Johan Lundquist
Course: FYST02 15HP Bachelor Thesis VT19 Lund University

Supervisors: Åke Andersson and David K. Olsson

Frequency Map Analysis of the MAX IV Storage Rings

August 2, 2019

Contents

List of Acronyms

- BPM - Beam Positional Monitor
- EVG - Event Generator
- R3 - The 3 GeV storage ring at MAX IV
- R1 - The 1.5 GeV storage ring at MAX IV

Abstract

As a beam of particles moves through a storage ring each particle will perform transverse oscillations as a result of the focusing magnetic fields it encounters. The amount of transverse oscillations performed in a single revolution of the ring is known as the particle's betatron tune. The goal of this project was to investigate the tunes by instigating an oscillation using a short-lived, strong magnetic field to cause a disturbance to the entire beam. This was performed and positional data was collected throughout the ring. This data, and its oscillating patterns, was analyzed using an algorithm similar to a Fourier transform in order to find the beam's tune. Furthermore, investigating the tune's change between turns travelled in the storage ring allowed for a rough measurement of the beam's stability. The resulting frequency maps, maps of the betatron tune in the two transverse planes, showed the effect on the beam when it is situated at certain tunes. This provides insight into areas of tune to avoid or seek out during operations. The project was successful in creating these maps, although further studies into areas of interest within the results could be advantageous.

1 Introduction

The basis of accelerator physics is the acceleration of charged particles using magnetic and electric fields. These particles are formed into what is known as a beam, a semi-continuous stream of particles moving at high speeds. When it comes to storing a beam, maintaining a set amount of charge and energy, a storage ring is a commonly used solution. By having the particles move in a closed orbit, they could theoretically be maintained forever, assuming energy is being provided to keep the energy of the particles constant. But of course, this is not the case in practice.

In reality there are many challenges to maintaining a beam in a storage ring, among them, maintaining a good tune. As the particles move through the ring, they oscillate around a nominal orbit. The number of transverse oscillations performed throughout one revolution is known as the rings betatron tune. If these oscillations resonate between rotations, even partly, the amplitude will increase and the beam's stability will eventually be lost. Therefore many such resonances must be avoided and this is where this project becomes relevant. The goal of this project is to affect the beam's position in some point, initiating oscillations of the beam, perform a frequency analysis of the oscillating positional data and read the resulting tune. From this, a so called frequency map will be constructed, mapping how the frequency of these oscillations in the transverse planes depends on their amplitude, along with how the varying tunes will affect the beam.

Knowing how the tune is altered by momentary disturbances, resulting in transverse oscillations of different amplitudes, can be useful in many ways. During normal operation and as the beam is used for scientific study, orbital disturbances are often unavoidable. Knowing how such disturbances may affect the tune of the beam can prevent possible interruptions to delivery, such as the beam becoming unstable or even lost as accelerated particles hit the walls of the beam pipe. These kinds of disturbances are costly, in money and time, as they can disturb or even ruin ongoing experiments at beamlines, the experimental stations. Furthermore, the information provided by this project can assist in future diagnostics of the machine, as knowing the areas where the tune makes the beam unstable means staff can alter the machine settings when nearing these areas, either to avoid them or suppress the effect they have on the beam, minimizing the risk of disturbances.

1.1 The Beta Function

After an initial kick instigates the movement of the beam, the steering and focusing performed by the dipole and quadrupole magnets causes the beam's trajectory to oscillate throughout a revolution. This oscillating trajectory is described in each plane by the beta function $\beta(s)$, where s is the longitudinal position of the beam, i.e. its position throughout the ring. When describing the amplitude $E(s)$ of these transverse oscillations, or betatron oscillations, throughout the ring, the beta function is used along with the emittance ϵ to form the function below.

$$E(s) = \sqrt{\epsilon\beta(s)} \tag{1}$$

The emittance ϵ is a constant of motion set by the initial conditions and the optics of the ring, which remains constant no matter the longitudinal position s. [1]

1.2 Tunes

The amount of betatron oscillations, transverse oscillations, performed in a single rotation of the ring is known as the betatron tune of the beam, Q. As mentioned in the introduction, resonances of these tunes can lead to major disturbances to the beam. This occurs due to *optical resonances* from the magnetic optics. As the beam will pass the same magnetic structures at the same positions with each rotation, resonance in tune will result in the particles passing each magnetic structure in a possibly resonant position with the field of the magnet.

For example, a full integer tune $Q = n$, the beam performing an integer number of betatron oscillations in a revolution, will result in an optical resonance with any dipole field errors in the magnetic optics. This occurs since the beam will always pass these magnets at the same phase of the oscillating path, whereupon the dipole errors will always kick the beam with the same error, adding the kicks constructively. This results in the positional error becoming more and more extreme with each rotation until the beam is lost. The optical resonances also couple higher order resonances to higher order magnetic fields, i.e. half integer tunes, $Q = 0.5 \cdot n$, result in an optical resonance with the quadrupoles and so on.

Furthermore, having the same tune in both transverse planes may also lead to increasing instabilities. In the higher multipolar fields we see that the field's effect on the beam in one plane will depend on the beam's position in the other, thus coupling the two betatron tunes and creating *coupled resonances* whenever the two tunes couple to form an integer. So whenever $mQ_x + nQ_y = p$, where m, n and p are all integers. Below in Figure 1, a full map of all the possible resonances up to the fifth order can be seen. It should be noted that these are the fractional tunes, the tune with the integer part subtracted. [1, 4]

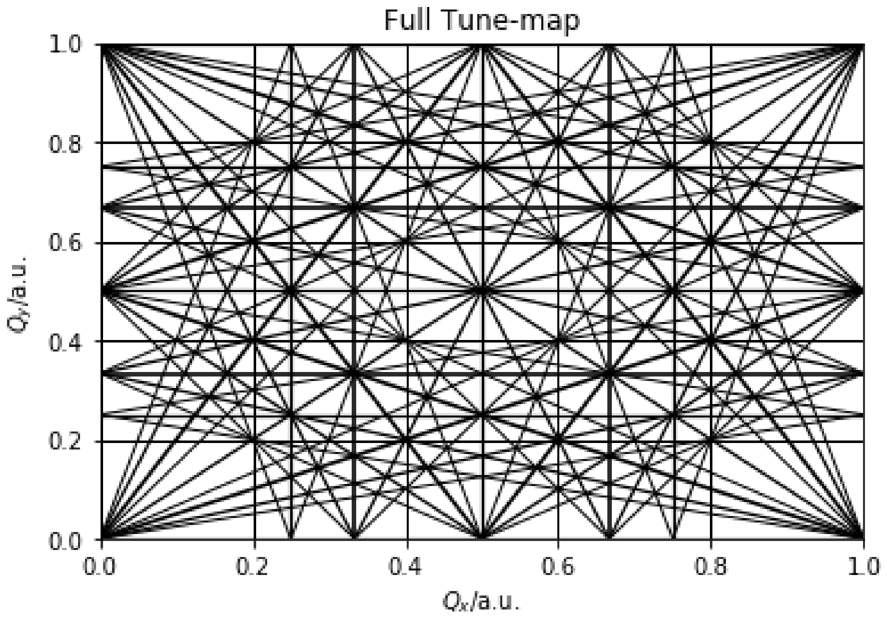

Figure 1: Full tune-map with resonance lines up to the fifth order. [4]

The effect a change of amplitude of the beam's oscillations on its tune is known within accelerator physics as *amplitude dependent tune shift*. As the position of the beam is displaced more and more from a perfectly closed orbit, we see a change in the tune follow as the beam's position effects its reaction to the magnetic optics throughout the ring. This is one of the qualities of the rings which can be extracted utilizing a frequency map. [5]

1.3 Chromaticity

Chromaticity describes the tunes dependence on the difference in momentum between particles of the beam and the nominal momentum. Say we have a beam with a momentum offset Δp from the nominal momentum of the beam, p. As this beam moves through the quadrupoles and sextupoles of the ring it will be affected differently from a beam of nominal momentum. Different amount of focusing results in different optics which results in a different tune. This resulting shift to the tune can be calculated using

$$\xi \equiv \frac{\Delta Q}{\Delta p/p} = \frac{1}{4\pi} \oint (m(s)D(s) + k(s))\beta(s)ds \tag{2}$$

Wherein $k(s)$ is the strength of the quadrupole field, $m(s)$ describes the strength of the sextupole fields, $D(s)$ is the dispersion function describing the closed orbit for a beam with unit relative momentum deviation and $\beta(s)$ is the beta function. This is what is known as the *chromaticity* of the beam. This can often be displayed in the form $(\xi_x \xi_y)$, representing the chromaticity in the horizontal and vertical planes respectively [1]. Under normal conditions the MAX IV storage rings both have a chromaticity of nearly (1 1), the chromaticity their optics were designed to have.

