

## Imperfections and corrections

R. Tomás, X. Buffat, J. Coello, E. Fol and L. Malina  
CERN, CH 1211 Geneva 23, Switzerland

### Abstract

The measurement and correction of optics parameters has been a major concern since the advent of strong focusing synchrotron accelerators. A review of typical imperfections in accelerator optics together with measurement and correction algorithms is given with emphasis on numerical implementations. Python examples are shown using existing libraries when possible.

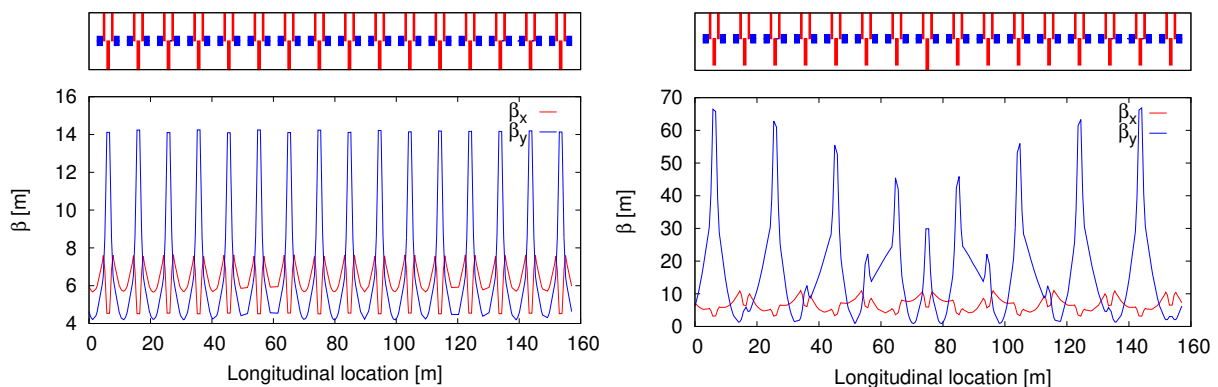
### 1 Introduction

Imperfections in accelerator lattices cause beam parameters to deviate from design. An illustration is shown in Fig. 1, where ideal and perturbed  $\beta$  functions are shown. The perturbation assumed is simply a 10% gradient error in the 8<sup>th</sup> defocusing quadrupole. This causes large relative deviations in  $\beta$  functions of up to 500% with respect to the design value. This is usually called  $\beta$ -beating and represented by  $\Delta\beta/\beta$ .

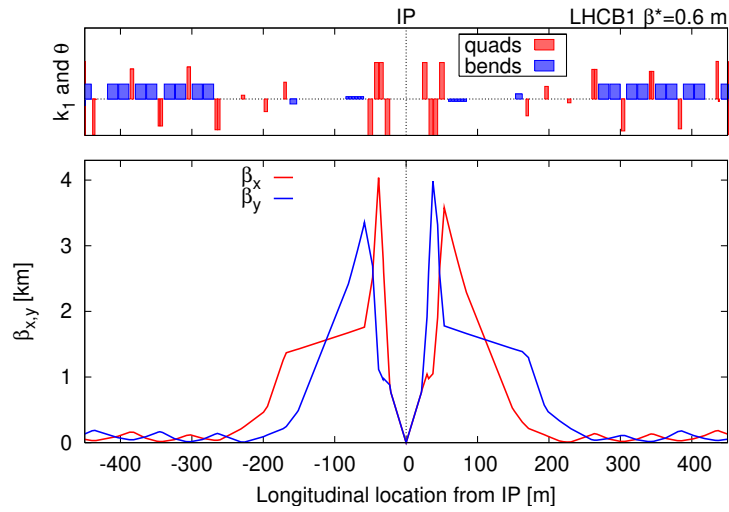
Perturbations from field imperfections and misalignments became a concern along with the conception of the strong focusing theory in 1957 [1]. However, the assumed approach was to specify design tolerances that would not impact machine performance. For example in [1] it is envisaged that with 1% rms gradient errors *any particular machine would be unlikely* to have more than 8% peak  $\beta$ -beating. At that time they did not foresee the great developments in optics that would push  $\beta$  functions to very large values, e.g., in the vicinity of collision points of collider accelerators. The LHC Interaction Region (IR) optics is shown in Fig. 2 as an illustration of optics designs reaching  $\beta$  functions of several km.

Modern accelerators have experienced  $\beta$ -beating values above 100% [2–4] in the initial commissioning phases. Figure 3 shows the initial  $\beta$ -beating measured in the LHC commissioning in 2016 with a peak value of 120%. The optics errors need to be corrected below specified tolerances for safe and efficient operation. The development of the optics measurement and correction techniques is illustrated by the evolution of the  $\beta$ -beating over time for many circular accelerators, see Fig. 4.

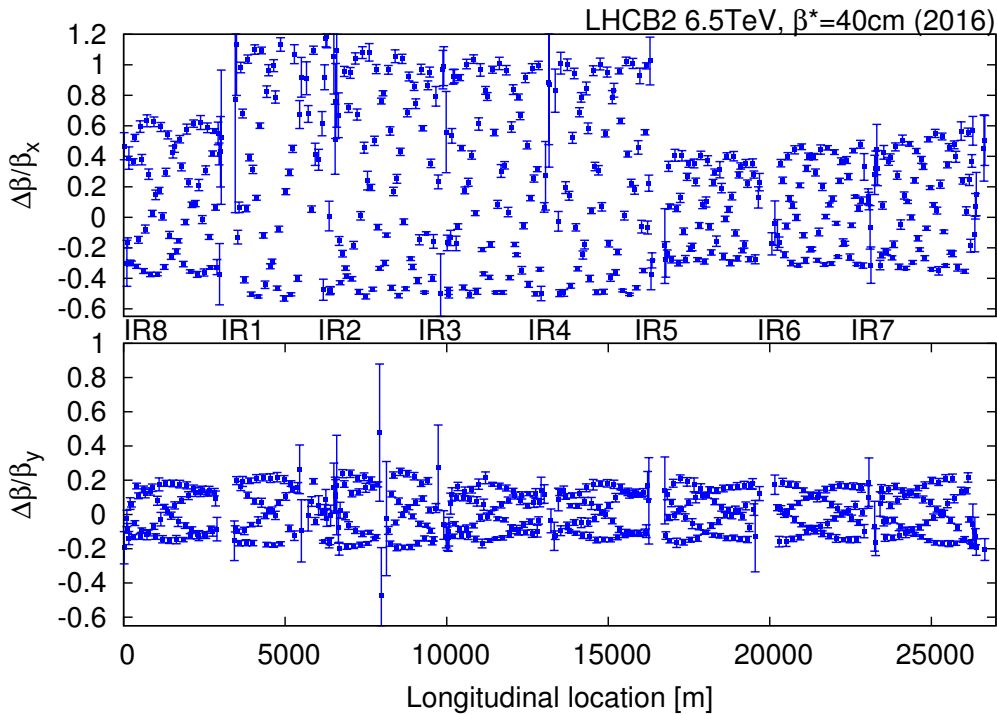
The techniques to measure and correct optics are described in the following with special emphasis on analysis algorithms and computing aspects. Section 2 gives the requirements to run the code examples below. Section 3 describes the most important imperfections present in accelerator lattices.



**Fig. 1:** Design  $\beta$  functions of the CERN Proton Synchrotron Booster featuring a triplet lattice (left) and the same lattice with a 10% gradient perturbation in the 8<sup>th</sup> defocusing quadrupole (right).



**Fig. 2:** Optics functions in the LHC IR for a  $\beta$  function at the interaction point of 60 cm.

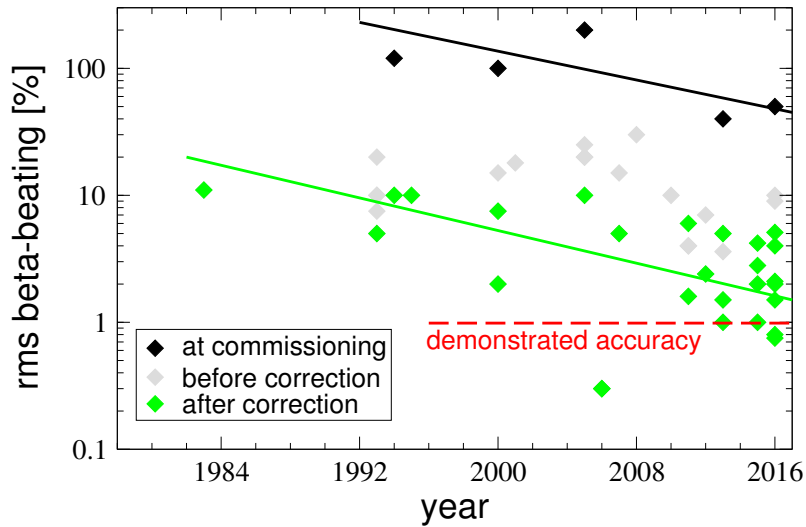


**Fig. 3:**  $\beta$ -beating measured in the LHC commissioning in 2016 with a  $\beta$  function at the interaction point of 40 cm.

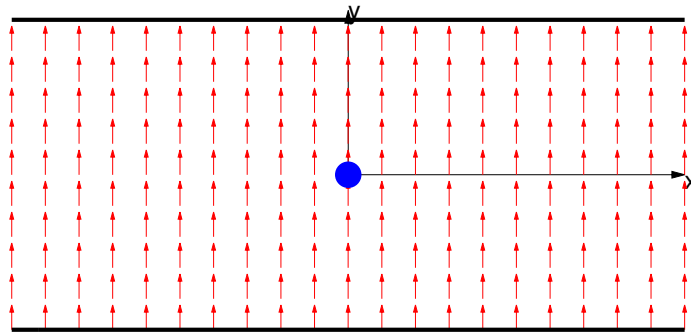
Section 4 describes the key particle dynamics used in optics measurements. Section 5 reports on the most used measurement techniques and data analysis techniques. Section 6 is an interlude devoted to the Farey sequences and how they can be used to describe the resonance diagram. Section 7 reports on optics correction techniques.

## 2 Requirements for code examples

Code examples below require [Python](#). The freely available and open source operative system [Ubuntu](#) has Python by default. The required plotting and numerical libraries can be installed with, e.g., the following shell command:



**Fig. 4:** Measured or inferred  $\beta$ -beating versus time for many circular accelerators as found in the bibliography of this paper. Three stages are differentiated: (i) during commissioning when magnet powering mistakes are expected, (ii) after fixing these mistakes but before careful optics corrections and (iii) after optics corrections. Taken from [5].



**Fig. 5:** The dipole magnetic field. The beam is represented in the center with a blue dot traveling perpendicular to the field.

---

```

1 python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy
  ↳ nose scikit-learn
2

```

---

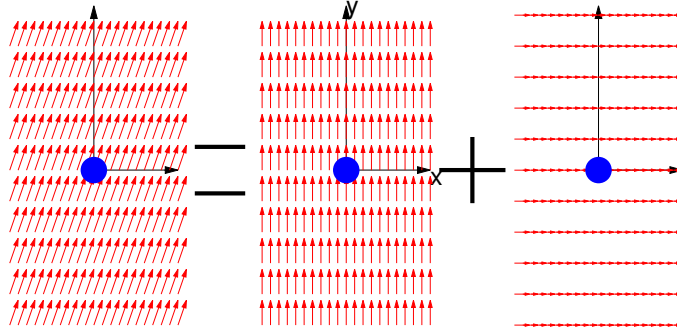
Alternatively it is also possible to install [Anaconda](#) which is a very complete Python free distribution with the required scientific packages.

### 3 Accelerator elements and their imperfections

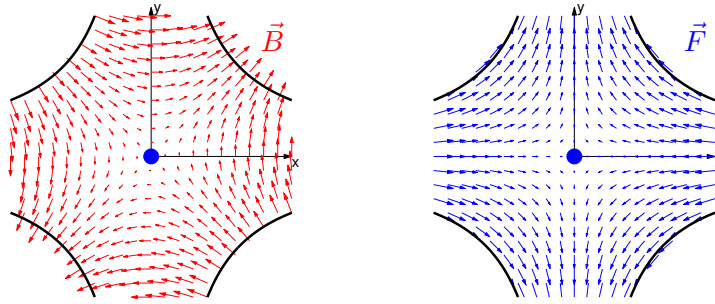
#### 3.1 Dipole

The simplest magnetic element in an accelerator is the dipole, which provides an homogeneous field as shown in Fig. 5.

The dipole features two main imperfections: a strength error and a tilt of the field around the beam axis. The tilt error is illustrated in Fig. 6 and can be interpreted as another dipole with a field orthogonal to the ideal dipole.



**Fig. 6:** A tilted dipolar magnetic field is seen as the sum of two orthogonal magnetic fields.



**Fig. 7:** Quadrupolar magnetic field (left) and force imparted to a particle traveling perpendicular to the figure (right).

Therefore dipole errors are seen as unwanted angular deflections in both transverse planes that distort the reference trajectories into new closed orbits. Assuming  $\theta_i$  to be unwanted angular deflections the closed orbit is given by

$$CO(s) = \frac{\sqrt{\beta(s)}}{2 \sin \pi Q} \sum_i \sqrt{\beta_i} \theta_i \cos(\pi Q - |\phi(s) - \phi_i|), \quad (1)$$

where  $s$  denotes the longitudinal location around the ring,  $Q$  is the tune and  $\phi$  is the betatron phase advance. The denominator  $\sin(\pi Q)$  makes closed orbit to diverge at the integer resonance  $Q \in \mathbb{N}$ . The effect of longitudinal misalignments is briefly described in Section 3.4. Another source of orbit errors is offset quadrupoles which is described in the following section.

