Drive and the Action Calculation of GetLLM, in Beta-Beat.src

Alex Sherman

Abstract

The Beta-Beat.src program is used to analyze data collected by Beam Position Monitors (BPMs) in accelerators at CERN. The Beams department at CERN uses it to study the behaviour of a beam as it traverses an accelerator, and in particular the LHC. Two pieces of code in Beta-Beat.src are "drive", a C/C++ program, and "GetLLM", a python program. This report described the modification of the drive code to be compatible with windows and take advantage of elements of C++, as well as a change in the calculation of the action in GetLLM to reduce its relative uncertainty. The dynamic aperture is recalculated with the new action and sees a reduction in its uncertainty.

Introduction

Drive is a C/C++ program which parses input files containing turn by turn data from formatted output files created by previous programs. It calls a FORTRAN function "SUSSIX" which analyses the data using Fourier analysis. The output of SUSSIX is then organized and placed into formatted output files by drive. Drive has now been made Windows compatible and cleaned up for better readability.

GetLLM is a python program which analyses the output data from drive, calculating many quantities of interest and creating several output files with the results. The action of a kicked beam is one such quantity which is in particular used in the calculation of the dynamic aperture and in the measurement of detuning with amplitude. This calculation was modified in order to reduce its relative uncertainty, changing the output of "getkick.out".

Drive

In drive, the lines of the "lin_x" and lin_y" output files are sorted by BPM index. The lines are placed into the files during a "pragma for loop", when multithreading occurs, so they must be sorted afterward. Previously, the C command "system" was being called to utilise the shell in sorting the lines. The use of the shell has now been replaced by code which sorts the files in C++. This code puts each line of the lin file into an element of a string array, where the index of the element corresponds to the BPM index. The lines are then read back into an empty line file in the correct order.

For compilation in Windows, the "Makefile" of drive also had to be changed as the different operating systems require different procedures when linking C and FORTRAN object files (see appendix A for the code). In addition, minor changes to the code had to be made due to differences in running drive on windows. These include changing the Windows convention of the output of scientific notation to match Linux's, naming and declaration conventions for calling an external FORTRAN subroutine, and changing the name of a file path (e.g. Windows uses '\' while linux uses '/'). Still, the Windows output files contain an

extra character of white space on each new line (it is a carriage return, '\r'), and a fix has not yet been found.

In addition to the changes regarding Windows compatibility, a number of minor changes were made to improve readability and simplify some operations being performed. Drive was previously a C program, and now it is a mixture of C and C++ code. The C++ objects "std::string" and "std::iofstream" are now used throughout the code, taking advantage of their member functions for performing various operations, in particular for parsing input files and creating output files, which is the bulk of the work done by drive. Along with the new use of C++, many of the variables in drive were organized into two "classes", one for input data and calculated data, respectively, and an array of "structs", which contain the name, position, plane, turn by turn data, and a Boolean "picked-up" value for each BPM.

Finally, some unused code was removed including three variables (hvt, allbpmphase, and allbpmamp), which were being declared and initialized but never used, and the printing (in standard output) of the pair of BPM indices for each time SUSSIX was called, as the output was random and unhelpful.

Action Calculation

At a point s along the ideal orbit, the amplitude of oscillation for a particle can be written $A_{x,y}(s) = \sqrt{2J_{x,y}\beta(s)}$, where $2J_{x,y}$ is the action and $\beta(s)$ is the beta function[1]. It follows that we can calculate the action as $2J_{x,y} = \frac{\left(A_{x,y}(s)\right)^2}{\beta(s)}$. In GetLLM, the calculation previously being done was $\sqrt{2J_{x,y}} = mean\{\sqrt{2J_{x,y}}\}$ where the mean was being taken over all BPMs and the model beta function was used. It followed that $\left(\sqrt{2J_{x,y}}\right)^2 = 2J_{x,y}$. A different method of calculation would be to let $2J_{x,y} = mean\{2J_{x,y}\}$, and this may be better for the calculation of $2J_{x,y}$ as it is truly the mean of its value over all the BPMs in the LHC. This calculation has now been implemented in GetLLM in the 'helper.py' submodule of the submodule algorithms. Currently it is not being placed in the output files while being tested.

