, SLAC-PUB-10493, 2004

[3] A.Seryi et al., Effects of dynamic misalignment and feedback performance of luminosity stability in linear colliders, SLAC-PUB-9896, 2003

[4] LIAR – A Computer Program for the Modeling and Simulation of High Performance Linacs, SLAC/AP-103, 1997

Contents

1. MATHEMATICAL BACKGROUND

a. The Kalman filter algorithm for optimal control

Kalman filters are based on linear dynamical system discretised in the time domain. They are modeled on a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the visible outputs from the hidden state. The Kalman filter may be regarded as analogous to the hidden Markov model, with the key difference that the hidden state variables take values in a continuous space (as opposed to a discrete state space as in the hidden Markov model). Additionally, the hidden Markov model can represent an arbitrary distribution for the next value of the state variables, in contrast to the Gaussian noise model that is used for the Kalman filter. There is a strong duality between the equations of the Kalman Filter and those of the hidden Markov model. A review of this and other models is given in Roweis and Ghahramani (1999).

In order to use the Kalman filter to estimate the internal state of a process given only a sequence of noisy observations, one must model the process in accordance with the framework of the Kalman filter. This means specifying the matrices $\mathbf{F}_k$, $\mathbf{H}_k$, $\mathbf{Q}_k$, $\mathbf{R}_k$, and sometimes $\mathbf{B}_k$ for each time-step k as described below.

Model underlying the Kalman filter is shown in Figure 1. Circles are vectors, squares are matrices, and stars represent Gaussian noise with the associated covariance matrix at the lower right.

The Kalman filter model assumes the true state at time k is evolved from the state at $(k - 1)$ according to

$$\mathbf{x}_\mathbf{k} = \mathbf{A}\ \mathbf{x}_\mathbf{k-1} + \mathbf{B}\ \mathbf{u}_\mathbf{k-1} + \mathbf{w}_\mathbf{k-1},$$

where

- $\mathbf{A}_k$ is the state transition model which is applied to the previous state $\mathbf{x}_{k-1}$;
- $\mathbf{B}_k$ is the control-input model which is applied to the control vector $\mathbf{u}_k$;
- $\mathbf{w}_k$ is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance $\mathbf{Q}_k$.

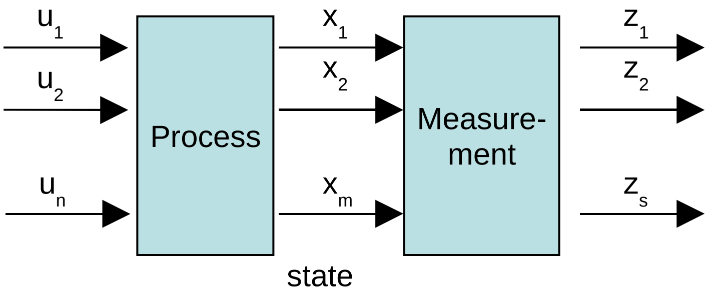
At time k an observation (or measurement) $\mathbf{z}_k$ of the true state $\mathbf{x}_k$ is made according to

$$\mathbf{z}_\mathbf{k} = \mathbf{H}_\mathbf{k}\ \mathbf{x}_\mathbf{k} + \mathbf{v}_\mathbf{k},$$

where $\mathbf{H}_k$ is the observation model which maps the true state space into the observed space and $\mathbf{v}_k$ is the observation noise which is assumed to be zero mean Gaussian white noise with covariance $\mathbf{R}_k$.

The initial state, and the noise vectors at each step $\{\mathbf{x}_0, \mathbf{w}_1, ..., \mathbf{w}_k, \mathbf{v}_1 ... \mathbf{v}_k\}$ are all assumed to be mutually independent.

Many real dynamical systems do not exactly fit this model; however, because the Kalman filter is designed to operate in the presence of noise, an approximate fit is often good enough for the filter to be very useful. Variations on the Kalman filter described below allow richer and more sophisticated models.



Predict the state ahead

$x_k = A\ x_{k-1} + B\ u_k$

Project the error covariance ahead

$P_k = A\ P_{k-1}\ A^T + Q$

Compute the Kalman gain

$K_k = P_k H^T (H\ P_k\ H^T + R)^{-1}$

Update estimate with measurement z_k

$x_k = x_k + K_k\ (z_k – H\ x_k)$

Update the error covariance

$P_k = (I – K_k\ H)\ P_k$

Figure 1. Computational scheme for the Kalman Filter algorithm.

The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state. In contrast to batch estimation techniques, no history of observations and/or estimates is required. It is unusual in being purely a time domain filter; most filters (for example, a low-pass filter) are formulated in the frequency domain and then transformed back to the time domain for implementation. In what follows, the notation represents the estimate of the state at time *n* given observations up to, and including time *m*.

The state of the filter is represented by two variables:

- $\mathbf{x_k}$, the estimate of the state at time *k*;

- **P$_k$**, the error covariance matrix (a measure of the estimated accuracy of the state estimate).

The Kalman filter has two distinct phases: **Predict** and **Update**. The predict phase uses the state estimate from the previous timestep to produce an estimate of the state at the current timestep. In the update phase, measurement information at the current timestep is used to refine this prediction to arrive at a new, (hopefully) more accurate state estimate, again for the current timestep.

Two feedback models were implemented in the code LFSC: general and exponential one.

b. General feedback model in LFSC

General model is described by the system of linear equations

$$\tilde{X}_{k+1} = \Phi\tilde{X}_k + \Gamma u_k + L(y - H\tilde{X}_k)$$

$$\tilde{u}_k = -K\tilde{X}_{k+1} + Nr,$$

$$u_{k+1} = u_k + g(\tilde{u}_k - u_k).$$

where

Φ – system matrix describes the dynamics of accel. Model;
Γ – control input matrix;
L – Kalman filter;
H – output matrix;
K – gain matrix;
N – controller-ref.-input matrix;
X – physical state vector;
u – actuator vector;
r – vector of system set points;
g – gain factor.

c. Exponential model

The corresponding equations for exponential model are

$$\tilde{X}_{k+1} = e_k + w*[\tilde{X}_k - (y - r)] + (y - r),$$

$$\tilde{u}_k = -K\tilde{X}_{k+1} + Nr,$$
$$u_{k+1} = u_k + g(\tilde{u}_k - u_k).$$

where

e$_{\mathbf{k}}$ – exponential change,
y – BPM reading,
w – exponential weight.

It will be shown later that exponential model for some cases has higher convergence rate than general one, but there is a correlation between gain and weight factors to provide the stability of iterations for this model.

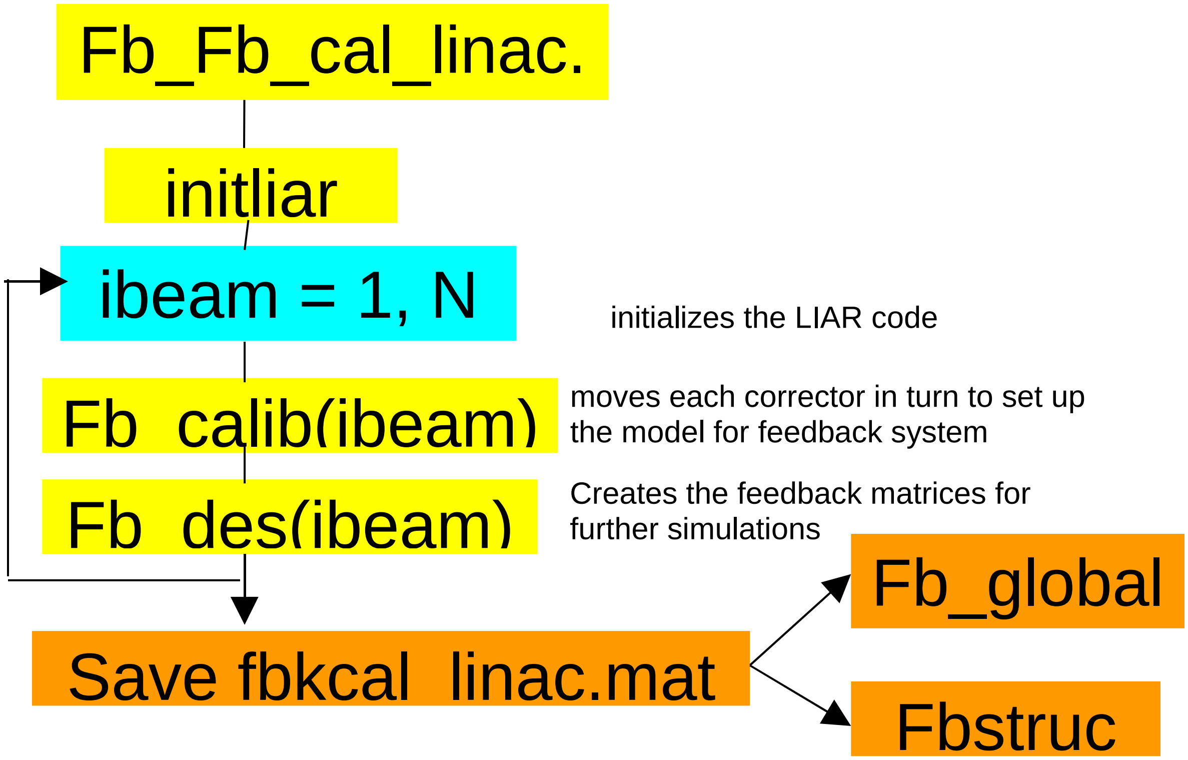
2. THE CODE STRUCTURE

The code LFSC has two separate modules. First one makes the calibration for each feedback loop. It saves the calibration results to some files are used in further simulations. Second one does the linac simulation for N beams, where N=1,2,3. First two beams correspond to electrons and positrons. Setting N=3 means that both beams and Interaction Point (IP) are included into the feedback loop. Each FB loop is an independent variable parameter for optimal control.

The current version of LFSC code has considerable difference from the original SLAC version in respect of data organization. Earlier version had two main disadvantages: 1) the code and data were not totally separated, many data items were embedded in the code; 2) too many input files, many of them contains just one number. Most part of these problems was resolved in current version, but user should make changes in the code, when he changes the BPM and corrector's arrangement in FB loops.

a. Calibration module FB_CAL_LINAC

The module reads a set of initial data using subroutine INITLIAR, then makes the calibration with Fb_CALIB, creates the feedback matrices with Fb_DES, and save the results into the file FBKCAL_LINAC.MAT.



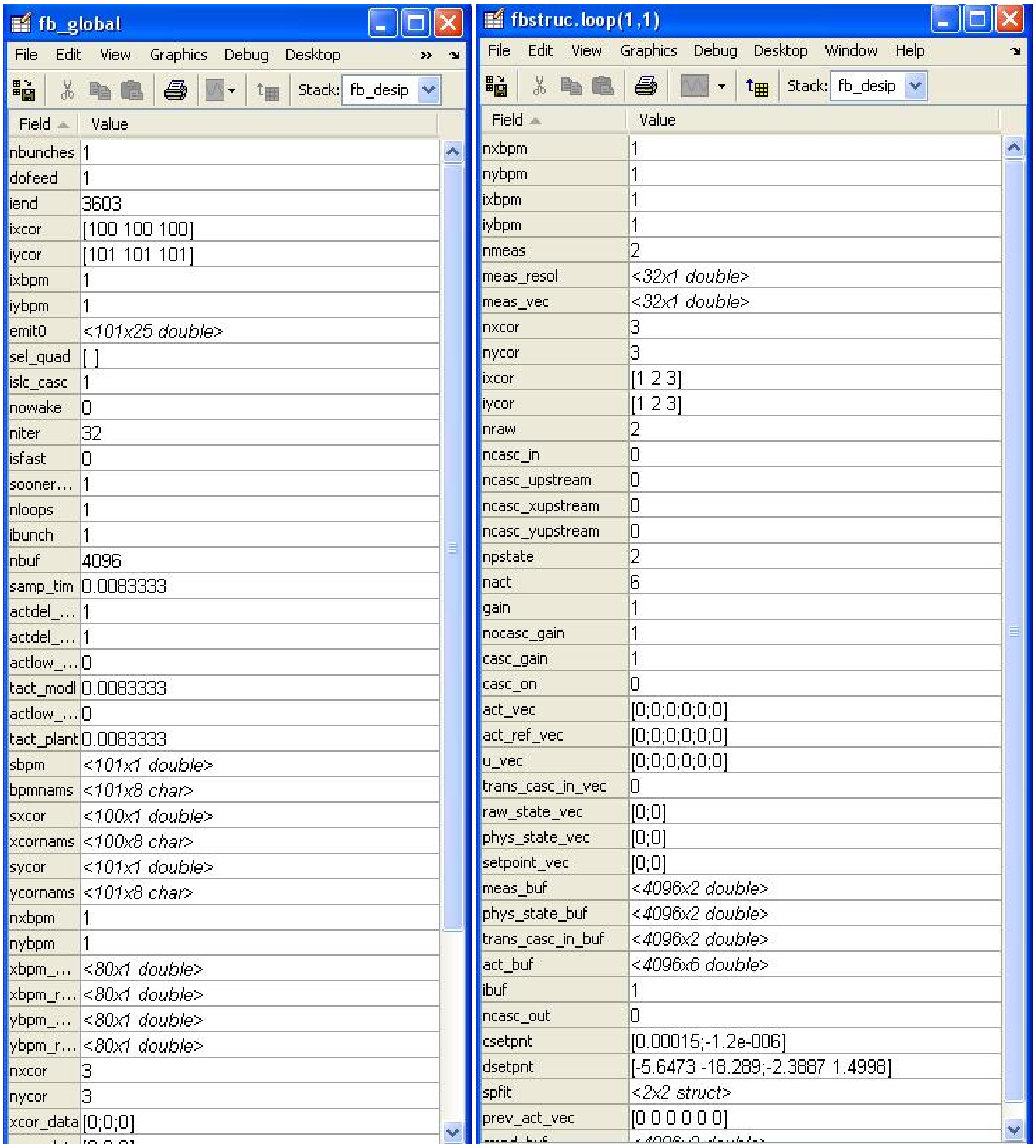
initializes the LIAR code

moves each corrector in turn to set up the model for feedback system

Creates the feedback matrices for further simulations

Figure 2. Flowchart for the calibration module FB_CAL_LINAC.