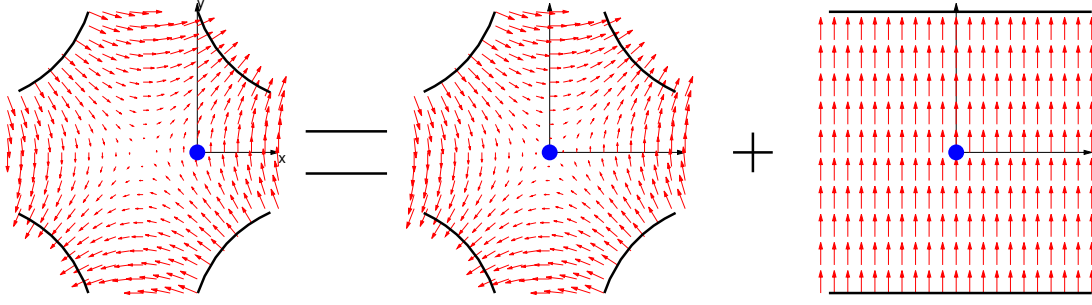
### 3.2 Quadrupole

Figure 7 shows the magnetic and the force fields inside a quadrupole. An offset quadrupole is seen as the superposition of a centered quadrupole plus a dipolar field, as shown in Fig. 8, hence introducing orbit deviations.

If the  $i^{\text{th}}$  quadrupole in a lattice has a gradient error of  $\Delta k_i$  it introduces horizontal and vertical tune deviations given by

$$\Delta Q_x \approx \frac{1}{4\pi} \overline{\beta_x} \Delta k_i L_i, \quad \Delta Q_y \approx -\frac{1}{4\pi} \overline{\beta_y} \Delta k_i L_i, \quad (2)$$

where  $L_i$  is the length of the quadrupole and  $\overline{\beta_{x,y}}$  stands for the average  $\beta_{x,y}$  function in the quadrupole. At the same time the gradient error also introduces  $\beta$ -beating. In presence of many quadrupolar errors



**Fig. 8:** An offset quadrupole is seen as a centered quadrupole plus a dipole.

the  $\beta$ -beating can be expressed as

$$\frac{\Delta\beta}{\beta}(s) \approx \pm \sum_i \frac{\Delta k_i L_i \bar{\beta}_i}{2 \sin(2\pi Q)} \cos(2\pi Q - 2|\phi(s) - \phi_i|), \quad (3)$$

where the positive sign stands for the horizontal plane, and negative for the vertical. The denominator  $\sin(2\pi Q)$  makes the  $\beta$ -beating diverge at the integer and half integer resonances,  $2Q \in \mathbb{N}$ . Betatron phase deviations between two locations in the accelerator,  $s$  and  $s_0$ , can be computed using the fundamental relation  $1/\beta = d\phi/ds$ , yielding

$$\Delta\phi(s_0, s) = \int_{s_0}^s \frac{ds'}{\beta(s')} \left( \frac{1}{1 + \frac{\Delta\beta}{\beta}(s')} - 1 \right). \quad (4)$$

More explicit first and higher order expansions of the phase beating can be found in [6–8]. Resonance driving terms,  $h_{jklm}$ , appear in the expansion of the Hamiltonian and characterize the strength of resonances,  $(k-j)Q_x + (m-l)Q_y = P$ , with  $P$  any integer.  $h_{jklm}$  are connected to the generating function resonance driving terms  $f_{jklm}$  via the following relation,

$$f_{jklm} = \frac{h_{jklm}}{1 - e^{i2\pi[(k-j)Q_x + (m-l)Q_y]}}, \quad (5)$$

where the denominator reveals the resonant behaviour. One way to explore higher order perturbations in the  $\beta$ -beating is via the the generating function resonance driving term  $f_{2000}$  (in the horizontal plane), that is defined as

$$f_{2000}(s) = \frac{\sum_j \Delta k_j L_j \bar{\beta}_{x,j} e^{2i\phi_{x,j}}}{1 - e^{4i\pi Q_x}} + \mathcal{O}(\Delta k^2), \quad (6)$$

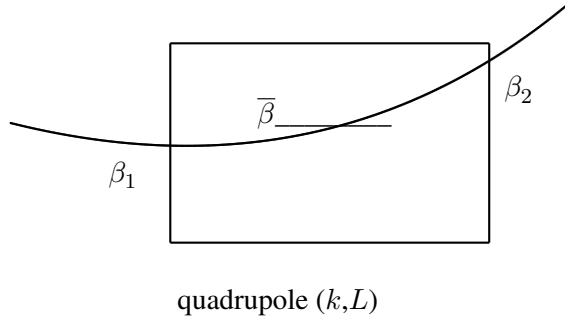
where  $\phi_{x,j}$  is cycled so to start from 0 at  $s$ . The  $\beta$ -beating can be expressed as function of  $f_{2000}$  via the following equation [8],

$$\frac{\Delta\beta}{\beta}(s) = 2 \sinh |f_{2000}| \left( \sinh |f_{2000}| + \cosh |f_{2000}| \sin \phi_{2000} \right), \quad (7)$$

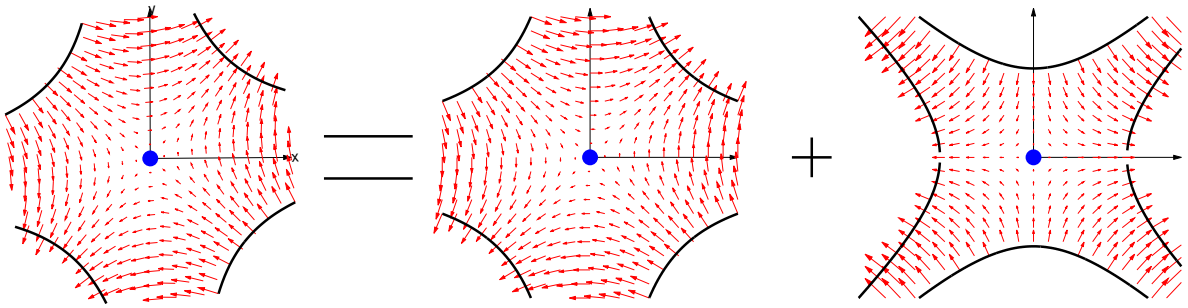
where  $\phi_{2000}$  is the phase of  $f_{2000}$ . Similar equations hold for the vertical plane with  $f_{0020}$ .

Figure 9 shows a sketch of the variation of  $\beta$  function along a quadrupole, displaying the the average  $\beta$  function as  $\bar{\beta}$  and the  $\beta$  at the edges as  $\beta_{1,2}$ . An approximation of  $\bar{\beta}$  is given in [9] as

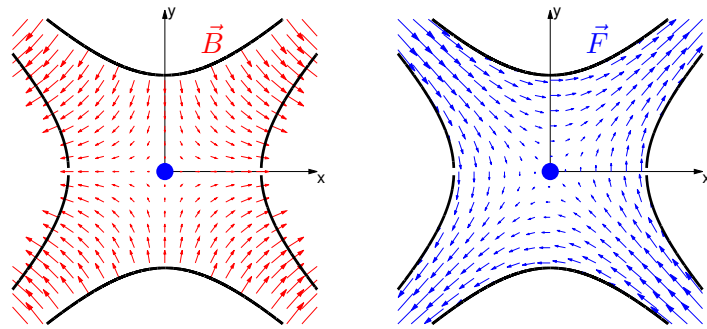
$$\bar{\beta} \approx \frac{1}{3} \left( \beta_1 + \beta_2 + \sqrt{\beta_1 \beta_2 - L^2} \right). \quad (8)$$



**Fig. 9:** Variation of  $\beta$  function along a quadrupole, displaying the the average  $\beta$  function as  $\bar{\beta}$ .



**Fig. 10:** A tilted quadrupole is seen as a normal quadrupole plus another quadrupole tilted by  $45^\circ$ .



**Fig. 11:** Skew quadrupole magnetic field (left) and force (right).

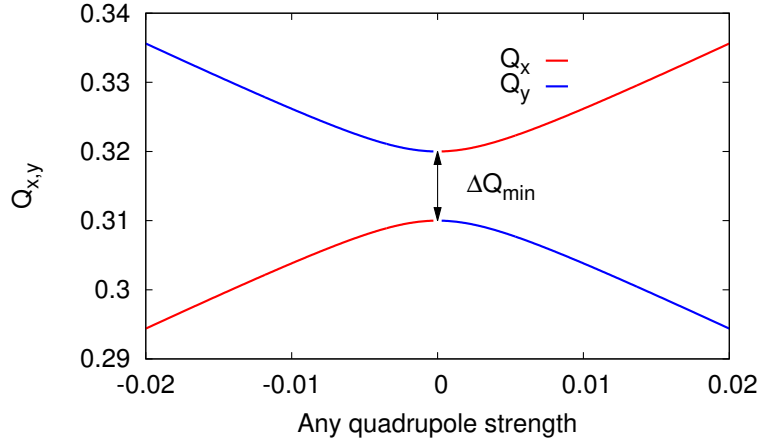
Exact equations for  $\bar{\beta}$  depend also on the quadrupole strength  $k$  as shown in [10, 11].

A tilted quadrupole is seen as a normal quadrupole plus another quadrupole tilted by  $45^\circ$ , which is called a skew quadrupole, see Fig. 10. The magnetic and force fields of a skew quadrupole are shown in Fig. 11.

As shown in Fig. 11 particles displaced horizontally in a skew quadrupole receive a vertical force. This causes the particle motion to couple between the horizontal and vertical planes. While the uncoupled betatron motion of the particle position at turn  $N$  and location  $s$  is simply expressed as

$$x(N, s) = \sqrt{\beta_x(s)} \epsilon_x \cos(2\pi Q_x N + \phi_x(s) + \phi_{x0}), \quad (9)$$

with  $\epsilon_x$  being the horizontal single particle emittance and  $\phi_0$  the phase at  $N = 0$  and  $s = 0$ , the coupled



**Fig. 12:** Approaching tunes in presence of coupling yielding to mode veering.

motion in presence of skew quadrupolar fields can be approximated as [12, 13]

$$x(N, s) \approx \sqrt{\beta_x(s)} \Re \left\{ \sqrt{\epsilon_x} e^{i(2\pi Q_x N + \phi_x(s) + \phi_{x0})} - 2i f_{1010} \sqrt{\epsilon_y} e^{-i(2\pi Q_y N + \phi_y(s) + \phi_{y0})} - 2i f_{1001} \sqrt{\epsilon_y} e^{i(2\pi Q_y N + \phi_y(s) + \phi_{y0})} \right\},$$

where  $\Re\{x\}$  stands for the real part of  $x$ , and  $f_{1010}$  and  $f_{1001}$  are the sum and difference generating function resonance driving terms given by

$$f_{1001} = \frac{\sum_j k_{s,j} L_j \sqrt{\beta_{x,j} \beta_{y,j}} e^{i(\phi_{x,j} \pm \phi_{y,j})}}{4(1 - e^{2\pi i(Q_x \pm Q_y)})}, \quad (10)$$

where  $k_{s,j}$  represents the  $j^{\text{th}}$  skew quadrupole gradient in the machine.  $f_{1001}$  drives the difference resonance  $Q_x - Q_y = P$  and  $f_{1010}$  drives the sum resonance  $Q_x + Q_y = N$ , for any  $P \in \mathbb{Z}$  and  $N \in \mathbb{N}$ .

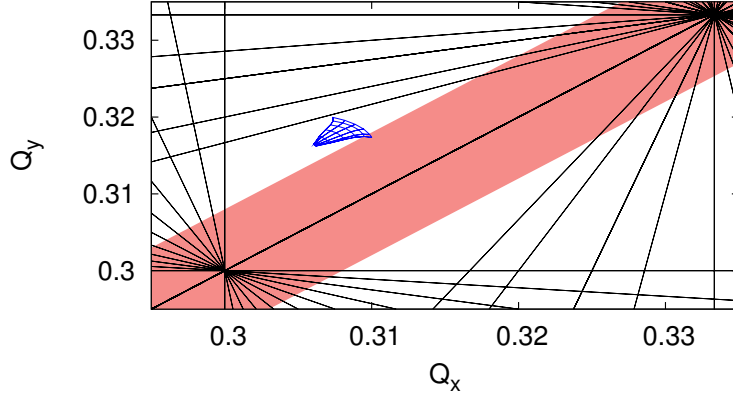
Another important feature of coupled motion is the appearance of a stopband around the difference resonance  $Q_x - Q_y = P$ ,  $P \in \mathbb{Z}$ . This implies that the fractional tunes cannot get closer than  $\Delta Q_{\min}$ , the closest tune approach, given by [14]

$$\Delta Q_{\min} = \left| \frac{1}{2\pi} \sum_j k_{s,j} L_j \sqrt{\beta_x \beta_y} e^{-i(\phi_x - \phi_y) + is(\hat{Q}_x - \hat{Q}_y)/R} \right|, \quad (11)$$

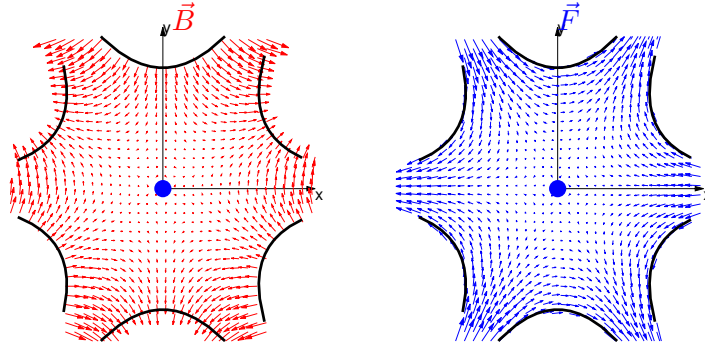
where  $k_{s,j}$  represents the skew quadrupolar gradients around the ring,  $R$  is the machine radius and  $\hat{Q}_{x,y}$  are the fractional tunes.  $\Delta Q_{\min}$  can also be computed from  $f_{1001}$  around the ring by [15, 16]

$$\Delta Q_{\min} = \left| \frac{4(\hat{Q}_x - \hat{Q}_y)}{2\pi R} \oint ds f_{1001} e^{-i(\phi_x - \phi_y) + is(\hat{Q}_x - \hat{Q}_y)/R} \right| \lesssim 4|\hat{Q}_x - \hat{Q}_y| \overline{|f_{1001}|}, \quad (12)$$

where  $\overline{|f_{1001}|}$  represents the ring average of  $|f_{1001}|$ . As an illustration, a hypothetical large coupling stopband (in red) would limit the tune space available for the LHC beam-beam tune footprint as shown in Fig. 13.



