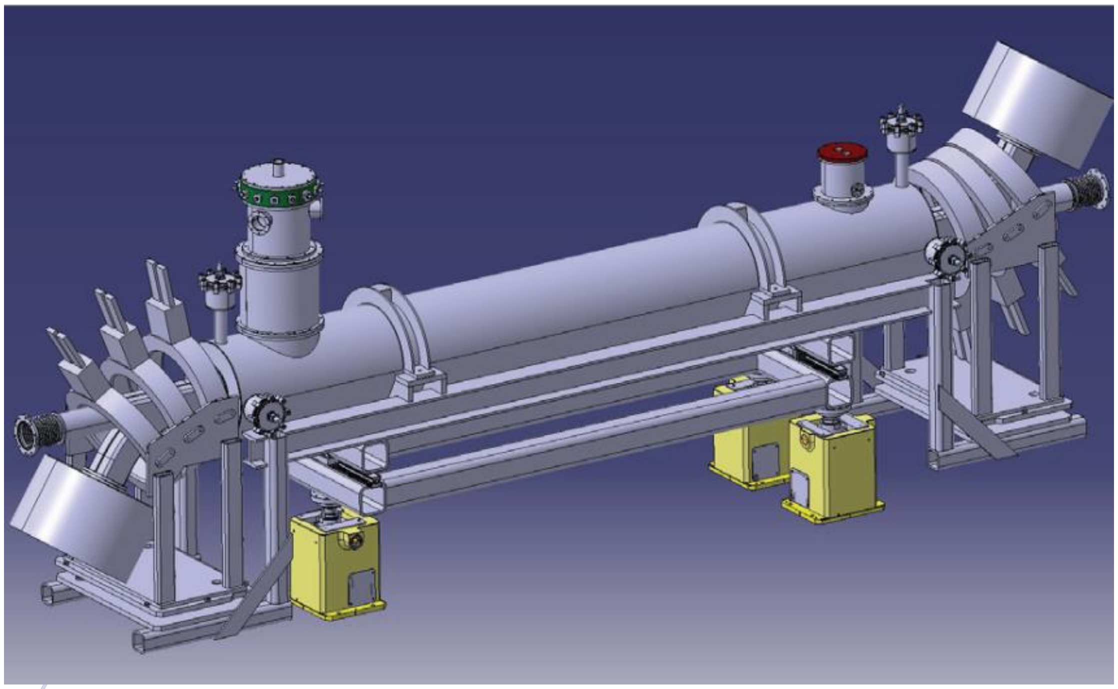
Mphys Project Report
Erasmus Year

Collimation of the High Luminosity Large Hadron Collider using a Hollow Electron Lens



Semester S7 (September 2017 – January 2018)

By Maxime Rey,
With Luke Dyks

Supervised by Rob Appleby, Haroon Rafique and Sam Tygier

Contents

Introduction

The LHC (Large Hadron Collider) is the biggest particle accelerator in the world and has the highest luminosity. It is circular with a perimeter of 26.7 km (we will use s or z to denominate this coordinate). It is used especially with protons and there are 2 beams in opposite directions, both in a different pipe. There is a ~10^{-10}mbar vacuum and the cryogenic parts (where the superconducting electromagnets are) are cooled down to 1.9 K with liquid Helium. The accelerator is composed of 8 octants as shown in figure 1. In this schema, the IP are the interaction points (Atlas, Alice, CMS and the LHCb are detectors). It was initiated in 2008 and will be stopped in 2019 in order to upgrade it to the HL-LHC (High Luminosity) with a luminosity 5 times higher (by increasing the proton population) and the beam energy will go from 362 MJ to 675 MJ per beam.

Some important parts to notice in the HL-LHC are the multipoles (bipoles (to bend), quadrupoles (to focus), sextupoles and octupoles), collimators (movable jaws in transverse axis made with a certain material designed to absorb beam photons) and RF cavities.

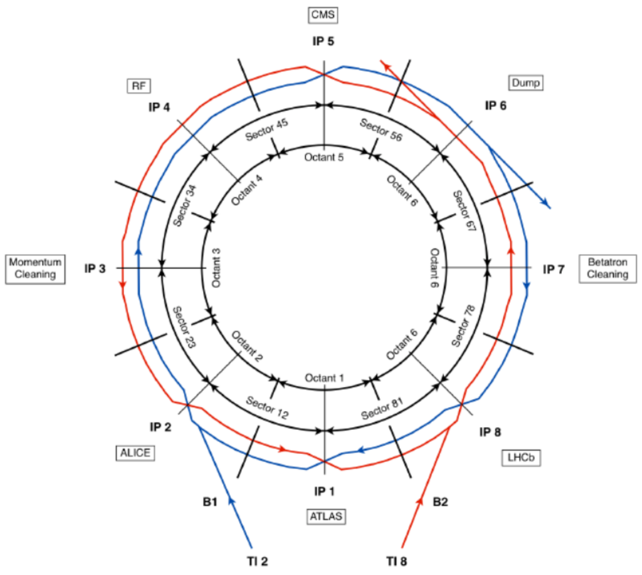


Figure 1: Design of the LHC

The proton beam is Gaussian and we distinguish "the core" $\in [\![-3\sigma , 3\sigma]\!]$ and the "halo" ($>3\sigma$), as shown in figure 2. The core (99.7% of the beam) is the part that we want to keep and the halo (0.3%) is the potentially dangerous particles that may scatter and damage the components of the accelerator. From past years experiments, it was concluded that the transverse profile population wasn't exactly Gaussian in the LHC and that we would have to remove even more particles (cf. blue lines in figure 2).

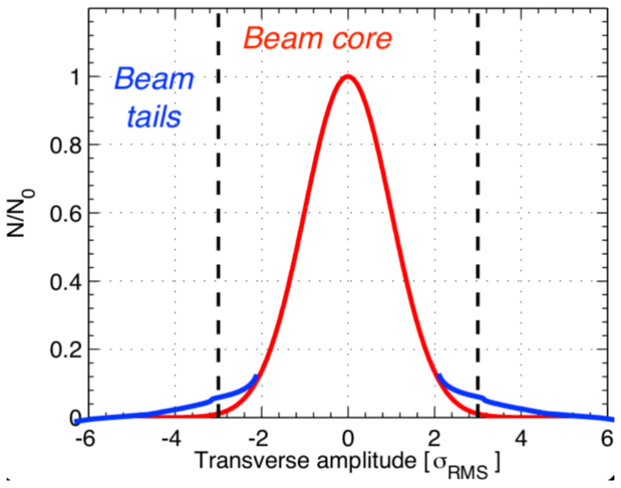


Figure 2: Beam population (image from [2])

Part I
Global understanding

1 The Hollow Electron Lens (HEL)

The HL-LHC will need a collimation system because of the Betatron oscillation and the intra-beam scattering (there will also be reasons for losses such as failure in different components and dust created by them, the imperfect vacuum, etc.).

The way to remove the halo is by making it crash into the collimators and it must have an efficiency of at least 99.99992% in order to be below the quench limit (47 mW.cm^{-3}, data from [1]). In 2008, there was a quench and the consequences were disastrous: tons of helium leaked out, the vacuum was broken, etc. Two ways have been thought of to clean this halo: the HEL (Hollow Electron Lens) and the crystal collimation. They work as shown in figure 3.

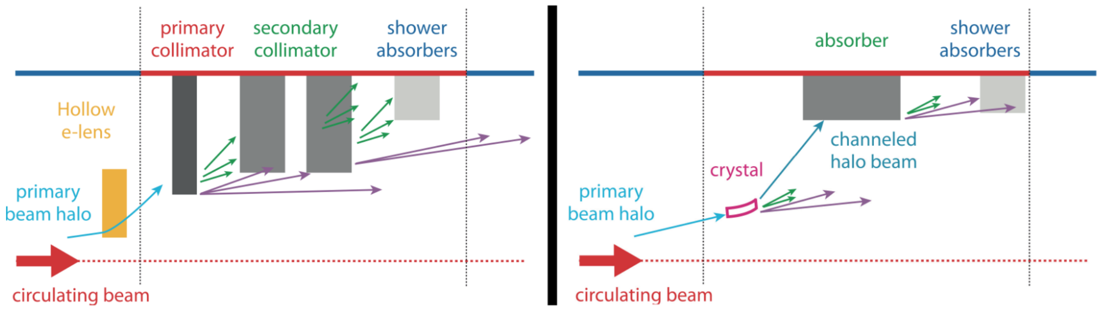


Figure 3: Two different ways of collimating: the HEL and the crystal collimation (schema from [1])

The key component that we studied to remove the halo is the HEL. It is a hollow cylinder (an annular beam of electrons) which will go around the beam halo and that will kick the particles out in the jaws. It was first put in the Tevatron Collider (the TEL, see [1]). The shape of the hardware is as shown in figure 4. The assumption was made that it would not affect the proton beam at the injection and extraction points as it didn't affect it in the Tevatron.