The file FBKCAL_LINAC.MAT includes two structures of dimension N: Fb_GLOBAL with the parameters set for each beam and FBSTRUC with the feedback data. Structure FBSTRUC of dimension Nloop includes the feedback matrices and other data for each loop.


fb_global
File Edit View Graphics Debug Desktop
Stack: fb_desip
fbstruc.loop(1,1)
File Edit View Graphics Debug Desktop Window Help
Stack: fb_desip

Field	Value
nbunches	1
dofeed	1
iend	3603
ixcor	[100 100 100]
iycor	[101 101 101]
ixbpm	1
iybpm	1
emit0	<101x25 double>
sel_quad	[ ]
islc_casc	1
nowake	0
niter	32
isfast	0
sooner...	1
nloops	1
ibunch	1
nbuf	4096
samp_tim	0.0083333
actdel_...	1
actdel_...	1
actlow_...	0
tact_modl	0.0083333
actlow_...	0
tact_plant	0.0083333
sbpm	<101x1 double>
bpmnams	<101x8 char>
sxcor	<100x1 double>
xcornams	<100x8 char>
sycor	<101x1 double>
ycornams	<101x8 char>
nxbpm	1
nybpm	1
xbpm_...	<80x1 double>
xbpm_r...	<80x1 double>
ybpm_...	<80x1 double>
ybpm_r...	<80x1 double>
nxcor	3
nycor	3
xcor_data	[0;0;0]

Field	Value
nxbpm	1
nybpm	1
ixbpm	1
iybpm	1
nmeas	2
meas_resol	<32x1 double>
meas_vec	<32x1 double>
nxcor	3
nycor	3
ixcor	[1 2 3]
iycor	[1 2 3]
nraw	2
ncasc_in	0
ncasc_upstream	0
ncasc_xupstream	0
ncasc_yupstream	0
npstate	2
nact	6
gain	1
nocasc_gain	1
casc_gain	1
casc_on	0
act_vec	[0;0;0;0;0;0]
act_ref_vec	[0;0;0;0;0;0]
u_vec	[0;0;0;0;0;0]
trans_casc_in_vec	0
raw_state_vec	[0;0]
phys_state_vec	[0;0]
setpoint_vec	[0;0]
meas_buf	<4096x2 double>
phys_state_buf	<4096x2 double>
trans_casc_in_buf	<4096x2 double>
act_buf	<4096x6 double>
ibuf	1
ncasc_out	0
csetpnt	[0.00015;-1.2e-006]
dsetpnt	[-5.6473 -18.289;-2.3887 1.4998]
spfit	<2x2 struct>
prev_act_vec	[0 0 0 0 0 0]

Figure 3. An example of feedback data FB_GLOBAL and FBSTRUC.

Procedure INITLIAR reads the lattice, loads wakefields and makes setup for beam and Twiss parameters. As the result it produces output files for emittance and rays.

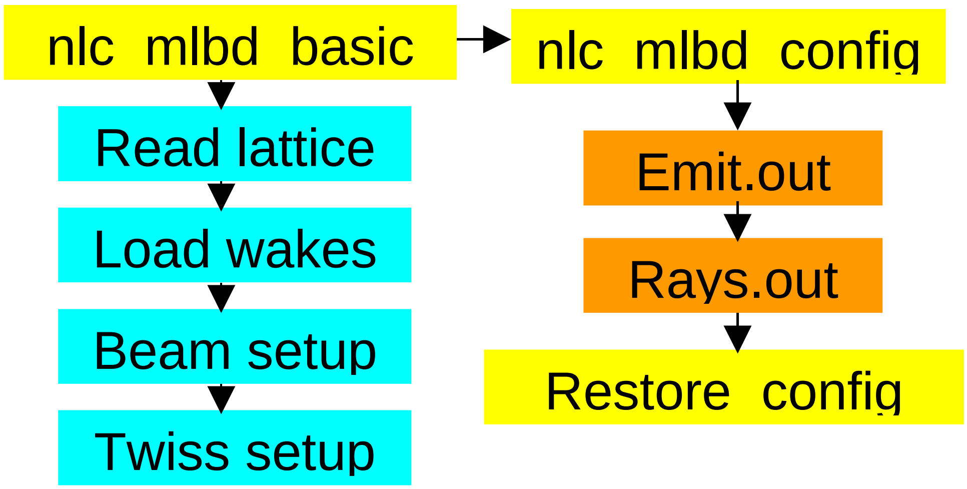


Figure 4. Initialization for the LIAR code.

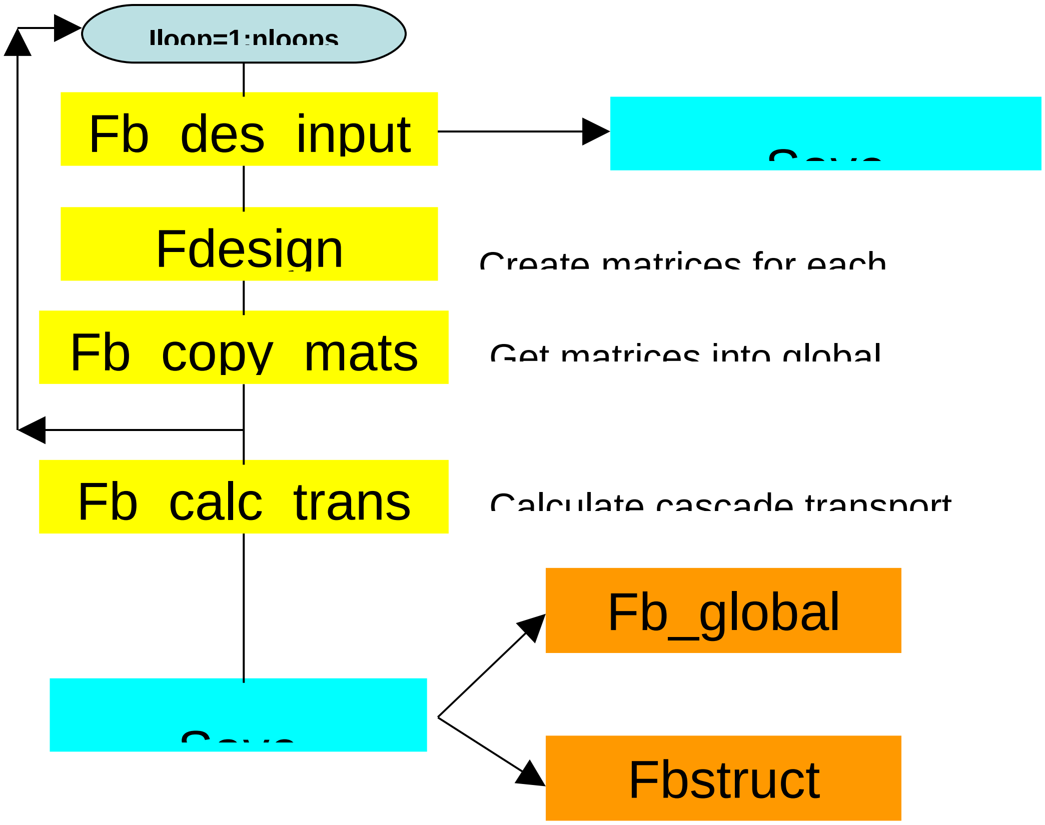


Figure 5. Procedure FB_DES creates the global matrices, and saves the results into structures FB_GLOBAL (global parameters) and FBSTRUCT (feedback structure).

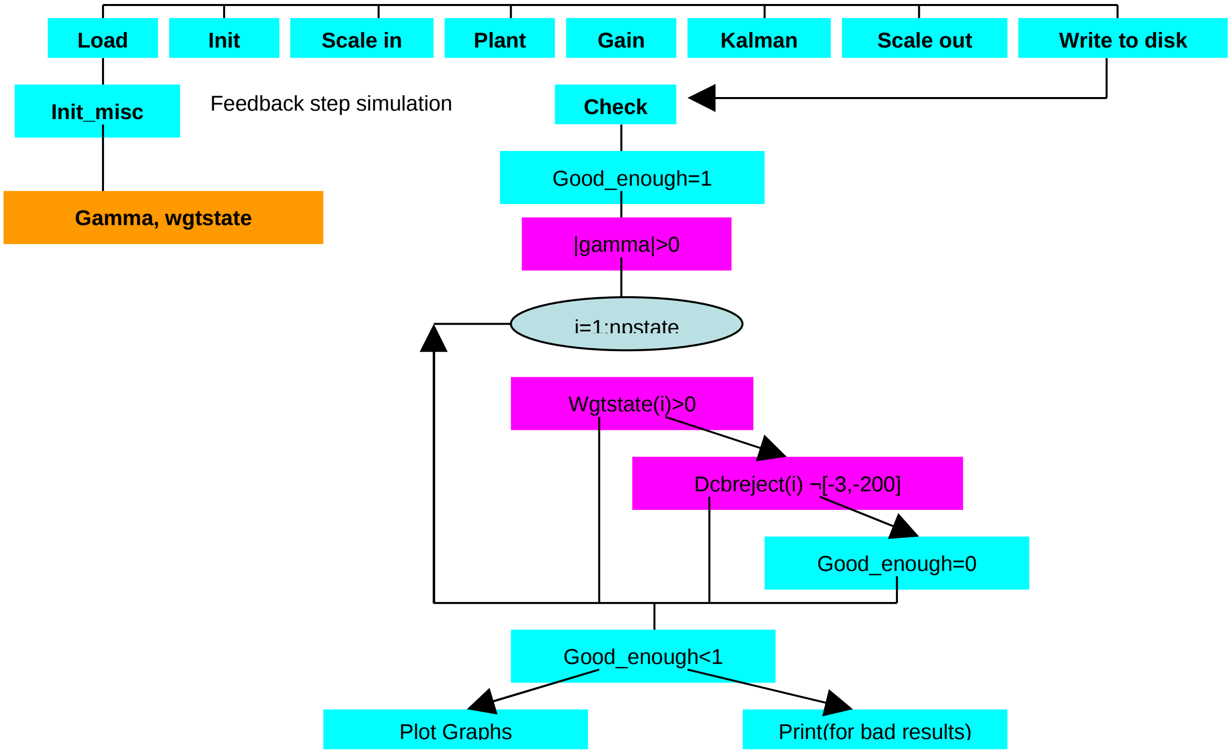


Figure 6. Routine FDESIGN creates the global matrices for FB simulations.

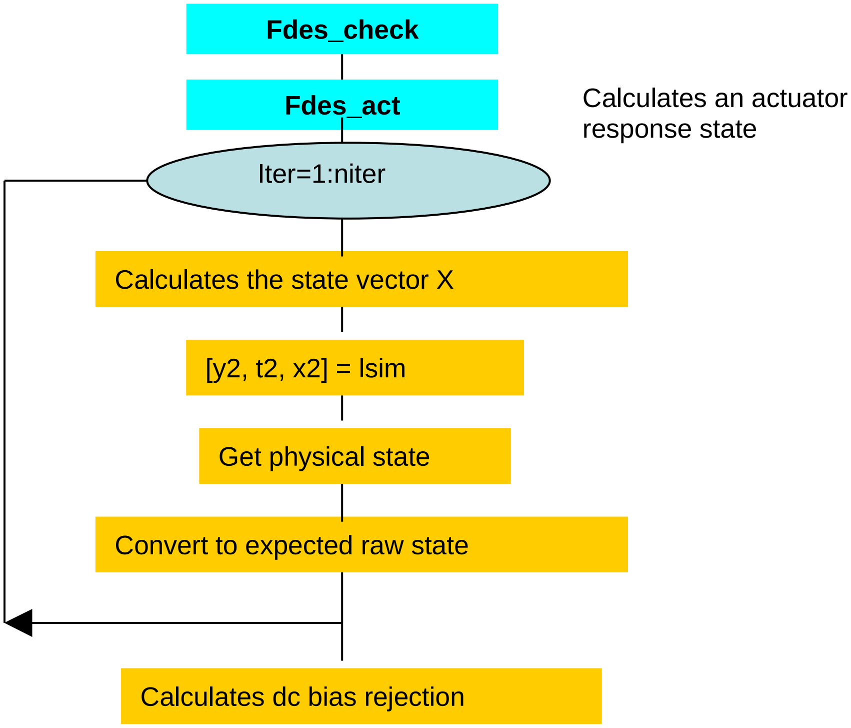

Fdes_check
Fdes_act
Calculates an actuator response state
Iter=1:niter
Calculates the state vector X
[y2, t2, x2] = lsim
Get physical state
Convert to expected raw state
Calculates dc bias rejection

Figure 7. Routine FDES_CHECK – feedback step simulation.

b. Simulation module SIM2

Module SIM2 provides the beam based feedback simulation for a linac with Nbeams. Parameter Nbeams=1 corresponds to a single beam linac, Nbeams=2 – electron-positron linac, Nbeams=3 – IP simulation to maximize the luminosity factor. The algorithm makes particle tracking pulse-by-pulse up to Npulses with using subroutine SIM2_ITER, then it saves the results into two files. File SAVESIM.MAT includes the results of last run: two structures FB_STRUC described above, and FB_RESULTS, which keep the data for beam position and emittance for each pulse. It will be renewed in the next run.