In addition to a change in the method of calculation, a new calculation is now performed for the action by using the beta function calculated from phase instead of the model beta function. In GetLLM, the uncertainty for the action is calculated as $\sqrt{RMS\left(\sqrt{2J_{x,y}}\right) - \left(mean\{\sqrt{2J_{x,y}}\}\right)^2}$ (where RMS is taken over all BPMs), so a higher deviation from the mean gives a larger uncertainty. Since the model beta function was previously used, the uncertainty in the action is high due to a known beta-beating of about 5% or more. Using the measured beta function from the data will be more accurate and therefore reduce the action uncertainty. But, if a BPM is malfunctioning or is near an

[1] Glenn Vanbavinckhove, [2012], *Optics measurements and corrections for colliders and other storage rings*, Ph.D, CERN, Switzerland

interaction point then the calculated beta function at its location will probably not be very accurate. Therefore thresholds were introduced on the beta-beating level ($\frac{|\beta_{model}-\beta_{meas}|}{\beta_{model}}$) and the relative uncertainty ($\frac{\beta_\sigma}{\beta}$) of the measured beta function. Any BPM whose measured beta-function fails either of these cuts is removed from the calculation of the action. Currently, the default level for these cuts is 15% each, but the user may specify them individually using -k (float) for the beta-beating threshold or –e (float) for the relative uncertainty threshold. When these thresholds were implemented, significant reductions were seen in the relative uncertainty in the action (see plots in appendix B).

Calculation of the DA

The DA was calculated with the new value of the action and compared with the old one. The DA is given by[2]

$$DA = KICK + \sqrt{-2ln\left(\frac{\Delta I}{I_{init}}\right)\left(\frac{\epsilon_{beam}}{\epsilon_{nominal}}\right)}$$

where the kick is related to the action by $KICK_{x,y} = \sqrt{\frac{2J_{x,y}}{\epsilon_{nominal}}}$ and $KICK = \sqrt{KICK_x^2 + KICK_y^2}$ (measured in number of $\sigma_{nominal}$'s, a measure of the size of the beam spread), $\frac{\Delta I}{I_{init}}$ is the fractional beam loss, ϵ_{beam} is the measured beam emittance (0.0035 ± 0.0005 μrad in the x-plane, 0.004 μrad in the y-plane, and $\epsilon_{beam} = \sqrt{\epsilon_{beam,x}^2 + \epsilon_{beam,y}^2}$) , and $\epsilon_{nominal}$ is the nominal beam emittance (defined to be 0.0078 μrad). The following table shows this calculation for two data sets, and the improvement in uncertainty.

Data set	β From	$\frac{\Delta I}{I_{init}}$	Kick ($\sigma_{nominal}$)	DA ($\sigma_{nominal}$)
Beam2@Turn@2012_06_25 @03_19_43_160turncutted_92_1630	Model	0.375	9.39 ± 0.37	10.55 ± 0.37
Beam2@Turn@2012_06_25 @03_19_43_160turncutted_92_1630	Phase	0.375	9.40 ± 0.30	10.56 ± 0.30
Beam2@Turn@2012_06_25 @04_12_38_806turncutted_92_1500	Model	0.24	9.62 ± 0.41	11.02 ± 0.41
Beam2@Turn@2012_06_25 @04_12_38_806_turncutted_92_1500	Phase	0.24	9.64 ± 0.32	11.04 ± 0.32