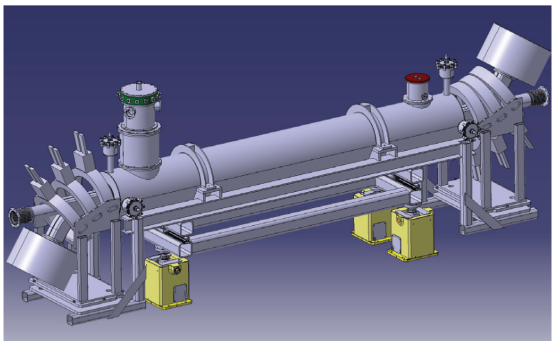


Figure 4: Design of the HEL

We had to study how it works, how different parameters influence it and how we could optimize it by simulating its use in the HL-LHC with Merlin 5 (a C++ Library for the HL-LHC).

We also had to consider that a cold section was necessary (because the HEL needs superconductivity), that the two beam pipes have to be separated to avoid beam-beam interactions and that it needs space (we did the tests with a length of 3m).

2 The Lossmaps

First, I had to get a grasp of how Merlin was working, how to output data from it and launch simulations so I first looked at the lossmaps. The lossmaps show where the particles are lost in the accelerator. For instance I learned how to output and plot the data out of the files, launch simulations, etc. I hence did plot some lossmaps as shown in figure 5. I ran many simulations with different number of turns and different number of particles. It showed clearly where most of the losses were and how long the simulations took depending on the number of turns/particles. It confirmed that most of the losses were effectively in the collimators (TCP, TCS, etc.). There were however a few unwanted losses in other spots.

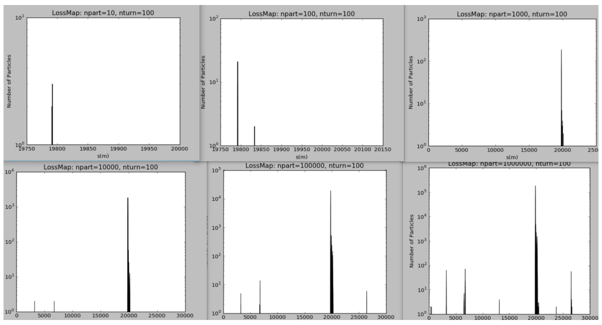


Figure 5: Lossmaps for 100 turns and different numbers of particles: from top left to bottom right: x10 particles between each plot (log scale)

The figure 5 shows that it takes a lot of particles in order to see where all the losses are. For each of these simulations, new peaks appear and if more were done, more peaks would have been seen as it is a log scale. Also, the more particles there are, the longer it takes. For instance, plotting 10^6 particles took about 6h when it only took 30s for 10 particles. A balance has then to be found between the time it takes and the precision wanted. Another parameter to take into consideration for the time of the simulation is the amount of data taken out (taking out the number of particles each turn, taking out only the position of the lost particles, both, etc.). So, for other simulations we have to choose carefully the amount of particles we want, the number of turns and the data we take out so that it doesn't take too much time.

3 The Distributions

The real space (x,y) and the phase spaces (x',y'), (x,x'), (y,y') particles distributions (x' $= \frac{dx}{ds}$ and y' $= \frac{dy}{ds}$) also had to be studied. Some were already created in the Merlin Library and I plotted them while Luke Dyks made a "Poincare Distribution" which will be explained in section §4. There were some which we didn't use, such as the Horizontal Halo Distribution (a horizontal line), but which were still useful to compare their codes and understand how they work. There is also the Normal Distribution which represents well enough the proton beam (Gaussian) but since we didn't intend to act on the core of the beam, we mostly used the HEL Halo Distribution shown in figure 6 as it would take less time. Working on the core beam would have made all the simulations really slow so this one was without the core of the beam: only the halo was populated. The limit of it is that we will eventually have to take it into account to check that we do not indeed affect the core of the beam.

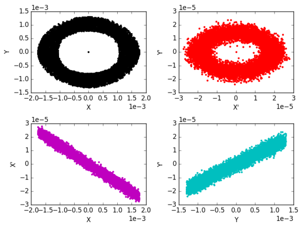


Figure 6: HEL Halo Distribution

4 The Poincare Section

The Poincare distribution is a bunch of particles, not many, whose evolution is studied through their trajectory. They are evenly dispatched on the transverse axis x with xϵ $[0\sigma\ ;\ 10\sigma]$ as shown in figure 7. Here, there are 64 particles with a constant space between them in the (x,y) plane.

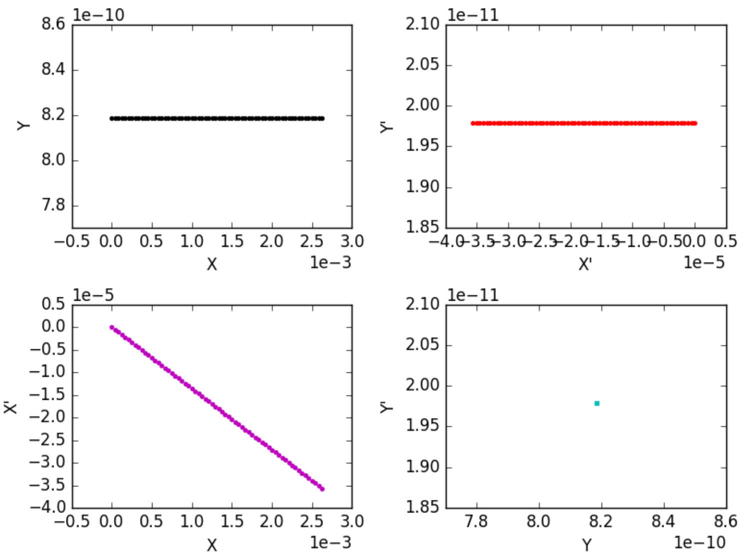
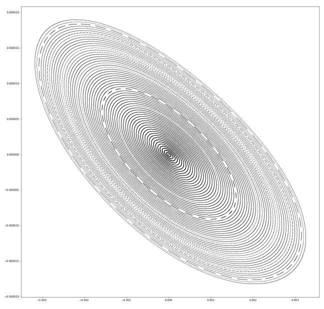
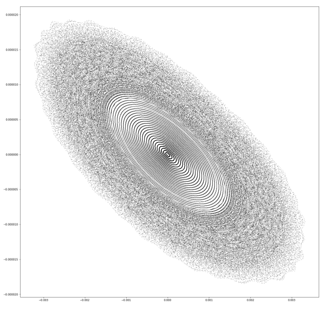


Figure 7: Poincare Distribution

The Poincare section will consist of plotting the (x, x') coordinates of the particles with the Poincare distribution and it will tell about the behaviour of the beam. For example, we can clearly see the difference between the stable and unstable orbits in the figure 8, which is a Poincare section with and without the HEL. The purpose of the Poincare section can also be noticed by changing the radius (either the inner or the outer radius) of the HEL as shown in figure 9. The "islands" that can be seen are resonances (I did put a bit more details on it in section §6)



(a) Without HEL



(b) With HEL

Figure 8: Poincare sections with and without HEL

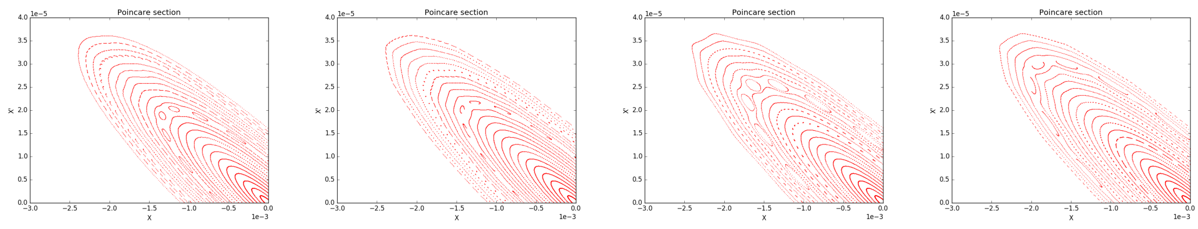


Figure 9: Poincare section for different outer radii (7.53m, 8m, 9m and 10m) for the HEL

5 The Beta Function

The beta function gives us the transverse amplitude of the particles throughout the accelerator. As the particles oscillations fluctuate with the magnets, they will have the same variation rates as of the lattice. It varies following the equation $\sigma(s) = \sqrt{\epsilon.\beta(s)}$, σ being the Gaussian width and ϵ being the beam emittance (almost a constant). It evolves as the figure 10 shows.