**Fig. 13:** A hypothetical large coupling stopband in red limiting the tune space available to the beam in presence of beam-beam tune footprint. Black lines represent tune resonances to be avoided.



**Fig. 14:** Sextupole magnetic field (left) and force (right).

### 3.3 Sextupole

Figure 14 shows the field and force fields in a sextupole. The equations describing the horizontal and vertical forces in a sextupole are given by

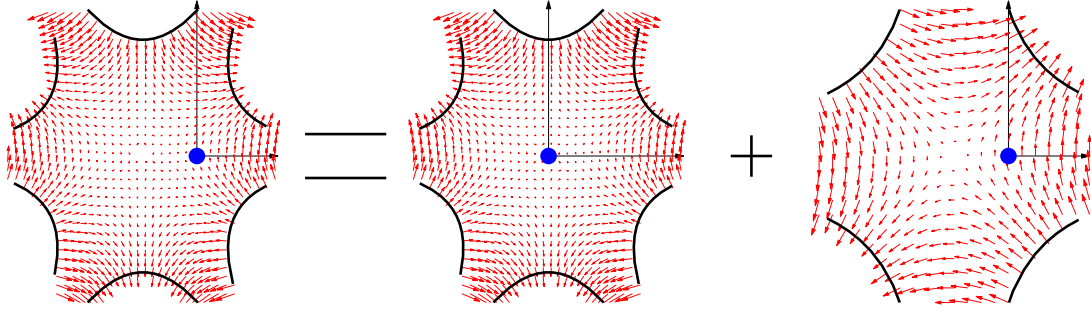
$$F_x = \frac{1}{2}K_2(x^2 - y^2), \quad F_y = -K_2xy, \quad (13)$$

where  $K_2$  is the integrated sextupolar gradient. Sextupoles are needed in accelerators to compensate chromaticity ( $Q'$ ), that describes the dependence of the tune with the relative energy deviation of the particle:  $Q' = dQ/d\delta$ , with  $\delta = (p-p_0)/p_0$ . An offset sextupole is seen as a centered sextupole together with an offset quadrupole, see Fig. 15. Horizontal offsets in sextupoles generate normal quadrupole perturbations while vertical offsets generate skew quadrupolar fields.

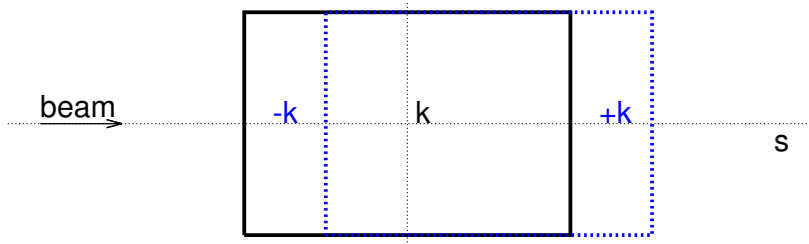
### 3.4 Longitudinal misalignments

Longitudinal misalignments can be approximated as thin perturbations at both ends of the ideal magnet with opposite signs as shown in Fig. 16. For a longitudinal misalignment by  $\delta$  of an element of strength  $k$  the thin perturbations have integrated strengths of  $\pm\delta k$ . Tolerances are generally larger for longitudinal misalignments as there is usually a partial compensation of the perturbations generated at both ends thanks to the opposite signs.





**Fig. 15:** A sextupole horizontally (vertically) displaced is seen as a centered sextupole plus an offset quadrupole (skew quadrupole).



**Fig. 16:** Approximating a longitudinal misalignment of an element (blue) by kicks at the edges of the unperturbed element (black).

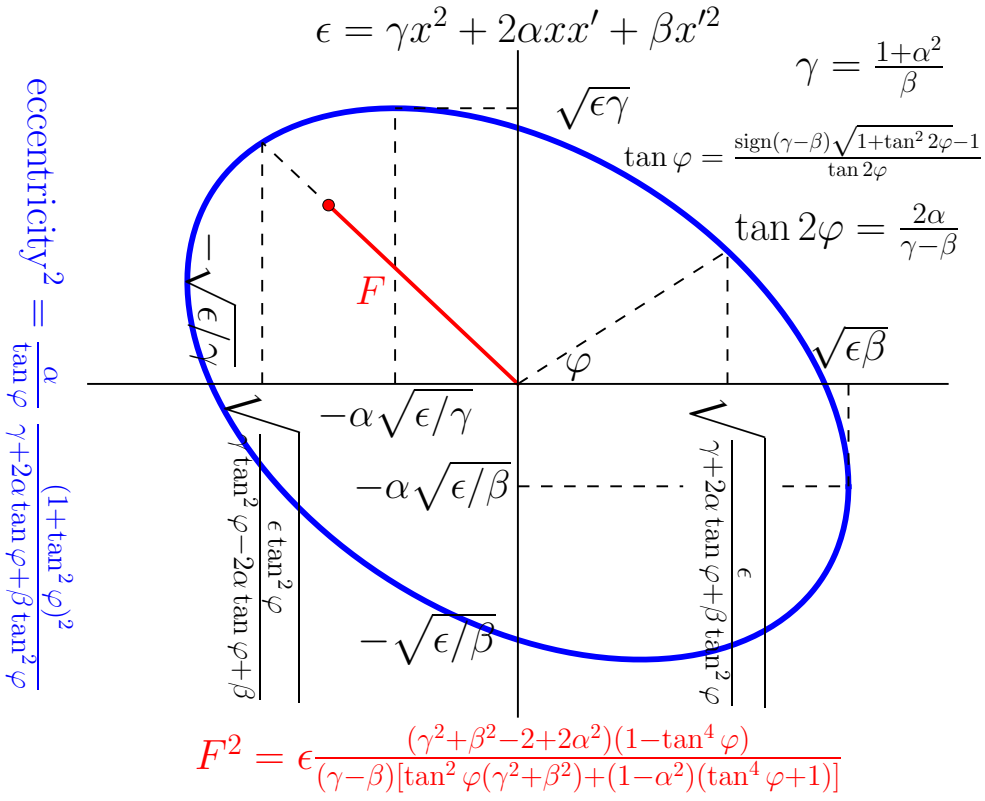
## 4 Phase-space and turn-by-turn motion

### 4.1 The transverse phase-space

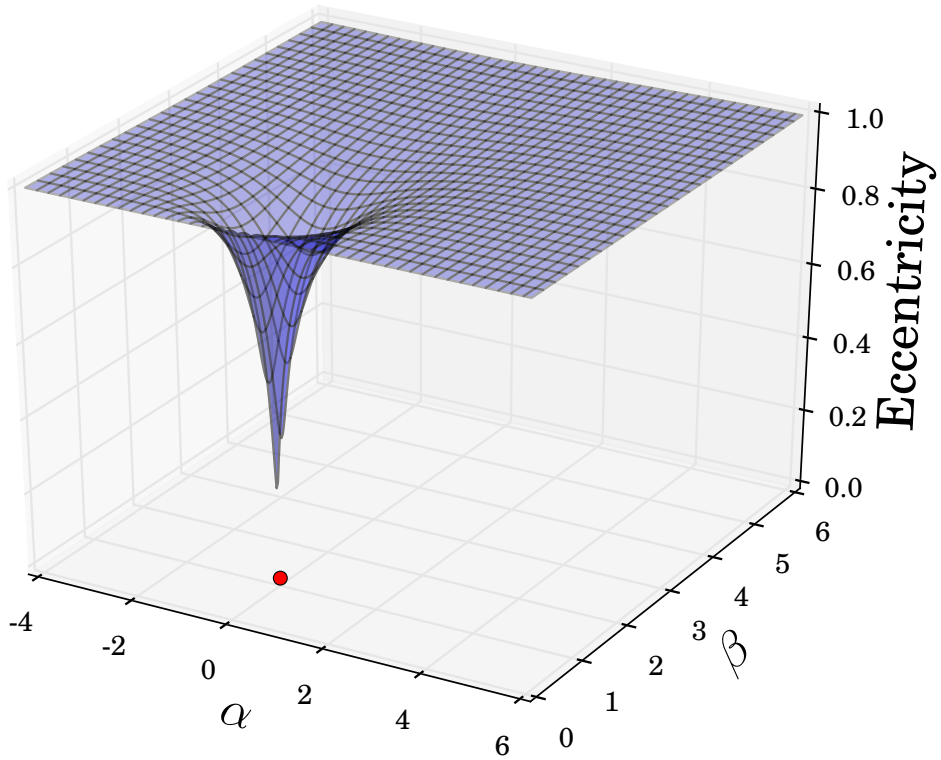
At any location of the accelerator the turn-by-turn uncoupled motion is represented by the position of the particle and its angle with respect to the longitudinal direction, i.e.  $x' = dx/ds$ , which are parametrized as follows

$$\begin{aligned} x(N) &= \sqrt{\epsilon\beta} \cos(2\pi QN + \phi_0), \\ x'(N) &= -\alpha\sqrt{\epsilon/\beta} \cos(2\pi QN) + \sqrt{\epsilon/\beta} \sin(2\pi QN + \phi_0), \end{aligned} \quad (14)$$

where  $\alpha = -\beta'/2$ . The particle trajectory stays within an ellipse in the phase-space  $(x, x')$ . This trajectory is shown in Fig. 17 along with several relevant parameters of the motion and the ellipse, as the angle of the principal direction of the ellipse  $\varphi$ , the coordinates of the intersections of the ellipse and the axes, the largest excursions in  $x$  and  $x'$ , the eccentricity of the ellipse and its focal length  $F$ . The eccentricity of a conic section is a non-negative real number that uniquely characterizes its shape. More formally two conic sections are similar if and only if they have the same eccentricity. One can think of the eccentricity as a measure of how much a conic section deviates from being circular. In particular the eccentricity of a circle is zero and the eccentricity of an ellipse is greater than zero and smaller than 1. Figure 18 shows the eccentricity of the betatronic ellipse versus  $\alpha$  and  $\beta$ . It is interesting to note that the motion is a circle only for  $\beta = 1$  and  $\alpha = 0$ .



**Fig. 17:** Phase-space ellipse  $x'$  versus  $x$ .



**Fig. 18:** Phase-space betatronic ellipse eccentricity versus  $\alpha$  and  $\beta$ . The eccentricity is zero only for  $\alpha = 0$  and  $\beta = 1$ , marked in red.

Expressing Eqs. (14) in matrix form illustrates the transformation that takes a circular motion into the elliptical one as follows,

$$\begin{pmatrix} x(N) \\ x'(N) \end{pmatrix} = \begin{pmatrix} \sqrt{\beta} & 0 \\ -\alpha/\sqrt{\beta} & 1/\sqrt{\beta} \end{pmatrix} \begin{pmatrix} \sqrt{\epsilon} \cos(2\pi QN + \phi_0) \\ \sqrt{\epsilon} \sin(2\pi QN + \phi_0) \end{pmatrix} \quad (15)$$



The circular motion is preferred in many studies for its simplicity and it is referred to as Normal form or Floquet Normal form.

## 4.2 Computing $\alpha$ , $\beta$ and $\epsilon$

When performing computer simulations of particles traveling in an accelerator subject to lattice imperfections or other electrodynamics interactions we have access to the turn-by-turn coordinates  $(x, x')$  and we want to study how the different phenomena perturb the phase-space ellipse. Evaluating  $\alpha$ ,  $\beta$  and  $\epsilon$  is possible by computing the singular value decomposition of the  $2 \times n$  matrix composed of the  $x$  and  $x'$  coordinates for  $n$  turns,

$$\begin{pmatrix} x(1) & x(2) & x(3) & \dots & x(n) \\ x'(1) & x'(2) & x'(3) & \dots & x'(n) \end{pmatrix}_{2 \times n} = U_{2 \times 2} S_{2 \times 2} V_{2 \times n}^T, \quad (16)$$

where  $U$  and  $V$  are unitary matrices and  $S$  is diagonal with non-negative real numbers on the diagonal. The diagonal entries of  $S$  are known as singular values. The columns of  $U$  and the columns of  $V$  are called the left-singular vectors and right-singular vectors respectively. The left-singular vectors are a set of orthonormal vectors and similarly for the right-singular vectors.

