New aperture model for the collimation routine BeamLossPattern

Andrea Santamaría García

May 4, 2015

Contents

1. BeamLossPattern

« BeamLossPattern is a program that calculates the location of particle losses along an accelerator. It relies on an aperture model of the full machine and takes as an input the particle coordinate as a function of the longitudinal coordinate (referred to as particle trajectories). Locations of beam losses are identified with a spatial resolution of 10 centimetres. For the 27 km long LHC ring, this is equivalent to checking 270000 loss locations! This is achieved by interpolating trajectories between to consecutive lattice elements (typical distance up to $>$ 100 m). The LHC aperture model has mostly been set by using that MADX interfaces with the LHC layout database. However, several aperture definitions required the preparation of dedicated MADX scripts in order to complement missing information on the database (see below). It is noted that the program BeamLossPattern is completely independent on MADX. MADX is only used to generate input files for the aperture program (aperture definitions versus longitudinal coordinate, survey files with the position of the design orbit, closed-orbit distortions, etc...). In particular, this means that the generated MADX scripts have not been conceived for MADX users: detailed MADX studies (e.g., particle tracking, APL aperture studies) might require different optimization such as positioning of aperture definition markers, definitions of aperture for specific elements, etc. For the moment, the aperture model for the LHC optics version V6.5 (sequence) contains:

1. »Beam screen location for all cold elements (form database)
2. »Aperture definition for the cold elements within beam screen markers (MADX script)
3. »Aperture definitions for warm elements (warm)
4. »Aperture of the various BPM types (BPM)
5. »Experimental regions (IR1, IR2, IR5, IR8)
6. »Standard aperture definition of the vacuum chambers - no flange position (drifts)
7. »Fixing by hand some special elements (fix holes)

»**Known missing elements**: Recombination chambers, position of vacuum chamber flanges of IR6 kickers.

»**Remark**: It is noted that the aperture of collimators and protection devices are not included in the aperture model of the ring. The reason for this choice is that the treatment of these elements is carried out within the tracking code, which provides directly the number of impacting particles on the collimators and the locations of inelastic impacts within the volume of the collimator jaws. Therefore, the aperture of the above 'injection' and 'lowb' lattices are identical (movable elements with different settings at injection or top energy are the only aperture differences that can arise). Different names for the two cases are kept for compatibility with the BeamLossPattern input.

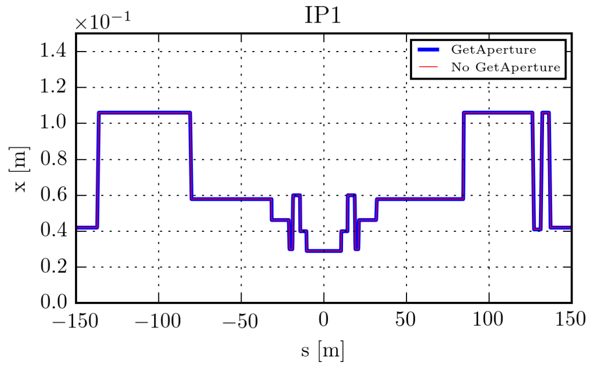
»The BeamLossPattern program does recognize square apertures and hence collimators can easily be modelled if needed.

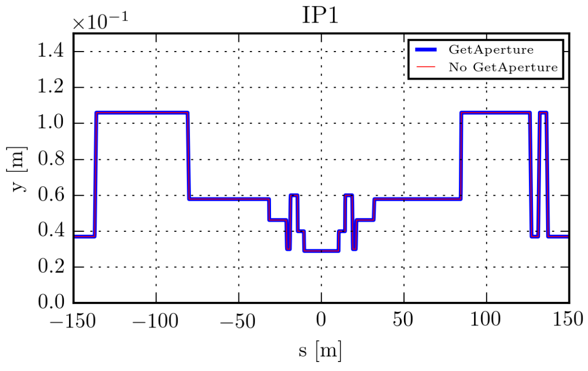
»Aperture types accepted by the program (RECTELLIPSE notation, see MADX manual): Circle, square, ellipse, LHC beam screen, RaceTrack. » [1]