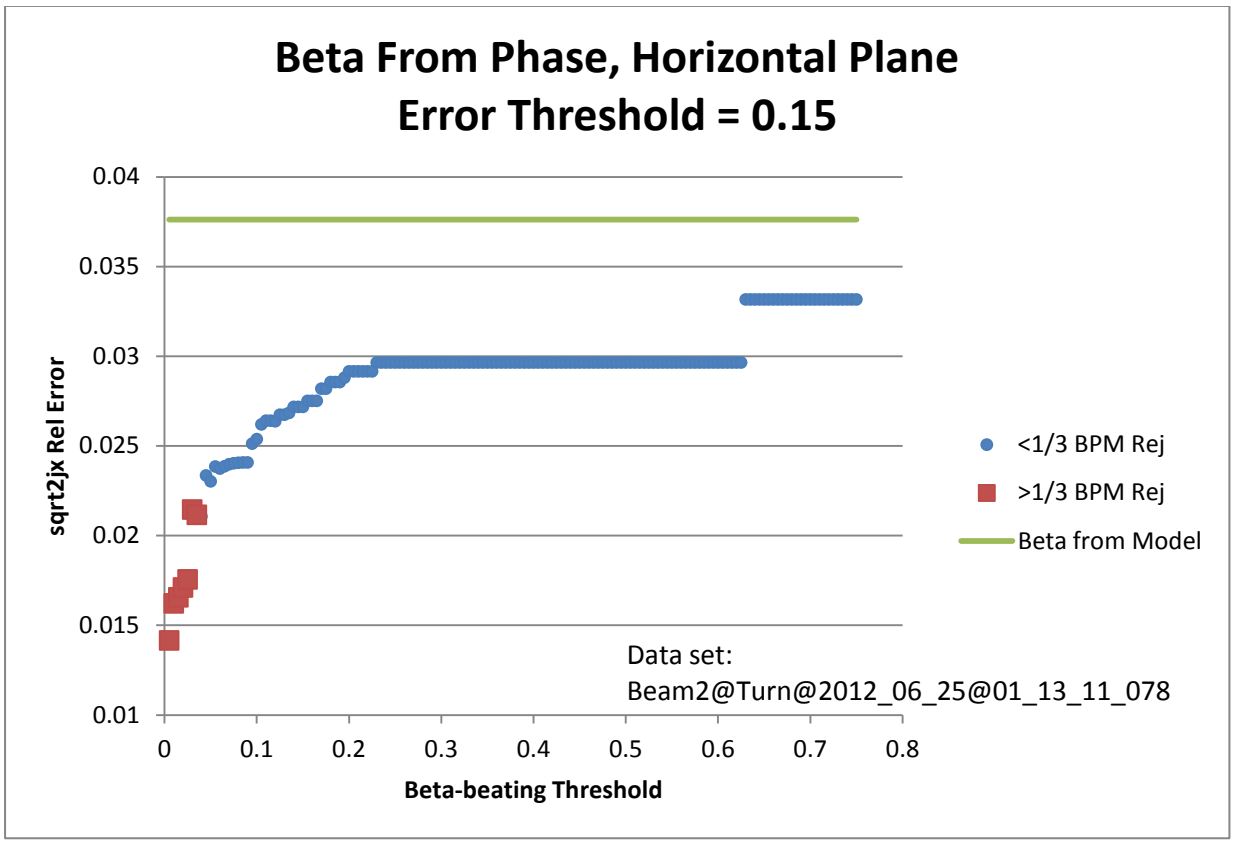
[2] *Measurement of LHC non-linear observables using kicked beams*, CERN, Unpublished