$V_{2 \times n}$  represents the turn-by-turn motion in a circle (like a Normal form) in an arbitrary phase origin. Therefore, there must be a rotation  $R(\theta)$  that can be inserted in the singular value decomposition as

$$USV^T = USR(\theta)R^{-1}(\theta)V^T, \quad (17)$$

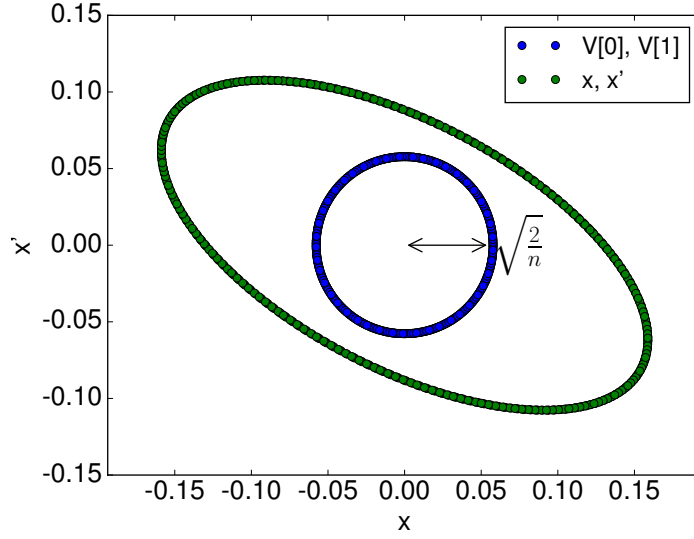
such that  $R^{-1}(\theta)V^T$  corresponds to the Floquet Normal Form, which has as main characteristic that the (1,2) element in the transformation of Eq. (15) is zero. We have to solve the following equation,

$$\frac{1}{\sqrt{\det(S)}}USR(\theta) = \begin{pmatrix} \sqrt{\beta} & 0 \\ -\alpha/\sqrt{\beta} & 1/\sqrt{\beta} \end{pmatrix}, \quad (18)$$

where  $\theta$  is determined to make zero the element (1,2) of  $USR(\theta)$ . The normalization factor  $\sqrt{\det(S)}$  is related to the single particle emittance as follows,

$$\epsilon = \frac{\det(S)}{n/2}, \quad (19)$$

where  $n$  is the number of turns. The following Python code implements the function `getbeta(x, px)` that computes  $\alpha$ ,  $\beta$  and  $\epsilon$  from turn-by-turn data together with an illustrative example. Figure 19 shows the  $(x, x')$  turn-by-turn data used in the code example together with the first two right-singular vectors of  $V$ .



**Fig. 19:** Illustration of the phase space ellipse with  $\alpha = 0.2$ ,  $\beta = 1$ , and  $\epsilon = 2 \times 10^{-3}$  used in the Python code together with the first two right-singular modes of the  $V$  matrix of the singular value decomposition of Eq. (16).

---

```

1 #Computing alfa , beta and epsilon using SVD
2 import numpy as np
3 def getbeta(x,px): # Function to return betx , alfx , ex
4     U, s, V = np.linalg.svd([x,px]) # SVD
5     N = np.dot(U,np.diag(s))
6     theta = np.arctan(-N[0,1]/N[0,0]) # Angle of R(theta)
7     co = np.cos(theta); si = np.sin(theta)
8     R = [ [co, si] , [-si, co] ]
9     X = np.dot(N,R) # Floquet up to 1/det(USR)
10    betx = np.abs(X[0,0]/X[1,1])
11    alfx = X[1,0]/X[1,1]
12    ex=s[0]*s[1]/(len(x)/2.) # emit = det(S)/(n/2)
13    return betx , alfx , ex
14
15 alpha = 0.2 #Example to use getbeta(x,px)
16 beta = 1.
17 ex = 2e-3
18 Q = 0.31
19 Nturns = 600
20 x = np.sqrt(beta*ex)*np.cos(2*np.pi*Q*np.arange(0,Nturns)) #easy tracking
21 px = -alpha*x/beta + np.sqrt(ex/beta)*np.sin(2*np.pi*Q*np.arange(0,Nturns))
22 betx , alfx , exc = getbeta(x,px)

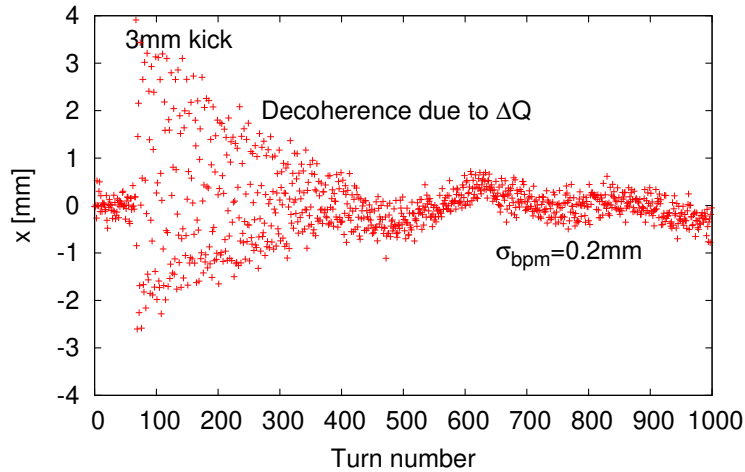
```

---

A first version of this code was developed in [17].

### 4.3 Excitation techniques

In real accelerators Beam Position Monitors (BPMs) measure transverse beam centroid position turn-by-turn, while the angle of the trajectory  $x'$  is not easily accessible. Betatron motion is usually excited via applying a single kick or via a resonant excitation. The single kick technique has the limitation that due to non-linearities not all the particles in the bunch oscillate with the same tune the motion eventually decoheres. The measured turn-by-turn data following a single kick is illustrated in Figure 20. The evolution of the decoherence process is illustrated in Figure 21, where the initial  $1\sigma$  envelope after the kick is shown in green and the deformation of this envelope with time emerges from the larger tune values for particles with larger amplitude (amplitude detuning is typically generated by sextupoles, octupoles or the beam-beam interaction). The decoherence of the beam limits the number of turns available for data analysis and therefore limits the accuracy of beam-based optics measurements.



**Fig. 20:** Illustration of turn-by-turn centroid data recorded by a BPM undergoing decoherence after a transverse kick and with a BPM Gaussian noise of 0.2 mm.

To avoid the limitation from decoherence it is possible to drive betatron oscillations with an AC dipole with a frequency close to the tune. Furthermore this forced oscillation can be ramped up and down adiabatically without causing emittance growth. The AC dipole cycle is illustrated in Fig. 22.

The adiabaticity of the ramping process of an AC dipole [18] can be easily simulated with the following code, which is used to produce the plot in Fig. 23 that compares the particle turn-by-turn motion for two AC dipole ramp-up lengths, 10 and 1000 turns, showing a lack of adiabaticity for the 10 turn ramp. The lack of adiabaticity implies energy transfer to the natural betatron motion with tune equal 0.31 as shown in the spectral components of particle motion in Fig. 24 as computed in the following example code.

---

```

1 #Simulating the AC dipole
2 from numpy import *
3 import matplotlib.pyplot as plt
4
5 Q = 0.31      # Machine tune (fractional part)
6 Qac = Q + 0.02 # AC dipole tune
7 q = 2*pi*Q
8 R = array([[ cos(q), -sin(q)],[ sin(q),  cos(q)]] #1 turn map
9 x=[[0.,0.]] # initial x, px
10 Nramp = 1000 # Number of turns to ramp up AC dipole strength
11 Nturn = 2048 # Number of turns to track
12
13 def ramp(j): # define the AC dipole linear ramp
14     return min(1, j*1.0/Nramp)
15
16 for i in range(Nturn): # tracking loop R with AC dipole kick
17     x.append( dot(R,x[-1]) + ramp(i)*array([0, 0.1*cos(Qac*i*2*pi)]) )
18 F = fft.fft(array(x)[Nramp:].T[0]) # FFT data after AC ramp

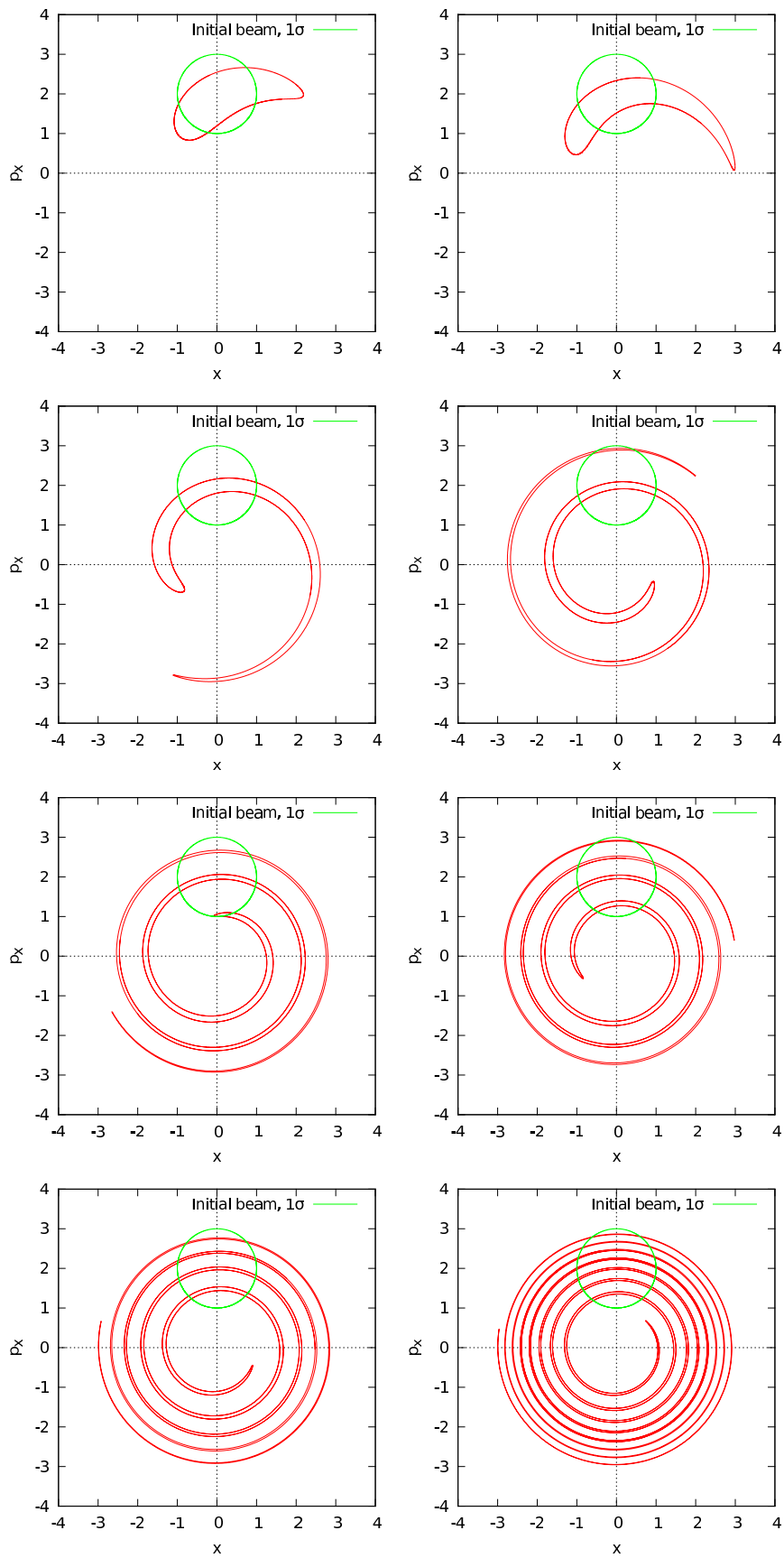
```

---

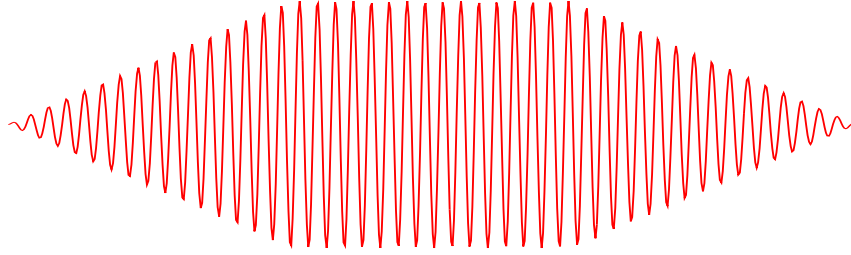
## 5 Measurement techniques and data analysis

### 5.1 Cleaning experimental BPM data

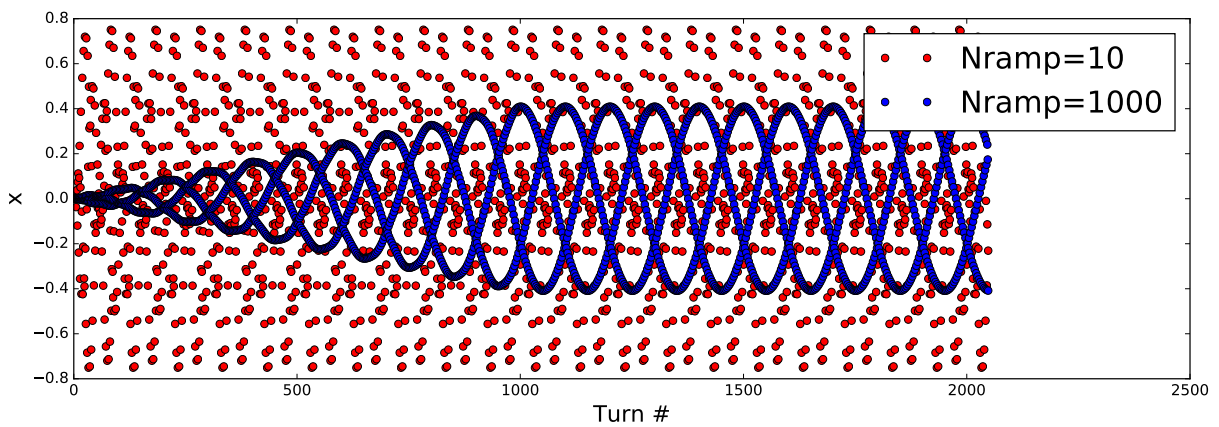
The BPM turn-by-turn data is fundamental to measure optics parameters around the accelerator. Betatron oscillations represent highly correlated signals among BPMs. This feature can be used to reduce the BPM noise by discarding the signals with low correlation levels. SVD is used for this purpose. Imagine  $R$  is



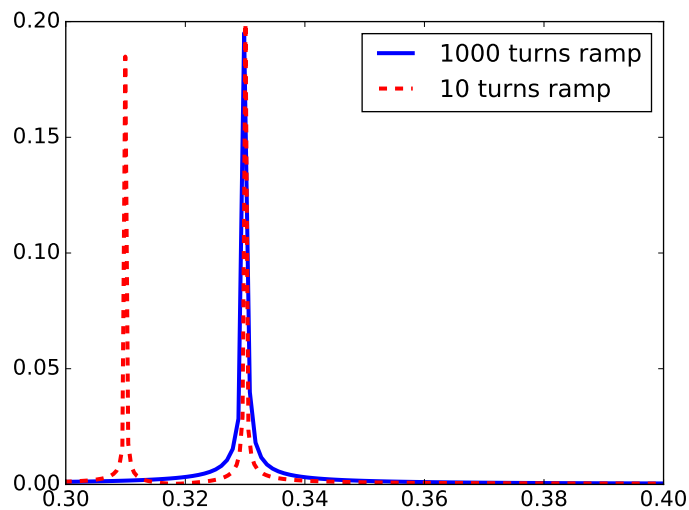
**Fig. 21:** Illustration of the decoherence process.