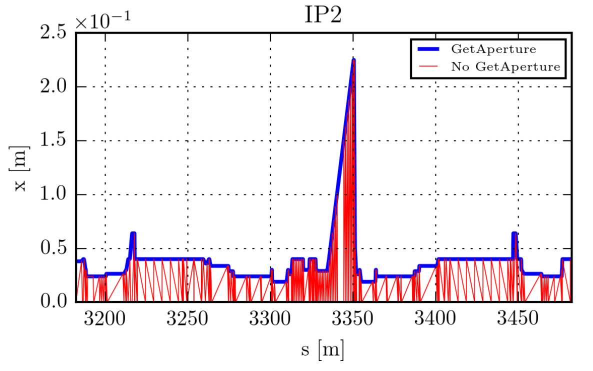
2. GetAperture

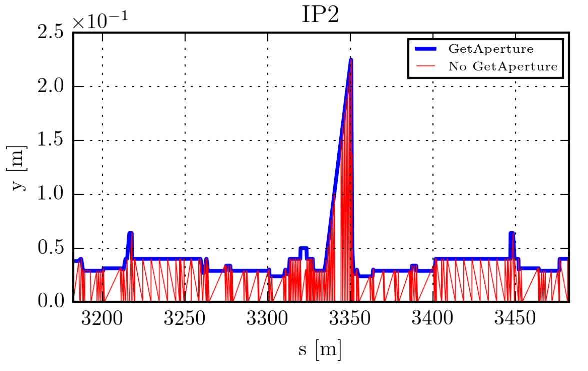
«For a given lattice and aperture model, this program produces an ascii files with the aperture every 10 cm all along the beam line. Aperture values are given at three different azimuthal angles: 0, pi/4 and pi/2. In order to be able to calculate the aperture in any position along the considered metre (and not only at the positions where the aperture is defined) the member function "GetAperture(double p)" can be used. It calculates the aperture as interpolation of the point where the aperture has been defined.» [1]

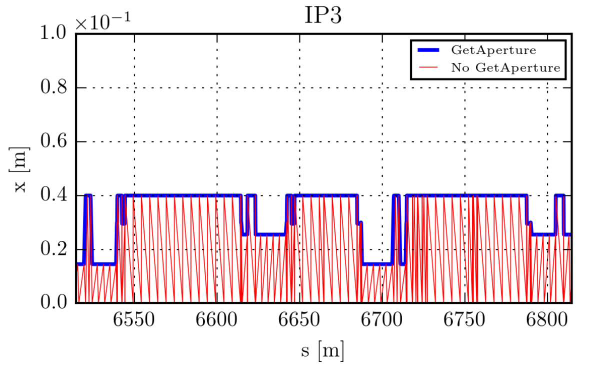
In order to test the GetAperture routine, the file `allapert_hllhc_2013.09.04.b1` was used. This file was originally generated in 2008 and includes the optics version V6.5. We can see from the plots that the GetAperture routine gets rid of the zero values of the aperture. We can also see that in interpolates the aperture at different values than the original one (see IP3 and 7 in the y coordinate).