1.4 The Synchrotron

There are many different uses of accelerators throughout the scientific community. A synchrotron, such as MAX IV, utilizes the accelerated particles to produce what's known as synchrotron radiation. According to Maxwell, radiation is produced when charged particles are affected by a force, accelerated, and a synchrotron utilizes this phenomenon along with the laws of special relativity in order to produce a highly intensive beam of light. The light is produced at any directional change of the beam's trajectory and energy losses to this "bending" radiation is one of the challenges to overcome when constructing a storage ring. The power of the emitted light P can be classically calculated using Equation (3) below.

$$P = \frac{e^2}{6\pi\epsilon_0 m_0^2 c^3} \left(\frac{d\vec{p}}{dt}\right)^2 \tag{3}$$

Wherein e is the charge of the accelerated particle, $\vec{p} = m_0\vec{v}$ is the momentum of the particle, ϵ_0 is the permittivity of vacuum and c is the speed of light. From this we can see that the power of the emitted light is proportional to the square of the change of momentum of the particle with respect to time, or more simply, its acceleration. We can also see that the power depends on a factor $\frac{e^2}{m_0^2}$, which suggests particles such as electrons are preferable for this form of accelerator, as they are very light relative to their charge compared with for example protons. In a completely classical scenario, i.e. $v << c$, the light would be emitted with the shape of a classical dipole emitter. However, these particles are moving much closer to the speed of light, and the laws of special relativity interfere, causing an angular distortion to the emitted light. This results in the light being emitted in a sweeping cone along the bending undergone. The light used in scientific research at a synchrotron facility is produced in a similar fashion, although one utilizes an insertion device, such as an *undulator*. An undulator uses a long series of oppositely polarized magnets set along a straight section of the ring in order to produce even more intense, forward directed light as the electrons are "shook" by the oscillating fields they move through. [1]

2 Method

2.1 MAX IV

The MAX IV facility houses two large storage rings, both of which will be investigated throughout this project, and they are classified by their stored beam energy, the kinetic energy of the stored electrons. There is the smaller, 1.5 GeV ring, referred to as "R1" in this report, and the larger 3 GeV ring, referred to as "R3" in this report. The particles stored in the rings are accelerated using a linear accelerator, or a "linac" for short. In order to guide the beam along circular paths of the rings, large bending magnets are used, dipole electromagnets with iron cores. Between these larger magnets are straight sections where corrections and focusing takes place.

The focusing is performed by quadropole magnets, electromagnets with four poles, the fields of which creates a focusing effect in one plane, de-focusing in the other. These are set up in a way to create a net focusing effect. Slight corrections to the beam's path is performed using dipole correction magnets, although the trajectory changes are much smaller than those made by the bending magnets. Furthermore, there are sextupole magnets present in both rings and octupole magnets present in R3. The sextupoles are used to correct the chromaticity, i.e. how the tune depends on a momentum deviation, brought up below. However, the sextupoles are non-linear magnets and will effect the tune of a beam with non-zero amplitude oscillations. The octupoles are introduced to counteract this effect. More detailed information on the use of higher pole magnets at MAX IV can be found in the Detailed Design Report, or DDR, for the facility, specifically in chapters 2.3 and 3.3. [7]

Both rings are built up of multiple *achromats*, R3 of 20 and R1 of 12. An achromat is a collection of magnets, and in the case of the MAX IV storage rings they are set up to be optically symmetrical, i.e. the magnetic setup is repeated with each following achromat. [7] This means multiple magnetic fields and diagnostic outputs are optically identical throughout the ring. The data collected from diagnostic machinery throughout the ring will repeat along with the achromats.

2.1.1 The Pingers

There are two main dipole pinger magnets utilized to disturb the beam's trajectory for the measurements performed for this project. There is one exciting the beam in the horizontal plane, referred to as the "kicker", and one in the vertical plane, referred to as the "pinger". Both are functionally the same, except for the orientation of their generated fields. They both function through charging up a capacitor with a high voltage on the kV scale, then releasing the current into two longitudinal conductors, about 0.1 m in length, creating a strong magnetic dipole field for a short time, in the microsecond range. This field disrupts the path of the beam suddenly and will initiate oscillations of the beam around its design orbit, for several hundred turns around the ring. [1]

When working with these sorts of extreme voltages, it is important to change things carefully, in order to avoid overheating and damaging the equipment. At MAX IV, these precautions are built into the machinery, and the scripts used to alter the voltage provided by the power supplies will set the device in an alarm state for steps higher than 2000 V, and also automatically de-charges the supplies slowly if the setpoint is suddenly lowered.

The kicker magnets are discharged and activated when sent a signal by the rings event generator (EVG). The EVG has three different timings for these signals. It can either send a single pulse on command, stop all pulses being sent or pulse with the frequency of arriving electrons from the linac, 2 Hz at the time of this project. For data collection it was important to avoid the more frequent pulsing, as it resulted in the beam being almost continuously kicked with very high amplitudes, resulting in any minor beamloss with each kick being rapidly repeated until the beam is completely gone.

2.1.2 The BPM

The Beam Position Monitor (BPM) is a device used for measuring the current position of the center of charge of the beam within the beam pipe. The device consists of four electrodes, measuring the electric and magnetic fields surrounding the beam. The induced field in these electrodes is used to locate the beam's transverse position within the beam pipe. [1] There are 198 BPMs within R3 and 35 within R1. For this project they will be enumerated according to their position within the ring, starting at the injection point with BPM nr. 0.

As the beam strays further from the center of the BPM, the fields it induces within the beam pipe are no longer linearly dependent on the position of the center of charge. This can be solved through a linearization of the positional data. A study [2] shows how using a few assumptions, the dependence of the induced signal in the electrodes of the BPM on the position of the center of charge of the beam can be reduced to a two dimensional electrostatic problem, with only unknown being the induced charge on the boundary, i.e. the beam pipe. This induced charge can be found using a boundary element method, using only two assumptions. Firstly that the charge density within the beam pipe is point-like, i.e. it has a negligible transverse size in relation to the size of the vacuum chamber. Secondly that the induced charge on the boundary is constant over each boundary element. The first assumption is sound for the MAX IV storage rings, which both hold a beamsize on the scale of a few hundred μm^2 and the beampipe has a diameter on the cm-scale. Then, knowing the geometry of the beam pipe and the BPM electrodes, a numerical linearization can be made. The code for this linearization, along with the plotting of all the data can be seen in Appendix C, which was done manually for this project, utilizing the theory presented in [2].

2.2 Data Collection

In order to get as even a kick as possible to the beam, as few bunches as possible were injected into the ring to be measured in. This was done since the strength of the kick to the beam has a half-sinusoidal shape with a width of twice the revolution time, meaning it will kick any off-crest bunches twice. Thus, few bunches were injected and the timing of the kicker and pinger magnets were altered until a single strong kick was performed, meaning all bunches were approximately on crest with the kick. Once injection was finished and initial correction to the orbit had been done, the BPMs were set to the correct settings for collecting turn-by-turn data of the beam. Once these settings were in place, kicks were applied to the beam. After each kick, 2000 turns of positional data was collected from each BPM in the ring, 198 for R3 and 35 for R1. The raw turn-by-turn data of R3 resulting from a kick of 2310 V in the horizontal plane and a 480 V kick in the vertical plane can be seen below in Figure 2. In the figure we see some BPMs have an offset applied to the measured position. This may be significant in normal operation of the machine, but for this project data with this kind of offset was ignored.

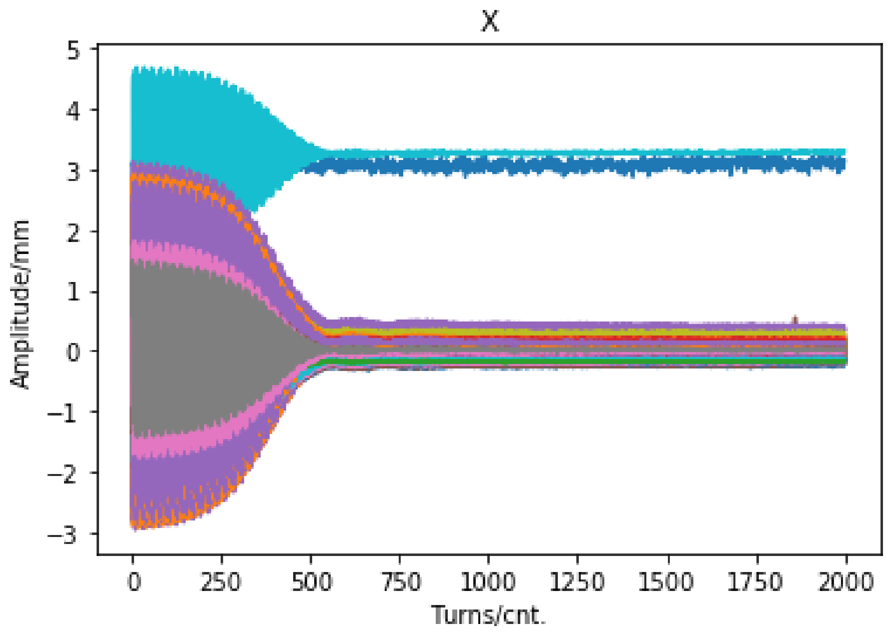

Figure 2: Raw turn-by-turn data collected by all BPMs in R3. The different colored plots signify the 198 different BPMs.