**Fig. 22:** Illustration of the AC dipole cycle including ramp-up, plateau and ramp-down.



**Fig. 23:** Simulated turn-by-turn beam data during the AC dipole excitation for two different ramp lengths of 10 and 1000 turns, showing the relevance of an adiabatic excitation.



**Fig. 24:** Spectrum of the simulated turn-by-turn beam data during the AC dipole plateau following two different ramp lengths of 10 and 1000 turns, showing the appearance of the natural tune for the non-adiabatic excitation. AC dipole tune is 0.33 and natural tune is 0.31.

the BPM matrix containing turn-by-turn data for all BPMs and his SVD is given by,

$$R = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \end{pmatrix} V^T. \quad (20)$$

If  $\sigma_3 \ll \sigma_2 \leq \sigma_1$ , then we can neglect  $\sigma_3$  by making  $\sigma_3 = 0$  and reconstruct  $R$  losing a negligible amount of information. Denoting the reconstructed matrix as  $R_{denoised}$ , it is given by the following equation,

$$R_{denoised} = U \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} V^T. \quad (21)$$

This technique is illustrated with the following Python code and in Fig. 25. In the code turn-by-turn data is simulated with very low tunes to produce pictures that can be easily visualized. Random Gaussian noise is added to mimic BPM noise with signal-to-noise ratio varying between 1:0.2 and 2:0.2. The SVD reconstruction is performed by keeping only the two largest singular values. The process is illustrated in Fig. 25 showing the 3 matrices in color code. It is impressive that the reconstructed matrix looks identical to the ideal one before adding the noise. Actually this technique is equally used to denoise digital pictures.

---

```

1 # Denoising BPM signal
2 import matplotlib.pyplot as plt
3 from scipy import misc, ndimage
4 import numpy as np
5 from numpy.linalg import svd
6
7 #Generating ideal Beam Position data
8 im = np.zeros((500, 500))
9 for i in range(500):
10     for j in range(500):
11         amplitudej = 1 + (np.cos(0.00678 * j * 2 * np.pi)) ** 2
12         im[i, j] = amplitudej * np.cos(i * 0.0137 * 2 * np.pi)
13
14 #Adding noise like measurement error
15 im = im + 0.2 * np.random.randn(*im.shape)
16
17 #Denoising with Singular Value Decomposition
18 k=2
19 U, s, V = svd(im, full_matrices=False)
20 rim = np.dot(U[:, :k], np.dot(np.diag(s[:k]), V[:k, :]))

```

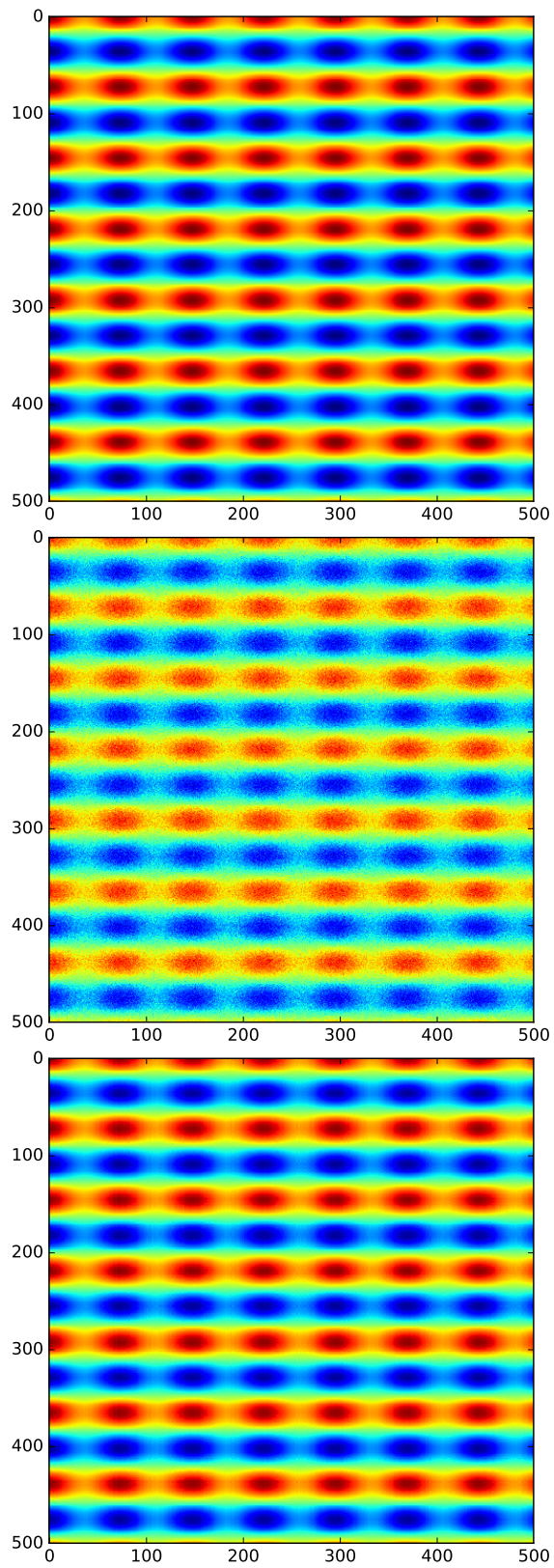
---

Large BPM systems always present some malfunctioning BPMs that need to be removed before the analysis. An example of good and bad BPMs is shown in Fig. 26 from the CERN SPS [19]. The plots in the bottom of the figure show how the Fourier spectrum can be used to identify bad BPMs by looking in regions of the spectra where no beam signal is expected. SVD has also been extensively used to identify bad BPMs [20, 21].

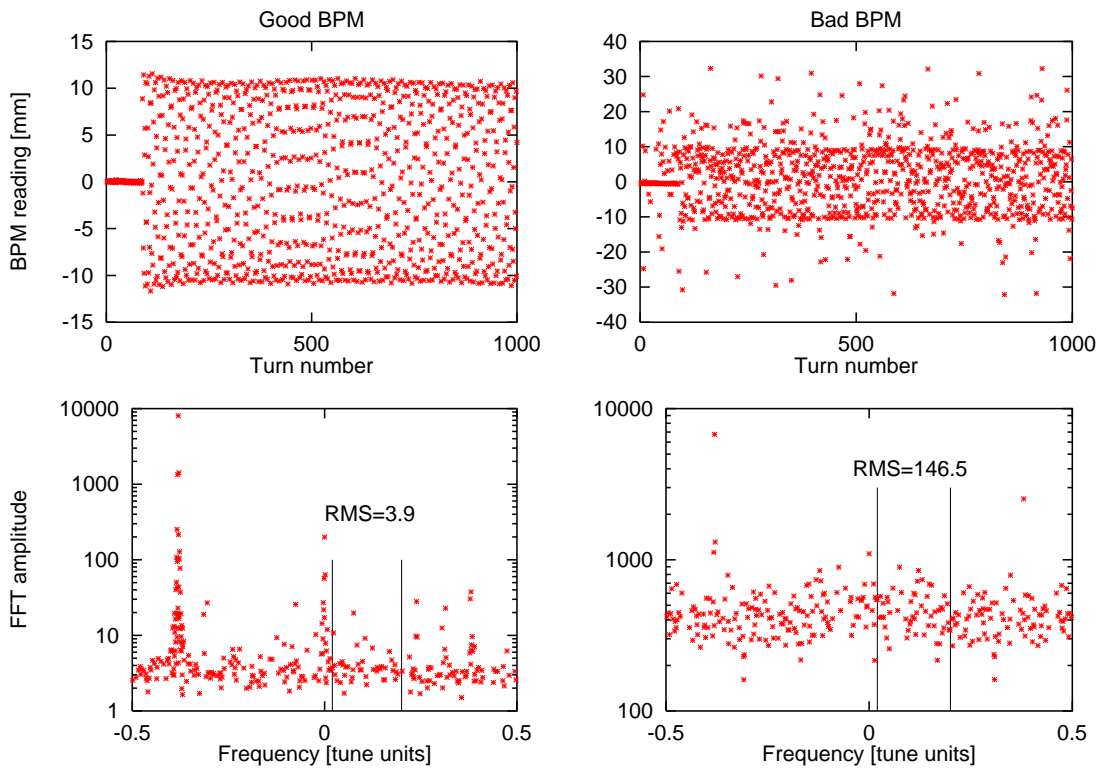
More recently Isolation Forest has been demonstrated to be very effective at finding malfunctioning BPMs as outliers within the distribution of selected features of the BPM data [22, 23]. Figure 27 illustrates the concept of the Isolation Forest algorithm, where random cuts are applied to the data for one randomly selected feature at a time until single data points are isolated. The basic concept is that anomalies require fewer number of cuts to reach isolation. A decision function is established using this number averaged over the number of trees.

The following Python code applies the Isolation Forest to simulated turn-by-turn BPM data with Gaussian noise and five bad BPMs. In this illustration the bad BPMs are chosen to have larger Gaussian noise and a different tune. The features chosen to compute the decision function are amplitude and frequency of the main spectral line. The features for all BPMs are shown in Fig. 28 together with the

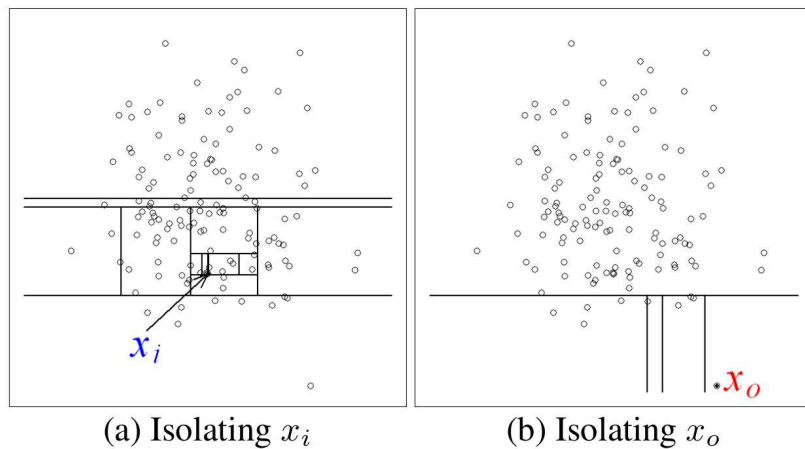




**Fig. 25:** Ideal beam data versus turn number and versus longitudinal location (top), same data adding Gaussian noise (middle) and after cleaning the noise with SVD (bottom).



**Fig. 26:** Good BPM (left) and bad BPM (right) with corresponding spectra (bottom), from [19].



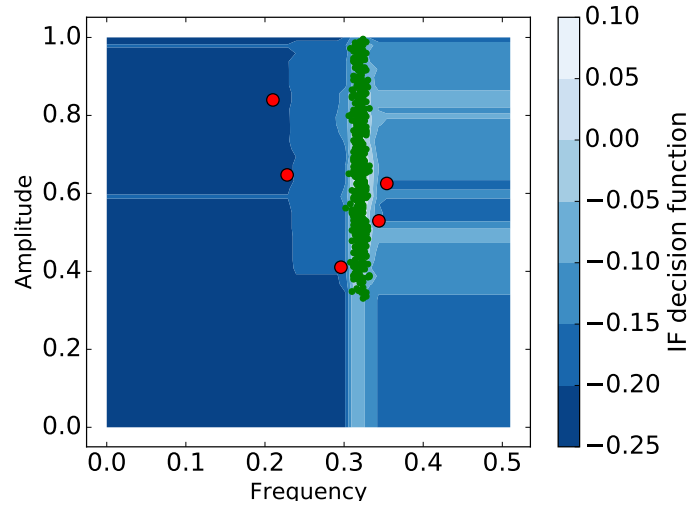
**Fig. 27:** Illustration of the Isolation Forest algorithm applied to a normal data point (left) requiring many cuts to reach isolation and to an anomaly (right) with fewer cuts.

decision function. The red BPMs are the BPMs identified as bad by assuming a contamination factor of 1%.

```

1 # Applying Isolation Forest to detect bad BPMs
2 import numpy as np
3 from sklearn.ensemble import IsolationForest
4
5 N_TURNS = 500
6 N_BPMS = 500
7 # generate bpm data with some bad signal - different tune, additional noise
8 bad_bpms_idx = [1, 10, 20, 30, 40]
9 im = np.zeros((N_TURNS, N_BPMS))
10 for bpm in range(N_BPMS):
11     err = 0.05 * np.random.randn()

```



**Fig. 28:** Isolation Forest applied to BPM data using frequency and amplitude of highest Fourier peak as features.

```

12 amp=(np.cos(0.00678 * bpm * 2 *np.pi) ** 2 + 1) # sqrt(beta e)
13 for turn in range(N_TURNS):
14     if bpm in bad_bpms_idx: # A bad BPM with different tune and noise
15         im[turn ,bpm]=amp*np.cos(turn*(0.32+err)*2*np.pi)+0.3*np.random.randn()
16     else: # Good BPM
17         im[turn ,bpm]=amp*np.cos(turn*(0.32+err/10)*2*np.pi) + 0.1*np.random.randn
18         ↪ ()
19 # extract frequency and amplitude – features – from bpm signal
20 amplitudes = [np.max(x) for x in np.abs(np.fft.rfft(im.T))/N_TURNS]
21 frequencies= np.array([np.argmax(x) for x in np.abs(np.fft.rfft(im.T))])*1.0/N_TURNS
22 features = np.vstack((frequencies , amplitudes)).T
23
24 # fit Isolation Forest model to the data and detect anomalies (contamination is the
25 ↪ fraction of anomalies)
26 iforest = IsolationForest(n_estimators=10, contamination=0.01)
27 outlier_detection = iforest.fit(features).predict(features) # Bad BPMs ==-1

```

## 5.2 Generic measurement cleaning

Most of the measured quantities are assumed to be normally distributed, however in case of failure or an artefact in data processing; outlying values may be produced and should be removed from the data sample. Finite-sized samples of a normal distribution follow a t-student distribution, which is also parametrised by a number of degrees of freedom.