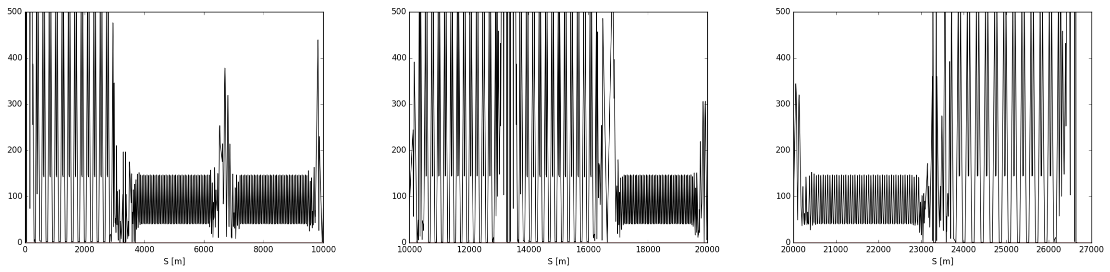


Figure 10: Beta function

6 The Tune

Another thing which needed to be analyzed was the tune. It is the frequency of the transverse oscillation. Resonances occur when a particle is at the same point in phase space through each turn; it may provoke huge oscillations and affect the beam core. Then, to avoid resonances, the tune has to be neither integer nor semi-integer (ideally, it is an irrational non-fractional number). Hence, when talking about the tune, only the decimals are given as the integer part will not change anything. Also, the horizontal and the vertical tunes (respectively Qx and Qy) have to be different to avoid coupling resonances. The following resonance condition has to be avoided: $m{\cdot}Qx + n{\cdot}Qy = p$ (m, n, p being random integers and |m|+|n| being the order of the resonance), which means it must not be where the lines cross on figure 11 (where it is plotted up to the twelfth order). I hence took the tune out of the code to check that they were indeed different and it was the case. However, for further studies, resonances may be induced to kick the beam halo.

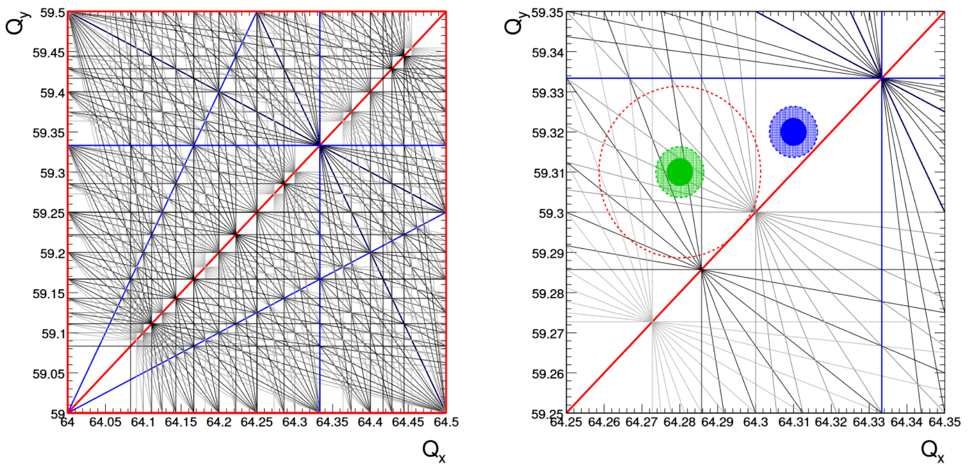
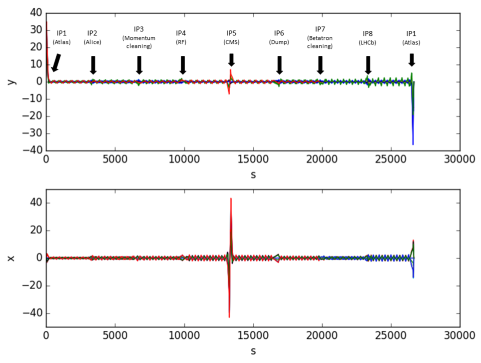


Figure 11: Full tune diagram and zoom on the injection and collision working point of the LHC (graph from [3])

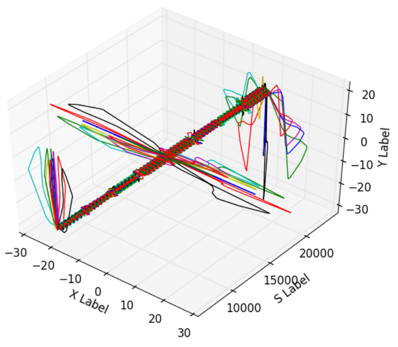
7 Particle Tracking

I could also track the particles throughout the accelerator, using Merlin to take out their (x,y,z) coordinates. It shows well how the particles behave and how big their oscillations may be. On the following graphs, you can see the total tracking of 10 particles throughout the HL-LHC in figure 12 and a zoom in in figure 13.

There are 8 slightly bigger fluctuations around the interaction points (IP); it is because before the IP, magnets are used to provoke a big oscillation which will be used to make the beams from the two pipes collide in a really precise point (the bigger the oscillation is, the more precise this interaction point will be) which will be really useful for example for detectors such as Atlas or Alice.



(a) Full trajectory in 2D



(b) Trajectories in 3D (should be circular but would not be readable)

Figure 12: Trajectories for 10 particles in the HL-LHC

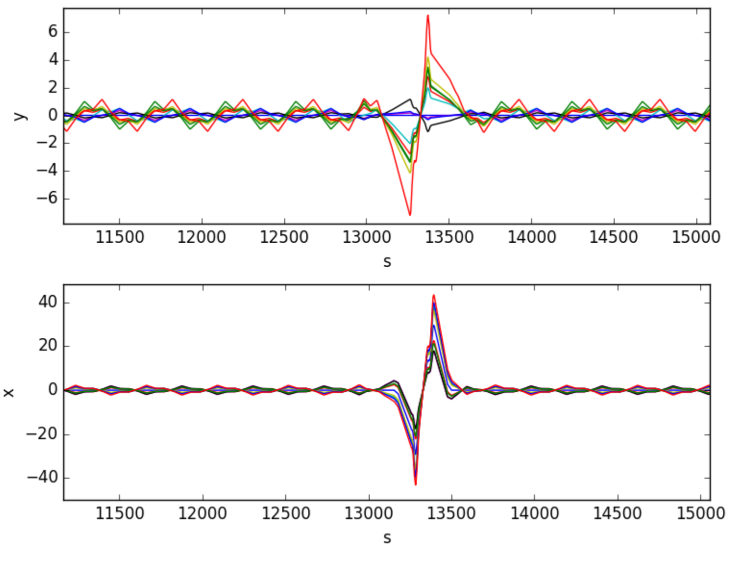


Figure 13: Zoom of the trajectories around the CMS (IP5)

Part II

Optimizing the HEL

8 The HEL kick

The force of the HEL on a proton is given by equation (1) (calculation in [1]), with v_e (resp. v_p) the velocity of an electron (resp. of a proton), $\beta_e\beta_p = \frac{v_e.v_p}{c^2}$ normal electron and proton velocities, I the HEL current and q the electron charge.

$$F(r) = \frac{Iq(1 \pm \beta_e\beta_p)}{2\pi.\varepsilon_0.v_e.r} \tag{1}$$

The angular kick of the HEL is then given by equation (3) (calculation detailed in [1]) with r the particle radius and $(B\rho)_p$ the beam rigidity. f(r) modulates the charge density and is given by equation (2) with R_{min} and R_{max} being the radii of the HEL.

$$f(r) = \begin{cases} 0 & r < R_{min} \\ \frac{r^2 - R_{min}^2}{R_{max}^2 - R_{min}^2} & R_{min} < r < R_{max} \\ 1 & R_{max} < r \end{cases} \tag{2}$$

$$\theta_{kick}(r) = f(r).\frac{1}{4\pi\varepsilon_0.c^2}\frac{2L_{HEL}I(1+\beta_e\beta_p)}{(B\rho)_p.\beta_e\beta_p}\frac{1}{r} \tag{3}$$

By removing the constants for a clearer vision, expressing the radii in σ units ($r = a\sqrt{\beta\varepsilon}$) and normalizing it with $\sigma_{x'} = \sqrt{\gamma\varepsilon}$ Luke Dyks simplified equation (3) to equation (4)

$$\theta_{kick}(\sigma_{x'})\alpha\frac{I}{\varepsilon\sqrt{\beta\gamma}}\left[\frac{A^2 - A_{min}^2}{A_{max}^2 - A_{min}^2}\right] \tag{4}$$

Considering the maximum kick and that under certain conditions ($\beta \gg \alpha$) $\gamma = \frac{1}{\beta}$, we have $\frac{1}{\varepsilon\sqrt{\beta\gamma}} \sim \frac{I}{\varepsilon\sqrt{\beta.\frac{1}{\beta}}}$ and since ε is quite constant, we get equation (5) : the kick is approximately proportional to the HEL current.

$$\theta_{kick}(\sigma_{x'})\alpha I \tag{5}$$

Moreover, by plotting the kick $\theta_{kick}(r)$ as in figure 14, it is possible to see how it varies depending on the position of the proton.