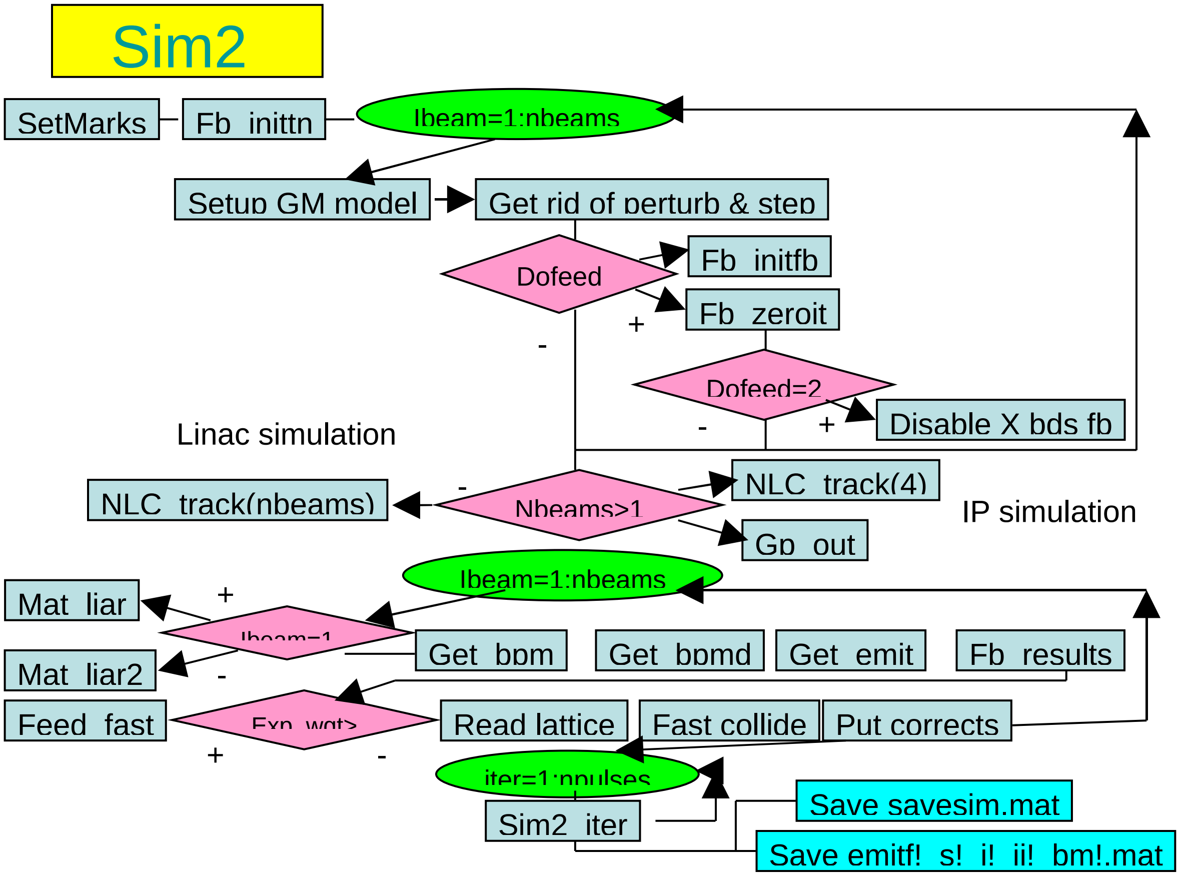


Figure 8. Flowchart for linac simulation module SIM2.

The file EMIT<…>.MAT keep the results for emittance evolution. It has the additional parameters in its name:

- Fn – n=0 – no feedback; n=1 – FB loops in main linac (default); n=2 – includes BDS cascades; n=3 – dispersion correction (loads the dispersion initial data); n=4 – measure dispersion every 40 pulses on 10th pulse, dispersion correction;
- Sn – n – number of GM seeds;
- In – mode of quad pulse jitter (n=0,…,18; 0 – no jittering);

- IJn – mode of klystron jitter (n=0,…,10; 0 means no jittering);
- BMn – n – number of beams.

This trick makes it possible to keep a bunch of different results for different set of input parameters with no rename of files.

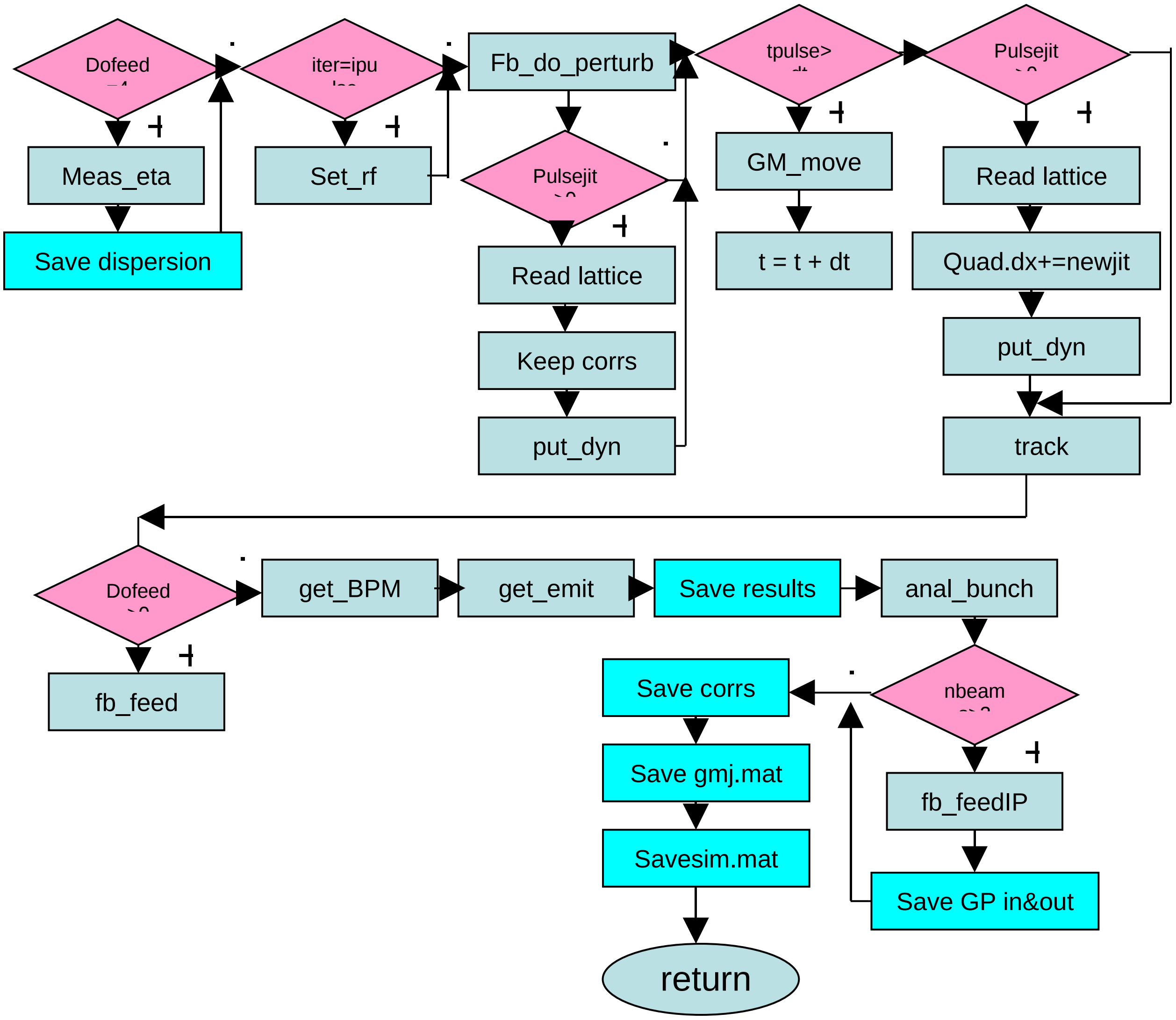


Figure 9. Routine SIM2_ITER provides the feedback iterations.

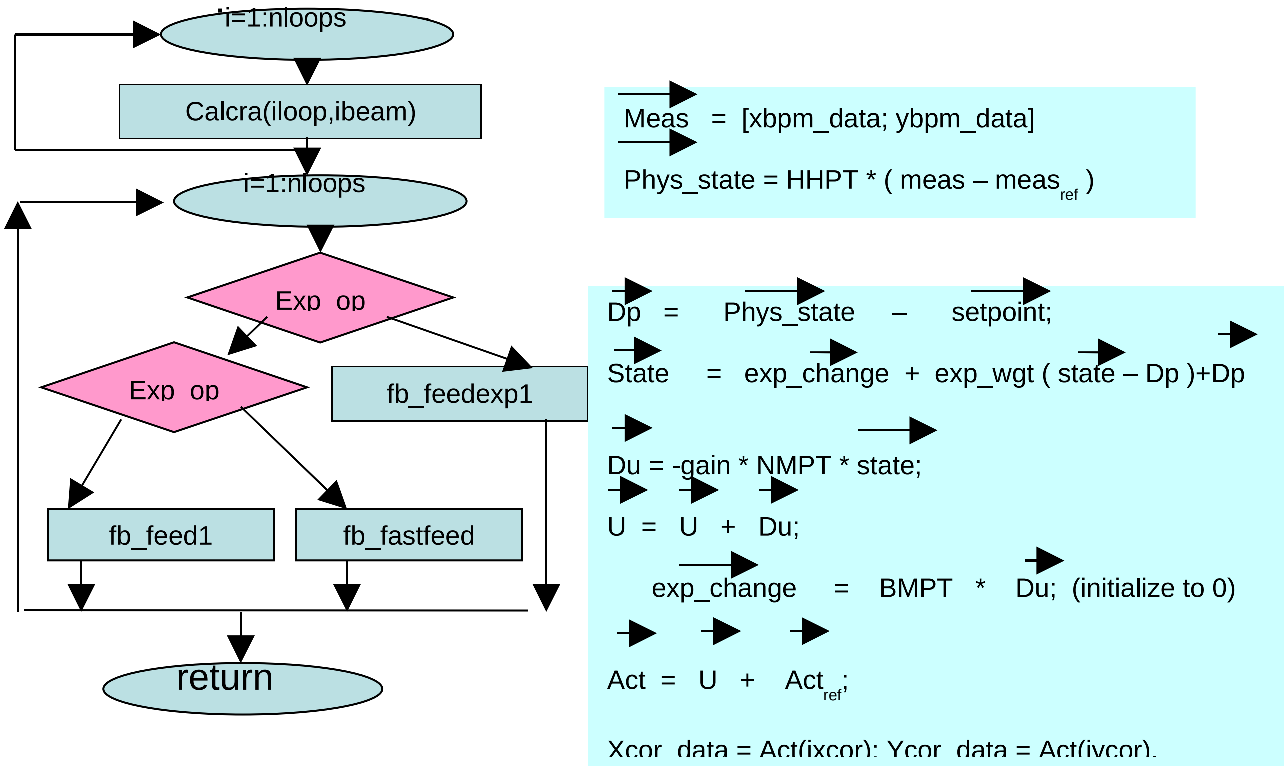

'i=1:nloops
Calcra(iloop,ibeam)
i=1:nloops
Exp op
Exp op
fb_feedexp1
fb_feed1
fb_fastfeed
return
Meas = [xbpm_data; ybpm_data]
Phys_state = HHPT * (meas – meas_{ref})
Dp = Phys_state – setpoint;
State = exp_change + exp_wgt (state – Dp)+Dp
Du = -gain * NMPT * state;
U = U + Du;
exp_change = BMPT * Du; (initialize to 0)
Act = U + Act_{ref};
Xcor_data = Act(ixcor); Ycor_data = Act(iycor).

Figure 10. Routine FB_FEED calculates the correction vector for different FB models.