An iterative cleaning procedure (developed in [24]) removes “tails” which are more populated than in the same-sized normally distributed quantity. In each iteration, values are tested for a hypothesis of belonging to a sample of normal distribution given the mean value, standard deviation and sample size (t-distribution). The algorithm represented by the following code can also operate onto two linearly dependent sets. In such a case, the fitted dependency on a second dataset is subtracted in every iteration.

```

1 # Iterative cleaning
2 import numpy as np
3 from scipy.stats import t
4 import matplotlib.pyplot as plt
5
6 def filter_mask(data , x_data=None, limit=0.0, niter=20):
7     mask = np.ones(len(data) , dtype=bool)
8     nsigmas = t.ppf([1 - 0.5 / len(data)], len(data))
9     prevlen = np.sum(mask) + 1
10    for _ in range(niter): # iterate
11        if not ((np.sum(mask) < prevlen) and (np.sum(mask) > 2)):

```

```

12         break
13     prevlen = np.sum(mask)
14     if x_data is not None: # linearly dependent data
15         m, b = np.polyfit(x_data[mask], data[mask], 1)
16         y, y_orig = data[mask] - b - m * x_data[mask], data - b - m * x_data
17     else: # independent data
18         y, y_orig = data[mask], data[:]
19     mask = np.abs(y_orig - np.mean(y)) < np.max([limit, nsigmas * np.std(y)])
20     return mask
21
22 x_data = 100 * np.random.rand(1000)
23 y_data = 0.35 * x_data + np.random.randn(1000) # create data
24 y_data[-100:] = y_data[99::-1] # corrupt some of the data
25 x_data[:50] = 38 + np.random.randn(50)
26 mask = filter_mask(y_data, x_data=x_data)
27 plt.plot(x_data, y_data, 'ro')
28 plt.plot(x_data[mask], y_data[mask], 'bo')

```

---

### 5.3 Fourier analysis

The Fast Fourier Transform (FFT) of a turn-by-turn data sample with  $N$  turns has the following tune ( $Q$ ), amplitude ( $A$ ) and phase ( $\phi$ ) resolutions, respectively

$$\sigma_Q \leq \frac{1}{2N}, \quad \sigma_A \approx \sqrt{\frac{2}{N}}\sigma, \quad \sigma_\phi \approx \sqrt{\frac{2}{N}}\frac{\sigma}{A}, \quad (22)$$

where  $\sigma$  is the BPM random error, assumed to follow a Gaussian distribution.

Many interpolation techniques have been developed to improve the frequency resolution of the FFT [25–27]. Zero-padding is a very simple approach that can significantly improve the determination of fundamental frequencies but is computationally expensive. A Python example using zero-padding follows.

```

1 #FFT with zero padding
2 import numpy as np
3 N = 4096
4 i = 2 * np.pi * np.arange(N)
5 data = np.cos(0.134 * i) + np.cos(0.244 * i) + 0.01 * np.random.randn(N)
6 f_zeropad=np.abs(np.fft.fft(data, n=10*N)/(N))

```

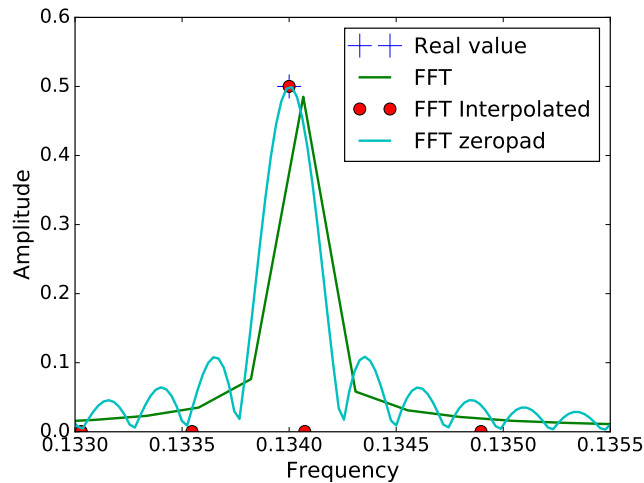
---

The algorithm NAFF [25] finds the frequency  $Q$  that maximizes  $|\sum x(N)e^{i2\pi QN}|$ , where  $x(N)$  is the sample data, and continues to find the next leading frequency after subtracting the found frequency component from  $x(n)$  and iterating. Python and Fortran implementations of NAFF can be found in [28, 29]. The following code, first version of Harpy [30], implements the NAFF algorithm but making a 3 point interpolation (Jacobsen method [31]) rather than maximizing  $|\sum x(N)e^{i2\pi QN}|$ . Figure 29 shows the spectrum of the signal in the Python example computed with different approaches around the main frequency 0.134. The plain FFT gives, as expected, the worst performance in identifying the spectral line. Interpolating with Jacobsen method [31] or zero padding give similar results in this example. The later version of Harpy [32] implements zero padding and reduces the computational costs by a combination with SVD. Moreover, the combination with SVD allows to estimate errors in the frequency spectra.

```

1 # First version of Harpy implementing NAFF with Jacobsen interpolation
2 import numpy as np
3 PI2I = 2 * np.pi * complex(0, 1)
4
5 def harpy(samples, num_harmonics):
6     n = len(samples)
7     int_range = np.arange(n)
8     coefficients = []
9     frequencies = []
10    for _ in range(num_harmonics):
11        frequency = _jacobsen(np.fft.fft(samples), n) # Find dominant freq.
12        exponents = np.exp(-PI2I * frequency * np.arange(n))
13        coef = np.sum(exponents*samples)/n # compute amplitude and phase
14        coefficients.append(coef)

```



**Fig. 29:** Illustration of different algorithms to find the main spectral frequencies.

```

15     frequencies.append(frequency)
16     new_signal = coef * np.exp(PI2I * frequency * int_range)
17     samples = samples - new_signal # Remove dominant freq.
18     coefficients, frequencies = zip(*sorted(zip(coefficients, frequencies),
19     key=lambda tuple: np.abs(tuple[0]), reverse=True))
20     return frequencies, coefficients
21
22 def _jacobsen(dft, n): # Interpolate to find dominant freq.
23     k = np.argmax(np.abs(dft))
24     delta = np.tan(np.pi / n) / (np.pi / n)
25     kp = (k + 1) % n
26     km = (k - 1) % n
27     delta = delta * np.real((dft[km] - dft[kp]) / (2 * dft[k] - dft[km] - dft[kp]))
28     return (k + delta) / n
29
30 N=4096
31 i = 2 * np.pi * np.arange(N)
32 data = np.cos(0.134 * i) + np.cos(0.244 * i) + 0.01 * np.random.randn(4096)
33 freqs, coeffs = harpy(data, 300)

```

### 5.3.1 Phase measurement

The phase advance between 2 BPMs  $\phi_{ij} = \phi_j - \phi_i$  is a fundamental optics observable, it is model and BPM calibration independent. Care with averaging several measurements is needed due to periodicity, i.e. circular mean has to be used. For  $n$  measurements of certain angle or phase,  $\alpha_i$ , the circular mean is defined as

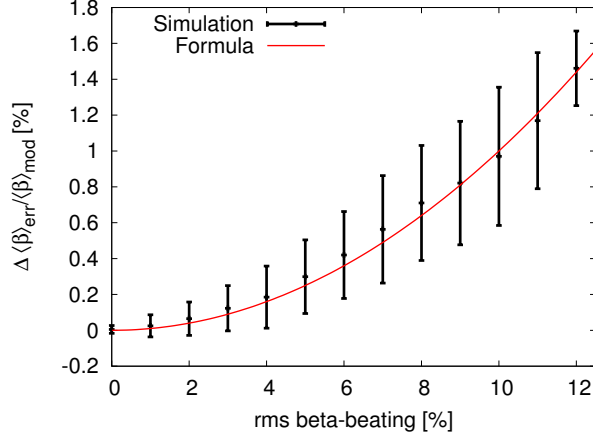
$$\bar{\alpha} = \text{atan2} \left( \frac{1}{n} \sum_i^n \sin \alpha_i, \frac{1}{n} \sum_i^n \cos \alpha_i \right), \quad (23)$$

and in Python it is simply computed using an existing function as shown in the following example code by computing the circular mean between 0 and  $2\pi$ , which is not  $\pi$  but 0.

```

1 #Computing the circular mean of 0 and 2pi
2 from scipy.stats import circmean
3 import numpy as np
4 circmean([0., 2*np.pi])

```



**Fig. 30:** Relative deviation of the average  $\beta$  function in presence of random errors versus the corresponding  $\beta$ -beating together with the prediction from Eq. (24).

#### 5.4 $\beta$ from amplitude

The average of the  $\beta$  function around the ring in presence of random errors is related to the rms  $\beta$ -beating via the following expression [33],

$$\left\langle \frac{\Delta\beta}{\beta} \right\rangle = \text{rms}^2 \left( \frac{\Delta\beta}{\beta} \right) \quad (24)$$

Figure 30 shows the ring average  $\beta$  function versus its rms value for many realizations of the LHC with random errors. In average random errors increase the beta-functions around the ring implying that random errors are defocusing.

As shown in Eq. (9) the amplitude of betatron oscillation at the location  $s$  is  $\sqrt{\beta(s)}\epsilon$ . Having enough BPMs around the ring allows to compute the average and rms of  $\beta\epsilon$  from the square of the FFT amplitude of the tune line.  $\epsilon$  can be computed with

$$\epsilon \approx \frac{\langle \beta\epsilon \rangle}{\langle \beta_{\text{model}} \rangle} \left( 1 - \text{rms}^2 \left( \frac{\Delta\beta}{\beta} \right) \right), \quad (25)$$

where the numerator comes from measurement, the denominator is the average model  $\beta$  function and the parenthesis corrects for the possible average  $\beta$ -beating. Biggest limitation of this technique is BPM calibration errors. After computing  $\epsilon$  it is possible to extract the  $\beta$  function at every BPM using the amplitude of the tune line. The main limitation of this method is relying on a good gain calibration of BPMs.

#### 5.5 $\beta$ from phase

It is possible to compute the  $\beta$  function at one BPM by using the phase advances between that BPM and another 2 BPMs as follows [34],

$$\beta_1^{\text{meas}} = \beta_1^{\text{mod}} \frac{\cot \Delta\phi_{1,2}^{\text{meas}} - \cot \Delta\phi_{1,3}^{\text{meas}}}{\cot \Delta\phi_{1,2}^{\text{mod}} - \cot \Delta\phi_{1,3}^{\text{mod}}}$$

This is known as the 3 BPM method, which was later extended to N BPMs in [35] relying on Montecarlo simulations and made fully analytical in [36] with a considerable improvement in speed.

## 5.6 Momentum reconstruction and resonance driving terms

BPMs only measure the centroid position. The angle of the trajectory can be computed from two BPMs separated by a drift, but there are usually very few BPMs placed in such configuration in an accelerator. Normalizing the turn-by-turn BPM signal by the amplitude of the tune line we define the normalized coordinate  $\hat{x}$ , which for two nearby BPMs can be parametrized as follows,

$$\begin{aligned}\hat{x}_1(N) &= \cos(2\pi Q_x N + \phi_1), \\ \hat{x}_2(N) &= \cos(2\pi Q_x N + \phi_2).\end{aligned}$$

We can reconstruct the normalized  $p_x$  at the first BPM as

$$\hat{p}_{x1}(N) = \sin(2\pi Q_x N + \phi_1) = \frac{\hat{x}_2(N)}{\cos \delta} + \hat{x}_1(N) \tan \delta, \quad (26)$$

with  $\delta = \phi_2 - \phi_1 - \pi/2$ . Note that when the phase advance between the 2 BPMs is  $\pi/2$  then  $\hat{p}_{x1}(N) = \hat{x}_2(N)$ , and when the phase advance is  $\pi$  the equation diverges.  $\hat{x}_1$  and  $\hat{p}_{x1}$  can be used to plot the particle trajectory in the phase space up to a constant. Non-linearities deform this trajectories from ellipses to possibly very complex shapes. Using Normal Form the turn-by-turn motion can be described in terms of the generating function terms  $f_{jklm}$  as [12]

$$\begin{aligned}\hat{x}_1 - i\hat{p}_{x1} &= e^{i2\pi Q_x N} - \\ &2i \sum j f_{jklm} \epsilon_x^{\frac{j+k-2}{2}} \epsilon_y^{\frac{l+m}{2}} e^{i2\pi N[(1-j+k)Q_x + (m-l)Q_y] + i\varphi}.\end{aligned}$$

This equation allows characterizing the non-linear beam dynamics experimentally by measuring the terms  $f_{jklm}$  from the complex Fourier analysis of  $\hat{x}_1 - i\hat{p}_{x1}$  as done in [13, 19, 37, 38].