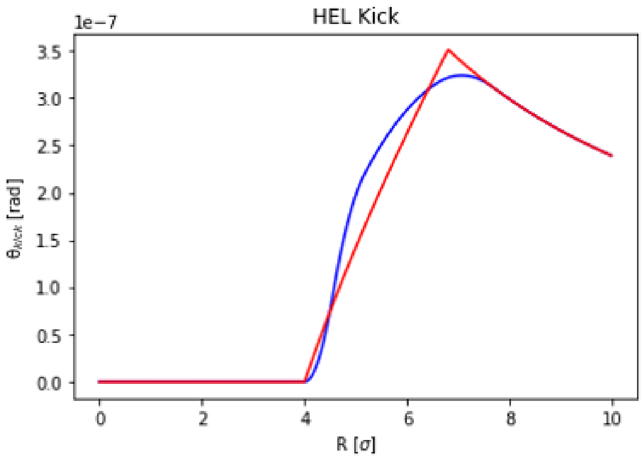


Figure 14: HEL kick depending on the position of the proton it affects (perfect model in red and real one in blue)

9 Tests over intrinsic parameters the HEL

9.1 Changing the radius

As shown in section §4, I tried to change the inner and outer radii of the HEL to check if the behaviour was as expected and it was: if the radius is changed, the area that the HEL affects changes as well; as shown in figure 15.

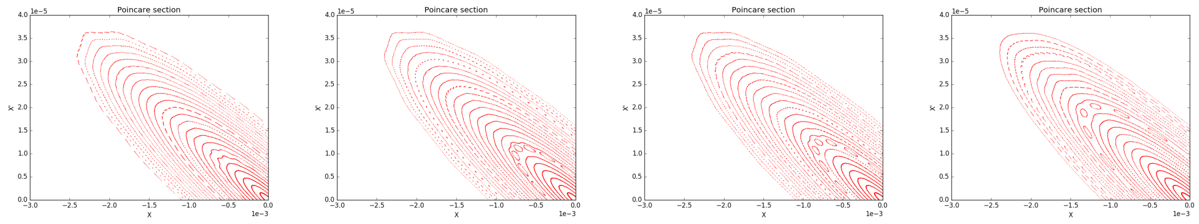


Figure 15: Poincare section for different inner radii (1.5m, 1.75m, 2m and 4m) for the HEL

9.2 Changing the current/length

I also tried to change the current of the HEL. It changes at a rate similar to the length of the HEL so I also tried with unreasonable current values.

The ones displayed in figure 16 are for 0m (no HEL), 3m, 15m and 30m which would approximately be corresponding to 0A, 5A, 30A and 50A. I could have done more tests to have a more precise correlation between length and current but that would not have been useful. The idea is that the bigger it is (or the more current there is) the further the particles will be kicked in the (x, x') space.

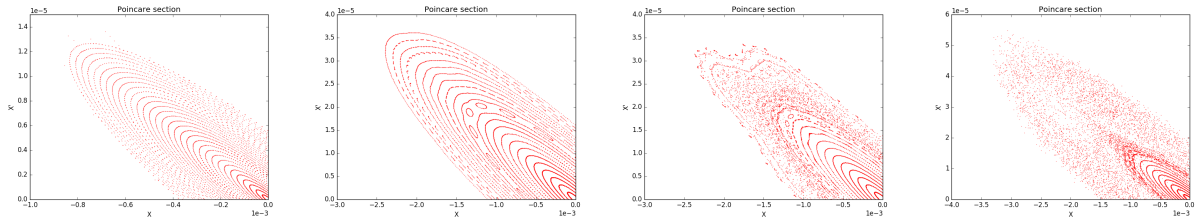


Figure 16: Poincare section for different length (1.5m, 1.75m, 2m and 4m) for the HEL

9.3 Changing the energy

The last intrinsic parameter I could analyze was the energy (I also tried with unreasonable energy values). As expected, the 0eV value is the same as without a HEL. The plot with 10 eV seems false but it makes sense. Since some approximations have been made, the results are not true anymore with small energies. Between 10^3eV, 10^4eV (not shown but similar) and 10^5eV, the results are similar: the trajectory in phase space for the particles the closest to the core ($x < 0.9$ eV) is unchanged and the differences for the other particles stays small.

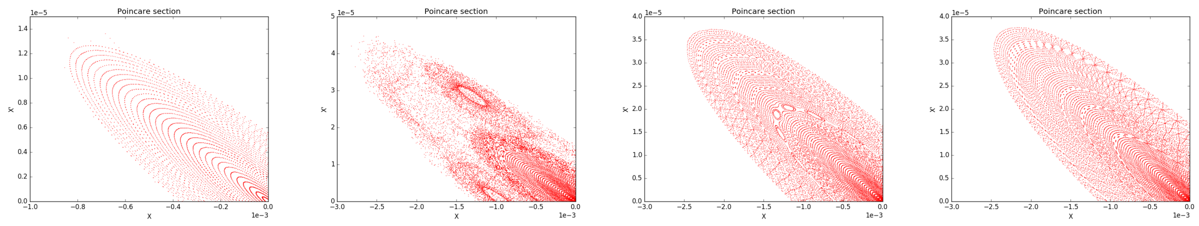


Figure 17: Poincare section for different energy values (0eV, 10eV , 10^3eV and 10^5eV)

10 The Phase Advance

I also had to consider the phase of the different elements, especially the phase advance between the primary collimators and the HEL. At first, it could be thought that, depending on the phase, more or less of the halo beam would be removed; as shown by the figure 18. For example, here the red phase is the ideal phase, the blue and dark ones are worse and the phase which would be in anti-phase with the red one would be as good as the red because the collimator is also on the opposite side.

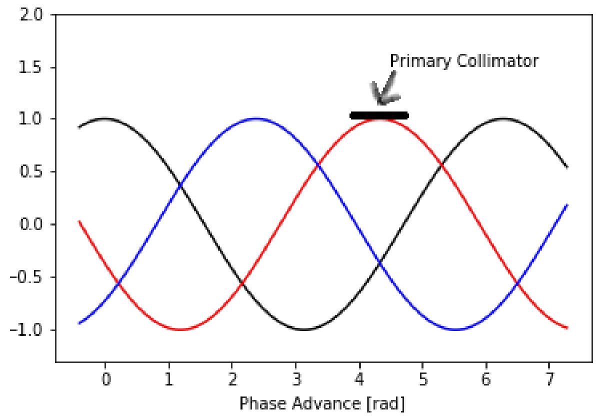


Figure 18: Naive example of the effect of different phase advances between the HEL and the collimators

However, I did a lot of tests because we thought it was an important factor for the collimation. Some of the results are shown in figure 19. For each graph, I took different positions where the phases in x (respectively y) were approximately constant and y (respectively x) had different values. It didn't go as predicted at all. In fact, there is another factor which is much more important with one HEL (which will be detailed in section §13).

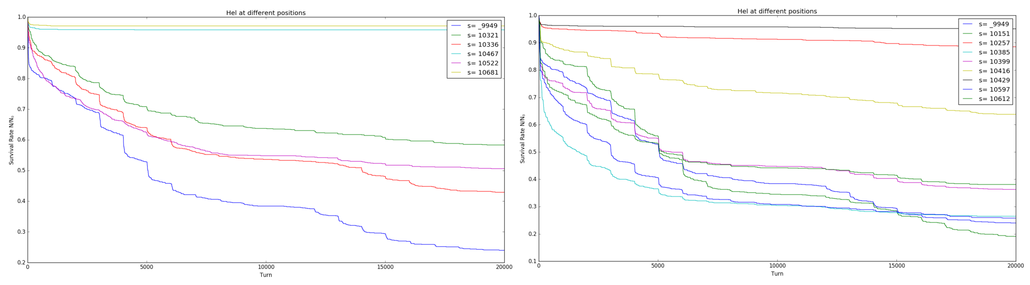


Figure 19: Different survival rates for different phase advances in x and y

Position	$\Delta\phi_x$	$\Delta\phi_y$
9949	0.39	0.12
10321	0.39	0.12
10336	0.74	0.13
10467	0.49	0.17
10522	0.27	0.11
10681	0.98	0.20

(a) Similar phase advance in y and different in x

Position	$\Delta\phi_x$	$\Delta\phi_y$
9949	0.39	0.12
10151	0.19	0.0004
10257	0.12	0.27
10612	0.13	0.96
10385	0.13	0.57
10399	0.12	0.42
10416	0.12	0.27
10429	0.10	0.24
10597	0.13	0.03

(b) Similar phase advance in x and different in y

Table 1: Phase advance values corresponding to the positions in the survival rates

The explanation is as follows: over the 20,000 turns, the phase of the HEL shifts bit by bit and creates a sort of average, as shown in figure 20. The first turn, the phase of the HEL corresponds to the blue curve, the next turn it is the red one, afterwards it is the black one and so on and so on. It would have been a problem if the machine tune was an integer or a half-integer but it is not, in order to avoid resonances. However, the phase advance has to be kept in mind as it may become important with 2 HEL.