Bunch.out	Bunch data: SLICE#, X, XANG, Y, YANG, EX, EY, E, Q, Z, NE, XX, XXA, XAXA, YY, YYA, YAYA, XSIZ, XPSZ, YSIZ, YPSZ
Emit.out	Tracking data for beam1: N, S, X, Y, EMX, EMY, PSIX, PSIY, BETAX, BETAY, BMAGX, BMAGY, XLUMR, YLUMR, E, SIGE/E, SIGX, SIGXP, SIGY, SIGYP, SIGE, ALPHAX, ALPHAY, BEAMYAW, BEAMPITCH
Emit0.out	Similar tracking data for unjittered reference beam (X=Y=0.0)
Train.out	Train data: # BUNCH, X, XANG, Y, YANG, JX, JY, EX, EY, E, Q, EX_ALL, EY_ALL, YNORM, YPNORM, XSIZ, YSIZ, XPSIZ, YPSIZ
Bpmnames.out	List of names for the BPM’s
Xcornames.out	List of names for X-correctors
Ycornames.out	List of names for Y-correctors
Matliarlog.out	Log-file for LIAR code running
XSIF.ERR	Xsif parser error-file
XSIF.STR	Xsif parser log-file
Xsiftemp.out	Temporary output file for parser
gm.out	Ground motion histograms

Figure 11. Structure of the output files for the code LFSC.

3. **GETTING START**

We assume that user can get all the information about the LIAR code from its User's Guide.

Typical configuration of file system for the code LFSC has the main folder which contains a set of subfolders:

- MFILES – Matlab source code for LFSC program;
- MFILES_WINDOWS – wrappers, interface for the LIAR code;
- FDESIGN – Matlab source code for the Kalman Filter algorithms;
- INFILES – common initial data (Ground Motion models A,B,C,J,K; lattice files with extension XSIF; longitudinal and transversal wakes; energy dispersion, GM seeds);
- RUN<n> - set of initial data and results for different runs.

You should create new folder for a new problem, then copy there the following set of files from any RUN-folder:

- FB_CAL_LINAC.M, SIM2.M – calibration and simulation modules;
- MATLIAR.DLL, MATLIAR2.DLL – dynamic linked libraries for the LIAR code;
- ELECTRONC.CFG, POSITRONC.CFG – configuration settings for electron and positron beams;
- TRLRFILE.DAT – path and name for the wakefield data;
- XSIFLINE.DAT – the name of beamline at corresponding lattice file;
- PARAMS.TXT – file with the global parameters of the problem.

We provide the example of PARAMS-file with brief comments:

```
jitter pulsejit model itime time_pulse wakes beams bunches pulses dofeed loops seeds BPMres gain exp_wgt espread bpmx bpmy
quadx quady
  0      0     "B"     0     0.2          0      1       1     50       1      5    10      0.1     0.8     0.33      1        0.0  300.0
  0.0    0.0
```

Here

- JITTER – klystron jitter model (0 – no jittering);
- PULSEJIT – pulse jitter model (0 – no jittering);
- MODEL – Ground Motion model (model "B");
- ITIME –
- TIME_PULSE – repetition rate for bunch train (0.2 sec = 5Hz);
- WAKES – number of wake files (0 – no wakes);
- BEAMS – number of beams (1 – main linac simulation);
- BUNCHES – number of bunches in a train (1 bunch);
- PULSES – number of pulses (50 pulses - 50*0.2 = 100 seconds);
- DOFEED – FB mode (1 – FB for main linac, no dispersion correction);
- LOOPS – number of FB loops (5 loops);

- SEEDS – number of GM seeds (10 seeds);
- BPMres – BPM resolution (0.1 um);
- GAIN – gain factor for FB model (0.8);
- EXP_WGT – exponential weight (0.33; zero value means general FB model);
- ESPREAD – energy spread (1% of nominal energy of beam);
- BPMX – x-BPM initial misalignment (0 – perfect alignment);
- BPMY – y-BPM initial misalignment (300 um);
- QUADX – x-quad initial misalignment (0 – perfect alignment);
- QUADY - y-quad initial misalignment (0 – perfect alignment).

The LFSC run includes the following steps:

1. Run the Matlab application;
2. Enter the appropriate Run directory;
3. Load FB_CAL_LINAC.M file;
4. Load INITLIAR1.M file and make needed changes in BPM/Corrector arrangement at lines 233-241, in FB_SETUP.M at lines 106-109 also;
5. Select the FB_CAL_LINAC.M file and run it using hot key F5 or menu item “Debug/Run”;
6. Select the file SIM1.M to run linac simulation or file SIM2.M to run IP simulation, then run it;
7. The simulation modules produce graphical results for beam position and emittance using PLOT_RESULTS.M. Load it and make needed changes when you want to get another or additional graphics.

4. THE RESULTS OF NUMERICAL SIMULATION FOR ILC PROJECT

The results presented here include some simple benchmarks and realistic simulations for short lattice (50 FODO cells) and entire ILC lattice (114 FODO cells). These calculations give an impression on the efficiency of beam based feedback algorithms implemented in the code LFSC.

Figure 12 presents typical layout for some NLC FB structure.

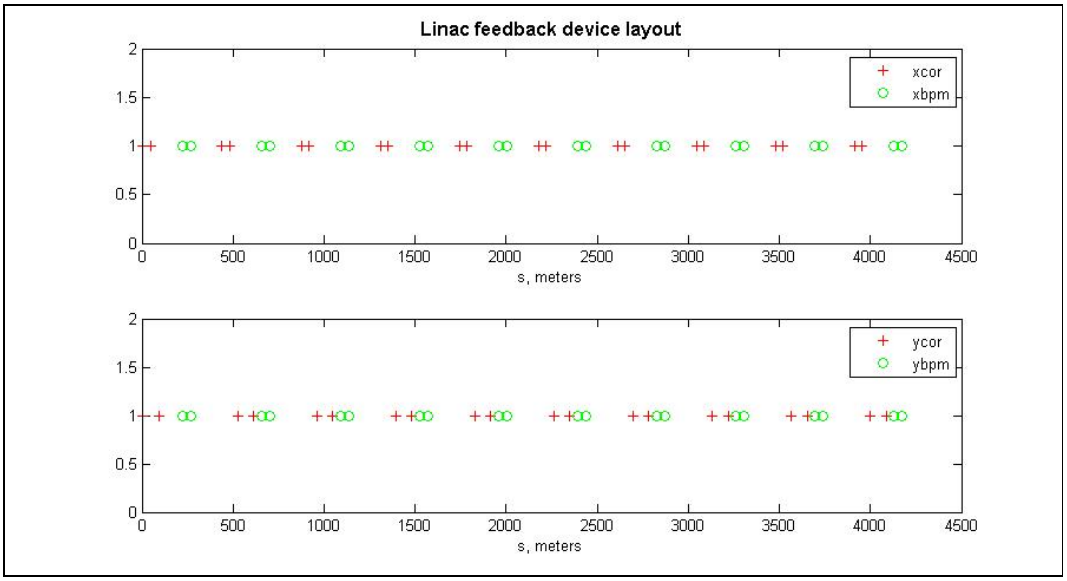


Figure 12. FB device layout for 10 loops of 2 correctors (red cross) and 2 BPMs (green circle) each in x and y directions.

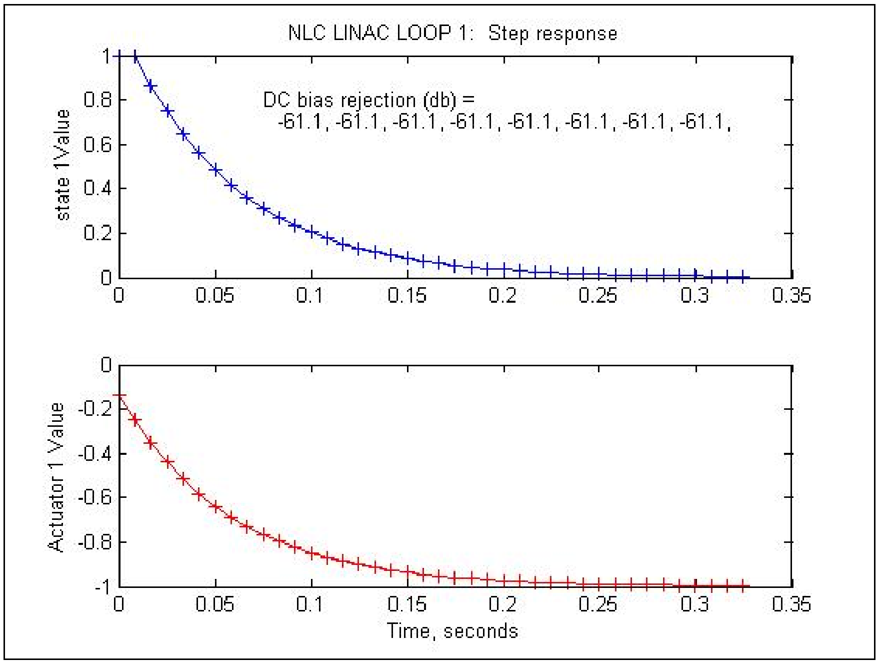


Figure 13. The response for stepwise perturbation for each FB loop.

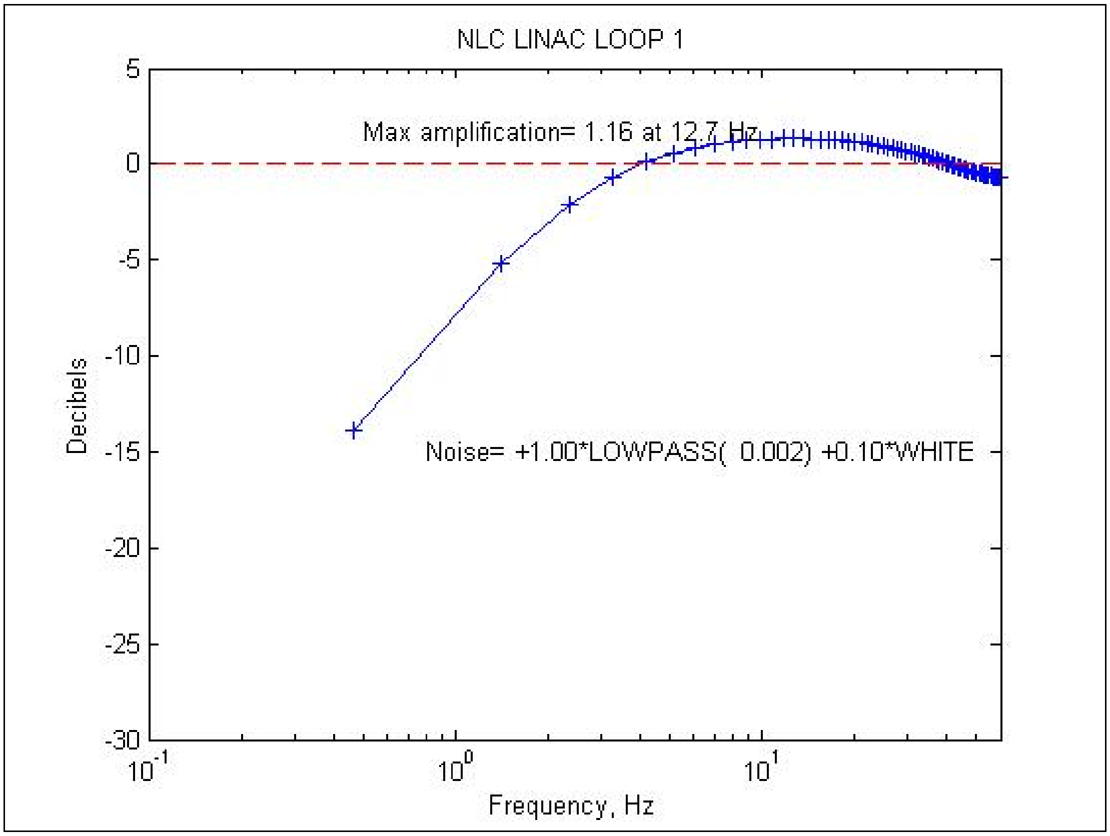


Figure 14. Parameters of low pass Kalman Filter.

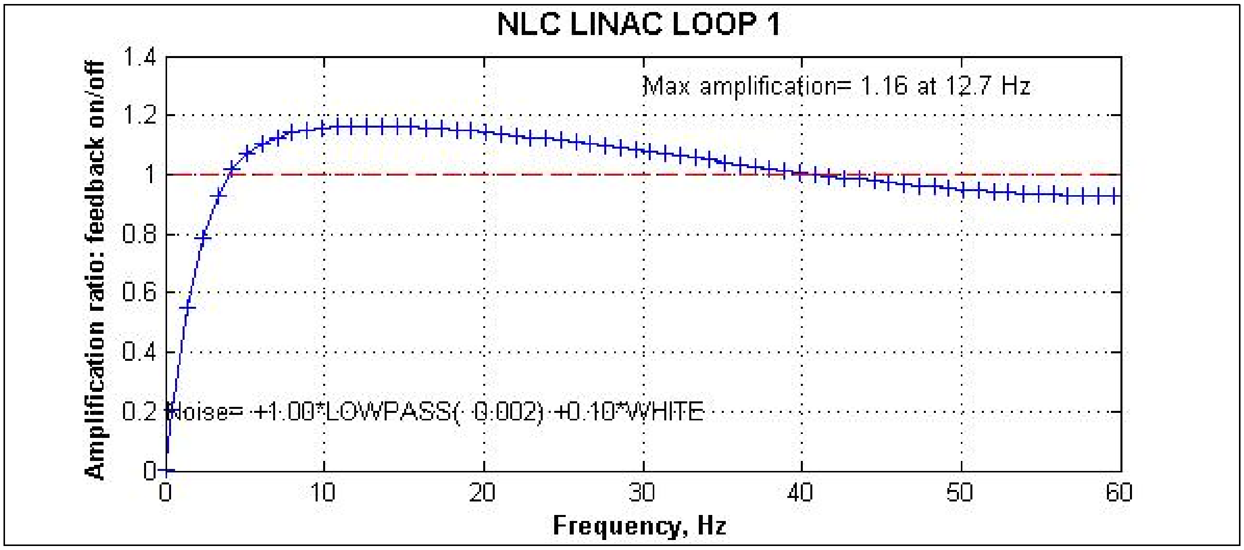


Figure 15. Amplification ratio. Red – FB off; blue – FB on.