## 6 Farey sequences

The Farey sequence  $F_n$  of order n is the sequence of completely reduced fractions between 0 and 1 which, when in lowest terms, have denominators less than or equal to  $N$ , which corresponds to the resonances of order  $N$  or lower (in one plane). The Farey sequence of order 5 is given by

$$F_5 = \left\{ \frac{0}{1}, \frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1} \right\} \quad (27)$$

Farey sequences have useful properties. The distance between neighbors in  $F_n$  (aka two consecutive resonances)  $a/b$  and  $c/d$  is equal to  $1/(bd)$ . The next leading resonance in between two consecutive resonances  $a/b$  and  $c/d$  is given by the mediant operation between these two fractions,

$$\frac{a+c}{b+d}.$$

The number of 1D resonances of order  $N$  or lower tends asymptotically to  $3N^2/\pi^2$ . The Farey sequence is efficiently computed in Python as follows,

---

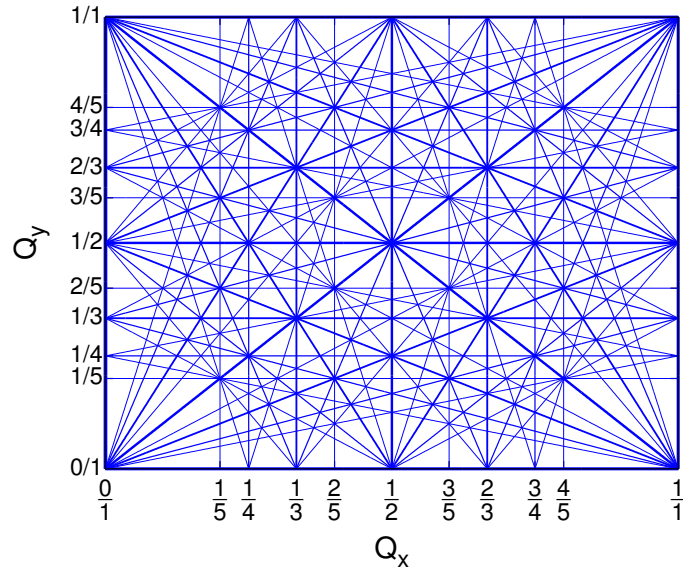
```

1 # The Farey sequence of order n
2 def Farey(n):
3     """Return the nth Farey sequence, ascending."""
4     seq=[[0,1]]
5     a, b, c, d = 0, 1, 1, n
6     while c <= n:
7         k = int((n + b)/d)
8         a, b, c, d = c, d, k*c - a, k*d - b
9         seq.append([a,b])
10    return seq

```

---





**Fig. 31:** Resonance diagram of order 5.

The 2D tune resonance diagram is defined by all solutions of the following equation,

$$aQ_x + bQ_y = p ,$$

with  $a$  and  $b$  and  $p$  integer numbers. These resonance lines are to be avoided in normal operation as some resonance driving terms diverge when approaching them. Figure 31 shows the resonance diagram of order 5. The resonance diagram is also connected to the Farey sequence. The lines going trough  $Q_x = \frac{h}{k}, Q_y = 0$  relate to the elements in  $F_N$  between 0 and  $\frac{1}{k}$  [39]. The number of resonance lines in the 2D diagram is [40]

$$\frac{2N^3}{3\zeta(3)} + O\left(\frac{N^3}{\log N}\right) , \quad (28)$$

where  $\zeta(3) \approx 1.20205$  is the Riemann zeta function evaluated at 3. The relation between the 2D resonance lines and the Farey sequence is most easily explained in the following code example to plot the resonance diagram.

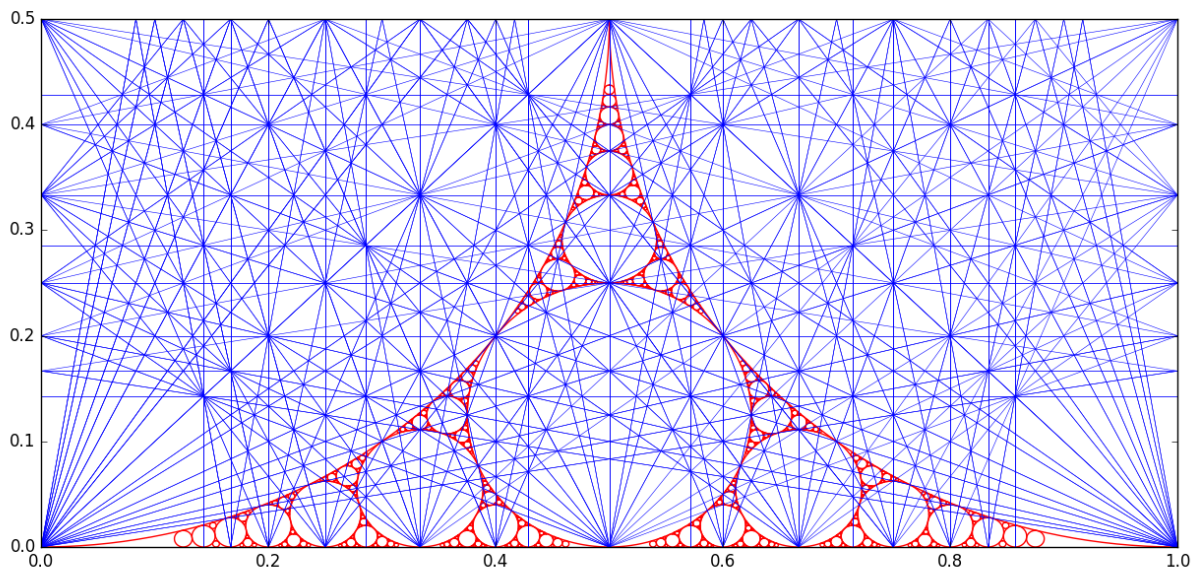
---

```

1 # Plotting the 2D resonance diagram with Farey sequences
2 import matplotlib.pyplot as plt
3 import numpy as np
4 fig = plt.figure()
5 ax = plt.axes()
6 plt.ylim((0,1))
7 plt.xlim((0,1))
8 x = np.linspace(0, 1, 1000)
9 FN = Farey(5) # Farey function defined in the previous code example
10 for f in FN:
11     h , k = f # Node h/k on the axes
12     for sf in FN:
13         p , q = sf
14         c=float(p*h)
15         a=float(k*p) # Resonance line a Qx + b Qy = c linked to p/q
16         b=float(q-k*p)
17         if a>0:
18             plt.plot(x, c/a - x*b/a, color='blue')
19             plt.plot(x, c/a + x*b/a, color='blue')
20             plt.plot(c/a - x*b/a, x, color='blue')
21             plt.plot(c/a + x*b/a, x, color='blue')
22             plt.plot(c/a - x*b/a, 1-x, color='blue')
23             plt.plot(c/a + x*b/a, 1-x, color='blue')

```





**Fig. 32:** Lower half of the resonance diagram and the Apollonian gasket (0,0,1,1).

```

24         if q==k and p==1: # FN elements below 1/k
25             break
26 plt.show()

```

The resonance diagram has also intriguing connections to the Apollonian gasket (0,0,1,1) as shown in Fig. 32.

## 7 Corrections

The goal of corrections is to bring machine optics parameters as close as possible to the design values to ensure machine safety and performance. Corrections are classified as local and global as discussed below.

### 7.1 Local correction

Local corrections are restricted within a predefined segment of the machine. They ensure that the perturbations from the errors within the segment are confined in the segment without significant leakage to the rest of the machine.

The most effective local correction is identifying the source and fixing it. However this can only be applied exceptionally.

In hadron colliders, it is fundamental to perform local corrections in the interaction regions. Two techniques have successfully demonstrated these local corrections: action and phase jump [41, 42] and segment-by-segment [3, 44].

### 7.2 Global corrections

Global corrections use distributed magnets around the ring to minimize deviations of optics parameters from design values. The simplest global corrections use a predefined set of magnets to control a single optics parameters without affecting the others. The set of magnets together with their strength variation is named as *knob* and it is computed using the ideal optics design. Precomputed knobs are primarily used to control orbit, tune and coupling deviations.

The most general correction approach is based on a response matrix between the available correctors and the optics parameters to correct. For efficient use, the measured values are weighted by their errors as well as by quantity-based weights [45]. Phase beating,  $\beta$ -beating, dispersion deviations and tune errors can be put in a vector connected to the normal quadrupole gradient changes  $\vec{k}$  via the matrix  $\mathbf{P}_{\text{theo}}$ . In [43] it is shown that using the normalized dispersion ( $D_x/\sqrt{\beta_x}$ ) in the calculation of corrections improves the correction performance. Coupling resonance driving terms and vertical dispersion connect to skew quadrupole changes  $\vec{k}_s$  via the matrix  $\mathbf{T}_{\text{theo}}$ . These two relations are given in the following equations,

$$\begin{pmatrix} \Delta \vec{\phi}_x \\ \Delta \vec{\phi}_y \\ \frac{\Delta \beta_x}{\beta_x} \\ \frac{\Delta \beta_y}{\beta_y} \\ \Delta \vec{D}_x \\ \Delta \vec{Q} \end{pmatrix}_{\text{meas}} = \mathbf{P}_{\text{theo}} \Delta \vec{k}, \quad \begin{pmatrix} \vec{f}_{1001} \\ \vec{f}_{1010} \\ \vec{D}_y \end{pmatrix}_{\text{meas}} = \mathbf{T}_{\text{theo}} \Delta \vec{k}_s.$$

$\mathbf{P}_{\text{theo}}$  and  $\mathbf{T}_{\text{theo}}$  can be computed by varying one gradient strength at a time and computing the new optics parameters or by collecting large statistics to train the linear regression model [46].  $\mathbf{P}_{\text{theo}}$  and  $\mathbf{T}_{\text{theo}}$  are pseudo-inverted to compute corrections from the measured optics deviations.

### 7.3 The best N corrector problem

The best N corrector problem consists in finding the best  $N$  correctors among the full set of  $M$  correctors, with  $M > N$ . This is very useful when the correction has some cost which increases with the number of correctors or when we aim to localize the error. It is likely that the best 1 corrector is near the error source, although there is no guarantee. The time required to find the exact solution to this problem scales rapidly with  $M$  and  $N$  as

$$t \propto \frac{M!}{(M-N)!},$$

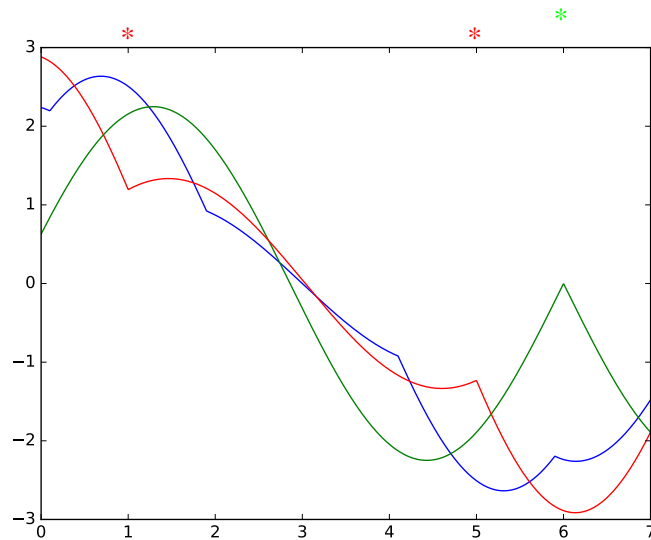
as all possibilities have to be explored and compared. An approximation algorithm to solve this problem is known as Micado [47] by iteratively finding a new best corrector in each step, starting by finding the best 1 corrector, then finding the second best corrector keeping the first one and so on. This problem is also considered in signal theory as a way to decompose signals into a weighted sum of finitely many functions. The algorithm to solve it is very similar to Micado and it is known as matching pursuit or orthogonal matching pursuit (OMP) [48].

The following Python code implements the exact solution of the best  $N$  corrector problem with 7 available correctors and for all values of  $N$  below 7. The orbit generated by the  $i^{\text{th}}$  corrector is approximated by  $\sin(|x-x_i|)$  and the target orbit to match is defined in the `measured_orbit(x)` function.

```

1 # Exact solutions of the best N corrector problem
2 import numpy as np
3 from scipy.optimize import least_squares
4 from itertools import product
5 import matplotlib.pyplot as plt
6
7 N_corr=7
8 s=np.linspace(0, N_corr, 1000) # 1000 observation points
9
10 def corrs(x,i): # Assume correctors at i=integer < N_corr
11     return np.sin(np.abs(x-i))
12
13 def model(x, c): # Orbit at x from corrector strengths as c
14     if len(x)==1:
15         return sum(c*corrs(x,np.arange(N_corr)))
16     return [model([y],c) for y in x ]
17

```



**Fig. 33:** Measured orbit versus longitudinal location (blue) together with resulting orbit using the best 1 corrector (green) and best 2 correctors (red). Location of best correctors is shown with the corresponding color. The best 1 corrector is not within the best two correctors.

```

18 def measured_orbit(x): # Target Orbit
19     return np.sin(np.abs(x-0.1)) + np.sin(np.abs(x-1.9)) - np.sin(np.abs(x-4.1)) -
    ↪ np.sin(np.abs(x-5.9))
20
21 def f(c): #Figure of merit for given corrector choice encoded in mask
22     return model(s, c*mask) - measured_orbit(s)
23
24 best=1e16*np.ones(N_corrs+1) ; bestmask=np.zeros([N_corrs+1,N_corrs])
25 for mask in product([0,1],repeat=N_corrs): # Try all corrector combinations
26     res = least_squares(f, x0=np.ones(N_corrs)) #Orbit correction
27     if res.cost < best[sum(mask)]:
28         bestmask[sum(mask)]=mask*res.x ; best[sum(mask)]=res.cost
29
30 plt.plot(s, measured_orbit(s))
31 plt.plot(s, model(s, bestmask[1])) #Plot best 1 corrector
32 plt.plot(s, model(s, bestmask[2])) #Plot best 2 correctors

```

Figure 33 shows the measured orbit together with the orbit generated by the best 1 and 2 correctors. The problem has been chosen to show that in the exact solution the best 1 corrector is not necessarily within the 2 best correctors.