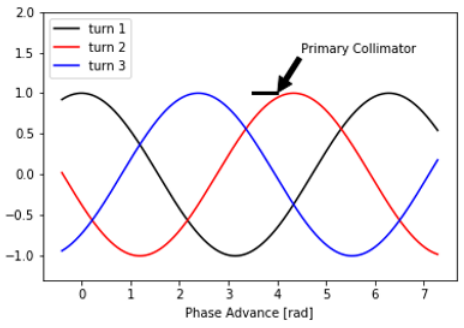


Figure 20: Schematisation of the averaging over the turns of the phase advance

11 The different modes

There are 4 ways to give current to the HEL. They are called operational modes (or just modes).

There is the DC mode which gives continuously the max current to the HEL, the Turnskip mode which switches to DC every n turns (2 by default), the AC mode which gives a modulated current (over time) and the Diffusive mode which gives either a randomly modulated current on a turn by turn basis or switches randomly the HEL on and off.

The 4 Poincare section are shown in figure 21. It can already be seen that the 2 most efficient seems to be the AC and the diffusive mode.

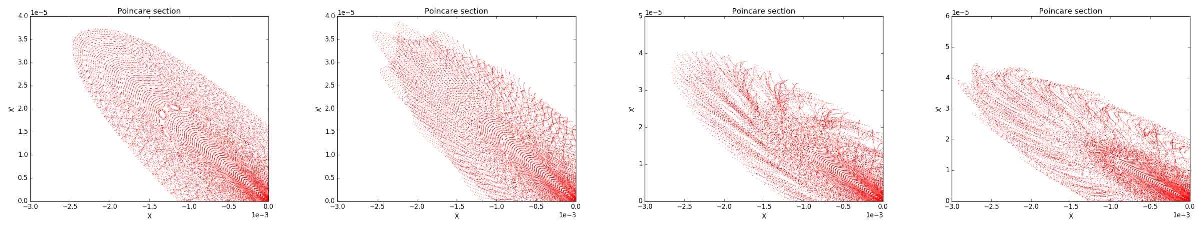


Figure 21: Poincare section for different modes (DC, Turnskip, AC and Diffusive) for the HEL

The fact that the two best modes were AC and Diffusive became obvious seeing the Survival rates for the different modes shown in figure 22. The DC mode was just a bit better than no HEL and the Diffusive mode may be harder to put in practice and could be damaging for the components (by randomly switching the HEL from off to max and then off again). Considering these and the fact that we have less control on the Diffusive mode than on the AC mode, we chose to study the AC mode.

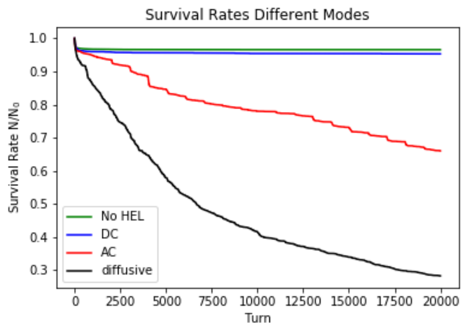


Figure 22: Survival rates for different modes

12 The AC mode

12.1 The HEL kick (AC mode)

The kick has now to be considered differently, as shown in equation (6) (cf [1]) where ν_{op} is the AC tune given by the user, T the turn and m an integer.

$$\theta = \theta.\frac{1}{2}(1 + cos(m.T.2\pi.\nu_{op})) \tag{6}$$

This is because of how the AC mode works: it attempts to be in resonance with the machine tune. The consequence is that the kick is modulated (see calculus in [4]).

The two things we now had to test were the HEL current and its tune.

12.2 Changing the Current

With this parameter, the result was quite predictable. A higher current provokes a stronger kick which will give a better collimation. The survival rates in figure 23 show this very clearly.

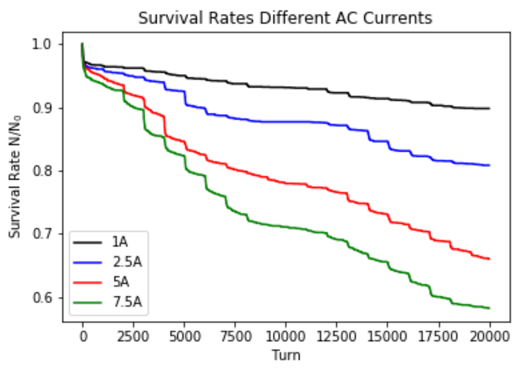


Figure 23: Survival rates for different currents

12.3 Changing the Tune

Now, an interesting parameter was to be considered: the tune. The AC tune is given by equation (7) (cf [1]) and is necessary to define an harmonic frequency $\varphi = m.T.2\pi.\nu_{op}$ which is used to modulate the AC kick. ν_{min} is a set minimal tune, w is a step and δ_{nu} is the corresponding tune step set by the user.

$$\nu_{op} = \nu_{min} + w{\cdot}\delta_{nu} \tag{7}$$

As told in section 12.1, for best results, the AC tune has to be the closest possible to the accelerator tune. The accelerator tunes are Qx = 62.31 and Qy =60.32 so we had to match them as much as possible to the AC tune. Since the default tune was 0.31, we tried the average of Qx and Qy: 0.315. As the figure 24 shows, the best kick is with the average of Qx and Qy. However, when Luke Dyks did plot the survival rates for some other tunes and checked them, we discovered that the best value wasn't the average 0.315 but the value 0.3125 as shown in figure 25.

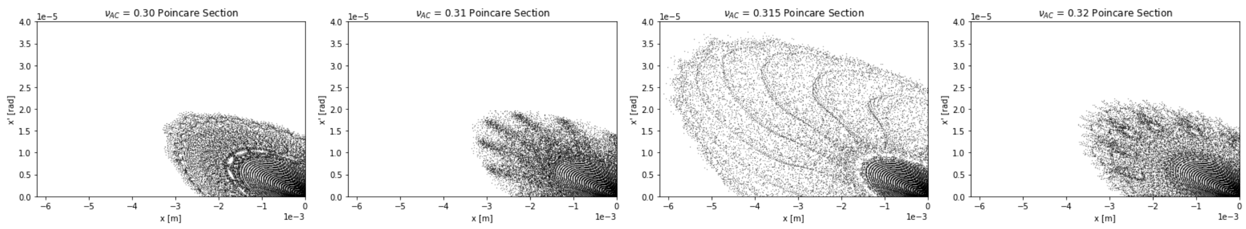


Figure 24: Poincare Sections for different tunes

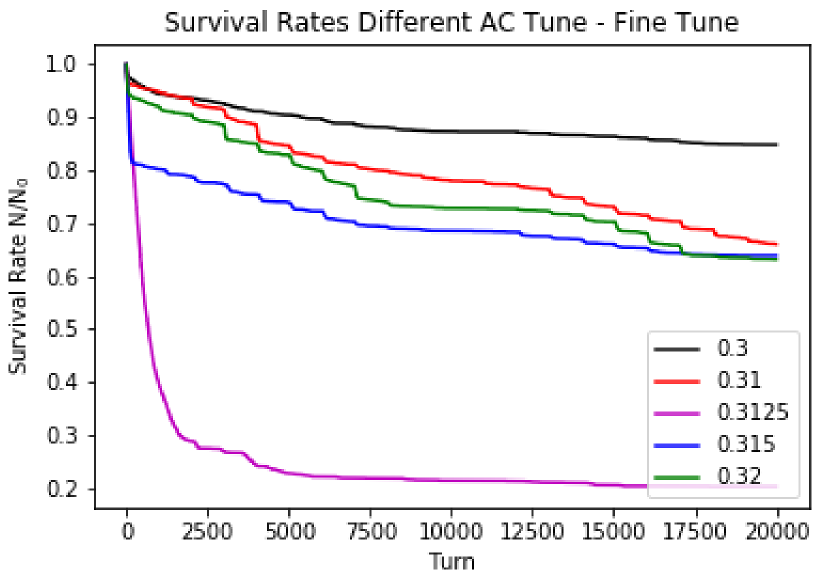


Figure 25: Survival rates for different tunes

13 The shape of the beam

I then looked at the shape of the beam. In order to do this, I chose either $\frac{\beta_x}{\beta_y}$ or $\frac{\beta_y}{\beta_x}$ (they have to be superior to 1) because I didn't want to focus on an axis in particular but on the overall shape. Whether the beam was thinner in x or in y was the same. It gave me access to the shape of the beam since β is proportional to the width of the beam. The plots I did, as in figure 26 for example, confirms well that the collimation rate for 1 HEL depends mostly on the shape of the beam. The proton beam has to match the shape of the HEL to be optimized: it has to be a round beam.