As kicks of increasing amplitude are applied to the beam, it will inevitably begin to lose current as the disturbance to the trajectory is too much for the machine to maintain. Therefore, the limits for kick voltages were set beforehand by kicking the beam with increasing voltages until it began to lose current. The limit was set wherever the beam lost 0.2 mA/kick and the limits for pure horizontal and vertical kicks were found at $V_x = 3900\text{V}$ and $V_y = 1400\text{V}$ for R3, and $V_x = 6300\text{V}$ and $V_y = 6900\text{V}$ for R1. However, when collecting data, kicks were applied in both planes simultaneously, and as such smaller limits than these were utilized.

In Figure 2 we can see a sudden decrease of amplitude around turn 300. This does not reflect the oscillating movement of the beam suddenly stopping, but is rather an artifact caused by the BPMs measuring the position of the center of charge, not the position of individual particles. The particles of the beam will diverge from the nominal momentum with varying Δp, and as can be seen in Equation (2), with non-zero chromaticity, this will lead to a variation in tune between the particles. This causes the particles to end up in different phase until at last the charges negate each other and the BPMs see a centralized beam. This process is known as *decoherence* and limits the resolution of the final analysis, the number of turns that can be used [3]. The code used for all the data collection can be seen in Appendix A.

In total, three data sets were collected for this project, listed below in Table 1. One data set of each ring in delivery settings, although the R3 data set was far smaller due to limited beam time availability of the ring. Beyond this a data set with R3 in close to (0 0) chromaticity was collected to investigate the effect on the decoherence. Setting the machine to this chromaticity meant altering the non-linear optics to some degree, affecting both the initial tunes as well as the beam's reaction to optical resonances.

Table 1: The three data sets and their parameters.

Data Set	Max. Pinger Voltages, X and Y (V)	dV (V)	# of Data Points
R3 Delivery	3200 and 1400	100	485
R3 (0 0) Chromaticity	3000 and 990	30	3398
R1 Delivery	5000 and 5000	50	10100

2.3 Data Analysis

After turn-by-turn data had been collected, some different forms of data analysis were performed to construct the desired frequency maps. The data analysis were in large parts identical for all three data-sets. Firstly, the points with the maximum kick amplitude was extracted for each BPM. This data was then linearized, as outlined in section 2.1.2, changing the more extreme points to an extrapolated "real" point. For the R3 delivery data set BPM nr. 20 was chosen for the frequency analysis, nr. 1 for the chromaticity (0 0) set and nr. 7 for the R1 set. These were chosen mostly due to their minimal need for linearization.

The amplitude oscillates between turns, as can be vaguely seen in Figure 2 and more clearly in Figure 3 below were the first 200 turns of the raw data are highlighted. It is from a frequency analysis of these oscillations the tune of the beam can be extracted. The frequency analysis was done using a NAFF algorithm. The algorithm is similar to a classic fast Fourier transform, but superior in precision, although with a longer computation time. The higher precision was necessary as the decoherence issues limited the number of turns available for the analysis.

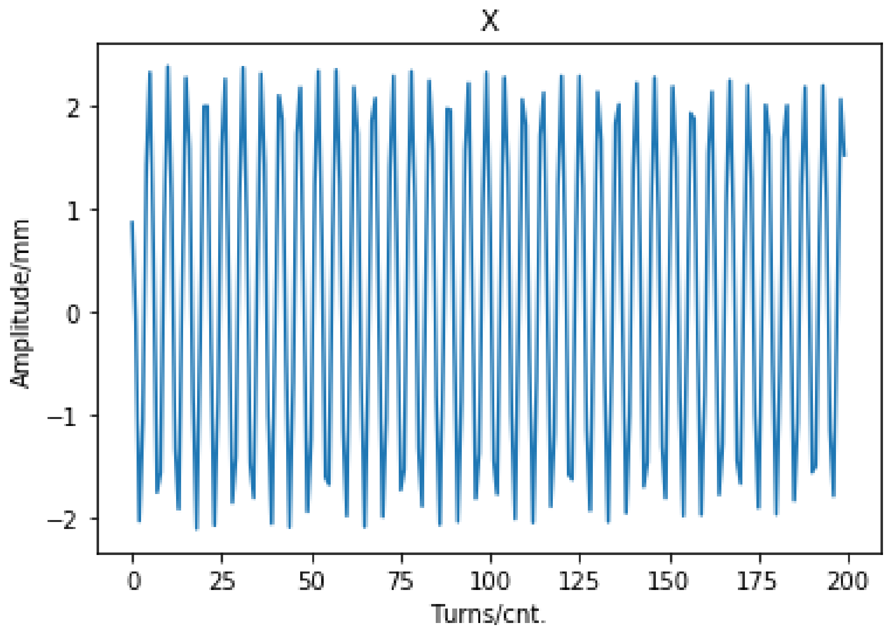

Figure 3: The positional data of the first 200 turns at BPM 1.

For each data set three iterations of analysis was done. For the R1 and R3 delivery sets the first 200 turns were analyzed, then the first 100 and then the following 100. For the R3 delivery data set, another analysis was made using only 100 turns for the initial analysis. The same process was done for the (0 0) chromaticity set, although 300 turns were used instead. The two data points, in the vertical and horizontal planes, from the total turns were used to present the data in tune space, Q_x plotted against Q_y. The four data points from the segmented versions of the analysis were then used to calculate the *diffusion parameter D*.

$$D = log(\sqrt{(Q_{x1} - Q_{x2})^2 + (Q_{y1} - Q_{y2})^2}) \tag{4}$$

Wherein Q_{x1} and Q_{x2} are the segmented frequencies extracted from the horizontal turn-by-turn data and Q_{y1} and Q_{y2} from the vertical. This parameter was then used as a color-code for the data maps in the results. It shows how fast the tune changes between turns. If the tune is changing very rapidly it is represented by a higher value of D.

Beyond this, the amplitude of each full frequency analysis was also recorded. This was squared and plotted against the BPM number as this should be of similar shape to the beta function, according to Equation (1). The code for all frequency analysis made can be seen in Appendix B.

3 Results

Below the three different data sets are displayed and briefly analyzed. Each data set is presented in positional space as well as tune space.

3.1 R3

Below in Figure 4 the positional map for the R3 delivery data set can be seen, color-mapped according to the diffusion parameter. The data presented is from BPM 20 in the ring and this plot presents the maximum amplitude of each kick, thus this map consists of 485 points. The few outliers in the bottom left of the diagram is caused by an error in the code where data collection did not wait for the kicker magnet to reset completely to 0 V before kicking again. This results in all later horizontal kicks having some residual voltage left in the magnet and a small kick always being present. This minor error is present in this and the R1 data set, but was reconciled for the (0 0) chromaticity data collection. We can see the diffusion increasing at the data points in the upper left of the map. This suggests the beam losing stability, but appears not to be directly connected to the amplitude of the kick, as the kicks in both planes in the upper right shows less diffusion. This is further explained by the tune map in the next figure.

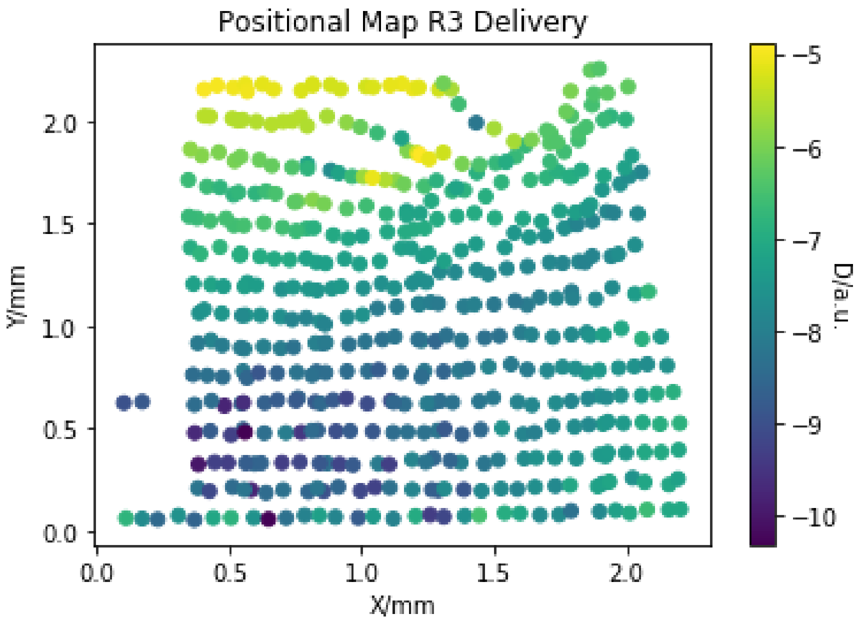

Figure 4: The measured positional disturbance of the beam in R3 with delivery settings at BPM 20, color-graded according to the diffusion parameter

Figure 5 displays the same data as Figure 4, but in tune space rather than positional. We also see the resonance lines within the figure. The fifth order resonance line can be seen on the right and two coupled resonances can be seen, one of the fifth order above and one of the fourth order below. We can see what may be an effect of the fifth order resonance, as the data points near it increase in diffusion dramatically. We also see some points becoming "stuck" before the fourth order coupled resonance, which may also be an effect of nearing this stronger resonance, discussed further with Figure 6 below. The original working point, the point in tune space the beam was originally located in, was at $Q_x = 0.1878$ and $Q_y = 0.2755$.