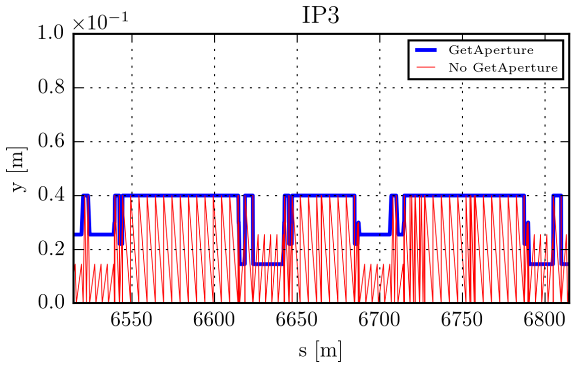


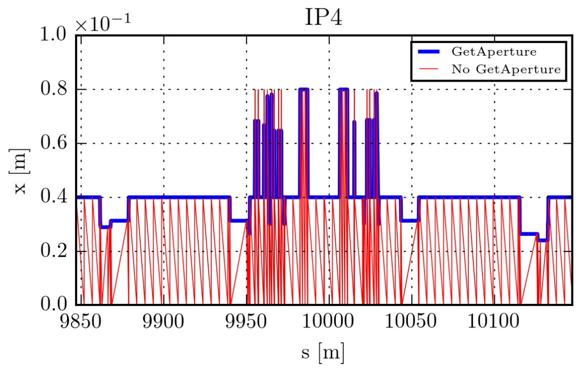
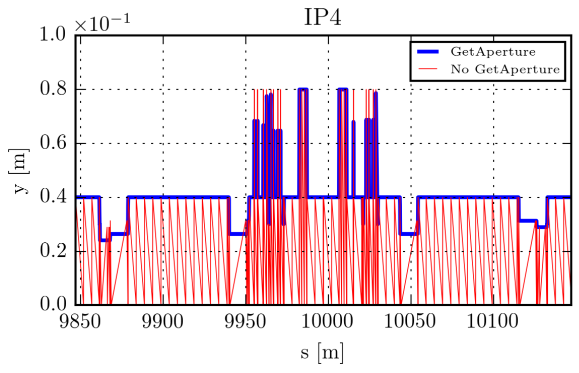
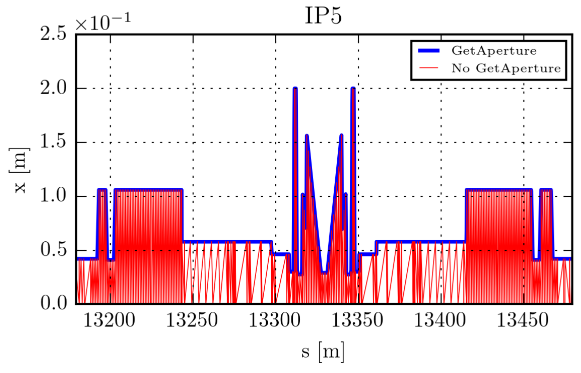
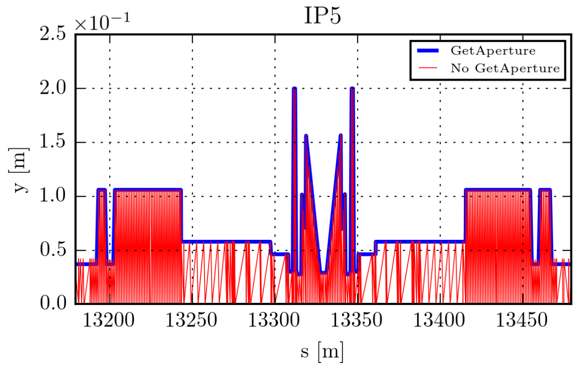
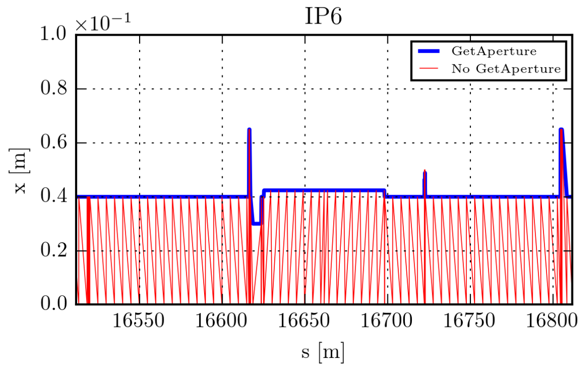
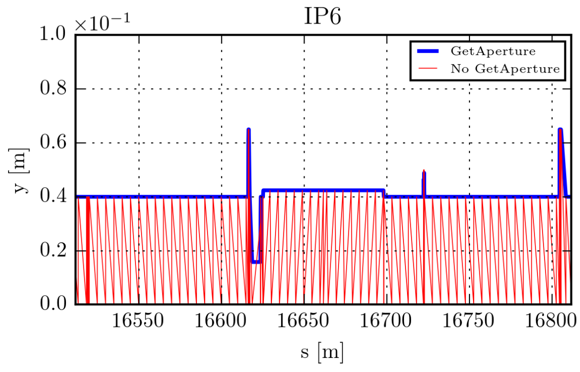
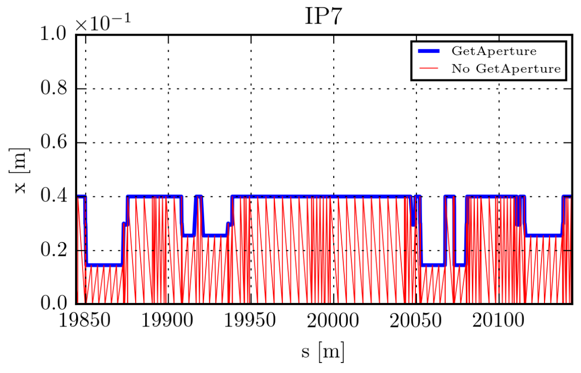
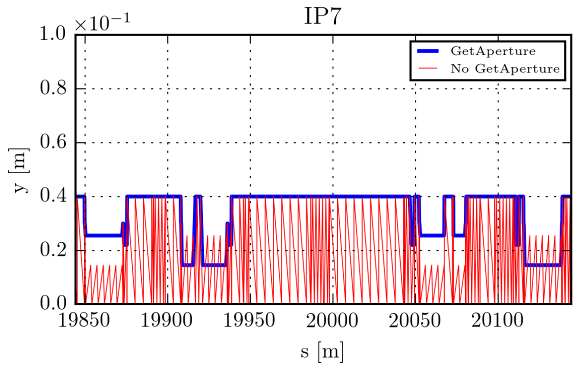
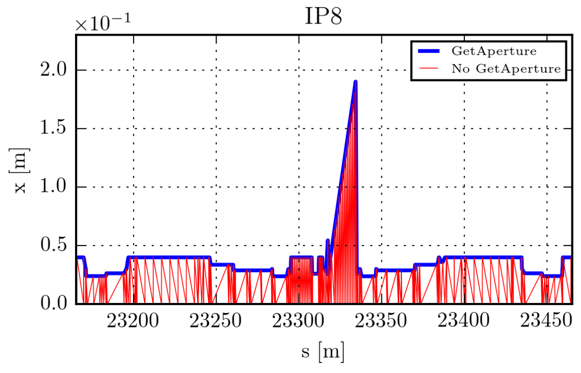
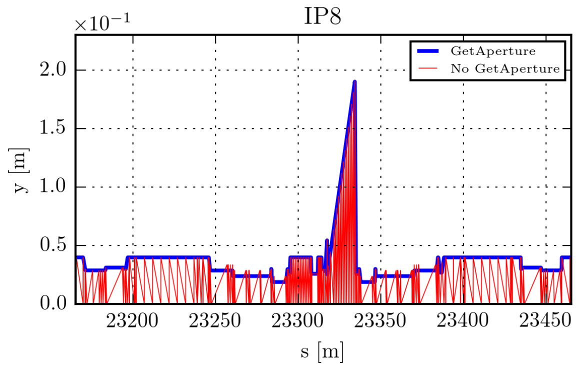