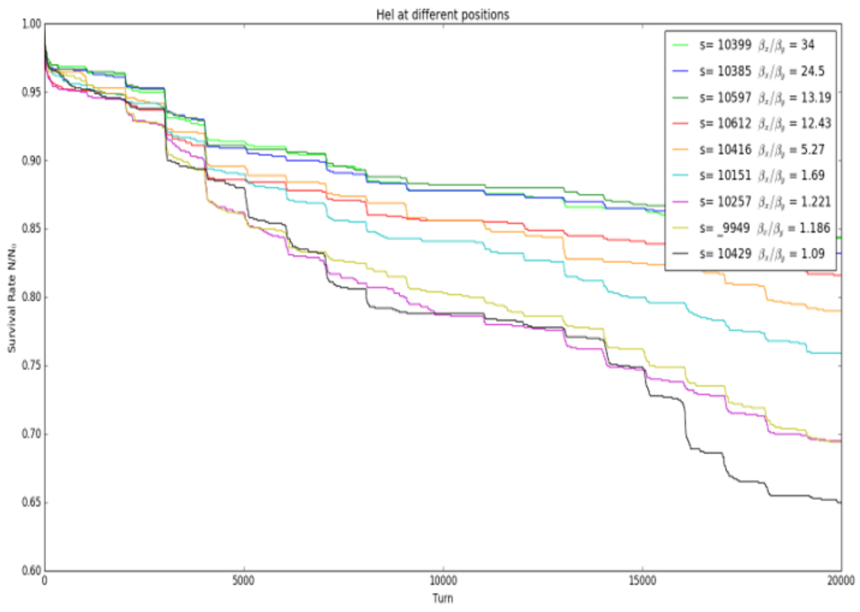


Figure 26: Different survival rates depending of the shape on the beam

Part III
Adding a second HEL

14 Finding spots for the 2^{nd} HEL

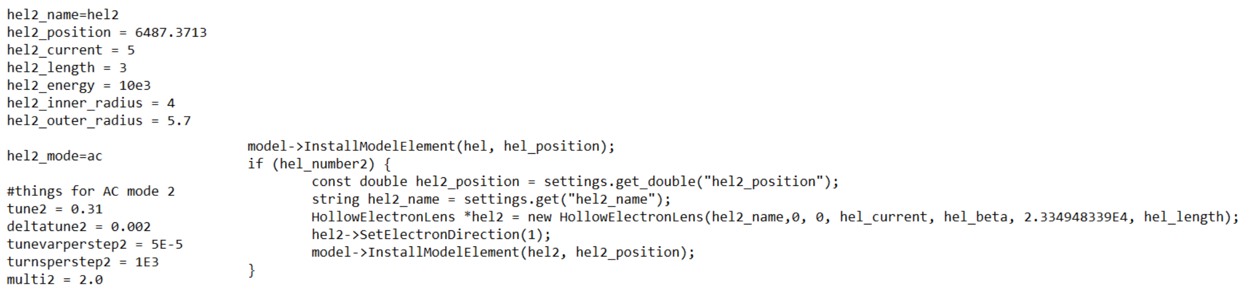
The first step to add another HEL is to find where to put it. I also did have to choose spots in the accelerator for a single HEL to see the influence of different parameters and a constraint was to find drift parts (i.e. places with free space in the accelerator) but for the second HEL, I wanted to look at it more precisely. I had to take into account that the spot had to be a drift part, a cold one (the HEL need superconducting magnets), that it had to have a length of more than 3m and that the beta function had to be small enough. Small enough first meant less than 1.05 but since there were only 2 possible spots (18 when only taking into account the drift parts, 2 in the end because some were in the detection parts), I changed it to 1.2 where I had 6 (or 30 unrealistic) possible spots. The code to take those spots (table 2) out of the lattice is displayed in the section §18.3.

Position	Ratio $\frac{\beta_x}{\beta_y}$ (or $\frac{\beta_y}{\beta_x}$)	β_x	β_x	Position near to
6487	1.0822	132.566	143.465	IP3: Momentum Cleaning (6,664)
6923.2	1.1625	84.0309	72.2838	IP3: Momentum Cleaning (6,664)
9853	1.0816	349.609	323.219	IP4: RF (9,997)
9981	1.15379	192.051	221.587	IP4: RF (9,997)
19891	1.0783	156.912	169.199	IP7: Betatron cleaning (19,994)
19986	1.08814	141.19	129.753	IP7: Betatron cleaning (19,994)

Table 2: Possible positions for a second HEL

15 Modifying and testing the code

We then had to add the HEL in the code. The changes weren't so big in the beginning but a good understanding of how they were implemented was necessary. I first added the HEL with the same parameters as the first one and then Luke Dyks added the possibility to put specific parameters.



```
hel2_name=hel2
hel2_position = 6487.3713
hel2_current = 5
hel2_length = 3
hel2_energy = 10e3
hel2_inner_radius = 4
hel2_outer_radius = 5.7

hel2_mode=ac

#things for AC mode 2
tune2 = 0.31
deltatune2 = 0.002
tunevarperstep2 = 5E-5
turnsperstep2 = 1E3
multi2 = 2.0
```

```
model->InstallModelElement(hel, hel_position);
if (hel_number2) {
        const double hel2_position = settings.get_double("hel2_position");
        string hel2_name = settings.get("hel2_name");
        HollowElectronLens *hel2 = new HollowElectronLens(hel2_name,0, 0, hel_current, hel_beta, 2.334948339E4, hel_length);
        hel2->SetElectronDirection(1);
        model->InstallModelElement(hel2, hel2_position);
}
```

Figure 27: Example of some code changes to add a second HEL

There are different ways of testing the behaviour of the second HEL. We can put the first one on and the second one off and vice-versa or put the second one where the first one was. We could also only turn the first one on with a certain current and compare it with the results from both at the same time with half the current. In all those tests, the HEL are at the same spot (which was doable because in the code the length is considered to be 0m even if its effective length is 3m). An example of one of those tests is figure 28 done by Luke Dyks. We have 4 curves: the first HEL at 5A and the other one at 0A, the same parameters but the other way around, both at 2.5A and the first HEL at 5A (a result we had from before we added the second one). Since they were all the same (the small differences are due to a different seed for a random number generation), our simulations were behaving coherently.

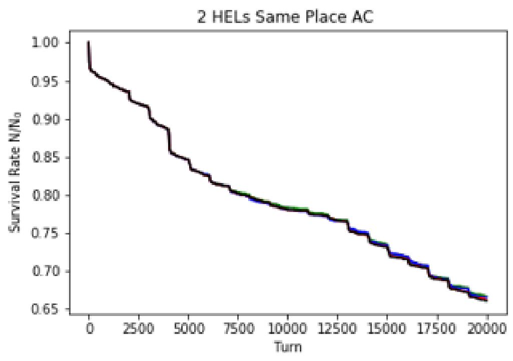


Figure 28: Example of a test for different settings

16 Compensation of the non-round beam shape

We have seen that for one HEL, the most important setting is not the phase advance but the shape of the beam. However, with the addition of a second HEL, we could now do a test: take a non-round beam position and clean the x part with one HEL and the y part with the other. As shown in figure 29, it is important to notice that it can go at least up to a collimation rate of 0.6 if the tune is well matched (we may probably even go further).

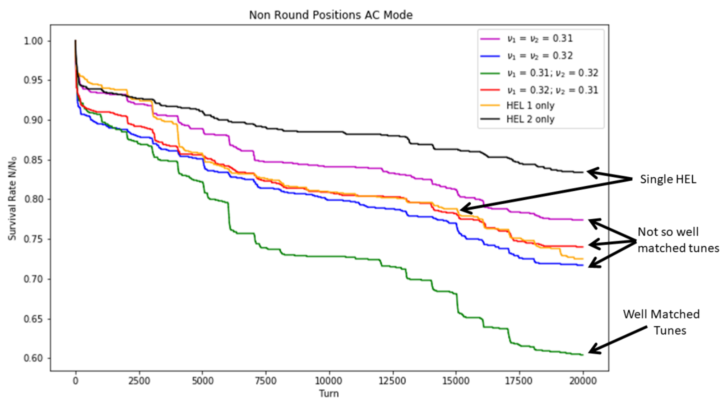


Figure 29: Tests for different tunes for each HEL. ν_1 (resp. ν_2) is the tune of the first HEL (resp. the second one)

Seeing this result, it had to be compared with the result for one HEL at a nearly round position. As the figure 30 shows, with two HEL, one acting on x and the other acting on y in a non-round beam position, we can have a better collimation than with one HEL acting on a round position. The restriction on the round position for the HEL may be lifted. But we will now have to consider the current: is it as good with a current halved on each HEL ? We could also probably increase the collimation even more using round positions.