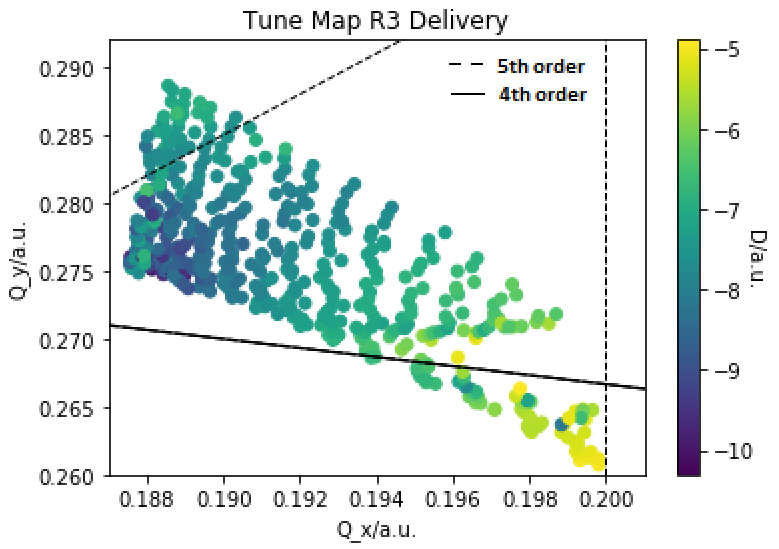

Figure 5: The measured frequencies of the R3 ring with delivery settings, displayed in tune-space and color-graded according to the diffusion parameter. Also displayed with fourth and fifth order resonance lines. The original working point was located at $Q_x = 0.1878$ and $Q_y = 0.2755$.

Below in Figure 6 the R3 delivery data set can be seen in tune space again, although this time, only the first 100 turns have been used for the frequency analysis. The same diffusion parameter is used. We can see here in this "earlier" measurement of the tunes that they appear much closer to the resonance lines than can be seen in Figure 5. This may suggest that the tunes do not get "stuck" before the resonance, but rather dampen away from it. We can see how the diffusion increases along the resonances, in line with the theory.

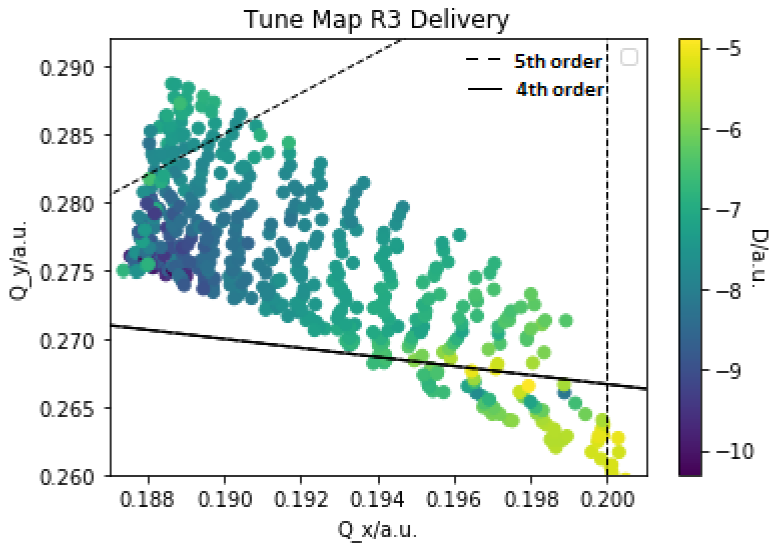

Figure 6: The measured frequencies of the R3 ring with delivery settings using a fewer amount of turns for the analysis. Displayed in tune-space and color-graded according to the diffusion parameter. Also displayed with fourth and fifth order resonance lines. The original working point was located at $Q_x = 0.1878$ and $Q_y = 0.2755$.

Figure 7 displays the positional map of the larger (0 0) chromaticity data set. As may be expected due to the change of the optical setup we see an increase to the sensitivity of the beam, causing it to be lost more easily and thus decreasing the possible amplitude at which data could be collected. Because of this, many of the areas of interest in Figures 4 and 5 above are not visible here, such as the higher diffusion when nearing the fifth order resonance, in the upper left of Figure 4. Most of this data appears quite homogeneous in respect to the diffusion. The suddenly increased diffusion in the lower left is most likely an artifact of running the frequency analysis without a kick in both planes. One can just barely see a line of lower diffusion on the right side of the figure.

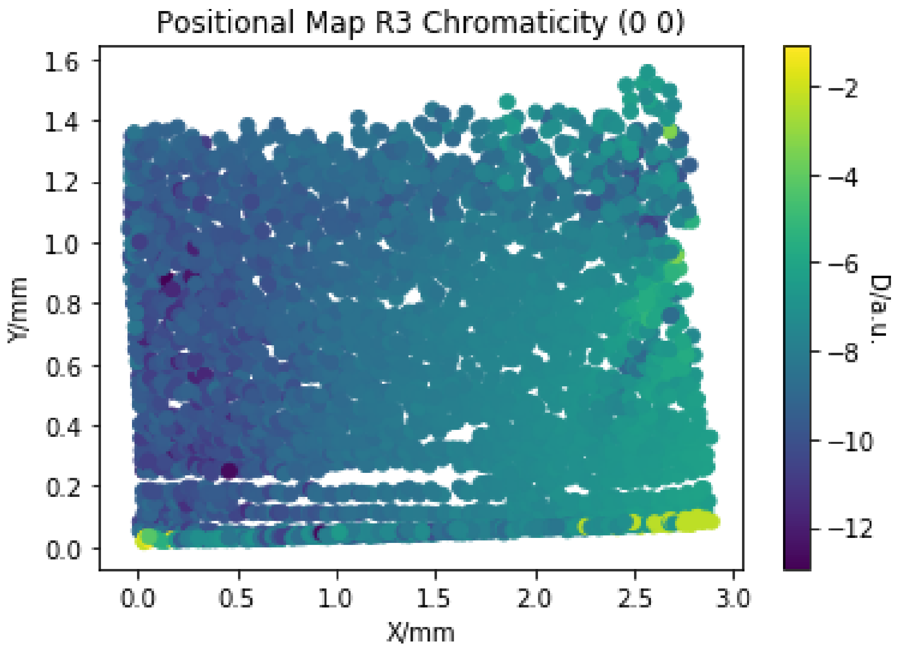

Figure 7: The measured positional disturbance of the beam in R3 with near (0 0) chromaticity settings at BPM 1, color-graded according to the diffusion parameter

The tune map of the (0 0) chromaticity data set can be seen in Figure 8. Here, some effect of altering the optics can be seen more clearly. The starting tunes have been altered such that the fourth order coupled resonance is never approached and the fifth order resonance is crossed completely. We see the fifth order resonance has little to no effect on the diffusion and that the fifth order coupled resonance can be seen to ever so slightly decrease the diffusion where it crosses the data. The original working point was located at $Q_x = 0.1942$ and $Q_y = 0.2791$.

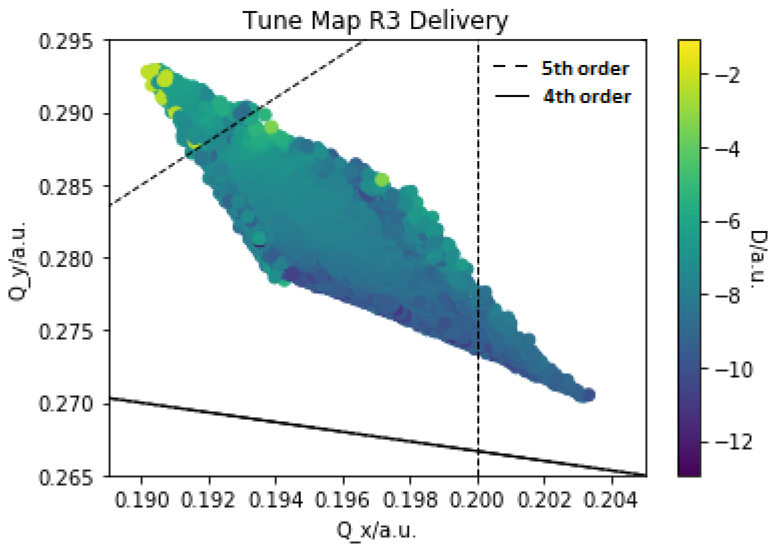

Figure 8: The measured frequencies of the R3 ring with chromaticity near (0 0) settings, displayed in tune-space and color-graded according to the diffusion parameter. Also displayed with fourth and fifth order resonance lines. Data from BPM 1. The original working point was located at $Q_x = 0.1942$ and $Q_y = 0.2791$.