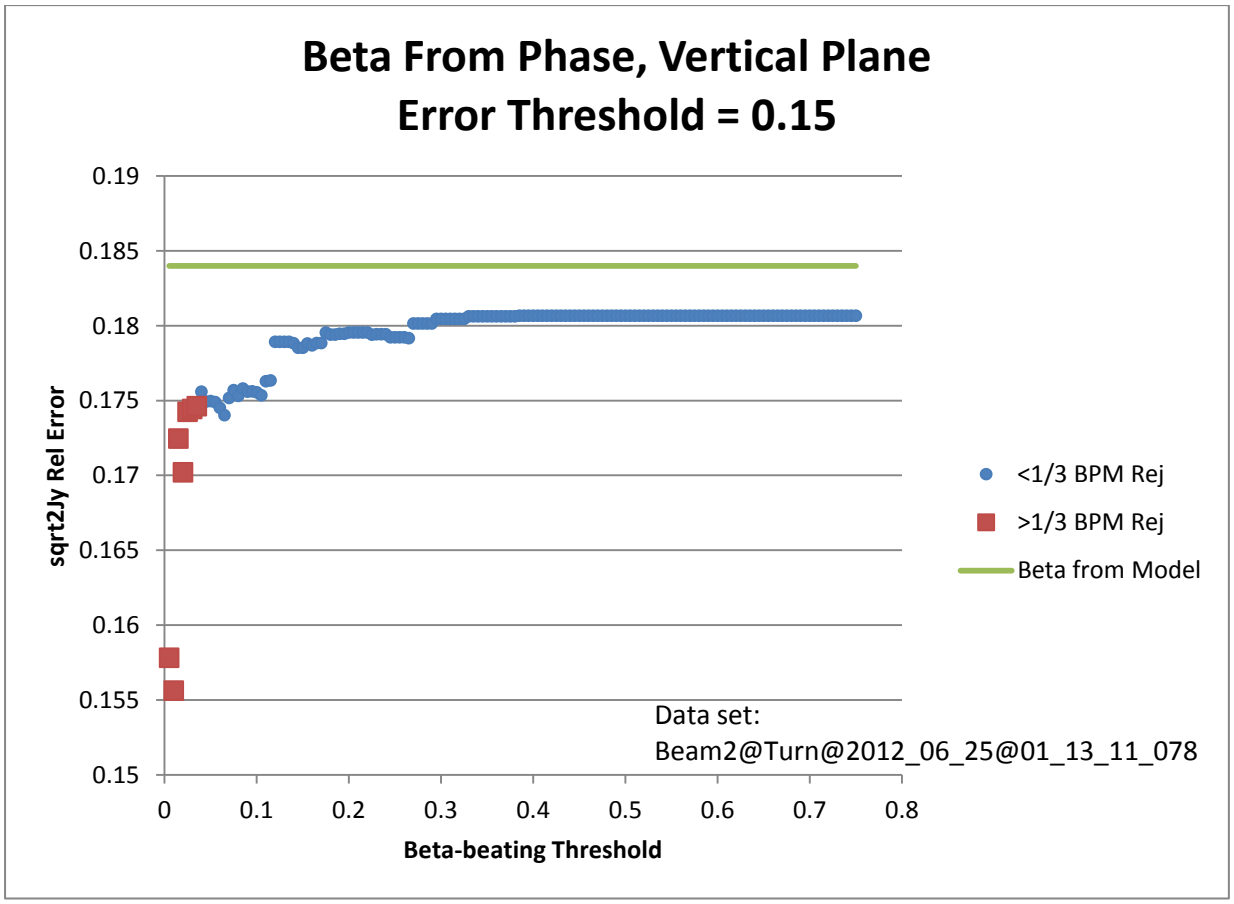
Appendix A

The following is the new Makefile for drive which allows it to be compiled on Windows:

```
FC = ifort

ifdef SystemRoot
    DRIVE = Drive_God_lin_win.exe
      CC = icl
      RM = del -f *.obj *.manifest
      FFLAGS = /O3 /Qopenmp /Qparallel /Qunroll /Qipo-c
      CCFLAGS = /c /Qopenmp /Qparallel /Qunroll /Qipo-c /Wall
      LINK = xilink -qipo /out:Drive_God_lin_win.exe
      FCL =
      DRIVE_OBJ = Drive_God_lin.obj
      SUSSIX_OBJ = sussix4drivexxNoO.obj
else
    DRIVE = Drive_God_lin
      CC = icc
      RM = rm -f *.o
      FFLAGS = -O3 -c -m32 -openmp -Bstatic -static -static-libgcc -
parallel -unroll -ipo
      CCFLAGS = -ansi -c -m32 -openmp -static-libgcc -parallel -unroll -ipo
-Wall -Wremarks -diag-disable 981
      LINK = ifort -o Drive_God_lin
      FCL = -lm -m32 -nofor_main -openmp -Bstatic -static-libgcc -static -
parallel -unroll -ipo -lstdc++
      DRIVE_OBJ = Drive_God_lin.o
      SUSSIX_OBJ = sussix4drivexxNoO.o
endif


all: $(DRIVE)

# make sussix object file from fortran source
$(SUSSIX_OBJ):
      $(FC) $(FFLAGS) sussix4drivexxNoO.f

# make drive object file from c source
$(DRIVE_OBJ):
      $(CC) $(CCFLAGS) Drive_God_lin.cpp

# link to binary
$(DRIVE): $(DRIVE_OBJ) $(SUSSIX_OBJ)
      $(LINK) $(DRIVE_OBJ) $(SUSSIX_OBJ) $(FCL)


clean:
      $(RM) $(DRIVE)
```

Appendix B


Beta From Phase, Horizontal Plane
Error Threshold = 0.15
sqrt2jx Rel Error
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Beta-beating Threshold
<1/3 BPM Rej
>1/3 BPM Rej
Beta from Model
Data set:
Beam2@Turn@2012_06_25@01_13_11_078


Beta From Phase, Vertical Plane
Error Threshold = 0.15
sqrt2Jy Rel Error
0.19
0.185
0.18
0.175
0.17
0.165
0.16
0.155
0.15
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Beta-beating Threshold
<1/3 BPM Rej
>1/3 BPM Rej
Beta from Model
Data set:
Beam2@Turn@2012_06_25@01_13_11_078