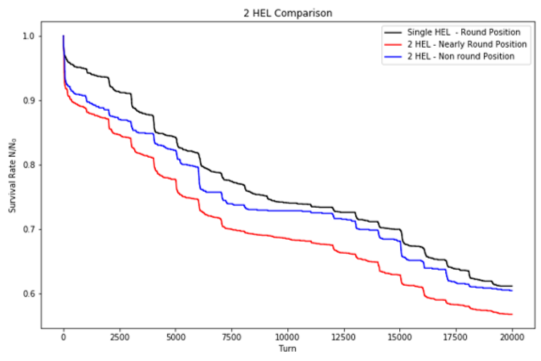


Figure 30: Survival rates for different HEL settings

17 Comparison of 2 HEL in a round beam position

While Luke Dyks was working on non-round positions, I was working on the round ones that I selected in section §14. I used the previous spot for the first HEL (9964m) and the ones displayed in figure 31 for the second HEL. There are here two phase advances between the HEL to consider: the one between HEL1 and HEL2 (phase advance 1-2 in the table) and the one from HEL2 to the accelerator end added to the one from the beginning of the accelerator to HEL1 (phase advance 1-2 in the table). It is the other way around if HEL2 is closer to the beginning of the accelerator than HEL1. However these curves don't seem to evolve according to any parameter I have listed in table 3: neither the shape nor the different phase advances or any other parameters that I tried. I even tried to check the distributions shape as shown in figure 32 but it didn't evolve accordingly. The 4 best collimation rates didn't have the roundest shape and none of the phase advances did vary in the same way either. I also tried to look at the value of the Beta function but it didn't fit either.

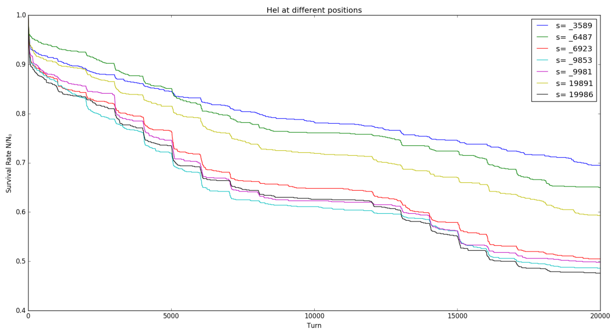


Figure 31: Different survival rates for the first HEL at 9964 and the second one at s (in the graphic)

Position	"Collimation ranking"	ratio $\frac{\beta_x}{\beta_y}$ (or $\frac{\beta_y}{\beta_x}$)	Phase advance 1-2	Phase advance 2-1
3589	7	1.01	0.29	0.66
6487	6	1.28	0.15	0.5
19891	5	1.07	0.32	0.19
6923	4	1.16	0.20	0.52
9981	3	1.15	0.14	0.13
9853	2	1.08	0.55	0.81
19896	1	1.08	0.25	0.38

Table 3: Specifications of the curves in figure 31

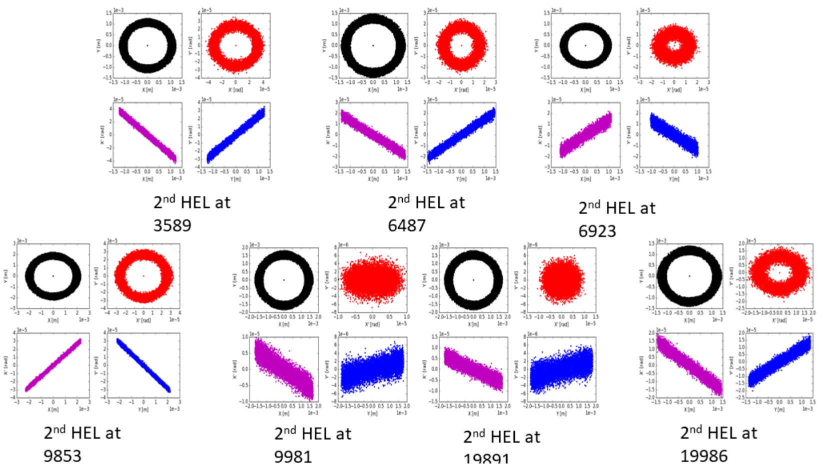


Figure 32: The distributions for the different positions in the phase spaces (x, y), (x, x'), (y, y') and (x', y')

Conclusion

We found out that we can change many parameters such as the length, the current or the energy and that the collimation will globally be better. The limit here is a technological limit: we probably won't have more than 5A, the length is very limited by other components in the LHC and it needs cold parts which makes it even more restrictive. However, we can improve the collimation by changing settings in the AC mode (changing the tune can give a better collimation), we have also seen that if we take a spot where the beam is round, the collimation will be better than in a less round spot. We have looked at this for 2 HEL and the beam shape which was the most important parameter for one HEL seemed to be less important with two because we have the possibility of compensating it by acting first on one of the transverse axis and then on the other one. Thus, it would be interesting to study how we could optimize the use of the two HEL even more. Also, the phase advance may be more important than before: we may put both HEL in resonance to try to improve the collimation (to see if this may work, we may try in a first place to cancel out the other HEL); this is an interesting parameter that we will have to look at.

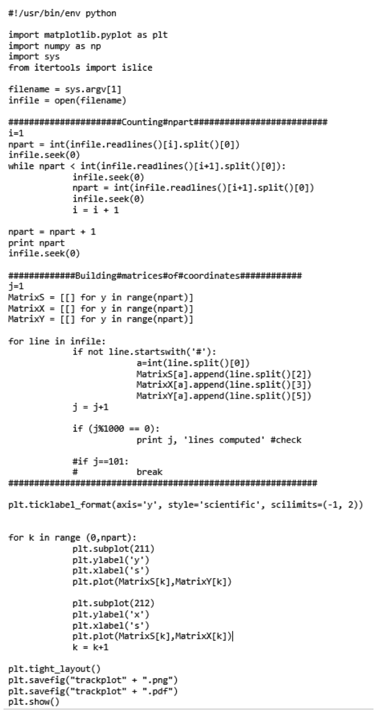
Also, another research that could be done is on the limits of the diffusive mode, if it is possible to optimize it and to see if this mode would be safe for the components (because it is a random switch from off to max current) or if it would damage them.

Appendix

18 Different codes

18.1 To track particles

This code is an example of a code which was used to take out the position of a certain number of particles npart at each step of their trajectories throughout the accelerator. In order to do this, I used an array of arrays: the coordinates in s, x and y. The 3D version is similar but a bit longer.

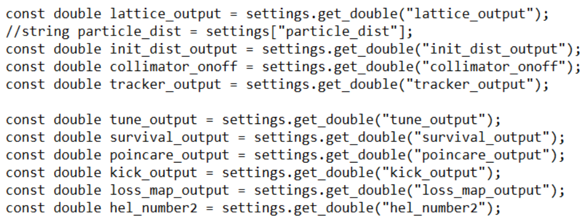


```
#!/usr/bin/env python

import matplotlib.pyplot as plt
import numpy as np
import sys
from itertools import islice

filename = sys.argv[1]
infile = open(filename)

######################Counting#npart##########################
i=1
npart = int(infile.readlines()[i].split()[0])
infile.seek(0)
while npart < int(infile.readlines()[i+1].split()[0]):
		infile.seek(0)
		npart = int(infile.readlines()[i+1].split()[0])
		infile.seek(0)
		i = i + 1

npart = npart + 1
print npart
infile.seek(0)

#############Building#matrices#of#coordinates###########
j=1
MatrixS = [[] for y in range(npart)]
MatrixX = [[] for y in range(npart)]
MatrixY = [[] for y in range(npart)]

for line in infile:
		if not line.startswith('#'):
				a=int(line.split()[0])
				MatrixS[a].append(line.split()[2])
				MatrixX[a].append(line.split()[3])
				MatrixY[a].append(line.split()[5])
		j = j+1

		if (j%1000 == 0):
				print j, 'lines computed' #check

		#if j==101:
		#		break
############################################################

plt.ticklabel_format(axis='y', style='scientific', scilimits=(-1, 2))


for k in range (0,npart):
		plt.subplot(211)
		plt.ylabel('y')
		plt.xlabel('s')
		plt.plot(MatrixS[k],MatrixY[k])

		plt.subplot(212)
		plt.ylabel('x')
		plt.xlabel('s')
		plt.plot(MatrixS[k],MatrixX[k])
		k = k+1

plt.tight_layout()
plt.savefig("trackplot" + ".png")
plt.savefig("trackplot" + ".pdf")
plt.show()
```