Figures 9 and 10 show the measured amplitudes squared in the horizontal and vertical planes which, according to Equation (1), should match the shape of the beta function. The shape of these plots suggests the data is sound as they match the general shape of the theoretical design beta function, plotted along with the values for each plane in the figures. Displayed is only a single achromat, achromat 4 to be specific, although as mentioned the optics should be symmetrical throughout the achromats. It should be noted that the beta function is usually presented with the unit meters, the right vertical axis which displays the theoretical values can be seen to follow this, while the left vertical axis presents the measured amplitudes squared.

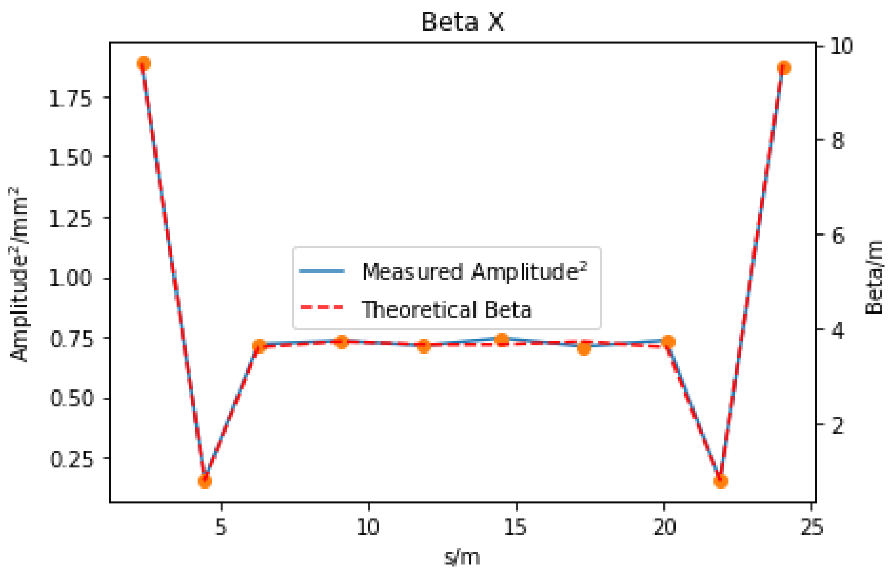

Figure 9: The squared amplitudes of the measured frequencies in achromat 4 in the horizontal plane along with the design beta function on the right-hand vertical axis, plotted against the longitudinal position.

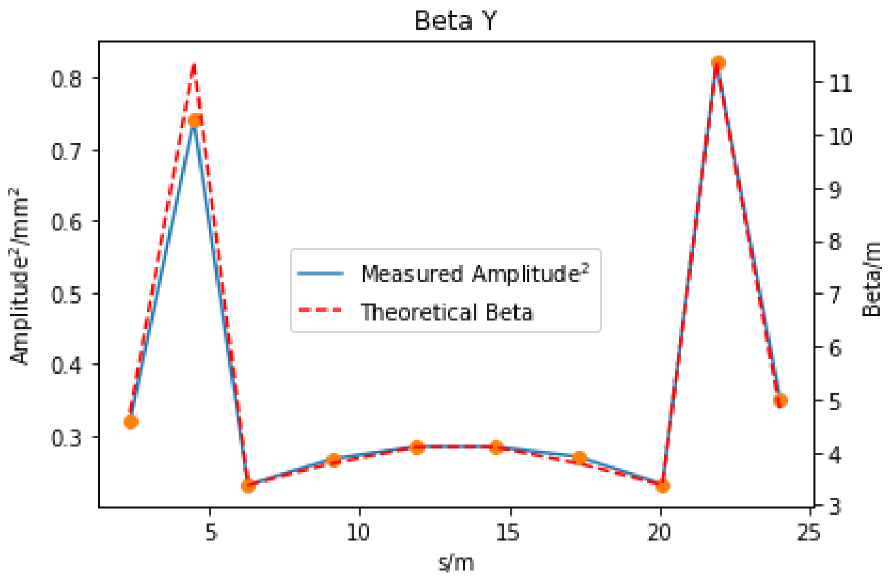

Figure 10: The squared amplitudes of the measured frequencies in achromat 4 in the vertical plane along with the design beta function on the right-hand vertical axis, plotted against the longitudinal position.

3.2 R1

Below in Figure 11, we can see the positional map for the R1 data set. It should be noted that the amplitude of these kicks are far higher than those for the R3 data sets. Instantly obvious is the strong streak of high diffusion running along the right side of the data. This is most likely an artifact caused by the coupling of the frequencies at high kicks along one plane, and low kicks along the other. Beyond this we also see an interesting pattern to the diffusion running throughout the data, as it appears to oscillate between low and high diffusion as the kicks increase in amplitude. The cause for this pattern is hard to determine. It is possible that it is merely an artifact of the frequency analysis, similar to the high diffusion area, although the complexity of the pattern could suggest it is a physical effect within the machine.

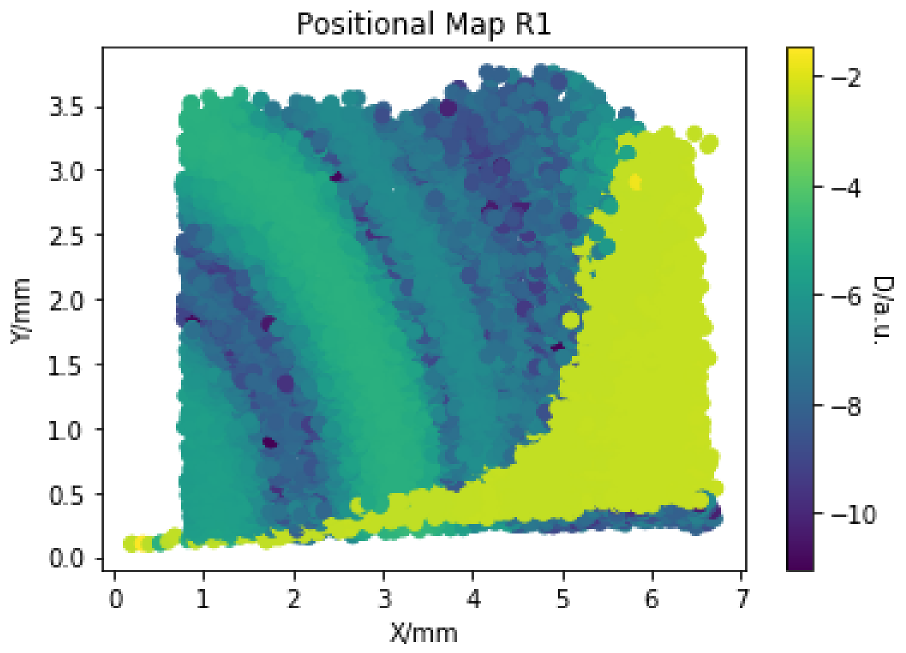

Figure 11: The measured positional disturbance of the beam in R1 with delivery settings at BPM 7, color-graded according to the diffusion parameter.

Figure 12 displays the data set of Figure 11 in tune space, along with two coupled resonance lines, both of the fifth order. We see the tunes cross one of these coupled resonances, which has little to no effect on the diffusion of the beam. We once again see the artifact in the part of the data with lower tunes. We also see the oscillating pattern, here appearing almost solely dependent on the horizontal tune. It should be noted that this data was also mapped along with much higher orders of resonance and no crossings following the oscillating pattern could be found. The original working point was at $Q_x = 0.2296$ and $Q_y = 0.1328$.

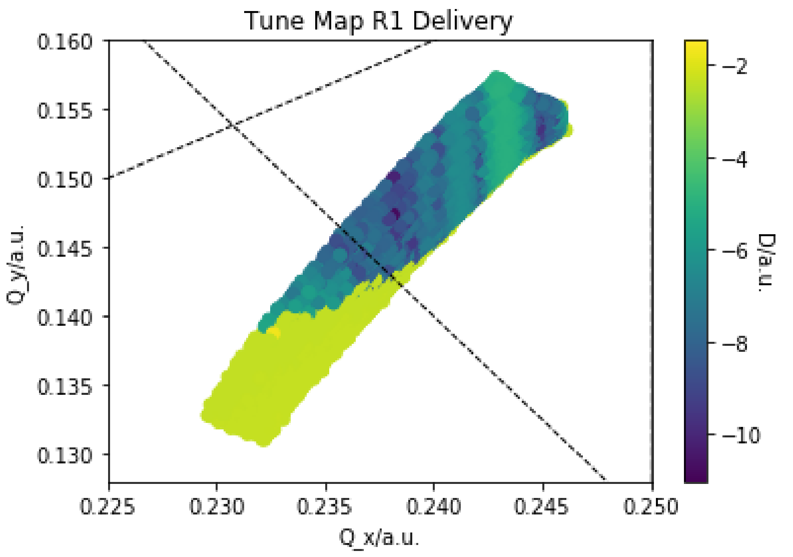

Figure 12: The measured frequencies of the R1 ring with delivery settings, displayed in tune-space and color-graded according to the diffusion parameter. Plotted along with two coupled resonance lines of the fifth order. The original working point was located at $Q_x = 0.2296$ and $Q_y = 0.1328$.