IP4
×10⁻¹
x [m]
s [m]
GetAperture
No GetAperture

IP4
×10⁻¹
y [m]
s [m]
GetAperture
No GetAperture

IP5
×10⁻¹
x [m]
s [m]
GetAperture
No GetAperture

IP5
×10⁻¹
y [m]
s [m]
GetAperture
No GetAperture

IP6
×10⁻¹
x [m]
s [m]
GetAperture
No GetAperture

IP6
×10⁻¹
y [m]
s [m]
GetAperture
No GetAperture

IP7
×10⁻¹
x [m]
s [m]
GetAperture
No GetAperture

IP7
×10⁻¹
y [m]
s [m]
GetAperture
No GetAperture

IP8
×10⁻¹
x [m]
s [m]
GetAperture
No GetAperture

IP8
×10⁻¹
y [m]
s [m]
GetAperture
No GetAperture

3. New aperture model with HLLHCV1.0 optics

First attempts of generating a new allapert file are described below. The code can be found here [2].

3.1. With vacuum markers and experimental beampipe

```
//**************************************
// IP1 AT COLLISION WITH HL-LHC VALUES
//**************************************
option, warn, info;

//------------------------------------------
// Shortcut creation for needed directories
//------------------------------------------
// Nominal
system, "ln -fns /afs/cern.ch/eng/lhc/optics/V6.503 db5";
// High Luminosity
system, "ln -fns /afs/cern.ch/eng/lhc/optics/HLLHCV1.0 hl";
// Beam parameters
system, "ln -fns /afs/cern.ch/user/a/ansantam/public/toolkit tkit";

//----------------------------------
// Load ATS macros and select beam
//----------------------------------
call, file = "hl/toolkit/macro.madx";
// on_disp = 1;  // parasitic dispersion
mylhcbeam = 1;

//--------------
// Verbose off
//--------------
option, -echo, -warn, -info;

//----------------------------------
// Redefinition of elements' length
//----------------------------------
REAL CONST l.TAN   = 3.7;
REAL CONST l.TANAL = l.TAN;
REAL CONST l.TANAR = l.TAN;
REAL CONST l.TANC  = l.TAN;
REAL CONST l.TCT   = 1.0;
REAL CONST l.TCTH  = l.TCT;
REAL CONST l.TCTVA = l.TCT;

//---------------------------------------------
// Load sequence files (position of elements)
//---------------------------------------------
if (mylhcbeam>2){
  call,file="hl/hllhc_thinb4.seq";
} else {
  call,file="hl/hllhc_thin.seq";
};

// call,file="hl/hllhc_sequence.madx"

//--------------------
// Load strength files
//--------------------
// beta*=15 cm, round optics
call, file = "hl/opt_round_thin.madx";

//------------------------------
// Load and define the aperture
//-----------------------------
```

```
call, file = "db5/aperture/aperture.b1.madx";
call, file = "db5/aperture/aper_tol.b1.madx";
 // TAN,D2,Q4,Q5 2010
call, file = "hl/aperture/aperture_upgrade_MS.madx"
//beam screen aperture 2009, doesn't compile
// call, file = "hl/aperture/aperture_upgrade_IT.madx";
call, file = "hl/aperture/aperture_upgrade_octagon.madx";
call, file = "hl/aperture/aperture_upgrade_rectellipse.madx";

// Not updated -- !Check where we can get the info?
//-------------------------------------------------
// Vacuum markers
call, file = "/afs/cern.ch/eng/lhc/optics/V6.503/aperture/as-built/layoutapertures.madx";
// Apertures around IPs
call, file = "/afs/cern.ch/eng/lhc/optics/V6.503/aperture/as-built/
    exp_pipe_model_after_LS1.madx";
// Apertures around IPs
call, file = "/afs/cern.ch/eng/lhc/optics/V6.503/aperture/as-built/
    exp_pipe_install_after_LS1.madx";

//-------------------------------
// Define the beam for the machine
//-------------------------------
exec, crossing_enable;
exec, mk_beam(7000); // Collision

//-------------------------------
// Define the crossings at the IPS
//-------------------------------
on_x1 := 1; on_sep1 := 0; on_sol_atlas := 1;
on_x2 := 1; on_sep2 := 0; on_alice:= 1; on_sol_alice := 1;
on_x5 := 1; on_sep5 := 0; on_sol_cms := 1;
on_x8 := 1; on_sep8 := 0; on_lhcb:= 1;
on_disp := 1;

//-----------------------------------------------------
// Load the beam parameters (to obtain sigmas as output)
//-----------------------------------------------------
 // Important to call these files first, if not Beam.madx doesnt work
call, file = "tkit/Beam_7_TeV.madx";
call, file = "tkit/Beam.madx";

//------
// IP1
//-----
//-------
// Beam 1
//-------
use, sequence = lhcb1;
select, flag = twiss, clear;
select, flag = twiss,
column = name, keyword, name, parent, s, l, aper_1, aper_2, aper_3, aper_4;
twiss, table = twiss, file = "twiss_ip1_b1.tfs";

exec,mk_irtwiss(1,b1)

// select, flag = aperture, clear;
// select, flag = aperture, column = keyword, name, parent, s, l, aper_1, aper_2, aper_3,
     aper_4;
// aperture, file = "allapert_twiss.b1";

stop;
```