Figures 13-15 represent the step response and parameters of Kalman Filter for each of feedback loops.

a. Simple benchmarks

We present one of the simplest tests, which include one FB loop with 2 correctors and 4 BPMs in each direction. The lattice includes total number of 100 correctors and 101 BPMs. Train repetition rate is 5 Hz.

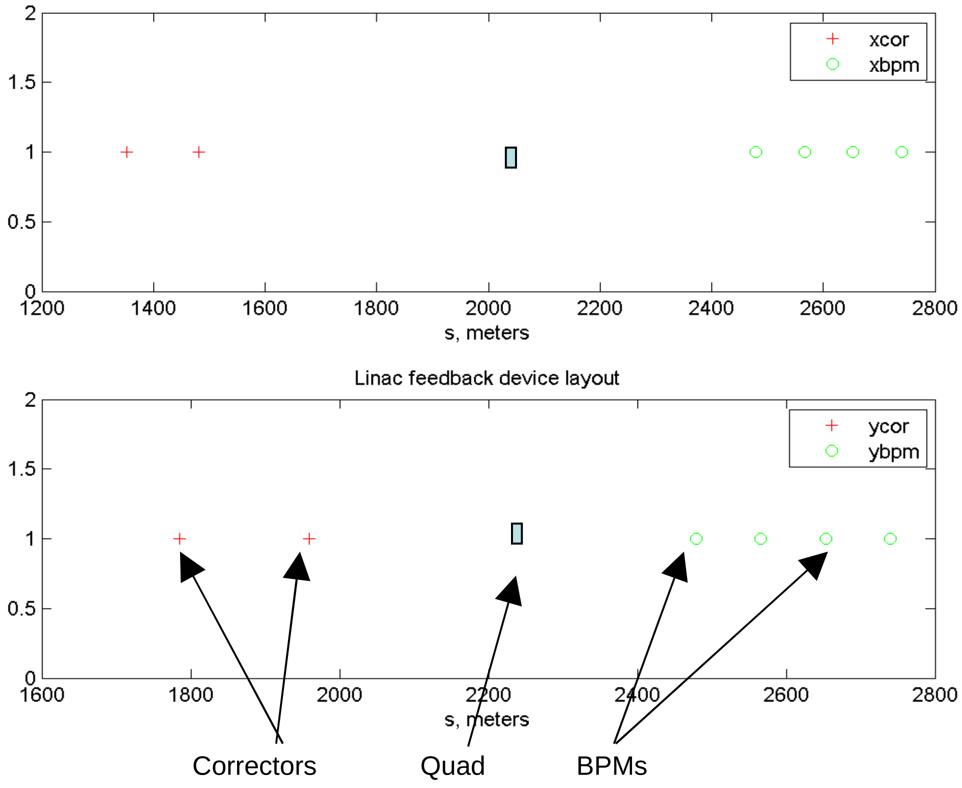


Figure 16. Simple test for short ILC structure with one FB loop.

Initially the lattice is perfectly aligned. The perturbation is made by the quad number 50. It is moving with harmonic law in y-direction as Y_{50} = A cos(2π F t) with amplitude A=25μm and varying frequency F=0-0.5Hz.

i. **Static perturbation, perfect BPM resolution**

Figure 17 shows static response (F=0) for stepwise perturbation depending of varying gain factor G of general Kalman Filter model. Optimal value G=1 takes 35-37 pulses to reach more or less good aligned state.

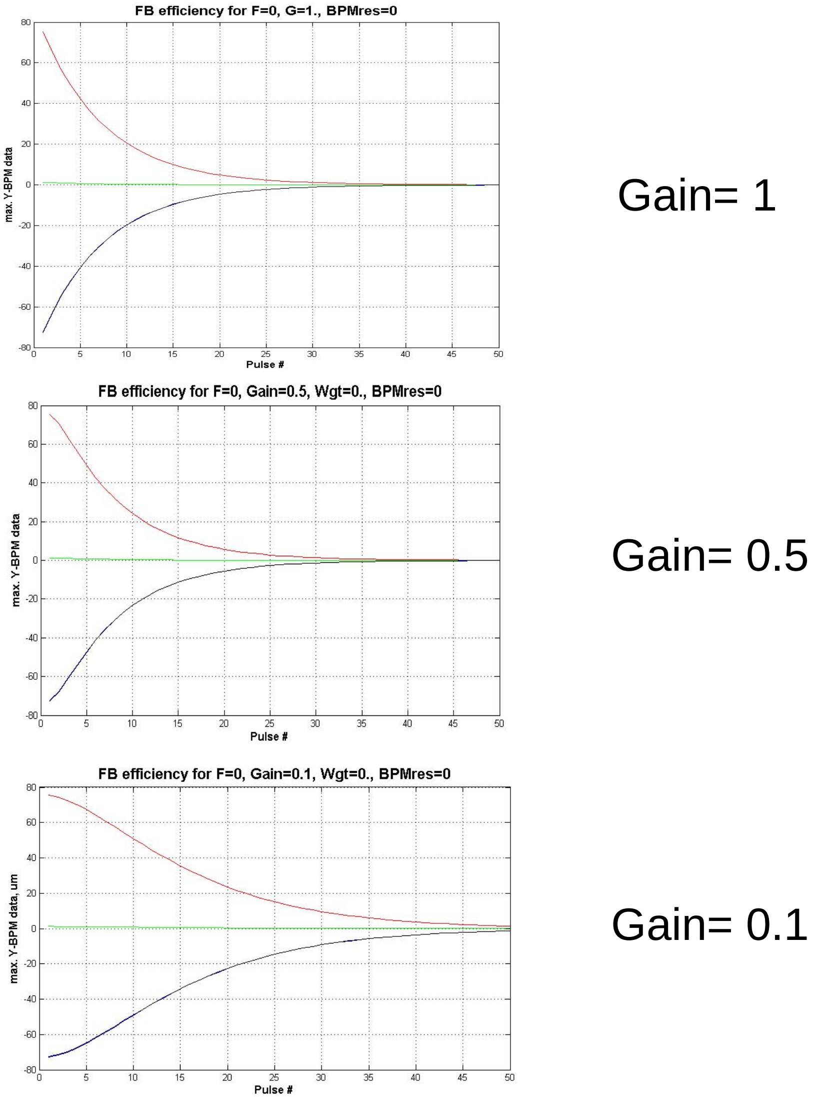


Figure 17. Static response vs. gain for general FB model. Different colors correspond to different BMPs. Two of them work in opposite phases.

The origin of FB delay is determined by the influence of limited pass band of used Kalman Filter. It couldn’t be substantially reduced. Our results are in good agreement with publications[2] and[5].

[5] I.Reyzl, Simulation of Feedback for Orbit Correction, EPAC’96.

ii. Dynamic response

By varying the perturbation frequency F, we can detect that upper limit for effective FB control is factor of 50 less than repetition rate, as it shown at Figure 18.

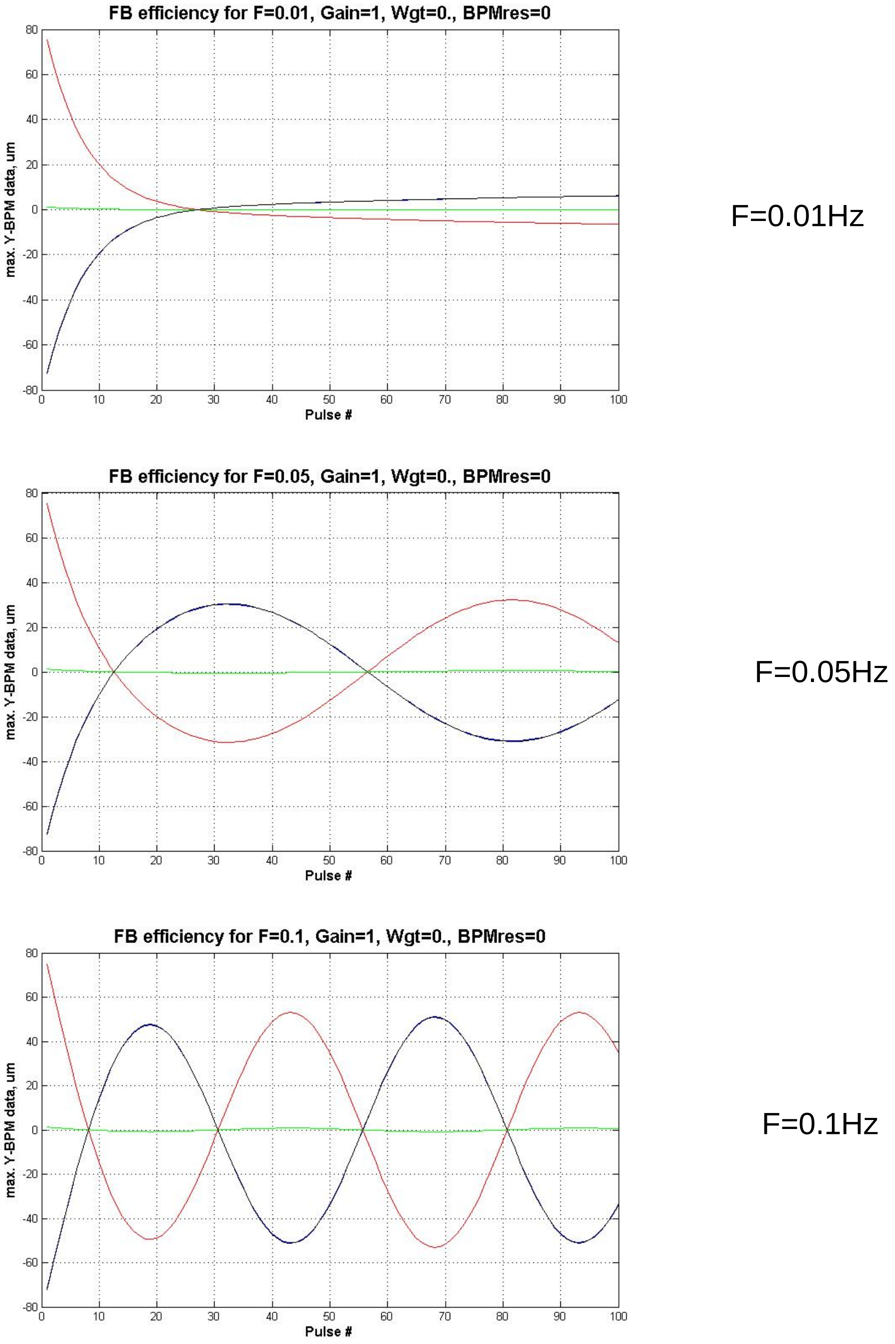
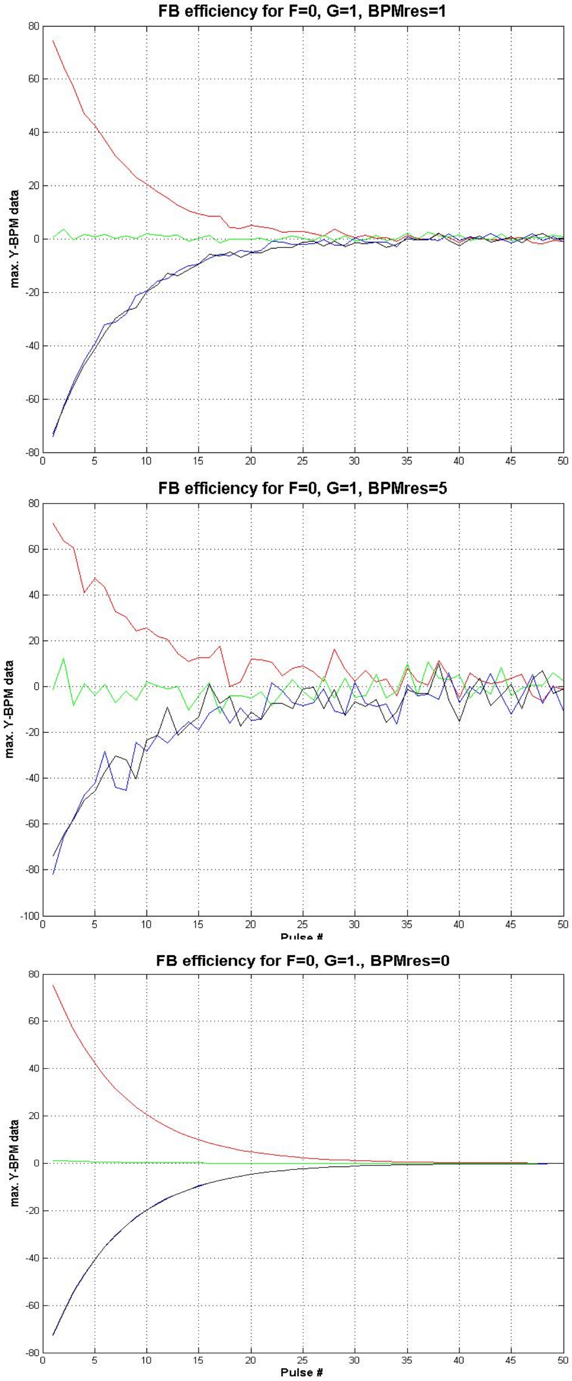


Figure 18. Dynamic response for general model of Kalman Filter.

iii. Effect of BPM resolution

Our simulations show that BPM errors can be effectively suppressed by FB system, when resolution value is 1 micrometer or less.