4 Conclusion

Frequency maps were constructed for both rings under varied conditions, and the response of the beam to certain tunes and resonances could be read from the results. This fulfills the basic goal of the work.

Furthermore, in R3, the connection of these responses to the set chromaticity of the machine can be gleaned somewhat by comparing the results of the delivery and (0 0) chromaticity data. We see, for example, that the instability caused by nearing or crossing the fifth order resonance line appears highly dependent on the kind optical changes made when altering the chromaticity. The results of this project also maps the amplitude dependent tune shift of the rings as a comparison between the positional and tune-space maps gives the tunes response to a oscillation of a certain amplitude.

Further analysis of the data could be made. For example, despite the achromat structure of the rings limiting the amount of unique data originating from the BPMs, this report only displays data from one BPM from each data set. More could be analyzed to see the effects throughout an achromat. Altering the chromaticity is a start, but the process of constructing the frequency maps could be repeated under very different optical setups to find how the different optics affects the tunes and the beam's response to optical and coupled resonances.

The project was just launched and needs improvements and developments when similar studies are to be made in the future. No stronger optical resonance was ever crossed during data collection. It could be interesting to see exactly how the beam reacts when nearing a resonance it will not be able to cross without beamloss. This could be achieved by changing the optics of the ring in order to move the initial tunes closer to one of the resonances, then using the amplitude dependent tune shift to pass over it, if possible. Furthermore, data collection was limited to the amplitudes wherein no major beamloss was present. This simplified collection of data a lot, but it could certainly be interesting to investigate the more extreme conditions for the beam. Looking at the R3 delivery data, its very possible that the beamlosses at high horizontal kicks were a result of nearing the fifth order optical resonance and if more beamloss was allowed, the interaction with of the beam with this resonance could be more closely studied. The oscillating pattern in the R1 data could also be studied further, perhaps performing the same analysis with some slightly different optical setup. Comparing such a data set with the one presented above, one could see how the pattern is effected by different changes to the optics of the ring.

Despite these possible improvements, the project in a whole fulfilled the basic goal of producing frequency maps for both rings, as well as observing some additional results. The tools and scripts developed as part of the project also hold more general use and could be utilized for other projects in the future. The diagnostic information gleaned in the results could be useful both in further developmental work at the facility as well as during delivery to scientific beamlines.

References

[1] Wille K, *The Physics of Particle Accelerators*, 2000, Oxford University Press

[2] Stella A, *Analysis of the DAϕNE Beam Position Monitor with a Boundary Element Method*, 1997, INFN-LNF, Accelerator Division

[3] Papaphilippou Y, Farvacque L, Plouviez E, Revol J-L, Ropert A, *Experimental Frequency Maps for the ESRF Storage Ring*, 2004, ESRF, Grenoble, France.

[4] Görgen P, *Tune Diagram in Python*, 28 November 2013, https://pgoergen.de/2013/11/tune-diagram-in-python/

[5] Gelfand NM, *Amplitude Dependence of the Tune Shift*, Conf.Proc. C870316 (1987) 1014.

[6] Tavares PF, Al-Dmour E, Andersson Å, Cullinan F, Jensen BN, Olsson D et al. *Commissioning and first-year operational results of the MAX IV 3 GeV ring*, Journal of Synchrotron Radiation. 2018;25(5):1291-1316.

[7] *MAX IV Detailed Design Report*, MAX IV Laboratory, Lund, Sweden, Aug. 2010.

Appendix A: Data Collector

```
import PyTango as PT
from PyTango import DeviceProxy as DP
import time
import matplotlib.pyplot as plt
import numpy
numpy.set_printoptions(threshold=numpy.nan)

print('Running')

''' DEVICES '''
#R3
db = PT.Database()
devR3 = db.get_device_exported_for_class('LiberaDeviceClass')[:]
bpms_allr31 = filter(lambda v: v.startswith('R3') and 'BPM' in v,devR3)
bpms_allr3=[]
bpms_allr3str=[]
for i in range(len(bpms_allr31)):
    bpms_allr3.append(DP(bpms_allr31[i]))
pingr3=[DP('R3-A110110CAB30/MAG/PSBA-01'), DP('R3-A111110CAB30/MAG/PSBB-01')]
timEVGr3=DP('R3-A101911CAB03/TIM/EVG-01')
timEVRr3H=DP('R3-A110111CAB04/TIM/EVR-01')
timEVRr3V=DP('R3-A111011CAB04/TIM/EVR-01')
DCCTr3=DP('R3-319S2/DIA/DCCT-01')
for i in bpms_allr3:
    bpms_allr3str.append(str(i))


#R1
bpms_allr11=[]
bpms_allr1=[]
bpms_allr1str=[]
for j in range(1,10):
    for i in range(1,4):
        bpms_allr11.append('R1-10'+str(j)+'/DIA/BPM-0'+str(i))
for j in range(10,13):
    for i in range(1,4):
        bpms_allr11.append('R1-1'+str(j)+'/DIA/BPM-0'+str(i))
for i in range(len(bpms_allr11)):
    bpms_allr1.append(DP(bpms_allr11[i]))
pingr1=[DP('R1-D110210CAB31/MAG/PSBC-01'),DP('R1-D111210CAB32/MAG/PSBD-01')]
timEVGr1=DP('R1-D110210CAB04/TIM/EVG-01')
timEVRr1=DP('R1-D110210CAB04/TIM/EVR-01')
DCCTr1=DP('R1-101S/DIA/DCCT-01')
for i in bpms_allr1:
    bpms_allr1str.append(str(i))
```

```
timSEL=DP('G/TIM/SEL')


''' Setting the machine up for measurements '''
def MachineSet(ring, buffersize):
    if ring=='r3':
        timSEL.Injection='SPF'
        timEVGr3.Inject_Stop()
        for bpm in bpms_allr3:
            bpm.AGCEnabled=False
            bpm.ConditionSwitching=False
            bpm.TDEnabled=True
            bpm.TDBufferSize=buffersize
        for i in [0,1]:
            pingr3[i].EnableTrigger()

    if ring == 'r1':
        for bpm in bpms_allr1:
            bpm.AGCEnabled=False
            bpm.ConditionSwitching=False
            bpm.TDEnabled=True
            bpm.TDBufferSize=buffersize
        timSEL.Injection='SPF'
        timEVGr1.Inject_Stop()
        timEVRr1.Output00Delay=18501
        for i in [0,1]:
            pingr1[i].EnableTrigger()

''' Reseting changes to the machine '''
def MachineReset(ring):
    if ring=='r3':
        for bpm in bpms_allr3:
            bpm.AGCEnabled=True
            bpm.ConditionSwitching=True
            bpm.TDEnabled=False
        timEVGr3.Inject_Frequency()
        timEVRr3H.Output00Delay=18241
        timEVRr3V.Output00Delay=17900
        for i in [0,1]:
            pingr3[i].DisableTrigger()

    if ring=='r1':
        for bpm in bpms_allr1:
            bpm.AGCEnabled=True
            bpm.ConditionSwitching=True
            bpm.TDEnabled=False
        timEVRr1.Output00Delay=18506
        timEVRr1.Output01Delay=18241
        timEVGr1.Inject_Frequency()
        for i in [0,1]:
            pingr1[i].DisableTrigger()

''' Collect TbT data '''
def TbT(ring, ping, pingvoltx, pingvolty):
    absx=[]
    absy=[]
    X=[]
    Y=[]
    Sum=[]
    if ping==1:
        if ring=='r3':
            pingr3[0].voltageSetPoint = pingvoltx
            pingr3[1].voltageSetPoint = pingvolty
            time.sleep(2)
            timEVGr3.Inject_Single()
            time.sleep(0.1)
            for bpm in bpms_allr3:
                X.append(bpm.XPosTD)
                Y.append(bpm.YPosTD)
                Sum.append(bpm.SumTD)

        if ring=='r1':
            pingr1[0].voltageSetPoint = pingvoltx
            pingr1[1].voltageSetPoint = pingvolty
            time.sleep(2)
            timEVGr1.Inject_Single()
            time.sleep(0.1)
            for bpm in bpms_allr1:
                X.append(bpm.XPosTD)
                Y.append(bpm.YPosTD)
                Sum.append(bpm.SumTD)
    else:
        if ring=='r3':
            for bpm in bpms_allr3:
                X.append(bpm.XPosTD)
                Y.append(bpm.YPosTD)
                Sum.append(bpm.SumTD)
        if ring=='r1':
            for bpm in bpms_allr1:
                X.append(bpm.XPosTD)
                Y.append(bpm.YPosTD)
                Sum.append(bpm.SumTD)
#    X=BPM.XPosTD
 #   Y=BPM.YPosTD
 #   Sum=BPM.SumTD
    t=range(len(X[0]))
    for i in range(len(X)):
        for j in range(len(X[0])):
            absx.append(abs(X[i][j]))
            absy.append(abs(Y[i][j]))
    return [t,X,Y,Sum,absx,absy]

''' Take maximums of multiple runs '''
def Stepper(ring, pingmaxx, pingmaxy, stepof):
    maxx=[]
    maxy=[]
    pingx=0
    pingy=400
    pingxl=[]
    pingyl=[]
    i=1
    while i==1:
        pingx+=stepof
        if ring=='r1':
            Curr1=DCCTr1.InstantaneousCurrent
        else:
            Curr1=DCCTr3.InstantaneousCurrent
        m=TbT(ring, 1, pingx, pingy)
        time.sleep(1)
        if ring=='r1':
            Curr2=DCCTr1.InstantaneousCurrent
        else:
            Curr2=DCCTr3.InstantaneousCurrent
    #    maxx.append(max(m[4]))
    #    maxy.append(max(m[5]))
        pingxl.append(pingx)
        pingyl.append(pingy)
        t1=time.localtime()
        numpy.savez('SAVE'+ring+' '+str(t1[0])+'-'+str(t1[1])+'-'+str(t1[2])
        +' '+str(t1[3])+':'+str(t1[4])+':'+str(t1[5])+', PINGS:
        pingx='+str(pingx)+','+' pingy='+str(pingy),bpms_allr3=bpms_allr3str,
        bpms_allr1=bpms_allr1str,pingxl=pingxl,pingyl=pingyl,
        t=m[0],X=m[1],Y=m[2],Sum=m[3],absx=m[4],absy=m[5])
        print('Saved data for PINGS: pingx='+str(pingx)+','+' pingy='+str(pingy))
        if Curr2/Curr1 < 0.9 or pingx==pingmaxx:
            pingy+=stepof
            pingx=0
        if pingy==pingmaxy+stepof:
            i=0

''' PLOTTING! '''
#timSEL.Injection='SPF'
#m=TbT('r1', 1, 0, 0)
Stepper('r1', 6300, 400, 100)
```

```
'''
plt.figure(1)
for i in range(34):
    plt.plot(m[0], m[1][i])
plt.title('X')
plt.figure(2)
for i in range(34):
    plt.plot(m[0], m[2][i])
plt.title('Y')
plt.figure(3)
for i in range(34):
    plt.plot(m[1][i],m[2][i], '.')
plt.title('All data')
plt.figure(4)
#plt.plot(maxx,maxy, '.')
#plt.title('Max points')


t1=time.localtime()
numpy.savez('SAVE'+'r1'+' '+str(t1[0])+'-'+str(t1[1])+'-'+str(t1[2])+' '
+str(t1[3])+':'+str(t1[4])+':'+str(t1[5]),
bpms_allr3=bpms_allr3str,bpms_allr1=bpms_allr1str,
t=m[0],X=m[1],Y=m[2],Sum=m[3],absx=m[4],absy=m[5])
'''
print('Complete')
```