The Twiss file generated with this command is not an appropiate input for GetAperture, since it gives segmentation fault when executed. It apparently doesn't support the CIRCLE aperture type as defined in MAD-X [3] (although the documentation says it does [1]).

```
ERROR: Invalid aperture definition!!
0.0235 0 0 0
ERROR: Invalid aperture definition!!
0.029 0 0 0

All aperture information has been read and the sequence has been created!

Length of the read sequence: 26659 metres.

Special case: aperture definitions at 0 and 0 are identical!
Segmentation fault
```

Further testing reveals that it doesn't support RACETRACK either:

```
./GetAperture allapert_final.b1
Reading aperture file allapert_final.b1

Reading from "allapert_final.b1"
Total number of read elements: 11446

ERROR: Invalid aperture definition!!
0 0.017 0.033 0
ERROR: Invalid aperture definition!!
0 0.01375 0.03375 0
ERROR: Invalid aperture definition!!
0 0.01375 0.03375 0

All aperture information has been read and the sequence has been created!

Length of the read sequence: 26659 metres.

Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0.5 and 0.5 are identical!
Special case: aperture definitions at 0.5 and 0.5 are identical!
Segmentation fault
```

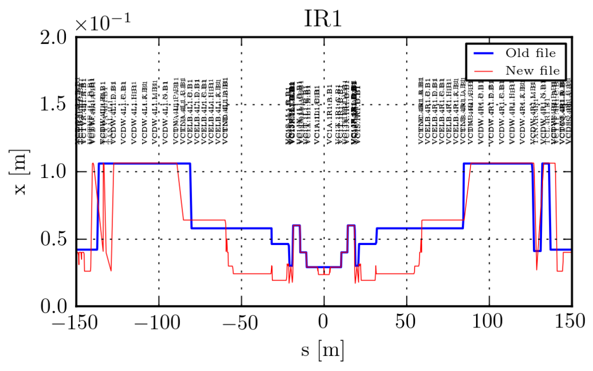
Since it only seems to support aperture with the four parameters defined, I create a Python script that will transform CIRCLE and RACETRACK into RECTELLIPSE:

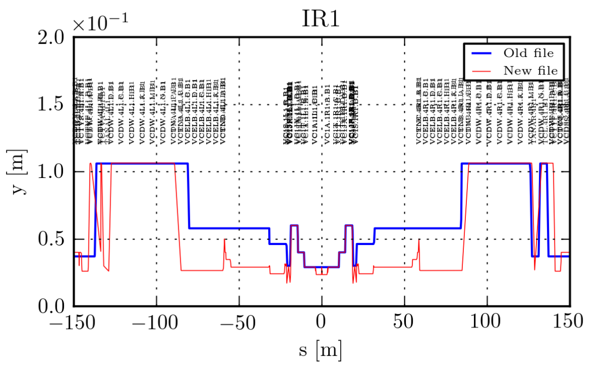
```python
import re

with open('twiss_ip1_b1.tfs', 'r') as infile:
    with open('allapert_final.b1', 'w') as outfile:
        line = infile.readlines()
        for item in line:
            if item[0] == '@' or item[0] == '$' or item[0] == '*':
                outfile.write(item)
    outfile.close()
infile.close()

with open('twiss_ip1_b1.tfs', 'r') as infile:
    with open('allapert_final.b1', 'a') as outfile:
        counter = 0
        for character in infile:
            columns = character.strip().split()
            if character[0] != '@' and character[0] != '$' and character[0] != '*':
                # Skip open apertures
                if float(columns[5]) == 9.999999 and float(columns[6]) == 9.999999 and float(columns[7]) == 9.999999 and float(columns[8]) == 9.999999:
                    continue
                # Skip closed apertures
                elif float(columns[5]) == 0 and float(columns[6]) == 0 and float(columns[7]) == 0 and float(columns[8]) == 0:
                    continue
                # Transform CIRCLE
                elif float(columns[5]) != 0 and float(columns[6]) == 0 and float(columns[7]) == 0 and float(columns[8]) == 0:
                    outfile.write('%20s %-17s %19s %23f %18f %15f %18f %18f %18f \n' % (columns[0], columns[1], columns[2], float(columns[3]), float(columns[4]), float(columns[5]), float(columns[5]), float(columns[5]), float(columns[5])))
                # Transform RACETRACK
                elif float(columns[5]) == 0 and float(columns[6]) != 0 and float(columns[7]) != 0 and float(columns[8]) == 0:
                    outfile.write('%20s %-17s %19s %23f %18f %15f %18f %18f %18f \n' % (columns[0], columns[1], columns[2], float(columns[3]), float(columns[4]), float(columns[6]), float(columns[6]), float(columns[7]), float(columns[7])))
                else:
                    outfile.write('%20s %-17s %19s %23f %18f %15f %18f %18f %18f \n' % (columns[0], columns[1], columns[2], float(columns[3]), float(columns[4]), float(columns[5]), float(columns[6]), float(columns[7]), float(columns[8])))
    outfile.close()
infile.close()

# with open('allapert_final.b1', 'r') as infile:
#     with open('plot_allapert.txt', 'w') as outfile:
#         for character in infile:
#             columns = character.strip().split()
#             if character[0] != '@' and character[0] != '$' and character[0] != '*':
#                 outfile.write('%0s %15f %15f %15f %15f \n' % (columns[3], float(columns[5]), float(columns[6]), float(columns[7]), float(columns[8])))
#     outfile.close()
# infile.close()
```

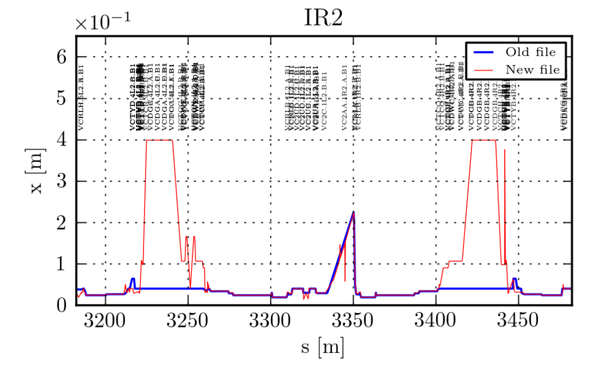
This time we get no segmentation fault:

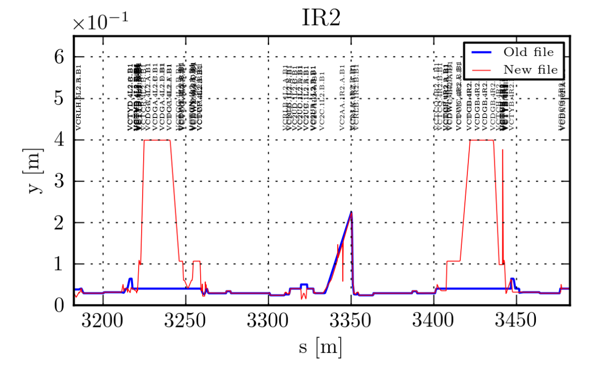
```
./GetAperture allapert_final.b1
Reading aperture file allapert_final.b1

Reading from "allapert_final.b1"
Total number of read elements: 11446


All aperture information has been read and the sequence has been created!

Length of the read sequence: 26659 metres.

Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0 and 0 are identical!
Special case: aperture definitions at 0.5 and 0.5 are identical!
Special case: aperture definitions at 0.5 and 0.5 are identical!
Warning: aperture definition added at the end of the metre!
```