FB efficiency for F=0, G=1, BPMres=1

Res = 1um

FB efficiency for F=0, G=1, BPMres=5

Res = 5um

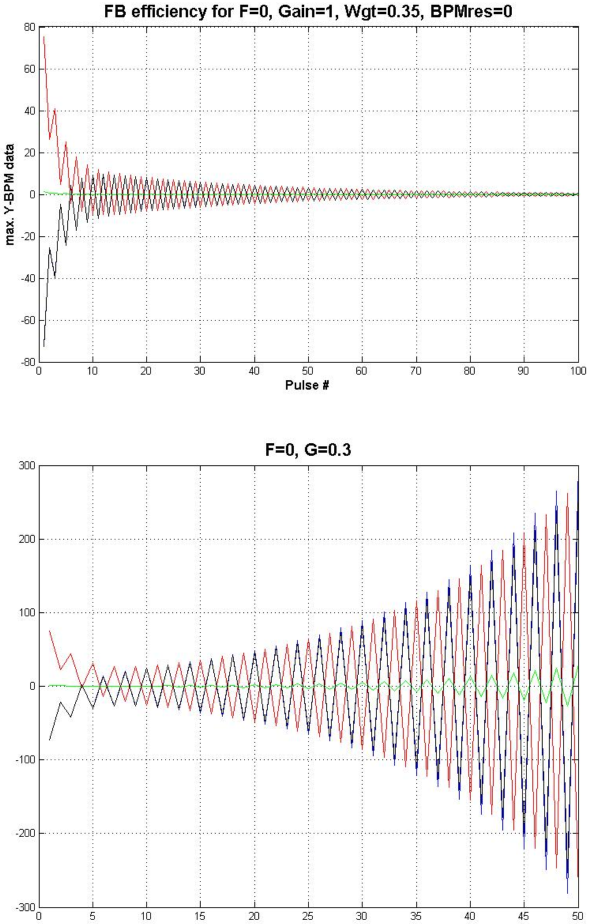
FB efficiency for F=0, G=1., BPMres=0

Res = 0.1um

Figure 19. FB efficiency vs. BPM resolution for static perturbation.

iv. Stability of exponential model vs. Gain and Weight factors

Our first runs for exponential FB model showed that unsuitable gain and weight factors can produce the divergence in FB control. From the other hand, converged process can produce big oscillations.



Convergence. W = 0.35

Divergence. W = 0.3

Figure 20. Different behavior of FB control for gain factor G=1.

Further study shows the lowering of G-factor can effectively suppress those oscillations, and there is strong correlation between gain and weight factors, which can provide stable control and effectively suppress the oscillations. Figures 21-22 demonstrate these features of exponential model. Comparing more flexible exponential model with general one, one can see it takes 13-14 pulses to reach good alignment for static response versus 35-37 pulses for general model.

Figure 23 demonstrate the efficiency of exponential model for harmonic perturbations of different frequency.

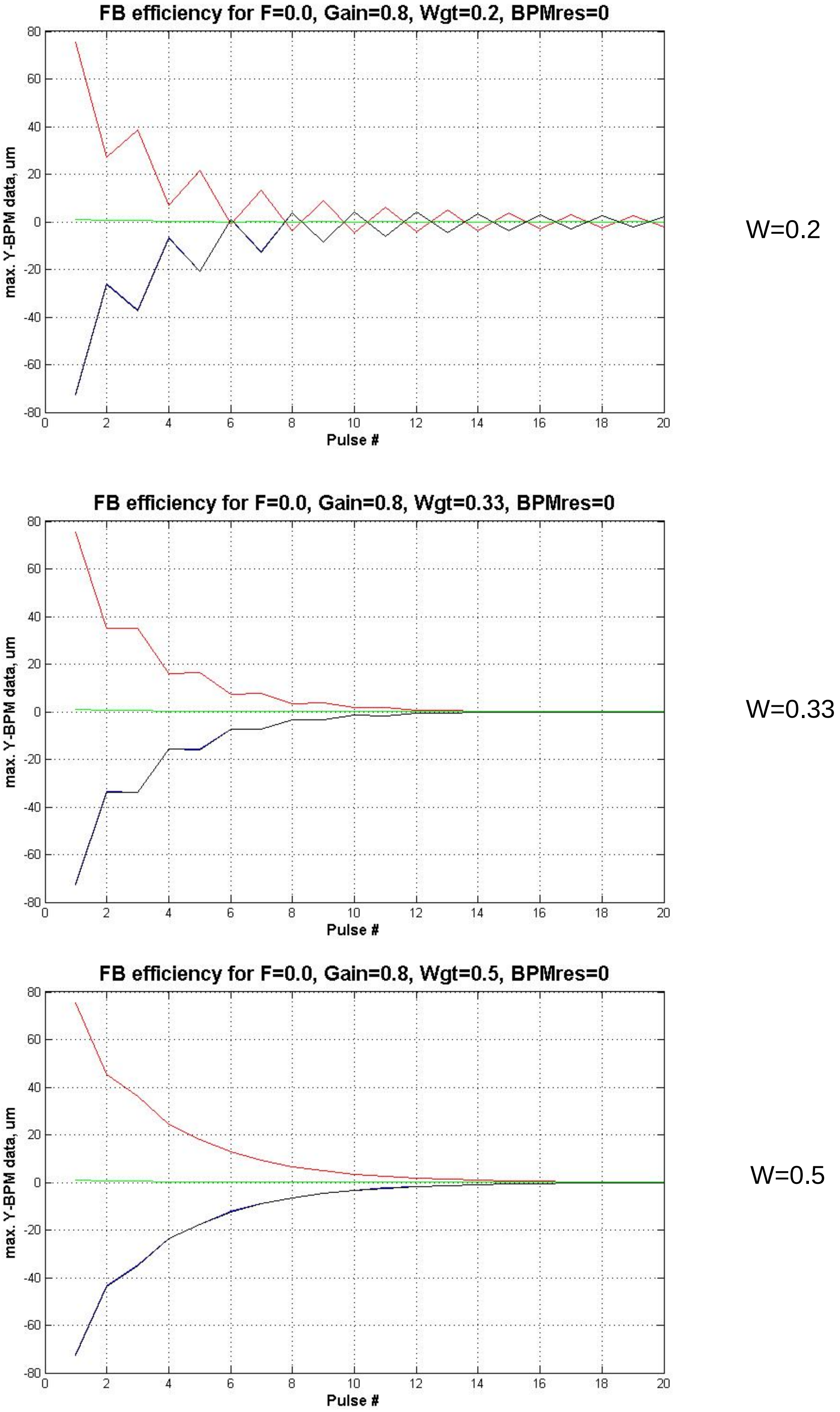
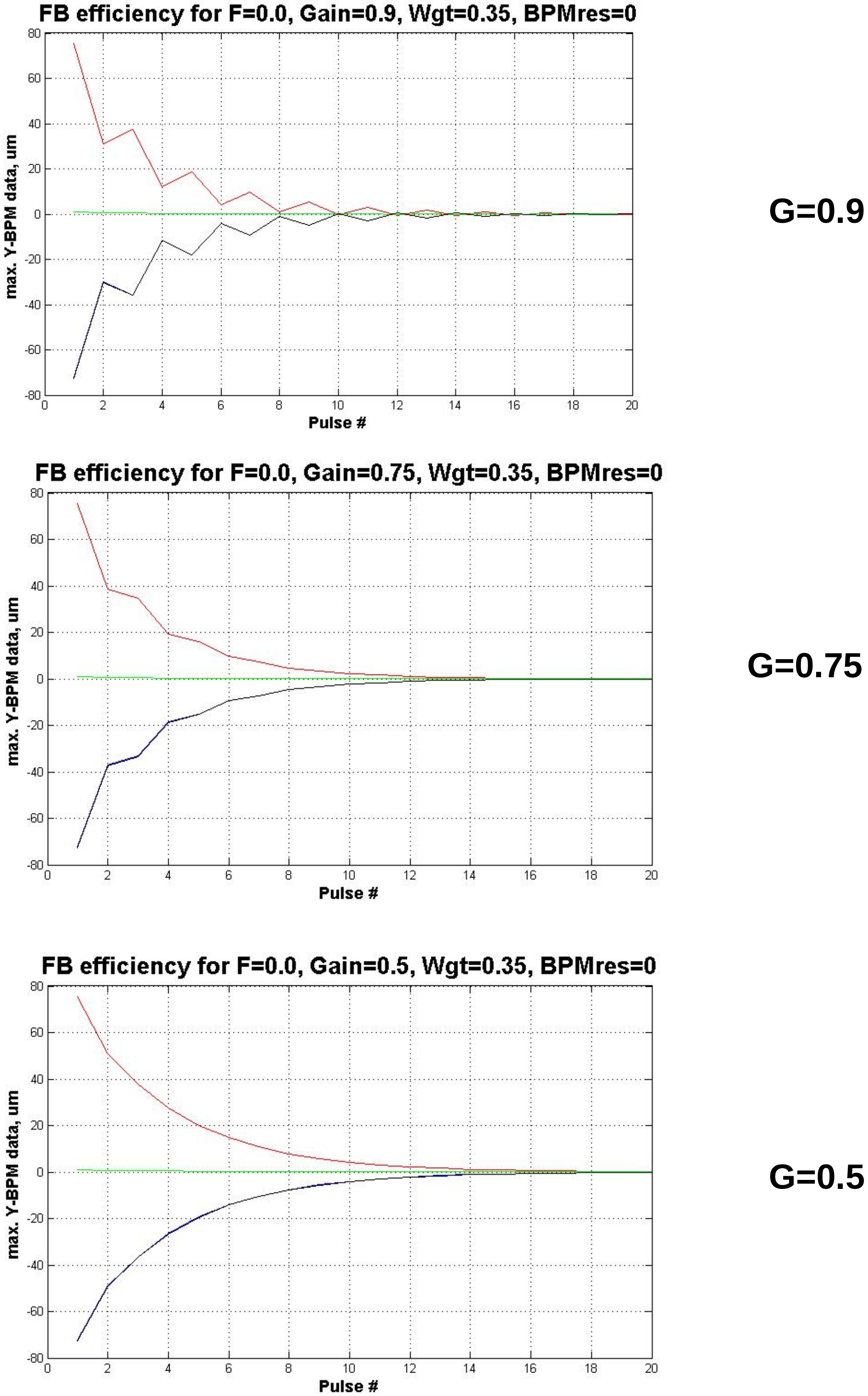


Figure 21. Effect of Weight factor for static perturbation. Gain factor G=0.8.


FB efficiency for F=0.0, Gain=0.9, Wgt=0.35, BPMres=0

FB efficiency for F=0.0, Gain=0.75, Wgt=0.35, BPMres=0

FB efficiency for F=0.0, Gain=0.5, Wgt=0.35, BPMres=0

Figure 22. Effect of Gain factor. Weight factor W=0.33.