Appendix B: Frequency Analyzer

```
    import time
import matplotlib.pyplot as plt
import numpy
import os
numpy.set_printoptions(threshold=numpy.nan)
from PyNAFF import *
#Maxx=[]
#Maxy=[]
#Minx=[]
#Miny=[]
k=os.listdir(r'DATA')
dat=numpy.load(r'DATA')
#datY=numpy.delete(dat['Y'],3,0)

def maxpoints(directory):
  Maxx=[]
  Minx=[]
  Maxy=[]
  Miny=[]
  for j in range(198):
    maxx=[]
    minx=[]
    maxy=[]
    miny=[]
    for i in k:
        loader=os.path.join(r'DATA', i)
        dat=numpy.load(loader)
        datX=dat['X']
        datY=dat['Y']
        maxx.append(max(datX[j]))
        minx.append(min(datX[j]))
        maxy.append(max(datY[j]))
        miny.append(min(datY[j]))
    Maxx.append(maxx)
    Minx.append(minx)
    Maxy.append(maxy)
    Miny.append(miny)
    print('saved bpm')
  #plt.figure(1)
  #plt.plot(Minx+Maxx,Miny+Maxy, '.')
  #plt.title('Max points')

  saver=os.path.join(r'DATA', directory)
  numpy.savez(saver, Maxx=Maxx,Maxy=Maxy,Minx=Minx,Miny=Miny)
  return 'DONE'
```

```
xfreql=[]
yfreql=[]
xfreq1l=[]
yfreq1l=[]
xfreq2l=[]
yfreq2l=[]

diff=[]

def FreqAnalysis(bpm, directory):
  for i in k:
    dire=os.path.join(r'DATA', i)
    dat=numpy.load(dire)
#    datX=numpy.delete(dat['X'],3,0)
#   datY=numpy.delete(dat['Y'],3,0)
    xfreq1=PyNAFF.naff(dat['X'][bpm],100,3,0,False,window=1)
    yfreq1=PyNAFF.naff(dat['Y'][bpm],100,3,0,False,window=1)
    xfreq2=PyNAFF.naff(dat['X'][bpm],100,3,100,False,window=1)
    yfreq2=PyNAFF.naff(dat['Y'][bpm],100,3,100,False,window=1)
    xfreq=PyNAFF.naff(dat['X'][bpm],200,3,0,False,window=1)
    yfreq=PyNAFF.naff(dat['Y'][bpm],200,3,0,False,window=1)
    if len(xfreq)==0:
      continue
    diff.append(log(((xfreq1[0][1]-xfreq2[0][1])**2
    +(yfreq1[0][1]-yfreq2[0][1])**2)**(1/2)))
    xfreql.append(xfreq)
    yfreql.append(yfreq)
#    xfreq1l.append(xfreq)
#    yfreq1l.append(yfreq)
#    xfreq2l.append(xfreq)
#    yfreq2l.append(yfreq)
#    if yfreq[0][1]>0.20 and len(yfreq)>1:
 #     yfreql.append(yfreq[1][1])
  #    yfreqa.append(yfreq[1][2])
   # else:
   #   yfreql.append(yfreq[0][1])
   #  yfreqa.append(yfreq[0][2])

  direc=os.path.join(r'DATA', directory)
  numpy.savez(direc, xfreql=xfreql, yfreql=yfreql, diff=diff)
#  plt.figure(1)
#  plt.scatter(xfreql,yfreql,c=diff)

def Beta(kick, dirname):
  xfreqa=[]
  yfreqa=[]
  for j in range(198):
    dire=os.path.join(r'DATA', k[kick])
    dat=numpy.load(dire)
    datX=dat['X']
    datY=dat['Y']
    xfreq=PyNAFF.naff(datX[j],250,3,0,False,window=1)
    yfreq=PyNAFF.naff(datY[j],250,3,0,False,window=1)
    if len(xfreq)==0:
      continue
#    xfreql.append(xfreq[0][1])
#    yfreql.append(yfreq[0][1])
    #xfreqa.append(abs(xfreq[0][2]))
    #yfreqa.append(abs(yfreq[0][2]))
    fftx=numpy.fft.fft(datX[j])
    ffty=numpy.fft.fft(datY[j])
    fftx1=numpy.delete(fftx, 0)
    ffty1=numpy.delete(ffty, 0)
    xfreqa.append(max(abs(fftx1)))
    yfreqa.append(max(abs(ffty1)))
  xfreqa.pop(0)
  xfreqa.pop(108)
  yfreqa.pop(0)
  yfreqa.pop(4)
  yfreqa.pop(107)
  direc=os.path.join(r'DATA', dirname)
  numpy.savez(direc, xfreql=xfreql, yfreql=yfreql,
  xfreqa=xfreqa, yfreqa=yfreqa)
  betax=[]
  betay=[]
  for i in range(len(xfreqa)):
    betax.append(xfreqa[i]**2/400)
  for i in range(len(yfreqa)):
    betay.append(yfreqa[i]**2/10)
  plt.figure(1)
  plt.plot(range(len(xfreqa)),betax)
  plt.title('X')
  plt.figure(2)
  plt.plot(range(len(yfreqa)),betay)
  plt.title('Y')

plt.figure(1)
for i in range(198): #r1 35 r3 198
    plt.plot(range(2000), dat['X'][i])
    #maxx.append(max(datX[i]))
    #minx.append(min(datX[i]))
plt.xlabel('Turns/cnt.')
plt.ylabel('Amplitude/nm')
plt.title('X')
plt.figure(2)
for i in range(198):
    plt.plot(range(2000), dat['Y'][i])
    #maxy.append(max(datY[i]))
    #miny.append(min(datY[i]))
plt.title('Y')
plt.figure(3)
for i in range(198):
    plt.plot(dat['X'][i],dat['Y'][i], '.')
plt.title('All data')
```