Figure 33: Code taking the coordinates of npart particles and plotting their trajectories in the accelerator

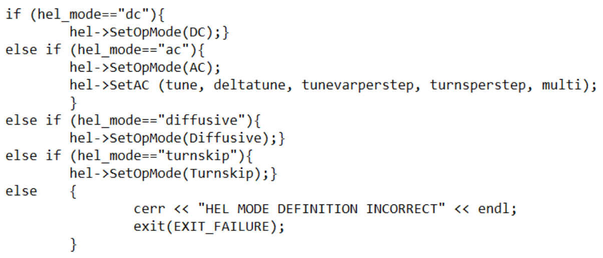
18.2 To simplify the change of parameters

Here are some parts of typical code necessary to be able to change settings (for example the AC tune) without having to re-build the code. To do this we used a file “.merlin”.



```
const double lattice_output = settings.get_double("lattice_output");
//string particle_dist = settings["particle_dist"];
const double init_dist_output = settings.get_double("init_dist_output");
const double collimator_onoff = settings.get_double("collimator_onoff");
const double tracker_output = settings.get_double("tracker_output");

const double tune_output = settings.get_double("tune_output");
const double survival_output = settings.get_double("survival_output");
const double poincare_output = settings.get_double("poincare_output");
const double kick_output = settings.get_double("kick_output");
const double loss_map_output = settings.get_double("loss_map_output");
const double hel_number2 = settings.get_double("hel_number2");
```

(a) Command to take the value of a setting in “.merlin” file



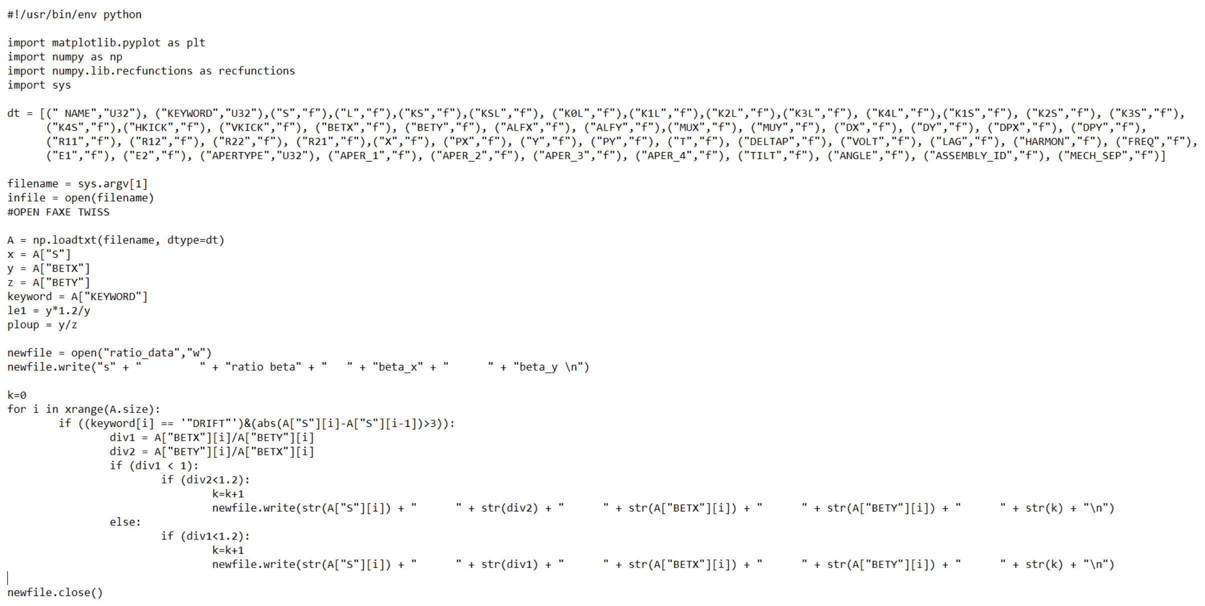
```
if (hel_mode=="dc"){
	hel->SetOpMode(DC);}
else if (hel_mode=="ac"){
	hel->SetOpMode(AC);
	hel->SetAC (tune, deltatune, tunevarperstep, turnsperstep, multi);
	}
else if (hel_mode=="diffusive"){
	hel->SetOpMode(Diffusive);}
else if (hel_mode=="turnskip"){
	hel->SetOpMode(Turnskip);}
else	{
		cerr << "HEL MODE DEFINITION INCORRECT" << endl;
		exit(EXIT_FAILURE);
	}
```

(b) Allows to change mode from another file so we don’t have to re-build the code

Figure 34: Parts of coding made to input parameters more easily

18.3 To find second HEL positions

This is a code I used to find some possible positions for the second HEL used in section §14. All the “if” commands are used for the restrictive conditions, for example it has to be in a drift part of at least 3m and the ratio $\frac{\beta_x}{\beta_y}$ (or $\frac{\beta_y}{\beta_x}$) has to be lower than 1.2.



```
#!/usr/bin/env python

import matplotlib.pyplot as plt
import numpy as np
import numpy.lib.recfunctions as recfunctions
import sys

dt = [(" NAME","U32"), ("KEYWORD","U32"),("S","f"),("L","f"),("KS","f"),("KSL","f"), ("K0L","f"),("K1L","f"),("K2L","f"),("K3L","f"), ("K4L","f"),("K1S","f"), ("K2S","f"), ("K3S","f"),
      ("K4S","f"),("HKICK","f"), ("VKICK","f"), ("BETX","f"), ("BETY","f"), ("ALFX","f"), ("ALFY","f"),("MUX","f"), ("MUY","f"), ("DX","f"), ("DY","f"), ("DPX","f"), ("DPY","f"),
      ("R11","f"), ("R12","f"), ("R22","f"), ("R21","f"),("X","f"), ("PX","f"), ("Y","f"), ("PY","f"), ("T","f"), ("DELTAP","f"), ("VOLT","f"), ("LAG","f"), ("HARMON","f"), ("FREQ","f"),
      ("E1","f"), ("E2","f"), ("APERTYPE","U32"), ("APER_1","f"), ("APER_2","f"), ("APER_3","f"), ("APER_4","f"), ("TILT","f"), ("ANGLE","f"), ("ASSEMBLY_ID","f"), ("MECH_SEP","f")]

filename = sys.argv[1]
infile = open(filename)
#OPEN FAXE TWISS

A = np.loadtxt(filename, dtype=dt)
x = A["S"]
y = A["BETX"]
z = A["BETY"]
keyword = A["KEYWORD"]
le1 = y*1.2/y
ploup = y/z

newfile = open("ratio_data","w")
newfile.write("s" + "          " + "ratio beta" + "   " + "beta_x" + "      " + "beta_y \n")

k=0
for i in xrange(A.size):
        if ((keyword[i] == '"DRIFT"')&(abs(A["S"][i]-A["S"][i-1])>3)):
                div1 = A["BETX"][i]/A["BETY"][i]
                div2 = A["BETY"][i]/A["BETX"][i]
                if (div1 < 1):
                        if (div2<1.2):
                                k=k+1
                                newfile.write(str(A["S"][i]) + "      " + str(div2) + "      " + str(A["BETX"][i]) + "      " + str(A["BETY"][i]) + "      " + str(k) + "\n")
                else:
                        if (div1<1.2):
                                k=k+1
                                newfile.write(str(A["S"][i]) + "      " + str(div1) + "      " + str(A["BETX"][i]) + "      " + str(A["BETY"][i]) + "      " + str(k) + "\n")

newfile.close()
```

Figure 35: Code going through the whole lattice and taking some parameters into account to select a second position for the HEL

References

[1] Rafique, Haroon (2017) MERLIN for High Luminosity Large Hadron Collider Collimation. Doctoral thesis, University of Huddersfield.

[2] Hilumi HL-LHC project: https://indico.cern.ch/event/647714/contributions/2646155/

[3] CERN - Tune and chromaticity diagnostics: https://cds.cern.ch/record/1213281/files/p317.pdf

[4] V. Previtali et al. Numerical simulations of a proposed hollow electron beam collimator for the LHC upgrade at CERN . Technical report, FERMILAB-TM-2560-APC, 2013.

[5] Implementation of Hollow Electron Lens in SixTrack and first simulation results for the HL-LHC: http://accelconf.web.cern.ch/AccelConf/ipac2017/papers/thpab041.pdf

[6] VladimirD. Shiltsev: Electron Lenses for Super- Colliders