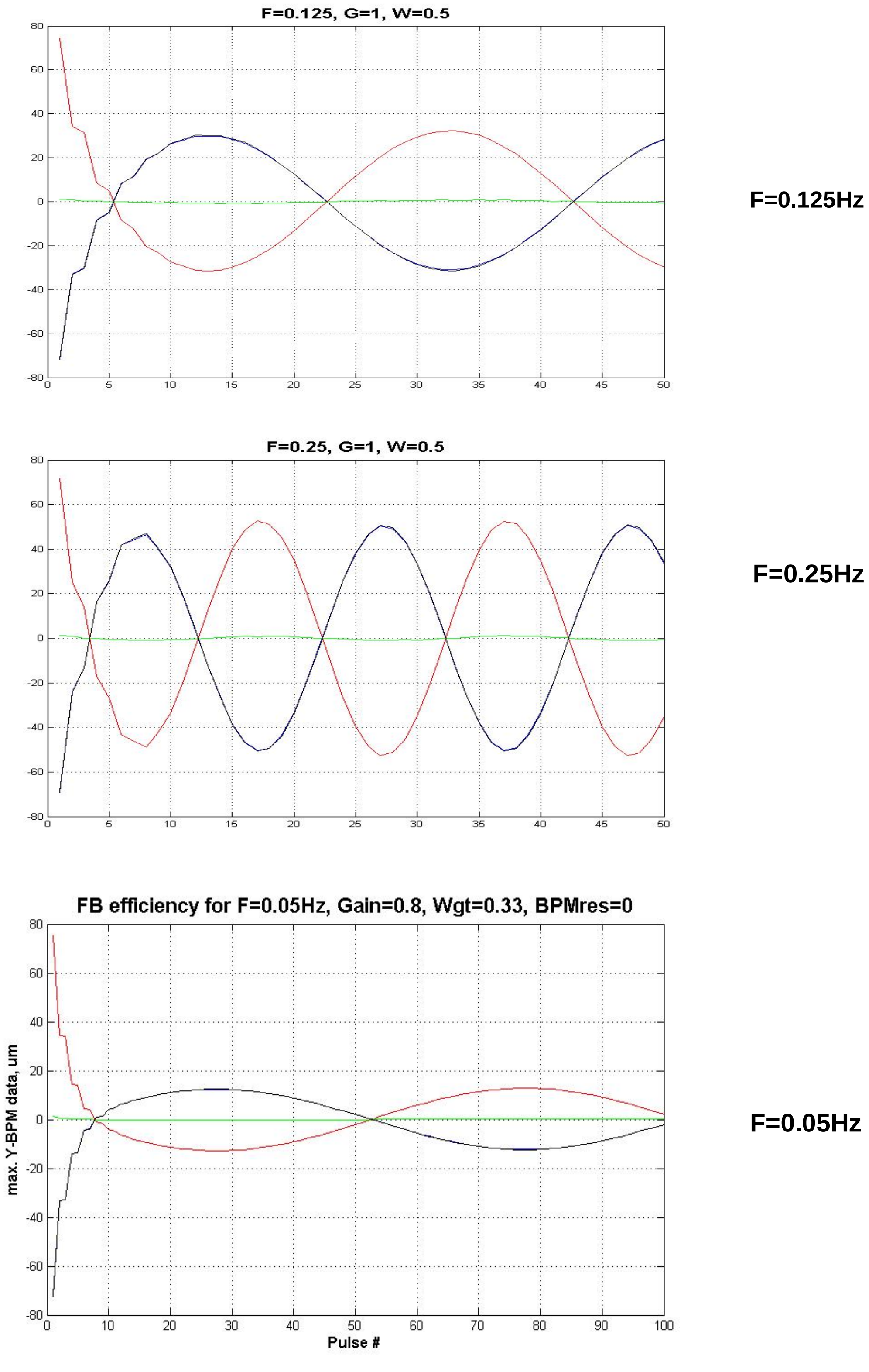


Figure 23. Frequency response for exponential FB model; G=1, W=0.5.

b. Short linac of 50 FODO cells

All further simulations for ILC lattice will have 5 FB loops. Each loop includes 2 x and y correctors with 90º phase shift between them to provide good control sensitivity for any BPM position. We varied the number of BPMs in a loop, BPM resolution and other parameters to study the FB efficiency.

i. Vertical emittance under GM with no feedback

We should simulate GM perturbation with no feedback control to get impression how does FB work in different conditions. The result of this simulation is presented in Figure 24.

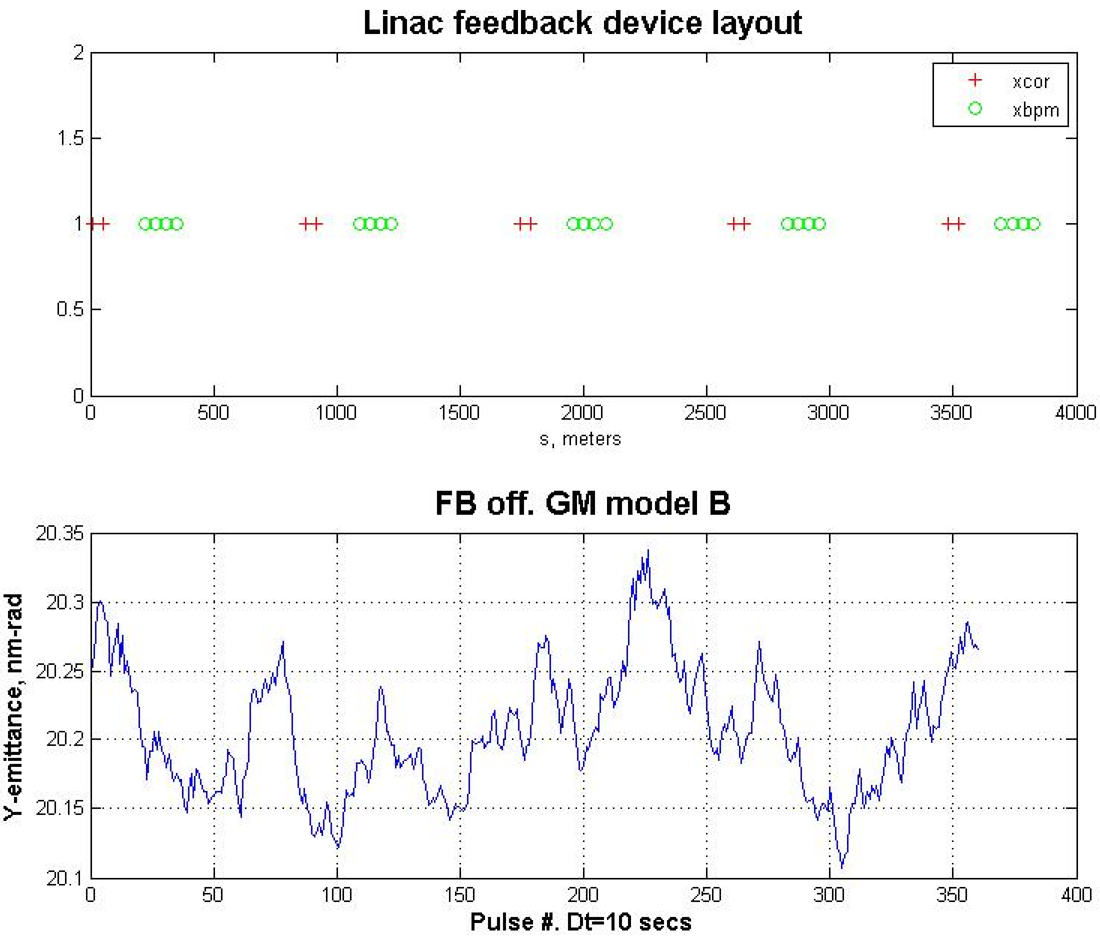
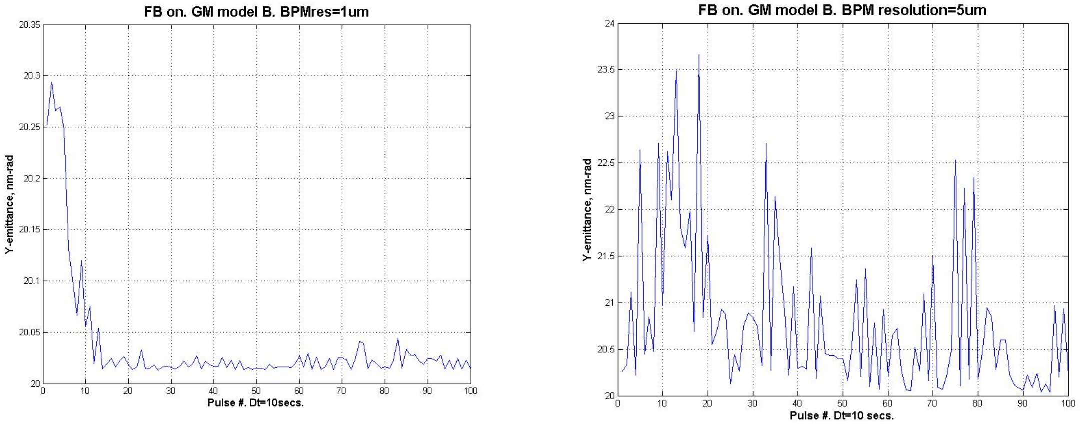


Figure 24. One our of Ground Motion perturbation with no FB control.

ii. Effect of BPM resolution

Figures 25-26 demonstrate the effect of BPM resolution for different number of BPMs in each loop. Our simulations show that vertical emittance < 20.7 for 8-bpm loop, and it is < 23.7 for 4-bpm loop. As the signal is averaged over all BPMs in a group, more BPMs can effectively suppress the BPM-reading errors.

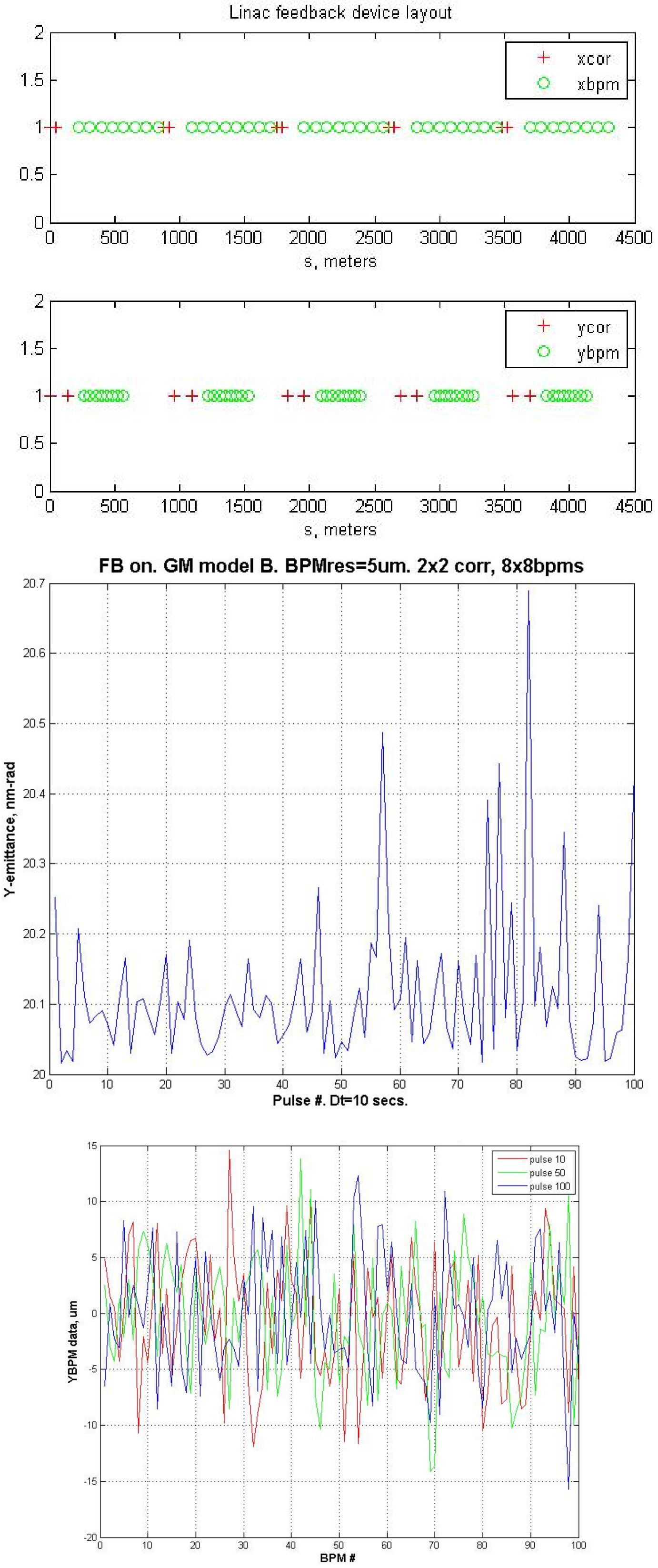


Res=1um

Res=5um

Figure 25. Effect of BPM resolution for 4 BPMs in each loop.

These results have good correlation with BPM effect for a described before simple static test.