The following Python code solves the same problem as above by implementing the OMP algorithm using existing Python libraries. The OMP results are shown in Fig. 34, to be compared to the previous exact solution in Fig. 33. Now the best 1 corrector is included in the best 2 correctors, as this solution is only an approximation.

```

1 # Best N corrector problem with Orthogonal Matching Pursuit
2 from sklearn.linear_model import OrthogonalMatchingPursuit
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 N_corrs=7
7 N_BPMs=1000
8 s=np.linspace(0, N_corrs, N_BPMs) # 1000 BPMs
9
10 def corrs(x, i):
11     return np.sin(np.abs(x-i))
12
13 def measured_orbit(x):
14     return np.sin(np.abs(x-0.1)) + np.sin(np.abs(x-1.9)) - np.sin(np.abs(x-4.1)) -
    ↪ np.sin(np.abs(x-5.9))

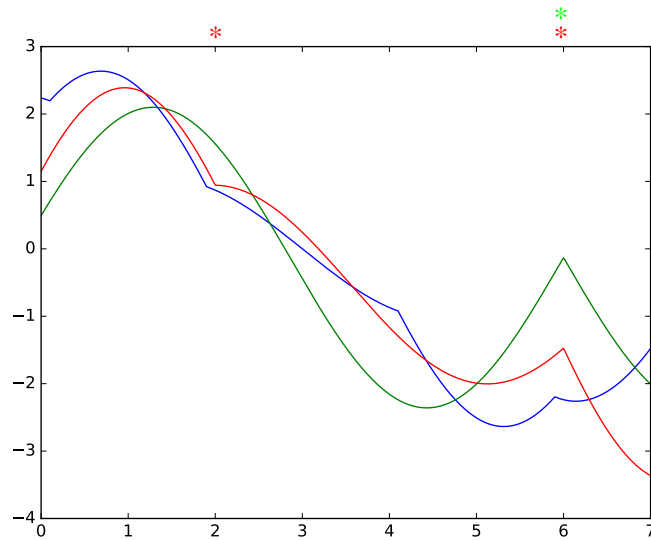
```

```

15
16 ##### New part for OMP #####
17
18 X=[]
19 for i in range(N_BPMs): # Prepare response matrix for OPM
20     X.append(corr(s[i], np.arange(N_corrs)))
21 y= measured_orbit(s)
22 reg = OrthogonalMatchingPursuit(n_nonzero_coefs=1).fit(X, y) #Run OMP for best 1
23     ↪ corr
24 print reg.coef_ # coefficient of best 1 corr
plt.plot(s, reg.predict(X))

```

---



**Fig. 34:** Measured orbit versus longitudinal location (blue) together with resulting orbit after using the best 1 corrector (green) and best 2 correctors (red) using Orthogonal Matching Pursuit. Location of best correctors is shown with the corresponding color. The best 1 corrector is within the best two correctors as this is the main approximation of the algorithm.

## References

- [1] E.D. Courant and H.S. Snyder, *Theory of the Alternating-Gradient Synchrotron*, Annals of Physics **281**, 360-408, received 1957.
- [2] G. Yocky, *Beta-beat correction using strong sextupole bumps in PEP-II*, SLAC-PUB-12523 (2007). <http://www.slac.stanford.edu/pubs/slacpubs/12500/slac-pub-12523.pdf>
- [3] M. Aiba *et al.*, J. Wenninger, F. Zimmermann, R. Calaga, and A. Morita, *First  $\beta$ -beating measurement and optics analysis for the CERN Large Hadron Collider*, Phys. Rev. ST Accel. Beams **12**, 081002 (2009). <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.12.081002>
- [4] R. Tomás, T. Bach, R. Calaga, A. Langner, Y. I. Levinsen, E. H. Maclean, T. H. B. Persson, P. K. Skowronski, M. Strzelczyk, G. Vanbavinckhove, and R. Miyamoto *Record low beta-beating in the LHC*, Phys. Rev. ST Accel. Beams **15**, 091001 (2012). <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.15.091001>
- [5] R. Tomás *et al.*, *Review of linear optics measurement and correction for charged particle accelerators*, Phys. Rev. Accel. Beams **20** 054801 (2017).
- [6] R. Miyamoto, *Diagnostics of the Fermilab Tevatron Using an AC Dipole*, PhD thesis, Uni. of Texas at Austin (2008). <https://repositories.lib.utexas.edu/handle/2152/18189>

- [7] N. Biancacci *et al.*, *Using AC dipoles to localize sources of beam coupling impedance*, Phys. Rev. Accel. Beams **19**, 054001 (2016).  
<http://journals.aps.org/prab/pdf/10.1103/PhysRevAccelBeams.19.054001>
- [8] A. Franchi, *Error analysis of linear optics measurements via turn-by-turn beam position data in circular accelerators*, arXiv:1603.00281 (2016). <https://arxiv.org/abs/1603.00281>
- [9] A. Hofmann and B. Zotter, *Measurement of the  $\beta$ -functions in the ISR*, Issued by: ISR-TH-AH-BZ-amb, Run: 640-641-642 (1975).  
<https://cds.cern.ch/record/1131122/files/CM-P00072144.pdf>
- [10] F. Carlier *et al.*, *Accuracy & Feasibility of the  $\beta^*$  Measurement for LHC and HL-LHC using K-Modulation*, Phys. Rev. Accel. Beams **20** 011005 (2017).
- [11] Paul C. Thrane, Project thesis: *Measuring  $\beta^*$  in SuperKEKB with K Modulation*, Norwegian University of Science and Technology, CERN-THESIS-2018-300.
- [12] F. Schmidt and R. Bartolini, *Normal Form via Tracking or Beam Data*, LHC Project Report 132 (1997).<http://cds.cern.ch/record/333077/files/lhc-project-report-132.pdf>
- [13] R. Tomás *et al.*, *Measurement of global and local resonance terms*, Phys. Rev. ST Accel. Beams **8**, issue 2, 024001. <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.8.024001>
- [14] M. Minty and F. Zimmermann, *Measurement and Control of Charged Particle Beams*, Springer, Berlin (2003).
- [15] Y. Alexahin *et al.*, *Determination of linear optics functions from turn-by-turn data*, Journal of Instrumentation **6**, P10006 (2011).  
<http://iopscience.iop.org/article/10.1088/1748-0221/6/10/P10006/pdf>
- [16] T. H. B. Persson *et al.*, *Improved control of the betatron coupling in the Large Hadron Collider*, Phys. Rev. ST Accel. Beams **17**, 051004.
- [17] P. Goncalves Jorge, *Computation of Optics Distortions due to Beam-Beam Interactions in the FCC-hh*, CERN-THESIS-2016-317.
- [18] R. Tomás, *Adiabaticity of the ramping process of an ac dipole*, Phys. Rev. ST Accel. Beams **8**, 024401 (2005). <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.8.024401>
- [19] R. Tomás, *Direct measurement of resonance driving terms in the SPS of CERN using beam position monitors*, PhD Thesis, Uni. of Valencia (Spain), CERN-THESIS-2003-010, 2003.  
<https://cds.cern.ch/record/615164/files/thesis-2003-010.pdf>
- [20] J. Irwin, C. X. Wang, Y. T. Yan, K. L. F. Bane, Y. Cai, F.-J. Decker, M. G. Minty, G. V. Stupakov, and F. Zimmermann, *Model-Independent Beam Dynamics Analysis*, Phys. Rev. Letters Vol. **82**, Num. 8 (1999). <http://journals.aps.org/prl/pdf/10.1103/PhysRevLett.82.1684>
- [21] R. Calaga *et al.*, *Statistical analysis of RHIC beam position monitors performance*, Phys. Rev. ST Accel. and Beams **7**, 042801 (2004).  
<http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.7.042801>
- [22] E. Fol *et al.*, *Application of Machine Learning to Beam Diagnostics*, presented at IBIC'18, Shanghai, China, Sep. 2018, paper TUOA02.
- [23] E. Fol, *et al.*, *Unsupervised Machine Learning for Detection of Faulty BPMs*, presented at IPAC'19, Melbourne, Australia, May 2019, paper WEPGW081.
- [24] L. Malina, *Novel beam-based correction and stabilisation methods for particle accelerators*, PhD thesis, CERN-THESIS-2018-426 (2018).
- [25] J. Laskar, *The chaotic motion of the solar system: A numerical estimate of the size of the chaotic zones*, Icarus **88**, Issue 2, 1990.
- [26] S. Cetinkaya, *DFT-Based High Resolution Frequency Estimation Using Three Samples*, Eastern Mediterranean University.  
<http://i-rep.emu.edu.tr:8080/xmlui/bitstream/handle/11129/1503/CetinkayaSadi.pdf>

- [27] R. Bartolini *et al.*, *Algorithms for a precise determination of the betatron tune*, CERN-SL-96-048. <https://cds.cern.ch/record/309235/>
- [28] <https://pypi.org/project/PyNAFF/>
- [29] R. Bartolini and F. Schmidt, *SUSSIX: A Computer Code for Frequency Analysis of Non-Linear Betatron Motion*, CERN SL/Note 98-017 (AP), 1998.
- [30] L. Malina, J. Coello de Portugal, J. Dilly, P.K. Skowronski, R. Tomás, M. Toplis, *Performance optimisation of turn-by-turn beam position monitor data harmonic analysis*, IPAC 2018.
- [31] Ç. Candan, *A method For Fine Resolution Frequency Estimation From Three DFT Samples*, IEEE Signal Proces. Lett. **18**, 351-354 (2011).
- [32] L. Malina, *Harpy: A fast, simple and accurate harmonic analysis with error propagation*, to be published.
- [33] R. Tomás *et al.*, *Average beta-beating from random errors*, CERN-ACC-NOTE-2018-0025.
- [34] P. Castro, *Luminosity and beta function measurement at the electron-positron collider ring LEP*, PhD thesis, CERN-SL-96-070-BI (1996). <https://cds.cern.ch/record/316609/files/Thesis-1996-Castro.pdf>
- [35] A. Langner and R. Tomás, *Optics measurement algorithms and error analysis for the proton energy frontier*, Phys. Rev. ST Accel. Beams **18**, 031002 (2015). <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.18.031002>
- [36] A. Wegscheider *et al.*, *Analytical N beam position monitor method*, Phys. Rev. Accel. Beams **20**, 111002 (2017).
- [37] M. Benedikt, F. Schmidt, R. Tomás, P. Urschütz, and A. Faus-Golfe, *Driving term experiments at CERN*, Phys. Rev. ST Accel. Beams **10**, 034002 (2007). <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.10.034002>
- [38] A. Franchi, L. Farvacque, F. Ewald, C. Le Bec, and K.B. Scheidt, *First simultaneous measurement of sextupolar and octupolar resonance driving terms in a circular accelerator from turn-by-turn beam position monitor data*, Phys. Rev. ST Accel. Beams **17**, 074001 (2014). <https://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.17.074001>
- [39] R. Tomás, *From Farey sequences to resonance diagrams*, Phys. Rev. ST Accel. Beams **17**, 014001 (2014). <http://journals.aps.org/prab/abstract/10.1103/PhysRevSTAB.17.014001>
- [40] R. Tomás, *Asymptotic behavior of a series of Euler's totient function  $\varphi(k)$  times the index of  $1/k$  in a Farey sequence*, arXiv:1406.6991. <https://arxiv.org/abs/1406.6991>
- [41] J. Cardona *et al.*, *Comparison of the action and phase analysis on LHC orbits with other techniques*, Proceedings of 2nd International Particle Accelerator Conference, San Sebastian, Spain, edited by C. Petit-Jean-Genaz, 2004-2006 (2011). <http://accelconf.web.cern.ch/accelconf/ipac2011/papers/wepc004.pdf>
- [42] J. Cardona *et al.*, *Local correction of quadrupole errors at LHC interaction regions using action and phase jump analysis on turn-by-turn beam position data*, Phys. Rev. ST Accel. Beams **20** 111004 (2017).
- [43] R. Calaga *et al.*, *BPM calibration independent LHC optics correction*, PAC 2007, p. 3693-3695.
- [44] R. Tomás, O. Brüning, M. Giovannozzi, P. Hagen, M. Lamont, F. Schmidt, G. Vanbavinckhove, M. Aiba, R. Calaga, and R. Miyamoto, *CERN Large Hadron Collider optics model, measurements, and corrections*, Phys. Rev. ST Accel. Beams **13**, 121004 (2010). <http://journals.aps.org/prab/pdf/10.1103/PhysRevSTAB.13.121004>
- [45] T. Persson, *et al.*, *LHC optics commissioning: A journey towards 1% optics control*, Phys. Rev. Accel. Beams **20**, 061002 (2017).
- [46] E. Fol *et al.*, *Optics Corrections Using Machine Learning in the LHC*, IPAC 2019.

- [47] B. Autin, Y. Marti, *Closed orbit correction of A.G. machines using a small number of magnets*, CERN-ISR-MA-73-17 ; ISR-MA-73-17 ; ISR-MA-73-17.
- [48] Y. Pati *et al.*, *Orthogonal Matching Pursuit: recursive function approximation with application to wavelet decomposition*, Asilomar Conf. On Signals, Systems and Comput: 40–44. CiteSeerX 10.1.1.348.5735.