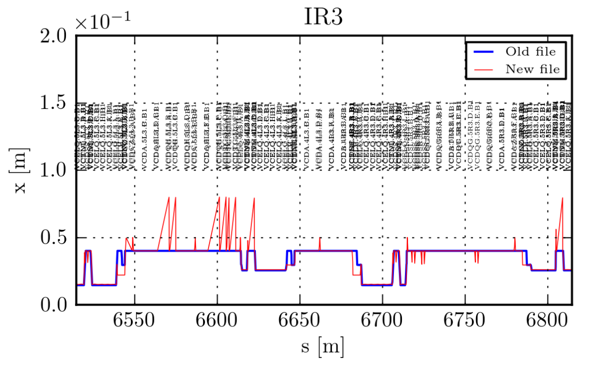
I compare the newly created aperture with the old one, after being processed by GetAperture:

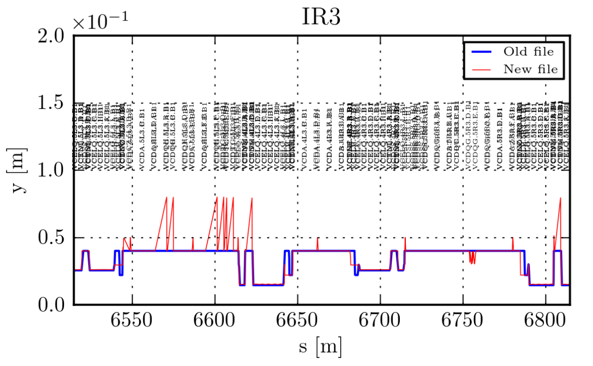


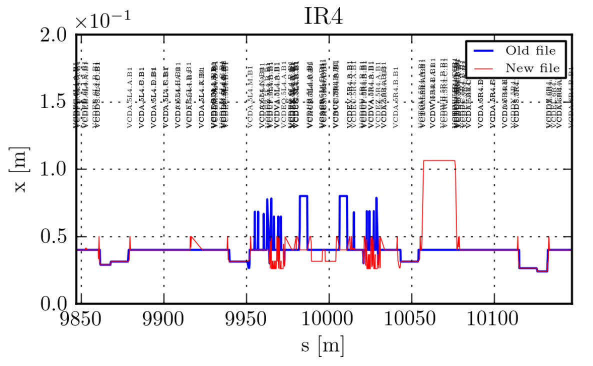


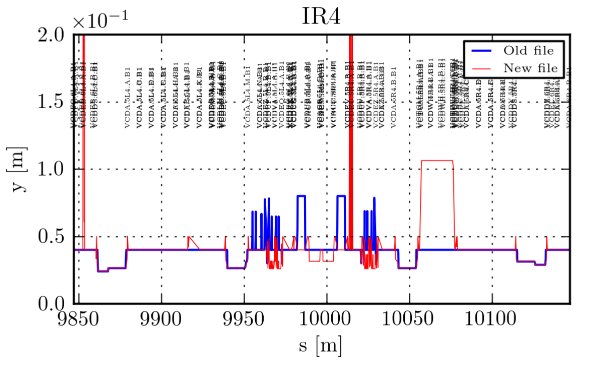


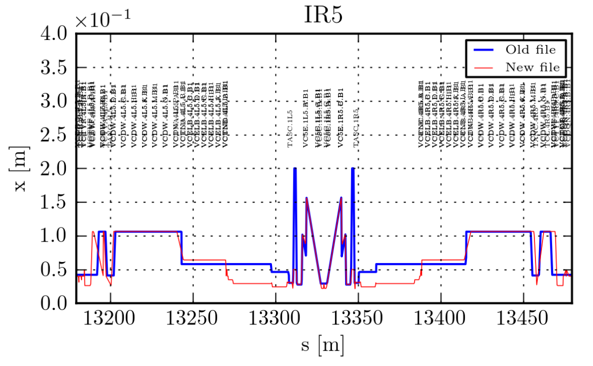


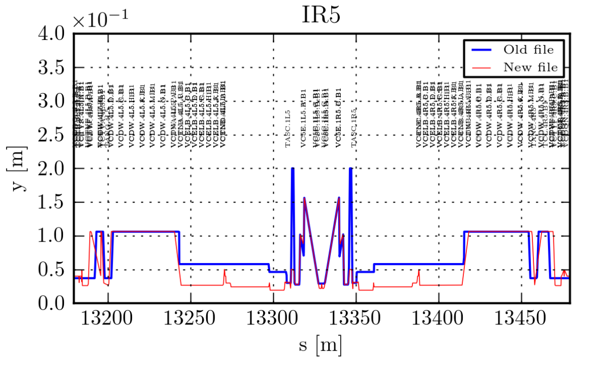


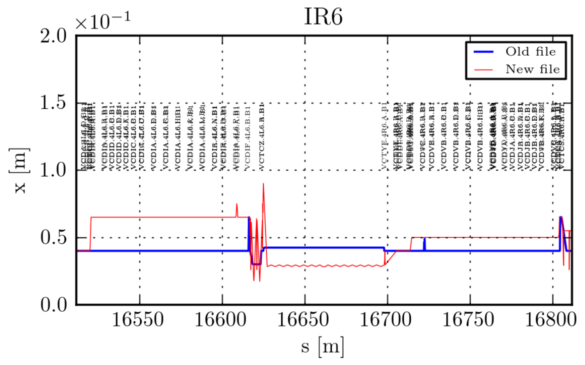


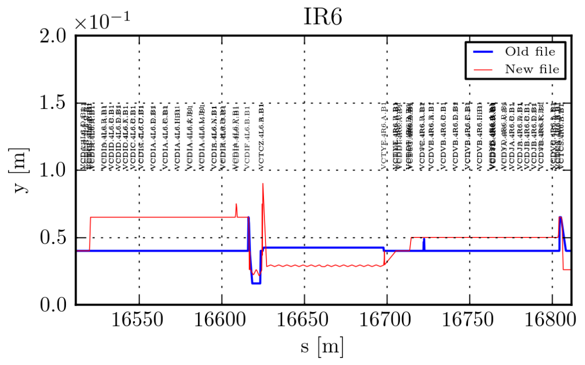


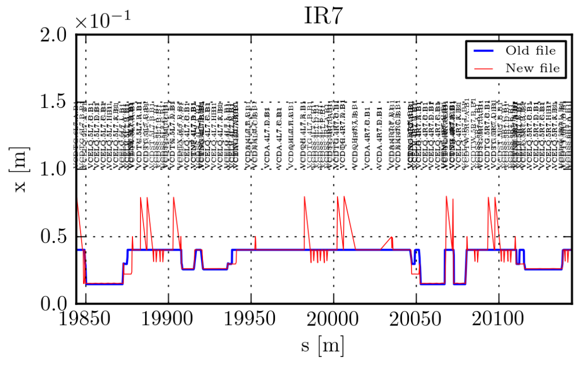


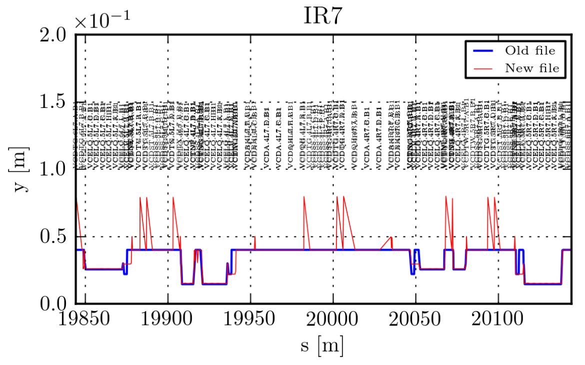