Vertical
emittance

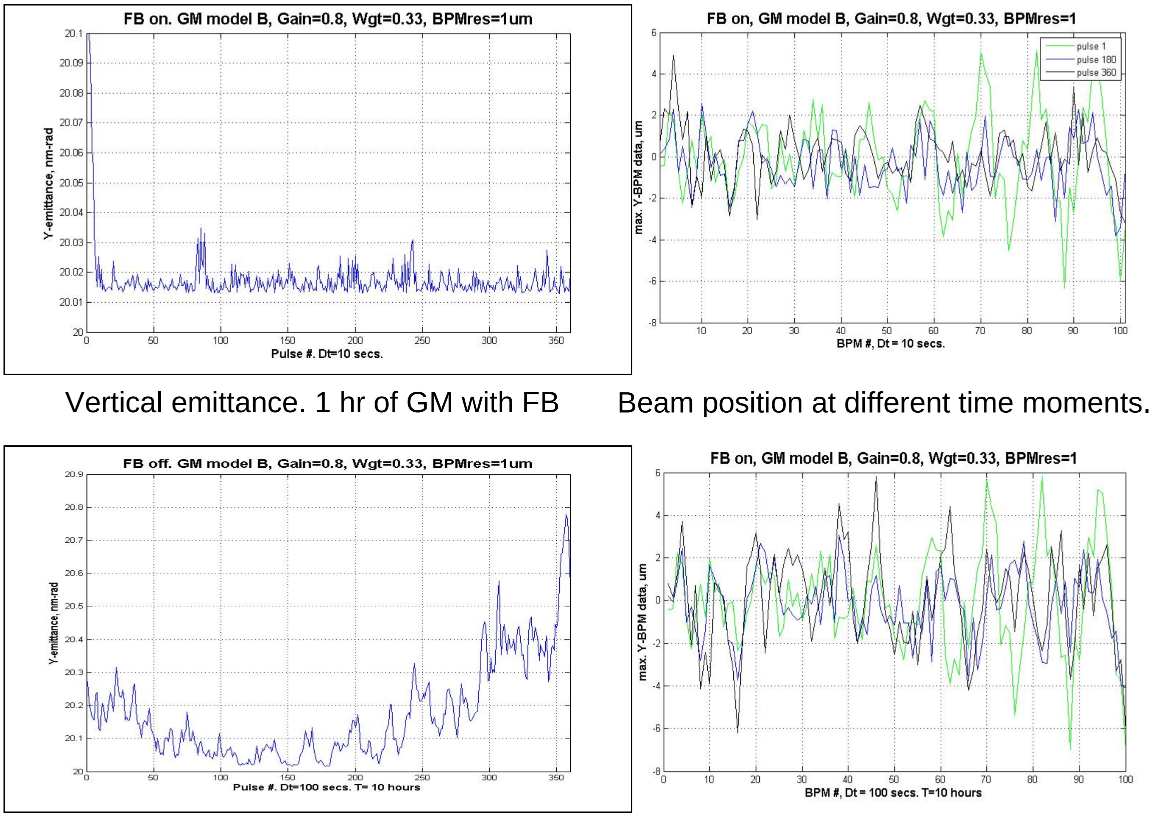
Beam position for
different time moments

Figure 26. Effect of BPM resolution for 8 BPMs in each loop.

iii. Effect of time intervals

Our simulations for 1 hour and 10 hours of ground motion perturbations (model “B”) and 1μm of BPM resolution shows that FB system can effectively neutralize the influence of low frequency parasitic oscillations.

7



Vertical emittance. 1 hr of GM with FB

Beam position at different time moments.

Vertical emittance. 10 hr of GM. No FB

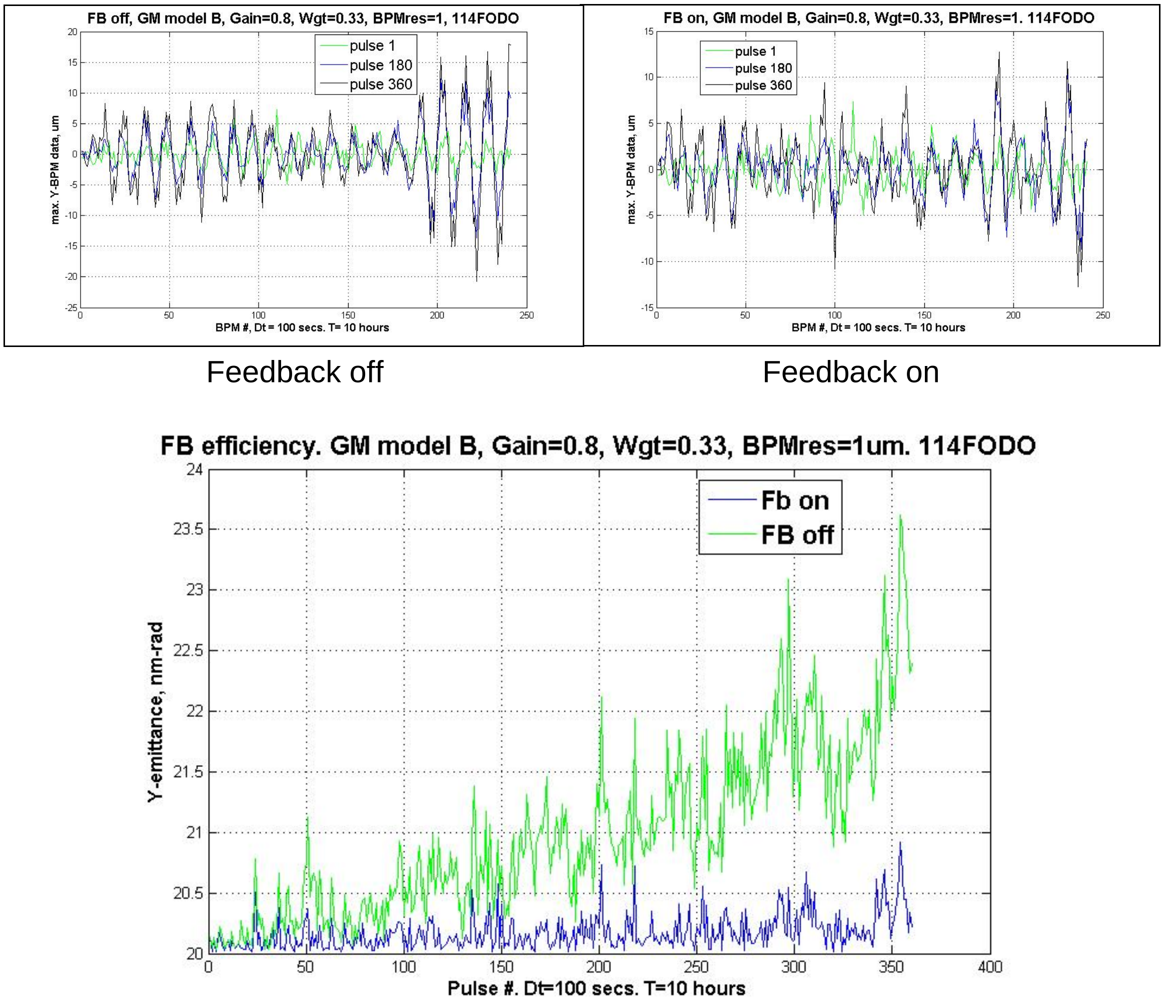
Beam position at different time moments. 10hrs

Figure 27. Vertical emittance and beam position for 1 hr and 10 hrs of GM simulation with FB on and off.

c. Entire ILC linac of 114 FODO cells

i. Effect of feedback control for 10 hrs of GM, model B

The efficiency of FB system for entire linac is demonstrated in Figure 28. Here we setup GM model “B”, simulation period is 10 hours and BPM resolution of 1µm.



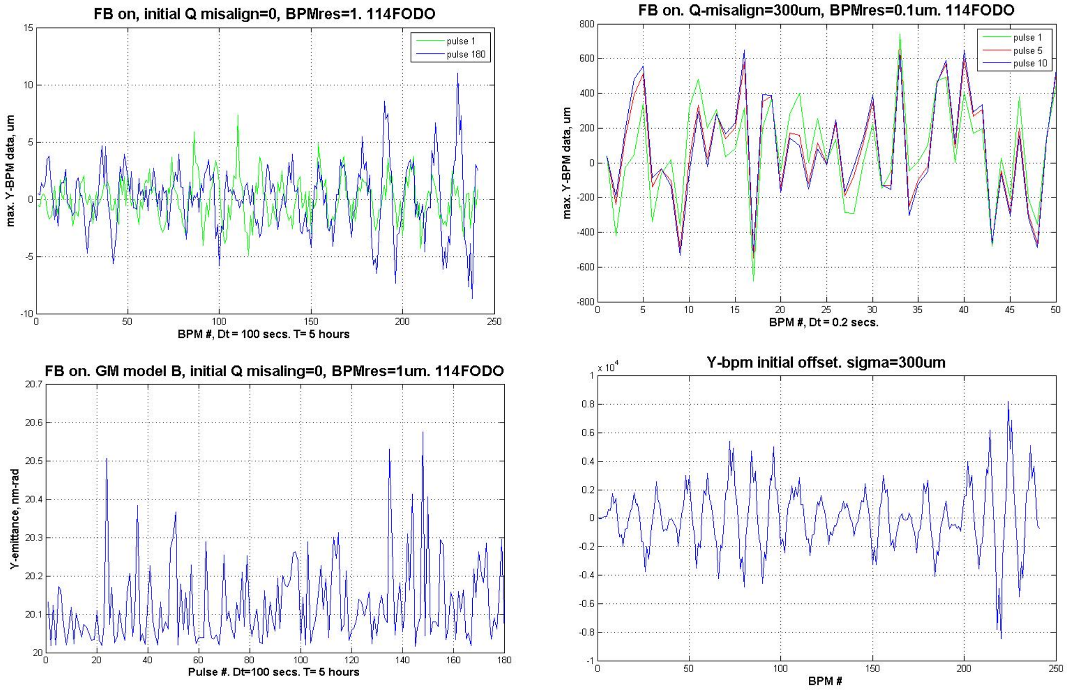
Feedback off

Feedback on

Figure 28. The efficiency of FB control for entire linac.

ii. Effect of initial misalignment

Figure 29 below shows the effect of random initial BPM-offset and quad misalignment of 300 µm for beam position and vertical emittance of entire linac. Simulation period for this case was 5 hours.

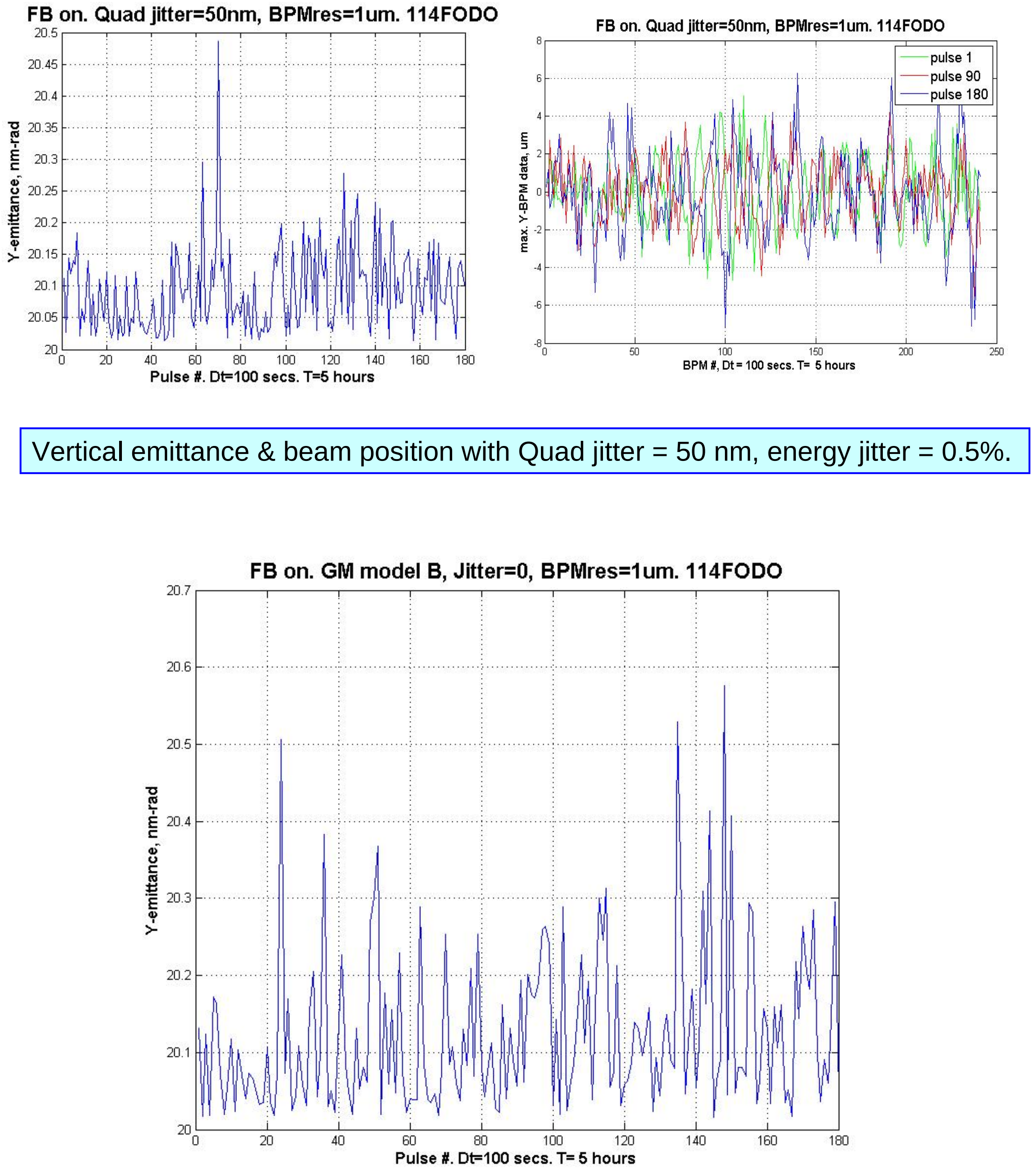


Beam position & V-emittance for perfect aligned linac **Random initial misalignment is 300 um**

Figure 29. Effect of initial random misalignment for entire linac.

iii. **Effect of beam energy and quad jittering**

The results of klystron and quad jittering simulations are presented in Figure 30. These results show that jittering effect just add the noise, which could not be in control.



Vertical emittance & beam position with Quad jitter = 50 nm, energy jitter = 0.5%.

Vertical emittance. No Jitter

Figure 30. Effect of quad and klystron jittering for entire linac.