Appendix C: Plotter/Linearization

```
import time
import matplotlib.pyplot as plt
import numpy
import pylab
import os
numpy.set_printoptions(threshold=numpy.nan)
from PyNAFF import *
from fractions import gcd
import scipy.io

dat1=numpy.load(r'DATA')
dat=numpy.load(r'DATA')
datM=numpy.load(r'DATA')

#xfreql=dat1['xfreql']
#yfreql=dat1['yfreql']
xfreqt=[]
yfreqt=[]
xfreql1=[]
yfreql1=[]
difft=[]
meanr3X=-45446.057000000001
meanr3Y=31835.623
mean00X=84874.739000000001
mean00Y=-22624.964
meanr1X=538854.13800000004
meanr1Y=-139346.432

def curver():
  return plt.scatter(datM['Maxx'][20], datM['Maxy'][20], c=dat['diff'])


def updateOrder(lines,leftY,rightY,order,label):
    if (leftY in lines):
        if (rightY in lines[leftY]) and (lines[leftY][rightY]["order"]<order):
            lines[leftY][rightY]={"order":order, "label":label}
        else:
            lines[leftY][rightY]={"order":order, "label":label}
    else:
        lines[leftY]={}
        lines[leftY][rightY]={"order":order, "label":label}

def plotTuneDiagram(maxOrder,xlim=[0,1],ylim=[0,1],
tickOrders=False,tuneLineColor="black"):
    ylim=numpy.array(ylim)
```

```
xlim=numpy.array(xlim)
lines={}
vlines={}
for a in numpy.arange(-maxOrder,maxOrder,dtype="float"):
    for b in numpy.arange(-maxOrder,maxOrder,dtype="float"):
        for n in numpy.arange(-maxOrder,maxOrder,dtype="float"):
            order=abs(a)+abs(b)
            if order <= maxOrder:
                if b != 0:
                    leftY  = (n - a*xlim[0]) / b
                    rightY = (n - a*xlim[1]) / b
                    if (ylim[0] <= leftY and ylim[1] >= rightY)or\
                       (ylim[1] >= leftY and ylim[0] <= rightY):
                        divisor=gcd(abs(n),abs(b))
                        updateOrder(lines,leftY,rightY,order,
                        "$\\frac{%d}{%d}$" % (abs(n/divisor),abs(b/divisor)))
                elif b==0 and a!=0:
                    if xlim[0]<=(1.*n/a)<=xlim[1]:
                        divisor=gcd(abs(n),abs(a))
                        if (1.*n/a) in vlines:
                            if vlines[1.*n/a]["order"]>order:
                                vlines[1.*n/a]={"order":max(vlines[1.*n/a]
                                ["order"],order),"label":"$\\frac{%d}{%d}$" %
                                (n/divisor,a/divisor)}
                        else:
                            vlines[1.*n/a]={"order":order,"label":"$\\frac{%d}
                            {%d}$" % (abs(n/divisor),abs(a/divisor))}
y2ticks=[]
y2labels=[]
for leftY in lines:
    for rightY in lines[leftY]:
        pylab.plot(xlim,[leftY,rightY],linewidth=5./lines[leftY][rightY]["order"]
        ,color=tuneLineColor)
        if rightY==leftY:
            y2ticks.append(rightY)
            y2labels.append(lines[leftY][rightY]["label"])
x2ticks=[]
x2labels=[]
for fraction in vlines:
    pylab.axvline(fraction,linewidth=5./vlines'
    [fraction]["order"],color=tuneLineColor)
    x2ticks.append(fraction)
    x2labels.append(vlines[fraction]["label"])
if tickOrders:
    pylab.xticks(x2ticks,x2labels)
    pylab.yticks(y2ticks,y2labels)
pylab.xlim(xlim)
pylab.ylim(ylim)
```

```
def truemapper():
  for i in range(len(xfreql)):
    xfreql1.append(xfreql[i][0][1])
    yfreql1.append(yfreql[i][0][1])
    #if yfreql[i][0][1]>0.2 or yfreql[i][0][1]<0.25:
    #  continue
    xfreqt.append(xfreql[i][0][1])
    yfreqt.append(yfreql[i][0][1])
    #difft.append(dat['diff'][i])
  '''
      if len(xfreql[i])==3:
        xfreqt.append(xfreql[i][2][1])
      if len(xfreql[i])==2:
        xfreqt.append(xfreql[i][1][1])
      if len(xfreql[i])==1:
        xfreqt.append(xfreql[i][0][1])
      if len(yfreql[i])==3:
        yfreqt.append(yfreql[i][2][1])
      if len(yfreql[i])==2:
        yfreqt.append(yfreql[i][1][1])
      if len(yfreql[i])==1:
        yfreqt.append(yfreql[i][0][1])
  '''
  plt.figure(1)
  plt.scatter(xfreql1, yfreql1, c=dat['diff'])
  #plt.ylim(0.260,0.292)
  #plt.xlim(0.187,0.201)
  plt.title('Tune Map R1')
  plt.xlabel('Q_x')
  plt.ylabel('Q_y')
  plt.colorbar()

  plt.figure(2)
  plotTuneDiagram(5)
  plt.scatter(xfreqt, yfreqt, c=dat['diff'])
  plt.title('Tune Map R3 Delivery')
  plt.xlabel('Q_x/a.u.')
  plt.ylabel('Q_y/a.u.')
  cb=plt.colorbar()
  cb.set_label('D/a.u.', rotation=270)
  #plt.ylim(0.26,0.292)
  #plt.xlim(0.187,0.201)

def linearizationr1(bpm, flank):
  x=datM['Maxx'][bpm]
  y=datM['Maxy'][bpm]
  lp_central=scipy.io.loadmat(r'linearizationPolynomscentre.mat')
  lp_flanking=scipy.io.loadmat(r'linearizationPolynomsflanking.mat')

  k_x13 = 6.753561
  k_y13 = 18.959501

  k_x2 = 12.146387
  k_y2 = 12.595727

  xn=[]
  yn=[]
  X=[]
  Y=[]
  for i in range(len(x)):
    if flank==1:
      xn.append((x[i]-meanr1X)/k_x13*10**(-6))
      yn.append((y[i]-meanr1Y)/k_y13*10**(-6))
      X1=0
      Y1=0
      for j in range(136):
        X1+=lp_flanking['xp'][0][0][1][0][j]*xn[i]**lp_flanking['xp']
        [0][0][0][j][0]*yn[i]**lp_flanking['xp'][0][0][0][j][1]
        Y1+=lp_flanking['yp'][0][0][1][0][j]*xn[i]**lp_flanking['yp']
        [0][0][0][j][0]*yn[i]**lp_flanking['yp'][0][0][0][j][1]
      X.append(X1)
      Y.append(Y1)
    if flank==0:
      xn.append((x[i]-meanr1X)/k_x2*10**(-6))
      yn.append((y[i]-meanr1Y)/k_y2*10**(-6))
      X1=0
      Y1=0
      for j in range(136):
        X1+=lp_central['xp'][0][0][1][0][j]*xn[i]**lp_central['xp']
        [0][0][0][j][0]*yn[i]**lp_central['xp'][0][0][0][j][1]
        Y1+=lp_central['yp'][0][0][1][0][j]*xn[i]**lp_central['yp']
        [0][0][0][j][0]*yn[i]**lp_central['yp'][0][0][0][j][1]
      X.append(X1)
      Y.append(Y1)
  return plt.scatter(X, Y, c=dat['diff']), X, Y


def linearizationr3(bpm):
  x=datM['Maxx'][bpm]
  y=datM['Maxy'][bpm]

  lp=scipy.io.loadmat(r'linearizationPolynoms.mat')
  k = 0.8671522000*1e1

  xn=[]
  yn=[]
  X=[]
  Y=[]
  for i in range(len(x)):
    xn.append((x[i]-meanr3X)/k*10**(-6))
    yn.append((y[i]-meanr3Y)/k*10**(-6))
  for i in range(len(x)):
    X1=0
    Y1=0
    for j in range(136):
      X1+=lp['xp'][0][0][1][0][j]*xn[i]**lp['xp'][0][0][0][j][0]*yn
      [i]**lp['xp'][0][0][0][j][1]
      Y1+=lp['yp'][0][0][1][0][j]*xn[i]**lp['yp'][0][0][0][j][0]*yn
      [i]**lp['yp'][0][0][0][j][1]
    X.append(X1)
    Y.append(Y1)
    '''
    X.append(numpy.polynomial.polynomial.polyval2d(xn[i],yn[i],
    lp['xp'][0][0][1][0]))
    Y.append(numpy.polynomial.polynomial.polyval2d(xn[i],yn[i],
    lp['yp'][0][0][1][0]))

    #X.append(numpy.polyval(lp['xp'][0][0][1][0],xn[i]))#[xn[i],yn[
    i]]))
    #Y.append(numpy.polyval(lp['yp'][0][0][1][0],yn[i]))
    #[xn[i],yn[i]]))
  linearizationr1(7,0)
plt.xlabel('X/mm')
plt.ylabel('Y/mm')
plt.title('Positional Map R1')
cb=plt.colorbar()
cb.set_label('D/a.u.', rotation=270)
  '''
  return plt.scatter(X, Y, c=dat['diff'])
```