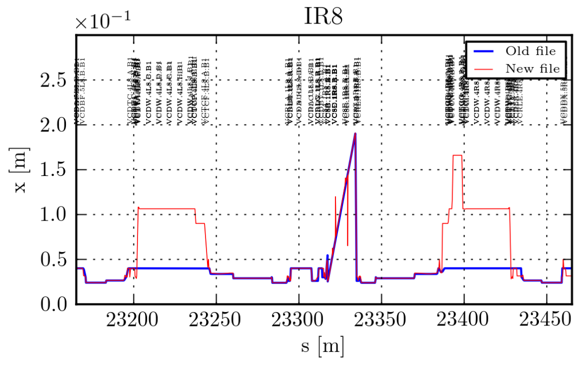


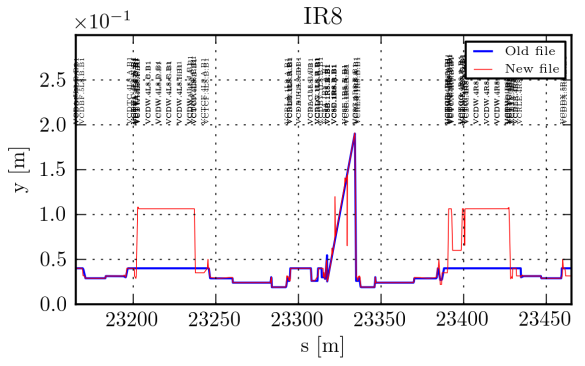












3.2. With vacuum markers, experimental beampipe and survey

3.3. Without vacuum markers or experimental beampipe

References

[1] LHC Collimation Project website. `https://lhc-collimation-project.web.cern.ch/lhc-collimation-project/BeamLossPattern.htm`.

[2] GitHub repository. `https://github.com/KFubuki/allapert_generator`.

[3] Defining aperture in MAD-X. `http://madx.web.cern.ch/madx/madX/doc/usrguide/Introduction/aperture.html`.